ExperiScope: an analysis tool for interaction data.

Conference Paper · January 2007

DOI: 10.1145/1240624.1240826 · Source: DBLP

Francois Guimbretiere
Cornell University

ExperiScope: An Analysis Tool for Interaction Data

François Guimbretière[a], Morgan Dixon[a], Ken Hinckley[b]

[a] Department of Computer Science
Human-Computer Interaction Lab
University of Maryland,
College Park, MD, 20742
{francois, mdixon3}@cs.umd.edu

[b] Microsoft Research
One Microsoft Way
Redmond, WA 98052
kenh@microsoft.com

ABSTRACT

We present ExperiScope, an analytical tool to help designers and experimenters explore the results of quantitative evaluations of interaction techniques. ExperiScope combines a new visualization incorporating aspects of the KLM and the three-state model with an interface helping users to rapidly cluster similar patterns of interactions. The tool makes it easy to identify and compare key patterns of use encountered during data collection. This promotes a deeper understanding of the results of a given evaluation.

We illustrate the advantages of this tool by revisiting the data collected for an experiment conducted by Hinckley et al. [19] which compared different mode switching techniques. Our results show that our tool complements the previously reported results by offering insights about error behavior and the impact of mode switching on user performance.

By providing a more fine-grained analysis of the data gathered during empirical evaluations, we hope that our tool will improve researchers' understanding of existing and newly developed interaction techniques.

Author Keywords

ExperiScope, Data analysis tool; Interaction design; Empirical evaluation.

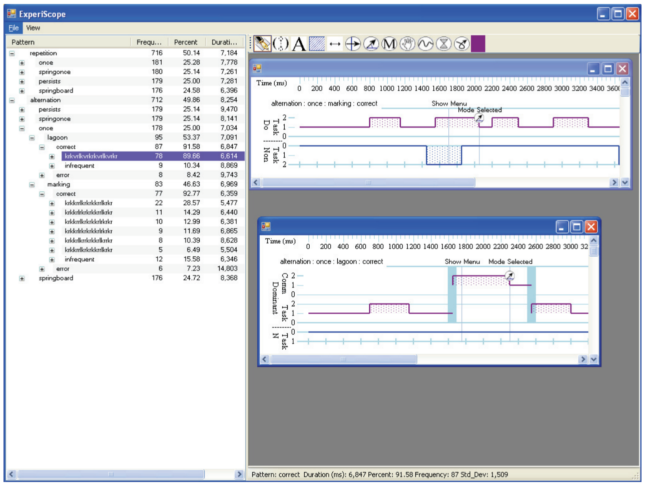
ACM Classification Keywords

H.5.2: User Interfaces— *Evaluation/methodology; input devices and strategies; interaction styles; prototyping.*

INTRODUCTION

As many experimenters can attest, capturing data is often the easiest step in the process of evaluating interaction techniques. The most difficult part is to make sense of the data. A typical controlled experiment can capture in excess of 2000 different interactions, and it is often impossible to check each interaction manually to ensure, for example, that participants followed instructions or that errors were detected correctly. Because analyzing each individual interaction is practically impossible, only gross outliers are usually identified. Once collected, the data is averaged and statistical tools are used to infer which interaction techniques or classes of techniques are faster in a given experimental context. While this approach is well understood and widely accepted, it provides limited insight about the underlying reasons for specific results and does not fully capture the diversity of user behavior. Like Vicente [31], we believe that this approach fundamentally limits the progress of the field. First, it makes it extremely difficult to notice that some interaction patterns are more common than others or that errors fall into different categories. More importantly, it also discourages researchers from capturing interaction data "in the wild" where patterns of interactions are more varied and, as a result, more difficult to analyze. Instead, when designing experimental protocols, researchers struggle to properly balance the need for external validity with the need to study interactions that are simple enough to be easily analyzed. Despite such efforts, controlled experiments are often dismissed as unrealistic.

To address this problem we are proposing ExperiScope, a new analysis tool to help interaction designers better



Pattern	Frequ...	Percent	Durati...
repetition	716	50.14	7,184
once	181	25.28	7,778
springonce	180	25.14	7,261
persists	179	25.00	7,281
springboard	176	24.58	6,396
alternation	712	49.86	8,254
persists	179	25.14	9,470
springonce	179	25.14	8,141
once	178	25.00	7,034
lagoon	95	53.37	7,091
correct	87	91.58	6,847
krkvrlkvrkrkvrlkvrkr	78	89.66	6,614
infrequent	9	10.34	8,869
error	8	8.42	9,743
marking	83	46.63	6,969
correct	77	92.77	6,359
krkkrrlkrkrkkrrlkrkr	22	28.57	5,477
krkkrlrkrkrkkrrlkrkr	11	14.29	6,440
krkkrrlkrkrkkrlrkrkr	10	12.99	6,381
krkkrlrkrkrkkrlrkrkr	9	11.69	6,865
krkkrlkrrkrkkrlkrrkr	8	10.39	8,628
krkkrrlkrkrkkrrlkrkr	5	6.49	5,504
infrequent	12	15.58	6,346
error	6	7.23	14,803
springboard	176	24.72	8,368

Pattern: correct Duration (ms): 6,847 Percent: 91.58 Frequency: 87 Std_Dev: 1,509

Figure 1 : ExperiScope showing several traces.

CHI 2007, April 28–May 3, 2007, San Jose, California, USA.
Copyright 2007 ACM 978-1-59593-593-9/07/0004...$5.00.

understand the results of empirical evaluations. Starting from a log of the interactions performed during an experiment or in actual use, our tool creates a synthesis of all the interaction patterns encountered during the experiment, highlighting for each class of interaction, the most common patterns of use as well as typical error scenarios.

To simplify the analysis of these patterns, we developed a new visualization technique shown in Figure 1. It combines aspects of the Keystroke-Level Model (KLM) [9], to describe the atomic tasks required by an interaction (such as pointing and entering a keystroke) and the Buxton three-state model [8], to describe the current state of the input system and to represent a user's level of engagement [30] with his or her input devices. We believe that this approach allows designers to understand the relationship between specific user actions and the input vocabulary offered by the input device(s) at any given point. Thus, our visualization helps designers appreciate what Buxton calls *pragmatic* considerations [6], such as deciding if it makes more sense to ask users to click a button at the periphery of the screen or to press a command button in order to enter a marking mode. It is also flexible enough to accommodate a large variety of techniques including two-handed interactions [5] and pressure-based interactions [28].

ExperiScope allows designers to rapidly identify key patterns that users exhibit during data capture. Experimenters can easily compare and, if needed, form higher order clusters of patterns which are equivalent from a user's point of view. For example, while users may have perceived that they pressed a modifier key and a mouse key "at the same time", the event stream will record that one of the two was pressed earlier than the other [25]. Regardless of the exact sequence, the two patterns would be considered semantically equivalent. Designers may also use our difference tool to identify and analyze timing differences between patterns.

We illustrate the potential of ExperiScope by revisiting the data collected for an experiment conducted by Hinckley et al. which compared different mode switching techniques [19]. Our analysis of the error patterns confirms the advantages of using the so-called "SpringOnce" as compared to the original "Springboard" design [19]. Our timing analysis also reveals that under certain circumstances, marking on a "lagoon" at the bottom of the screen can be as fast as marking in place. Our visualization suggests possible reasons for this result.

RELATED WORK

Our visualization draws on several visualization techniques including the three-state model [8] and the keystroke-level model (KLM) [9]. It is also related to interaction technique analysis tools such as the family of GOMS models [9].

The three-state model [8] generalizes the states sensed by input devices (0=out-of-range, 1=tracking, 2=dragging). This model has been used extensively to describe and analyze new interaction techniques, such as a two-handed touch sensing technique [18] and Tracking Menu [13]. Unfortunately, three-state models describe all possible interactions simultaneously, making it difficult to understand typical syntactical constructs encountered by users. This neglects important aspects of command selection design such as phrase structure and chunking [7]. Like an oscilloscope (or a logic analyzer) presents a temporal description of the behavior of a complex state machine, our notation provides a description of the evolution of input states of a given three-state model in the context of the sequence of interactions performed by users. This approach will help designers to better understand how users actually employ novel interaction devices or input techniques and thus pave the way for creative insights into how techniques could be improved or extended. Our approach also extends the three-state model to more complicated state models, including multi-level buttons [32] or pressure sensing techniques [28].

The keystroke-level model (KLM) [9] approximates the time it takes a user to perform an interaction using a series of pre-defined atomic steps, or *operators*, with constant values representing average task times. KLM operators include *P* (pointing to a target on the screen), *H* (homing the hands to an input device), and *M* (mental preparation), among others. KLM has been applied to studies of command selection [11, 19]. In the present system, we use KLM notations to provide a high level description of user interactions. We present this information in the context of the interactive system state model to make it easier to identify specific pragmatic considerations [6] that might influence the design.

Several notations and simulators have been proposed to study possible designs of interactive techniques. The GOMS model proposed by Card et al. [9] was used successfully to predict task time in applications such as text editors. Its successors, including CPM-GOMS, [14] and EPIC [2], were used in increasingly more complex scenarios. These models and their corresponding implementations (such as GLEAN [22] and APEX [21] for CPM-GOMS) focus on simulating users' behavior assuming known interaction patterns. The tool described here is complementary as it helps researchers to better understand real users' behaviors. It can also be used in conjunction with GOMS-based models to test hypotheses or calibrate the models for a given experimental setting although this possible extension is not considered in the present paper.

Our work is also related to recent work comparing usage behaviors for single and multiple monitors [20]. Like Hutchings et al., we are interested in capturing and

Action	Symbol	Description
Mark	L	User performs a structured gesture or a mark.
Gesture	G	User performs an unstructured, free form gesture.
Point	P	User aims the pointer at a specific position on the screen.
Cross	C	User crosses a specific goal on the screen [1].
Timeout	T	User dwells at the same location for some time.
Mental Preparation	M	User pauses to think before acting.
Home	H	User homes (moves hand(s) to acquire an input device)
Button Press	K,R	User presses and releases a button or keyboard key [29]
Visual shift	V	User shifts visual attention to a different area of the screen
2-Handed Input	2	Coordination between pointing devices controlled by each hand.

Figure 2 : The set of actions used in our notation.

analyzing patterns of user interactions. While Hutchings et al. focused on high level tasks, our work focuses on lower level interactions. Both approaches are complementary and could be integrated (e.g., through the use of semantic zooming [4]).

INTERACTION VISUALIZATION

Our visualization design uses KLM as a starting point. This is a natural choice since the KLM has been used extensively to analyze the results of experiments and convey the structure of command interactions [9, 19]. In the KLM, a given interaction is decomposed into a series of discrete events representing actions such as pointing at a button, marking a command, and pressing a key or a button. This notation is easily extensible beyond the original set of events; Figure 2 presents the set we currently support in our system.

While very powerful, the purely textual KLM descriptions are often difficult to read. In particular, when looking at a given event sequence, it is challenging to understand the system input stream state and the pragmatic aspects of the interaction. For example, it is often difficult to determine readily which hands are being used or which button is pressed without retracing the sequence step by step. This can be very tedious and error prone. To address this problem, we rely on a two-dimensional notation presented in Figure 3. On the horizontal axis, we present the phraseology of the interaction using the KLM action symbols (Figure 2) as they appear over time. On the vertical axis, we represent the pragmatic aspects of the interaction itself including the locus of interaction and the state of each device at a given time.

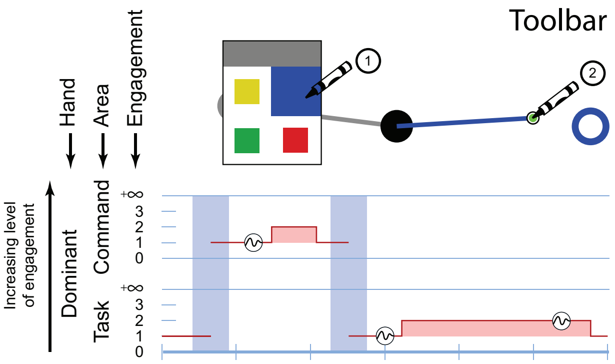


Figure 3 : Connecting two dots using a tool palette. Top: illustration of the interaction being performed (adapted from [17]); Bottom: the corresponding visualization in our system.

Pragmatics

While there are many pragmatic aspects to the design of interactions, our notation focuses on 2 principal aspects: where users direct their visual attention, and the input configuration.

Where to look?

In most current graphical user interfaces, the screen can be partitioned in two main areas: the *Task Area* which contains the users' work, such as the drawing area of a drawing application or the text area of a word processor; and the *Command Area* dedicated to command selection mechanisms such as the menu bar and different toolbars. Our notation segregates activities in these areas into two different "bands" (see Figure 3). Transitions between Task and Command Areas are highlighted in light blue indicating a shift in visual attention. It is important to note that this information can be inferred without the use of eye-tracking technology by simply noting where pointing interactions are taking place. However, this may not always be reliable. For example, novices may look at the keyboard to execute a shortcut.

What to press?

In each of the two bands, we represent the state of each input stream with a notation similar to a timing diagram. Many command selection techniques use simple two-state buttons. However, multi-level buttons [32] or pressure sensing techniques [28] require that our notation represents more than three levels of interaction. For multi-state selections we add additional levels to the "3-state" representation (level 3, level 4 etc.). For continuous sensors we adopt the Card et al. [10] convention of using +∞ to indicate a continuum of states. Multiple buttons can be represented by different line styles and colors, allowing even complex chording interactions to be represented. To visually emphasize changes in levels of engagement, the area beneath the line is lightly shaded (see Figure 3).

Level of engagement

An interesting aspect of our layout is that it provides a rendering of the changing levels of engagement as the user progresses through the command selection steps with "no engagement" (out of range, state 0) as a base level. Following Sellen et al [30], we adopt a notion of engagement which reflects both the haptic and visual channels. For each step, our visual representation illustrates the user's level of engagement:

Haptic engagement reflects the muscular tension required at any particular step. For example maintaining the tip of the pen on the screen and pressing the command button requires a deeper level of engagement than simply moving the pen on the screen.

Visual engagement reflects steps that require users to switch their locus of attention away from the current task to another part of the interface. Techniques that let the user interact at the current locus of attention may have significant advantages, as argued by many authors [15, 23, 29].

First example: Palette

Let's consider a simple "Connect the Dot" task in which users are asked to select a color using a tool palette and then asked to connect the two dots using a simple rubber band interaction (Figure 3 top). The visualization of this interaction is shown in Figure 3 bottom. Starting from the drawing area, users first switch their attention to the command area (as indicated by the blue highlight), they aim (∿) for the correct color on the toolbar and perform a click (⎍) in which the falling edge validates the tool selection. Then, they switch their attention back to the drawing area (as indicated by the blue highlight) and aim (∿) at the first dot before pressing the pen tip (_|‾). Finally, they start dragging while aiming at the second dot (∿) before lifting (‾|_) the pen on top of this dot.

Two-handed techniques

So far we have only considered one-handed techniques, but, of course, many interactions such as ToolGlass [5] or simply the use of modifier keys involve both hands. In two-handed interactions, both the dominant and the non-dominant hand may travel between the task area and the command area (for example to select a given ToolGlass [5]). We present the diagrams for the dominant and non-dominant hand alongside each other. For each hand, the band for the task area is shown closer to the baseline (the dotted line at the center of the notation) than the band for the command area. In Figure 4, we illustrate the use of our notation for two-handed techniques using the example of a "Connect the Dot"-task performed with ToolGlass. The non-dominant hand controls a semi-transparent tool palette. To select a tool or, in this example, a color, users need to click through the corresponding color on top of the starting point of the line to be drawn. They then connect the two dots using a simple rubber band interaction (see Figure 4, top). Our visualization of this interaction is shown in Figure 4, bottom. Users first aim the proper color square of the ToolGlass on top of the first dot (∿ in the task area for the non-dominant hand). Next, they use their dominant hand to place the pen on top of the first dot (∿ in the task area for the dominant hand). While the movement of both limbs can be carried out independently, they will need to be coordinated upon clicking the mouse button (_|‾). Our notation reflects this fact with the two-handed synchronization symbol (⋮). Users can then start dragging while aiming at the second dot (∿) before lifting (‾|_) the pen on top of this dot.

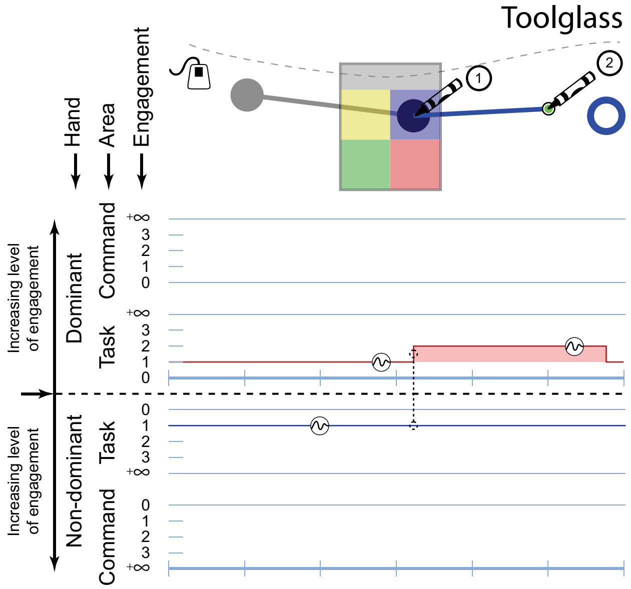
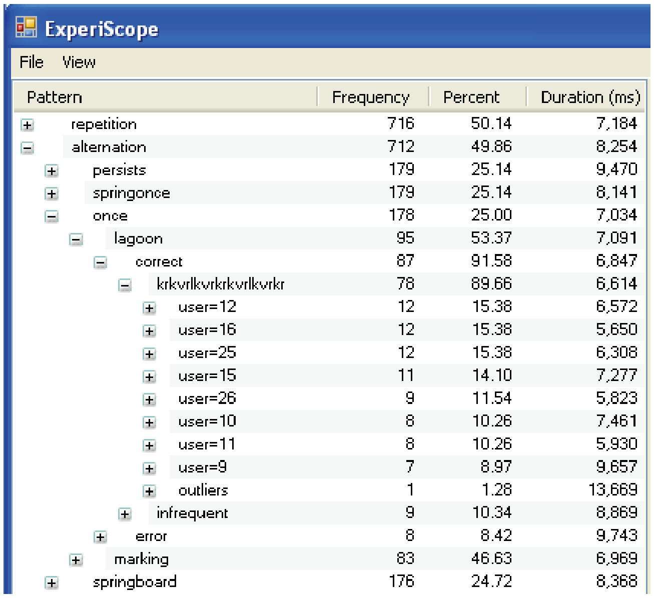


Figure 4 : Connecting two dots using a ToolGlass. Top: illustration of the interaction being performed (adapted from [17]); Bottom: the corresponding visualization in our system.

The goal of our visualization was to make key aspects of interaction design more salient. The representation of elapsed time along the horizontal axis makes it easy to pinpoint parts of an interaction sequence that slow down users. On the vertical axis, we emphasize the engagement level required at any given time, and highlight key aspects influencing users' engagement including the need to switch one's locus of attention, or the need for synchronization between two hands. It is clear that a systematic manual analysis of a log could provide similar results. We believe, however, that our visualization will help novices to rapidly pinpoint problems, and allow experts to conduct analyses that are more fine-grained than the ones based on currently available models.

EXPERISCOPE

The visualization presented above is very useful to explore and compare a small set of interactions. Of course, given the sheer number of interactions recorded, it is impractical to observe each interaction individually. ExperiScope therefore combines the visualization described above with a hierarchical visualization of all the patterns encountered during an experiment.



Pattern	Frequency	Percent	Duration (ms)
⊞ repetition	716	50.14	7,184
⊟ alternation	712	49.86	8,254
⊞ persists	179	25.14	9,470
⊞ springonce	179	25.14	8,141
⊟ once	178	25.00	7,034
⊟ lagoon	95	53.37	7,091
⊟ correct	87	91.58	6,847
⊟ krkvrlkvrkrkvrlkvrkr	78	89.66	6,614
⊞ user=12	12	15.38	6,572
⊞ user=16	12	15.38	5,650
⊞ user=25	12	15.38	6,308
⊞ user=15	11	14.10	7,277
⊞ user=26	9	11.54	5,823
⊞ user=10	8	10.26	7,461
⊞ user=11	8	10.26	5,930
⊞ user=9	7	8.97	9,657
⊞ outliers	1	1.28	13,669
⊞ infrequent	9	10.34	8,869
⊞ error	8	8.42	9,743
⊞ marking	83	46.63	6,969
⊞ springboard	176	24.72	8,368

Figure 5 : Clustering of the results of an experiment

To use ExperiScope, experimenters must first describe each interaction sequence using ExperiScope's simple XML format. This format describes the type of the interaction sequence (or the corresponding experimental cell) as well as key events occurring during the interaction. These include three-state model transitions (button pressed, pen in range…), KLM events (crossing, pointing…), error conditions, and any other user defined annotations that can help to interpret the corresponding interaction (name of the command selected, menu displayed…). These annotations can be either generated automatically or created manually when automatic detection is difficult (e.g., when a Mental Preparation takes place). Each interaction record can also include "private" data which will be ignored by ExperiScope, but will be kept with the corresponding record during all processing.

Upon reading the data from an experimental run, our tool first creates a synthetic hierarchical overview of the interaction patterns encountered during the experiment. An example of this overview (using data from a study that compared different mode switching techniques [19], and omitting the baseline condition) is shown in Figure 5. The first levels of the hierarchy reflect the structure of the experimental design. They are built automatically using the interaction type data provided by the experimenter. In this example, the analysis considered 2 different tasks (Alternation, Repetition), 4 task structures (Persist, SpringOnce, Once, Springboard), and two command selection techniques (Lagoon and Marking). Within each experimental cell, the system further groups correct trials and trials containing any errors.

Clustering

Even in this hierarchical presentation, too many interaction sequences do not allow for quick exploration. To address this problem, our system further clusters similar interaction sequences. For each sequence, the system first constructs a string representing the sequential succession of events during that interaction. This string is then matched against existing patterns to determine if such a sequence has already been seen. If so, the new sequence is placed in the corresponding bin for further processing before visualization. If not, a new bin is created and assigned the corresponding pattern. While the use of a serial representation may be limited when compared to more complex representations such as Partially Ordered Sets [26], it was sufficient to describe all the cases we encountered in the present example.

To further limit screen clutter, our tool only shows patterns which appear more than 5% of the time for a given interaction type. Less frequent patterns are grouped into a folder labeled "infrequent" (Figure 5). If necessary, the tool also creates a subfolder called "outliers" for sequences with a total completion time more than three standard deviations above the mean for the corresponding experimental cell. For each pattern and folder, statistical information such as average time and frequency is presented to the right of each item. The data from the "outliers" folder is not considered when computing these values. As shown in Figure 5, this approach allows for a quick overview of the general trends of an experiment but also provides more details if needed.

Dealing with unimportant differences

Sometimes, patterns with a slightly different serialization of events are considered equivalent in practice. This is best illustrated by interactions which involve pressing a modifier key while clicking the mouse button. In such interactions, it is often difficult for users to respect a strict order (e.g., pressing the modifier key before clicking [16, 24, 25]). Therefore, interfaces often adopt a lax synchronization policy which does not require a specific ordering. Our system offers the option to cluster apparently different but functionally equivalent patterns into one group by a simple drag and drop operation.

Pattern visualization

Upon clicking on any pattern, the pattern is shown in a new window using the visualization described above. For simple patterns, the time between events is computed by averaging the corresponding times across all sequences that match a given pattern. For sets of functionally equivalent patterns that were grouped manually, the visualization reflects possible variations in the sequence of events by creating hatching marks in areas where the different patterns disagree (Figure 6). Currently, our system assumes that the main interaction patterns (actions by the dominant hand excluding modifiers) are identical within a set. More complex composition methods are possible and will be considered in future work.

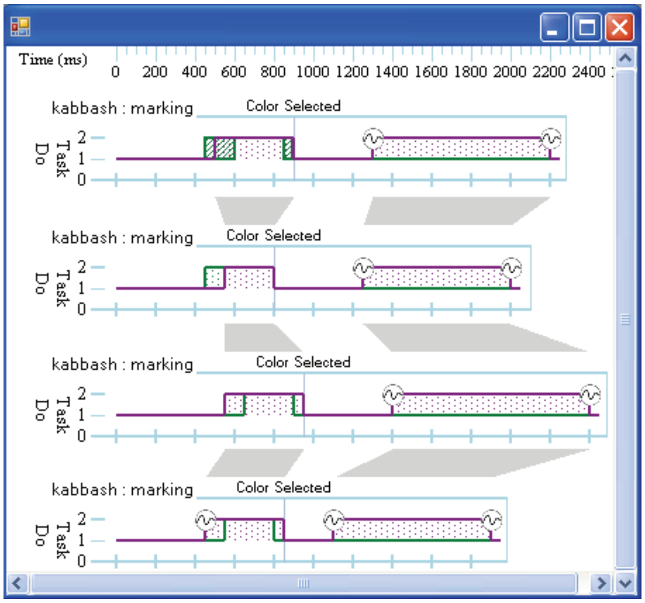


Figure 6 : The aggregate patterns for simple marking menu interactions. The purple line represents the activity of the pen tip, the green line the activity of the pen button. The top pattern represents the aggregate view of the bottom patterns using hatching where patterns disagree. The tool creates grey bands between timelines to simplify comparison.

To simplify pattern comparison, users can also open several patterns in the same window - either by selecting several patterns to open at once, or by simply dragging a new pattern on top of an existing window. In such cases, our tool graphically highlights the timing differences between each segment of the main interaction patterns (i.e., actions by the dominant hand excluding modifiers) as shown in Figure 6.

To further simplify comparisons, our system also offers an alignment tool based on instrumental interactions [3]. The tool consists of a vertical ruler that can be set at any position on the time scale. Once set, any location on a diagram can be snapped to the ruler by using a snapping tool. To simplify the analysis of successive segments, users can use the horizontal arrow keys to shift all the patterns that are currently aligned by one transition to the right or left.

If needed, any diagram can be edited to reflect information that was not captured during the logging process but may be helpful for data analysis. This includes adding text tags to identify events of interest such as the name of the currently selected tool. Users may also create a diagram from scratch as a reference point. This feature can be useful to compare expected patterns to observed patterns.

Exporting data

The information captured during the analysis can be either saved using the same XML format[1] or using a tabular format suited for pivot table processing. This makes it easy for example, to restrict a statistical analysis to a specific set of interaction patterns or to interactions annotated with a certain flag. While it would have been possible to include advanced statistical capabilities in ExperiScope, we found it more convenient to conceptualize our tool as a filter between the collected data and traditional data analysis packages such as SPSS.

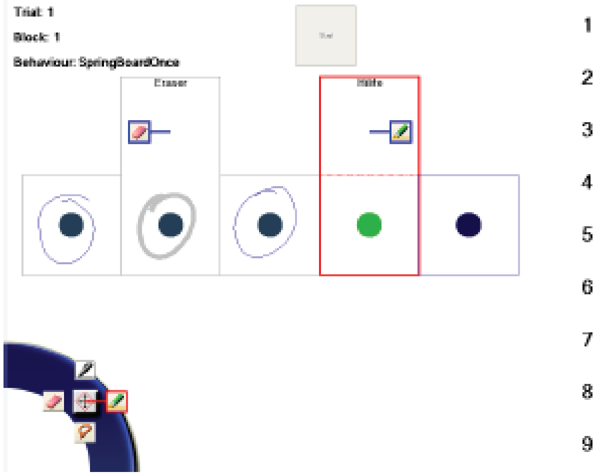


Figure 7 : Screen layout for the Hinckley et al. experiment (from [19]). The Alternation condition is shown. The five targets that users are asked to circle are shown in the middle of the screen. The lagoon is shown in the lower left corner.

CASE STUDY

We used the tool presented above to revisit data from Hinckley et al. [19], a study which examines the performance of “Springboard” (Figure 7) and other mode switching techniques[2].

The Springboard technique [19] explores the use of spring-loaded modes (also called quasi-modes [29]) in pen interfaces. Springboard interactions require users to press a command button with their non-dominant hand in order to switch between inking and command gestures. This approach can be used in local settings where the command selection is performed at the locus of attention (e.g., for a marking menu), or in non-local settings where a special area for command selection (called the “lagoon”), becomes active at the lower left of the screen upon pressing the command button. Upon activation, the lagoon area can be used to mark commands (see bottom left of Figure 7).

To evaluate the potential for the Springboard technique, Hinckley et al. considered 3 main variables:

- **Task type**. All tasks were derived from a simple basic task: Using a pen to circle five large dots shown on the screen. In the basic reference task, all the dots were to be circled using standard ink. In the *Alternation* task (shown

[1] The saved format includes preprocessed information to speed up the opening of the log.

[2] ExperiScope, the original dataset, as well as the ExperiScope formatted dataset are available at http://www.cs.umd.edu/hcil/experiscope

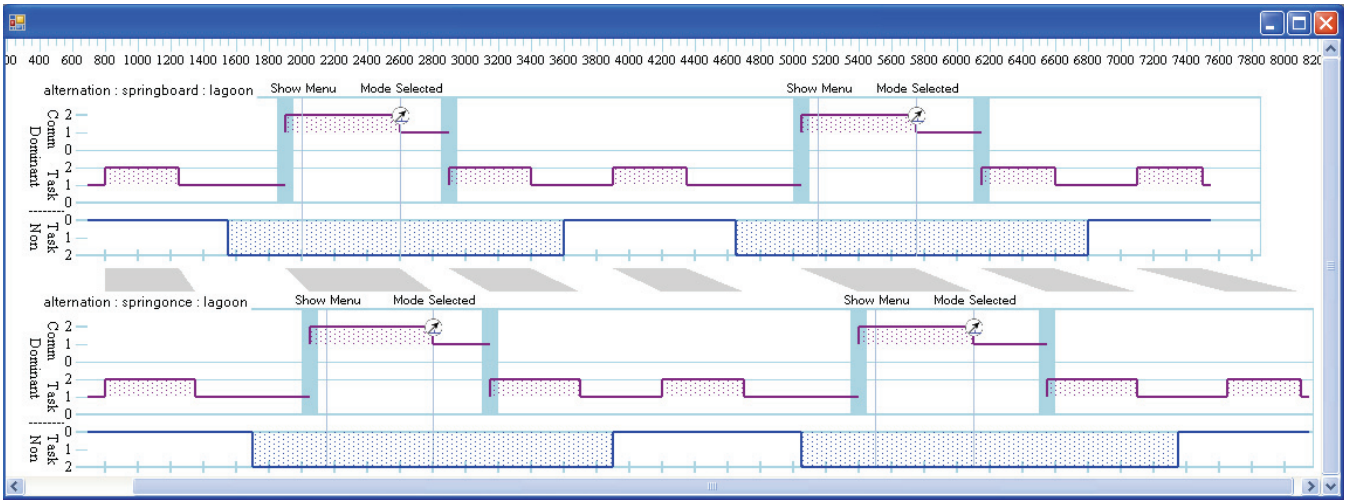


Figure 8 : Patterns of use in the Alternation Lagoon condition for Springboard (top, 84% of all correct sequences) and SpringOnce (bottom, 73% of all correct sequences). The visualization illustrates that the two interaction patterns are identical.

in Figure 7), participants needed to select special tools before circling the second and fourth dot. A different tool was required for each dot. This task was designed to simulate commands interleaved with inking. In the *Repeat* condition, the 3 center dots required participants to select the same tool, once, for all three targets. This task examined the influence of the amortization of command selection [27].

- **Menu type** indicated where the menu could be triggered. While all tasks used marking menu [23] as a menu system, in the *Marking* setting, the menu could be triggered at any location of the screen simply by pressing a command button. In the *Lagoon* setting, the menu could only be triggered inside a lagoon (shown at the bottom left of the screen in Figure 7).
- **Modal behavior** encompassed four different settings. In the *Persist* setting, the command button only needed to be pressed during the tool selection, and the tool selection persisted until a new tool was selected. This reflects the behavior of a modal system. In the *Once* setting, the command button only needed to be pressed during the tool selection, and the tool selected was available for one stroke. Thereafter, the pen reverted to inking. In the *Springboard* setting, the command button needed to be pressed as long as the tool was needed (including potential repeat invocations). Once the command button was released, the pen reverted automatically to the inking. Finally, in the *SpringOnce* setting, users could either keep the button pressed as long as the tool was needed, or, if they released the button, the tool stayed active for at least one stroke. This mode was introduced because for simple commands, users sometimes released the command button too soon.

To perform our analysis, we first created a parser to transform the original experimental logs into a format recognized by ExperiScope. The log of one participant was not available to us, and the logs did not contain pen locations, so the transitions between task and command areas were inferred. Next, we used our visualization tool to analyze the data. In the following, we illustrate how our tool can aid the analysis and interpretation of this type of dataset. Because we focus on the use of the visualization tool, we did not carry out any analyses to evaluate the statistical significance of the findings. In practice, of course, such analyses should be performed to validate the insights gathered by ExperiScope.

Checking user behavior

Our first task was to verify that user behavior reflected the instructions provided during the experiment. For example, we examined user behavior in the Alternation Lagoon condition and compared interaction sequences for different modal settings. Our tool illustrated (Figure 8) that the most dominant pattern of interaction for the SpringOnce setting (73% of all correct sequences) was identical to the most dominant pattern in the Springboard condition (84% of all correct sequences). This confirmed that users did not mind keeping their finger pressed during the full duration of the command. In fact, only 3 samples followed a strict SpringOnce interaction (e.g. releasing the button as soon as the command is selected for both tool selections). At the same time, the SpringOnce design appeared to help users to reduce their errors. For SpringOnce, the error rate was only 10% (compared to 17% for Springboard). When examining the 12 different error patterns, we found that 37% of them were caused by Springboard users releasing the button too soon.

It is interesting to note that the pattern seen in the *Lagoon* condition is not repeated in the *Marking* condition. Here, only 58% of the patterns observed in the SpringOnce setting resembled Springboard patterns. Nevertheless, the SpringOnce setting had the same beneficial influence on

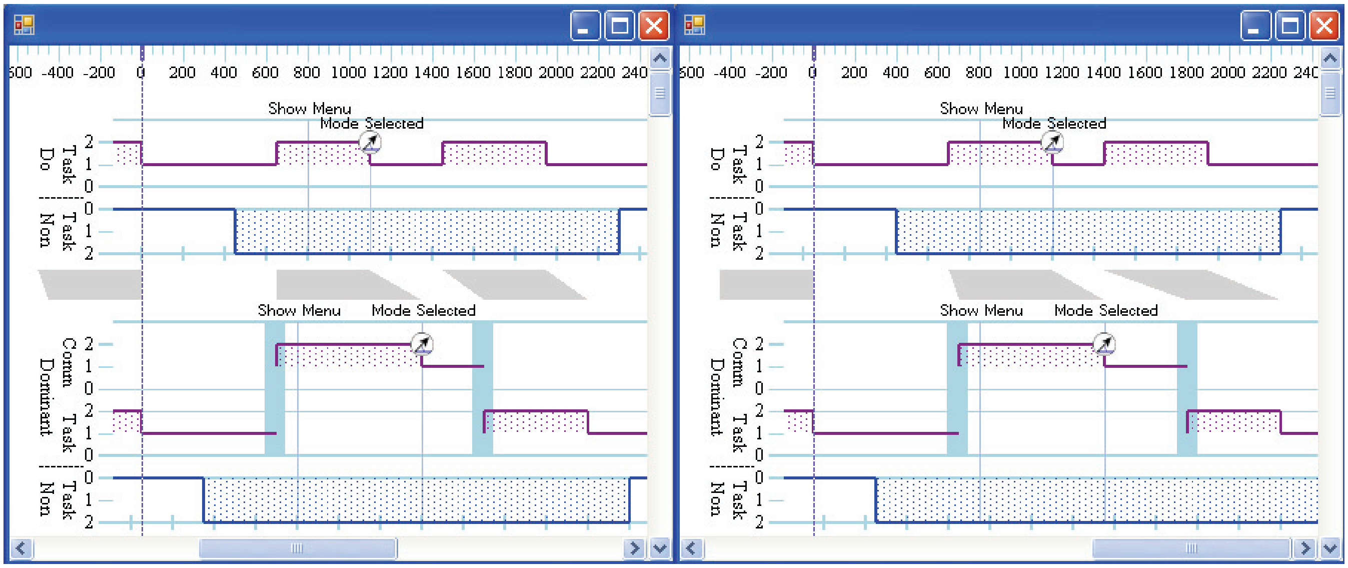


Figure 9 : Comparison of the most common patterns of use for Alternation/Springboard/Marking (85% of correct sequences, top graph) and Alternation/Springboard/Lagoon (84% of correct sequences, lower graph). Left: first command invocation; Right: second command invocation. In both cases, the alignment tool (the dotted vertical dotted line) is used to align the graph at the end of the previous interaction (inking around a dot).

errors as in the *Lagoon* condition. About 50% of the errors were caused by Springboard users releasing the command button too early. Using ExperiScope, we were able to arrive at these findings through simple visual inspection of about 16 patterns.

Comparing interaction patterns

One of the key findings of Hinckley et al. was that the difference between local interactions (Marking) and non-local interaction (Lagoon) was smaller than expected.

Our tool provides some interesting findings in that regard. First, we found that for the *most common* interaction pattern (more than 84% of correct interactions) in the Alternation Springboard condition, the average interaction time for Lagoon was about 150 ms faster than for Marking. Given the opposite pattern for the average completion times across *all* interactions in this condition (Lagoon being about 300 ms slower), this suggests that the slower performance of Lagoon was due to the cost of error correction. This is not surprising, since in the Lagoon setting, an error could cause a full round trip between the task area and the lagoon area.

Our tool also shows that the two techniques do not proceed at the same speed, but distribute the interaction times differently. First we observed that the in the Marking condition, the first interaction occurred about 300 ms later than in the Lagoon condition, probably reflecting a longer initial thinking time. We then carried out a side by side comparison of the timing for the two techniques during the first (Figure 9, left) and second command selection (Figure 9, right). We used our alignment tool to synchronize the two patterns at the point where the user completes the previous interaction (i.e., drawing a circle using ink). Figure 9, left, illustrates that the initial handicap of Marking is recovered by the time the command selection is performed because command selection is faster in the Marking condition than in the Lagoon condition. It also shows that for the first command selection, which takes place at the left of the screen (i.e., the leftmost dot to be circled), there is virtually no difference in travel time for the dominant hand between both conditions (as seen by the comparison tool showing white parallelograms between these two segments). This is somewhat surprising since the two paths are very different. In the Marking condition, participants first needed to mark anywhere on the screen and then go to the target. In the Lagoon condition, participants first needed to go to the Lagoon and then to the target. By the time the target is circled, Marking is ahead of Lagoon, yet most of this speed advantage is lost during the release of the command button, which seems to incur a higher cognitive load.

A very similar pattern is found for the second selection task (Figure 9, right) with one major difference: Because the second selection task is taking place further right on the screen (i.e. at the location of the second dot), we can now see a slightly longer traveling time for the Lagoon condition (as illustrated by the white areas in the comparison tool which are no longer parallelograms). This reflects the influence of Fitts' law [12] on the overall performance of the task.

The cost of a button press

We also used our tool to explore the cost of pressing a button. We first compared the aggregate patterns for the Marking-Once condition (84% of correct patterns, 6361ms), the Lagoon-Once condition (89% of correct patterns, 6612ms), and the Lagoon-Springboard condition (84%, 7524ms) during the first command selection in the

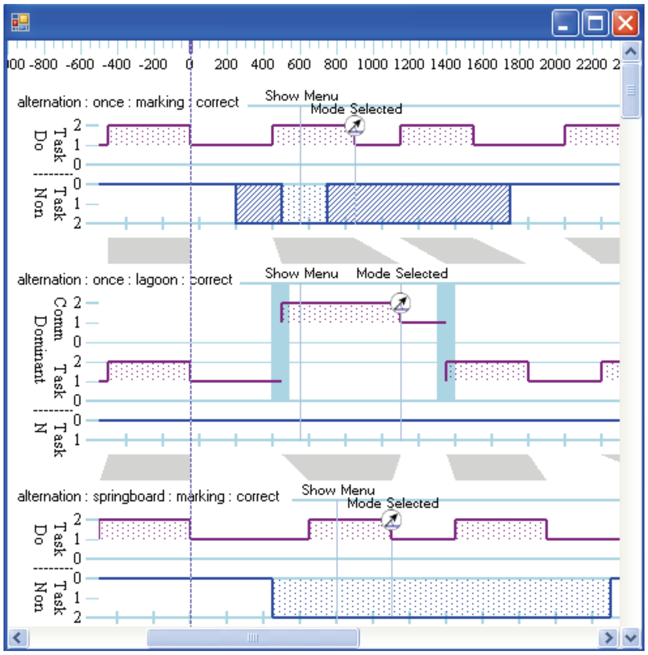


Figure 10 : Comparison of 3 Alternation techniques (Marking Once, Lagoon Once, Marking Springboard) showing the different costs of pressing a command button.

Alternation task (Figure 10). In the Lagoon-Once condition, no button press is necessary to activate the Lagoon (straight blue line), so it serves as a useful reference. A comparison of the menu invocation times for Marking-Once (Figure 10, top) and Lagoon-Once (Figure 10, middle), shows that for Marking-Once, the ability to invoke the menu at a convenient location may completely offset the cost of pressing the command button. The Marking-Once pattern also exhibits a faster command selection time and a short delay between command selection and drawing. It is important to note that in the Marking-Springboard condition (Figure 10, bottom) the exact same actions take considerably longer. This may highlight the higher mental cost of the Springboard approach as compared to the simpler Once approach. Since the pattern of use for the Once condition is so regular, it can probably be chunked completely [7] into a broader pattern, while the Springboard requires a small amount of mental preparation that cannot be chunked.

DISCUSSION AND FUTURE WORK

These examples illustrate how ExperiScope lets researchers quickly inspect their data to discover the most common patterns of use and major causes of errors. It helps designers gain a more in-depth understanding of their data to better comprehend how users distribute their time during a given task. As a result, researchers will be able to answer more complicated questions and conduct more meaningful statistical tests. The tool also illustrates the diversity of user behavior for any given technique. Thus, it supports Vicente's call for a more detailed analysis of usability data [31] to ensure that average performance measures do not hide key disparities in users' behaviors.

From one experimental cycle to the next

Experimental protocols often require several iterations to be successful. For this reason, we consider it a very important feature of our tool to allow for seamless transfer of analytical approaches and insights from one round of analysis to the next. For example, during the early phases of experimental design, researchers may identify patterns of interactions which are equivalent and subsequently cluster them together to create a filter. Although this often represents tedious work, it may help researchers to gain key insights about the data set. It is therefore important that the information gathered by manual modifications of the clustering can be easily transferred from one dataset to the other (either between different versions of the same experiment or from one experiment to another). We are currently exploring solutions to export cluster patterns from one cycle of analysis to the next. This mechanism can also help researchers to create a library of similar patterns to simplify the analysis of future experiments.

From analytical tool to teaching tool

So far, we have presented experimental analysis as the main focus of ExperiScope. In addition, we believe that it has great potential as a teaching tool. We foresee that it can be used both by teachers to illustrate key aspects of interaction techniques and by students to carry on their own investigations. We are planning to integrate this tool into future HCI curricula and will evaluate its impact on students' understanding of key concepts in human computer interaction design.

CONCLUSION

In this paper we have presented a new visualization technique and a new tool for experimental analysis designed to help users rapidly review and analyze data captured during research on human computer interaction. We have demonstrated how ExperiScope can be used to revisit data presented Hinckley et al. [19] to provide a more in-depth understanding of the results of this experiment. The design of our tool makes it easy to use in a wide variety of contexts. We believe that it will help both interaction designers and experimenters to evaluate and compare interaction techniques more accurately and provide a deeper understanding of experimental results. It can also be used in teaching to illustrate key aspects of interaction techniques.

ACKNOWLEDGEMENTS

This work was supported in part by NSF Grant IIS-0414699 and by the Microsoft Research Center for Interaction Design and Visualization at the University of Maryland. We would like to thank Corinna Löckenhoff, Ben Bederson, and Mary Czerwinski for providing many useful comments. Our tree viewer is based on Thomas Caudal's TreeListView (http://www.codeproject.com/cs/miscctrl/treelistview.asp). Nick Chen provided the data used in Figure 6.

REFERENCES

1. Accot, J. and S. Zhai. Refining Fitts' law models for bivariate pointing. *Proceedings of CHI'03*, pp. 193 - 200.
2. Anderson, J.R. and M. Matessa, An Overview of the EPIC Architecture for Cognition and Performance With Application to Human-Computer Interaction. Human-Computer Interaction, 1997. **12**(4): p. 391-438.
3. Beaudouin-Lafon, M. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. *Proceedings of CHI'00*, pp. 446 - 453.
4. Bederson, B.B. and J.D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. *Proceedings of UIST'94*, pp. 17-26.
5. Bier, E.A., M.C. Stone, K. Pier, W. Buxton, and T.D. DeRose. Toolglass and magic lenses: the see-through interface. *Proceedings of SigGraph'93*, pp. 73 - 80.
6. Buxton, B., Lexical and Pragmatic Considerations of Input Structures. Computer Graphics, 1983. **17**(1): p. 31 - 37.
7. Buxton, B. Chunking and Phrasing and the Design of Human-Computer Dialogues. *Proceedings of IFIP World Computer Congress*, pp. 475 - 480.
8. Buxton, W. Three-State Model of Graphical Input. *Proceedings of INTERACT'90*, pp. 449 - 456.
9. Card, S.K., T.P. Moran, and A. Newell*, The psychology of human-computer interaction*. 1983: Erlbaum Associates.
10. Card, S.K., J.D. Mackinlay, and G.G. Robertson, A morphological analysis of the design space of input devices. ACM Transactions on Information Systems, 1991. **9**(2): p. 99 - 122.
11. Douglas, S.A. and A.K. Mithal. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. *Proceedings of CHI'94*, pp. 411 - 416.
12. Fitts, P.M., The infomation capacity of the human motor system in controlling amplitude of movement. Journal of Experimental Psychology, 1954. **47**: p. 381 - 391.
13. Fitzmaurice, G., A. Khan, R. Pieké, B. Buxton, and G. Kurtenbach. Tracking Menus. *Proceedings of UIST'03*, pp. 71 - 80.
14. Gray, W.D., B.E. John, and M.E. Atwood, Project Ernestine: Validating a GOMS Analysis for Predicting and Explaining Real-World Task Performance. Human-Computer Interaction, 1993. **8**: p. 237 - 309.
15. Grossman, T., K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. *Proceedings of CHI'06*, pp. 861 - 870.
16. Guimbretiere, F., Fluid Interaction for High Resolution Wall-size Displays, PhD thesis, Stanford University. 2002
17. Guimbretière, F., A. Martin, and T. Winograd, Benefits of Merging Command Selection and Direct Manipulation. Transactions on Human-Computer Interaction, 2005. **12**(3): p. 460 - 476.
18. Hinckley, K., M. Czerwinski, and M. Sinclair. Interaction and modeling techniques for desktop two-handed input. *Proceedings of UIST'98*, pp. 49 - 58.
19. Hinckley, K., F. Guimbretiere, P. Baudisch, R. Sarin, M. Agrawala, and E. Cutrell. The springboard: multiple modes in one spring-loaded control. *Proceedings of CHI'06*, pp. 181 - 190.
20. Hutchings, D.R., G. Smith, B. Meyers, M. Czerwinski, and G. Robertson. Display space usage and window management operation comparisons between single monitor and multiple monitor users. *Proceedings of AVI'04*, pp. 32 - 39.
21. John, B., A. Vera, M. Matessa, M. Freed, and R. Remington. Automating CPM-GOMS. *Proceedings of CHI'02*, pp. 147 - 154.
22. Kieras, D.E., S.D. Wood, K. Abotel, and A. Hornof. GLEAN: a computer-based tool for rapid GOMS model usability evaluation of user interface designs. *Proceedings of UIST'95*, pp. 91 - 100.
23. Kurtenbach, G., The Design and Evaluation of Marking Menus, PhD thesis, University of Toronto. 1993
24. Lank, E., J. Ruiz, and W. Cowan. Concurrent bimanual stylus interaction: a study of non-preferred hand mode manipulation. *Proceedings of GI'06*, pp. 17 - 24.
25. Li, Y., K. Hinckley, Z. Guan, and J.A. Landay. Experimental analysis of mode switching techniques in pen-based user interfaces. *Proceedings of CHI'05*, pp. 461 - 470.
26. Luckham, D. and J. Vera, An Event-Based architecture Definition Language. IEEE Transaction on Software Engineering, 1995. **21**(9): p. 717 - 734.
27. Mackay, W.E. Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses support different task strategies. *Proceedings of AVI'02.*, pp. 203 - 209.
28. Ramos, G., M. Boulos, and R. Balakrishnan. Pressure Widgets. *Proceedings of CHI'04*, pp. 527 - 534.
29. Raskin, J.*, The Humane Interface: New Direction for Designing Interactive Systems*. 2000: ACM Press.
30. Sellen, A., G. Kurtenbach, and B. Buxton, The Prevention of Mode Errors Through Sensory Feedback. Human-Computer Interaction, 1992. **7**(2): p. 141 - 164.
31. Vicente, K.J. and G.L. Torenvliet, The Earth is spherical ($p < 0.05$): alternative methods of statistical inference. Theoritical Issues in Ergonomics Science, 2000. **1**(3): p. 248 - 271.
32. Zeleznik, R., T. Miller, and A. Forsberg. Pop through mouse button interactions. *Proceedings of UIST'01*, pp. 195 - 196.