

Interfaces for Staying in the Flow

Benjamin B. Bederson
Computer Science Department
Human-Computer Interaction Lab
3171 A.V. Williams Building
University of Maryland
College Park, MD 20742
bederson@cs.umd.edu

ABSTRACT

Psychologists have studied “optimal human experience” for many years, often called “being in the flow”. Through years of study, the basic characteristics of flow have been identified. This paper reviews the literature, and interprets the characteristics of flow within the context of interface design with the goal of understanding what kinds of interfaces are most conducive to supporting users being in the flow. Several examples to demonstrate the connection to flow are given.

Author Keywords

Flow, User Experience, Interface Design.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g., HCI): User Interfaces.

INTRODUCTION

Hiking in the woods, writing, programming, rock-climbing. What do these activities have in common? They are all things that people regularly describe as being activities likely to result in “being in the flow”. When we are fully engaged and in control of an activity, we sometimes sense that time passes more quickly and we feel immersed in that activity to the exclusion of all else. Furthermore, people regularly describe these experiences as some of the best of their lives.

Yet, all too often, computer users have distinctly different experiences – where they are frustrated, lack control, are distracted and interrupted, and feel anything but in the flow.

Is there anything that can be done to create user interfaces that are more likely to help computer users have these highly positive experiences? To try and answer this question, this paper summarizes some of the literature about

“flow”, and interprets it with the goal of providing a language, and specific design approaches to create better computer user experiences.

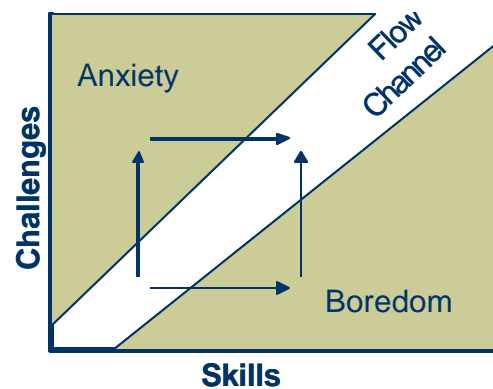


Figure 1: Balancing challenge and skills is key to flow (from [7]).

Mihaly Csikszentmihalyi [7] wrote a popular book about what he called “optimal experience” that summarized a huge amount of work in the field of understanding flow. He uses this term because these flow experiences were so frequently described as being so positive. In this book, Csikszentmihalyi describes how he and many other researchers around the world applied the “experience sampling method” to try and understand and characterize this elusive human experience.

He gave thousands of individuals a pager and then randomly beeped them approximately 8 times a day at which time they were instructed to write down what they were doing and how they were feeling at that moment. Because so many researchers in so many places were involved, participants from all walks of life were surveyed. This includes people from all around the world, from different social and economic backgrounds, from different fields of work, including professionals and laborers. Over a period of several years, hundreds of thousands of human experiences were sampled in this way.

Csikszentmihalyi then processed and grouped the optimal experiences that were reported, and he reported the broad characteristics that held them together. While he presented

many characteristics of human optimal experience, he did not apply it to the field of HCI. This paper attempts to bridge his work to interface design. It is a review of the literature based on Csikszentmihalyi's framework for being in the flow. This paper focuses on five of the characteristics of flow that Csikszentmihalyi observed along with interfaces that exemplify those characteristics:

- Challenge and require skill
- Concentrate and avoid interruption
- Maintain control
- Speed and feedback
- Transformation of time

One of the goals of this paper is to encourage our discipline to consider the qualities of interfaces more broadly. By focusing too closely on narrow quantitative measures, our field risks missing out on other important characteristics of what makes an interface "good".

1. CHALLENGE AND REQUIRE SKILL

One of the first observations Csikszentmihalyi made was that in order for someone to have an optimal experience, they must expend effort to acquire skills, and then apply them. So, while playing tennis or programming are activities conducive to flow, passive or relaxing activities such as lying on the beach aren't (Figure 1).

While it may well be reasonable for a task to be challenging, it doesn't seem reasonable that we should encourage interfaces and tools to be hard to use. Yet, some of the best tools that humans use, on or off the computer, require significant learning. Even a tool as simple as a hammer is non-trivial to learn, as one must move one's arm so that the hand hits the surface below the target.

While it has become nearly dogma that computer interfaces should be so simple to use that ideally, no documentation should be necessary, there are several counterexamples of tools for expert users that should perhaps cause us to reconsider this approach. For example, both Adobe Photoshop and the Emacs text editor are tools that require significant effort to learn effectively. And yet, those people that do put in that effort often love those tools, sometimes fanatically, and use them in countless ways for years.

Other hard-to-learn tools, on the other hand, don't typically engender widespread approval. This implies it isn't just the difficulty of learning a tool that is important. But rather, that it is possible to learn a tool well enough so that it becomes almost an extension of one's body, like a hammer. Part of the reason Photoshop and Emacs work so well is that once an expert has mastered them, they can focus on the task at hand without interruption. This is due in my opinion to their general efficiency, powerful macro capabilities that allow automation of repetitive tasks, and to the robustness and predictability of the software. Other, more problematic tools, on the other hand, may have usability problems that make it so users never get to the point where the application reaches tool-like status. If users

have to re-figure something out, or fight with an unstable feature regularly, they won't be able to get to the point of concentrating on the task.

In general, balancing the needs of novices and experts remains a daunting problem. But, it is crucial to support experts – something that is often overlooked, or left just to shortcut key accelerators. Many computer users become experts at specific programs over time, and providing ways for them to be extremely efficient must not be ignored.

Stages of Skill Acquisition

To understand the pathway from novice to expert user, let us look at how people learn to use interfaces. By thinking more broadly about skill acquisition in general, we can learn from the psychology literature where Anderson summarized the following three levels of skill acquisition [2].

1. **Cognitive stage:** learner works from instructions or examples.
2. **Associative stage:** learning replaces general problem-solving methods with domain-specific problem-solving methods. Learner can apply a skill directly without having to first think through how they are going to do it.
3. **Autonomous stage:** Skill becomes automated and rapid, and there is minimal cognitive involvement in application of skill. Also called automatized or routinized.

Interpreting these stages within the context of user interfaces, we can see that stage 1 (*cognitive* stage) applies to novice users that use a GUI to explore the interface. Having some way for users to discover what is possible is crucial. GUIs rely on visual affordances [17] and command line systems rely on some kind of help system or documentation. In a GUI for example, users can click on the *Edit* menu, to discover that a *Copy* command is available.

Stage 2 (*associative* stage) applies to intermediate users who know what interface elements are available. For example, they may know that the *Copy* command is within the *Edit menu* and thus can go directly to it. However, the nature of visual pointer-controlled interfaces is such that users must get visual feedback from the interface in order to execute the task. While skilled users may not have to think much about it, they still must consciously follow the position of the pointer on the screen to see when it is over the *Copy* button before they select it.

The final *autonomous* stage applies to expert users that can execute an interface element *without feedback* from the interface. This is commonly found in GUIs with keyboard shortcuts. An expert user with touch typing skills might press the "ctrl-c" key combination to execute the *Copy* command without waiting for or receiving any feedback from the interface. These kinds of interfaces are extremely

fast not only because users do not need to move their hands between keyboard and pointing device, but also because they do not require feedback from the interface. The user just thinks and performs. And for very practiced users, the thinking part can be so automatized that the user is not even consciously aware how they apply the command. This is evidenced, for example, by Emacs users who frequently can not tell what keyboard combination they use to apply a command, but can do so quickly when their hands are on the keyboard.

Unfortunately, aside from keyboard shortcuts, there are few other interfaces that enable users to operate in the third autonomous stage. While at first glance, it may seem impossible to operate a visual pointer-based interface without feedback, but in fact, it is possible. One example is Guimbretièrre’s Flow Menus [11] which use a combination of radial layout and *crossing* rather than clicking to activate an interface element (Figure 2). This interface, similar in spirit to pie menus [5] and quikwriting [19], is elegant in that a single interface brings users through the three stages as they start out as novices, and become experts over time. It works in stage 1 by letting users see the visual affordances and explore the interface. It works in stage 2 where users know where the menu elements are, but still get feedback from the interface. Finally, the same interface works in stage 3 because of its radial and crossing-based design. The radial approach lets users learn a direction to move the pointer. People can aim pointers reliably within these 45 degree segments reliably. People can’t, on the other hand, reliably select from a linear menu without feedback because of the variability and acceleration in mappings from device to pointer motion. The crossing-based design works by activating an item when a line is crossed, rather than when a specific visual object is clicked. Coupled with the radial layout, this enables experts to learn gestures and execute pointer-based commands without feedback from the interface.

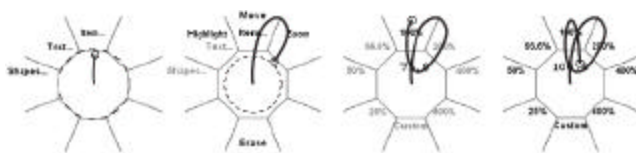


Figure 2: FlowMenus enable autonomous interaction with a pointer-based GUI (from [11]).

2. CONCENTRATE AND AVOID INTERRUPTION

The next primary characteristic of optimal experience is that for people to reach this state, they must be able to focus their attention at length on the task at hand. Concentrating on the task further enhances one’s focus, often enabling them to tune out other input. Activities such as reading or painting, for example, are good at encouraging this kind of concentration because they involve very little interaction with anything outside the immediate task.

This naturally brings up the question of what kind of interfaces can best support concentration and avoid

interruption. There are a few ways of looking at this. The first might be to look at when is a good time to interrupt users. Horvitz studied this in the context of when the best time is to automatically suggest something about scheduling an appointment after interpreting an incoming email [12].

Others have observed that it is essentially never a good time to interrupt the user. Thus, dialog boxes that pop up in the center of the screen, especially when they are modal, should be used with extreme caution. In fact, some current interface designers have made efforts to reduce the number of dialog boxes. Windows XP, for example, puts non-critical messages in “bubbles” linked to icons in the system tray in a corner of the screen. They are in the user’s visual periphery, they fade in and out, and they can be ignored if desired.

Some researchers have taken this approach further by investigating the use of “ambient displays”. These are displays off the traditional computer monitor. For example, Ishii has used bottles, projected light, and pinwheels to show a range of peripheral information [9, 13]. Greenberg has used small puppets to display whether or not a colleague is available [15]. More simply, users often are aware of the noise that their personal computer hard drive makes, and use that awareness to help understand whether their computer is busy processing. These and other similar kinds of approaches have potential in reducing interruptions, but there remains an open question of how successful they are.

Given more traditional interfaces, some users do their best to customize the interfaces they are given to allow them to focus on their task. For example, programs that enable users to hide unused toolbars and other interface items or let users work in full-screen mode are good in this respect. Users that put in the effort to configure their interface can focus more fully on a specific task. Some interface designers may argue that users shouldn’t be required to configure interfaces and that configuration is the antithesis to flow. Alternatively, configuration can be seen as an unfortunate, but necessary precursor to flow. This is similar in spirit to turning off a telephone ringer, or clearing a desk – in order to focus. As such, giving users control over their environment is a good thing.

Another approach that can potentially be used to reduce users’ sense of interruption is to use animation between screen states. While animation in general can be used in ways that are very disruptive, we have found that animation can be helpful if used to help users understand how the interface changes. The potential for this kind of animated transition is that it can reduce the cognitive overhead of understanding of the relationship between two screen states, thus enabling users to stay focused on the task. The trade-off is that animations fundamentally require time. So a question is whether the time spent on such animations is well spent.

We attempted to understand this issue by conducting a study looking at how users responded to animated transitions within a visual family tree application [3]. Tasks included finding a person’s grandfather or sibling. There were visual links around the person that navigated through the tree. The transitions were controlled to be instantaneous or animated over one second. The animations did not affect overall primary task time or accuracy. But animation did help users reconstruct the tree afterwards. Others have designed animation into their visual applications in similar ways for the same reason since the earliest visualizations [6, 20].

Reducing the need for users to consciously make connections between different interface states has the potential for reducing their short-term memory load. We believe this is the reason for the effectiveness of DateLens, a fisheye calendar we built for Pocket PC PDAs [4]. DateLens presents days within the context of other nearby days and uses animated transitions to change focus days (Figure 3). We performed two studies comparing DateLens to Microsoft Calendar 2002. In both cases, we found that users performed significantly faster and with higher accuracy using DateLens on tasks that required analysis of longer time periods (such as finding a time to meet in the next few weeks). Showing days within the context of surrounding days enabled users to navigate and compare and contrast nearby days with a reduced memory burden because they could use spatial memory to aid navigation.

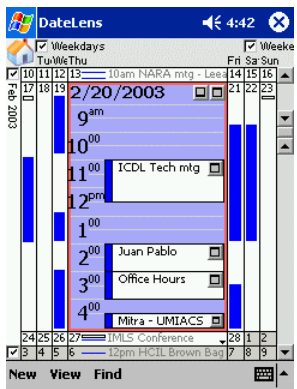


Figure 3: DateLens calendar shows details in context.

3. MAINTAIN CONTROL

The third characteristic of activities that frequently result in flow is that the person must be able to maintain control over the activity. Lack of control, such as when driving in traffic is a sure way to destroy flow.

Applying this lesson to interface design, we can see that it is important that the user feels in control of the application. So, systems that make decisions for the user and modify the interface based on those decisions are at risk of making the user feel out of control. Adaptive interfaces, which change based on the user’s behavior, are a prime example of this category.

Let us look at a specific example. There are many times when interface designers must balance the needs of novices and experts. This commonly comes out when deciding how to present menu items since it may be better for novices to see only a few items while experts may want access to the full palette of possible actions. Some applications solve this problem by using adaptive menus that start with most menu options hidden, available by clicking on a special button. Then, items that haven’t been activated are eventually hidden, and items from the hidden area that are activated are brought into the initial view. While this approach works, it is potentially disruptive to flow because users don’t know what to expect. They are effectively brought back to the first stage of interaction because they must always learn from the interface since it sometimes changes. This interpretation is backed up by Kaptelinin’s study that showed the importance of users’ memory of menu item locations [14].

Adaptive menus are an attempt to solve a real problem – the difference in needs between novices and experts. But it does so in a way that removes control from the user. An alternative approach that we developed is to show some items and hide others behind a special button, as with the adaptive windows. But instead of having the computer decide when to move items between the visible and hidden areas, we let the user make the decision.

We implemented this approach for a similar task, which is browsing of files. We observed that we rarely want to examine all file system folders, and that many get in the way of accessing the small number of folders that we do want to find. So we developed *Favorite Folders* (Figure 4), a plug-in for Windows Explorer that lets users hide specific folders behind ellipses (“...”) [16]. Users can manually control which folders are “favorite” and appear in the primary view, and which are “archived” and hidden beneath the ellipses, requiring an extra click to access.

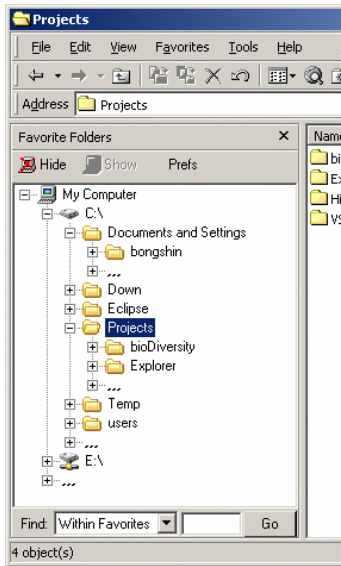


Figure 4: Favorite Folders, a prototype file browser that enables users to hide unwanted folders behind ellipses.

4. SPEED AND FEEDBACK

Csikszentmihalyi showed that flow-related activities must allow the person to clearly set goals and receive feedback about their progress towards those goals. Novak, Hoffman, and Yung [18] furthered this recognition in the context of web design, showing that speed had a direct influence on users' experience of flow when navigating the web.

This obviously isn't news for interface designers. We know that the more responsive the interface the better, and that users must consistently receive prompt feedback in response to their actions.

But Csikszentmihalyi's identification of the importance of setting goals is more challenging. Recent consumer-oriented software often has task-based interface elements, such as the common tabbed pane design of many e-commerce websites where each tab represents one task. But this is only the first step. Can we go further by helping users recover from lost flow when they go down a wrong path or get interrupted? Having good "undo" support is important in this regard, but perhaps we also should consider more substantive approaches to helping users understand where they are within a task.

5. TRANSFORMATION OF TIME

The final characteristic of flow-related activities that we discuss in this paper is the transformation of time. People regularly report that their perception of time changes when they are in the flow. This offers a direction to a possible metric for understanding flow.

It turns out that there is a psychological principle summarized by Weybrew [21] that says that when interrupted during an activity, people will overestimate the time that activity took.

Czerwinski, Horvitz and Cutrell showed how this can be harnessed [8]. They ran a study where people performed web-based tasks of various difficulties, and were given varying amounts of assistance. Each participant was asked to estimate how long each task took, and the study compared this subjective estimate of task duration to the actual duration.

They found that the more difficult the task, the longer the participants thought the task took relative to the actual task time. They called this measure Relative Subjective Duration (RSD), and suggested it be used as a general metric to accompany traditional self-reported subjective satisfaction as a way to avoid the positive bias commonly associated with subjective preference reports.

The connection of RSD to flow is admittedly loose, but we know that RSD is a reliable measure of interruption and task complexity, and we know that being interrupted is negatively associated with flow. So we leave it as a hopeful, but open conjecture as to whether RSD can be used to measure flow.

EXAMPLE APPLICATION – NOTABLE

While many applications exhibit many elements of the flow characteristics described in this paper, the interfaces described so far were largely written before I started thinking about flow in these terms. More recently, I tried to build an interface that explicitly supports the characteristics of flow identified here. I describe it here with the goal of further understanding how the concepts of flow can be practically applied.

The interface I describe is for "Notable", a note-taking program (Figure 5). I initially built Notable for my personal use as I wasn't satisfied with existing note-taking software precisely because I felt that existing tools got in the way of flow. I designed Notable to be as simple, lightweight and as fast as possible to support the tasks of writing, finding, and modifying short notes. Notable is perhaps most notable for its lack of innovative features. It is a very traditional GUI-based application. Yet, it combines many of the characteristics described in this paper in a way that results a simple, but useful example.

Notable works as follows. Users create new notes through the menu or a keyboard shortcut. The first line of the note is special and is considered to be the note "subject". All note subjects are displayed in the list box at the near-top right of the interface. Typing in the find box (just below the menu bar) immediately starts filtering the note list to show only those notes that contain the word(s) entered in the find box. Or, if "Search Subject Only" is checked, only the subject of each note is searched. This is like a common incremental search, but it shows all the search results dynamically as users type. This enables a search strategy of typing until few enough results are shown so they can be seen all at once without scrolling.

The tree widget on the left shows categories that can be manually assigned to notes, and can be used to further filter which notes are shown. Finally, to make it easier to integrate into possible existing note-taking strategies, Notable syncs with Microsoft Outlook notes, or with file system text files. Notable also syncs with Microsoft Outlook email and contacts since those data sources contain a text field that is commonly used to store generic note-style information.

Notable shares many characteristics with Stuff I've Seen (SIS) [10] and Microsoft's new OneNote product. Like SIS, Notable uses a full-text index to enable instantaneous search on any word in any note. Like OneNote, it supports management of notes without giving notes explicit filenames or locations.

Notable was designed with the characteristics of flow discussed in this paper in mind as follows.

Challenge & Require Skill: While Notable is very simple to get started using, it also supports some "expert" features that require learning. However, use of these features have the potential to increase the user's efficiency and focus. One simple example is that the first character of the search word the user enters can filter note types. For example, if the first character is '@', only emails are shown, or if it is '/', only file system notes are shown. These filters are learned in a similar way to how keyboard shortcuts are learned. There is a standard dialog to control these filters with information about the character filters.

By offering the ability to restrict the search to the subject, users are encouraged to put keywords in the subject field which will aid future note finding. This is an implicit encouragement for the user to modify their behavior to fit the interface, which is not usually considered a good design approach. But if users do this, they will be able to retrieve their notes more efficiently.

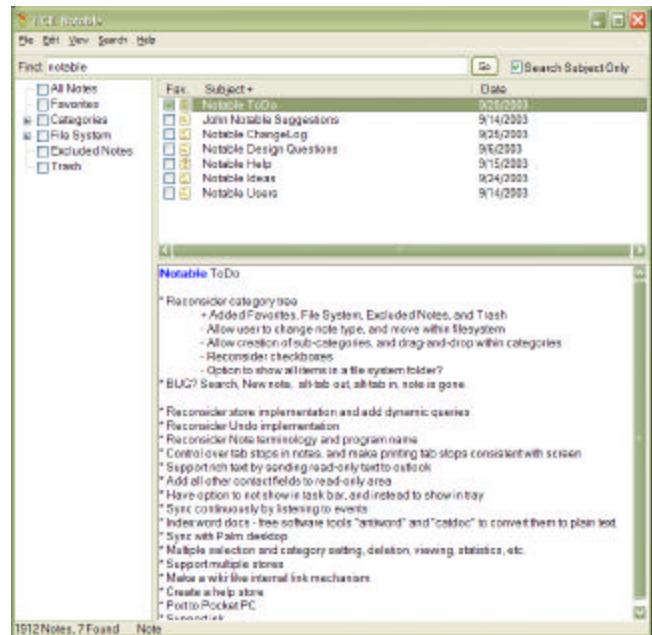


Figure 5: Screenshot of Notable, a note-taking program we developed with explicit consideration of the characteristics of flow described in this paper.

Concentrate and Avoid Interruption: Notable is designed with traditional GUI widgets, but it has keyboard shortcuts for every command so that users can apply stages 1, 2 and 3 of skill acquisition to the interface. In addition, special care has been made to allow non-standard but natural key presses to navigate. For example, pressing the down arrow within the find box moves the cursor focus to the result list, and pressing the enter key in either the find box or the result list selects the focused note, and moves the keyboard focus into the note for immediate typing. This offers single keystroke mechanisms to perform common activities.

Maintain Control: The search algorithms are straight forward and easy to understand so that users can learn how what they type in the find box will affect what shows up in the result list. By giving users the option of searching just within the subject (with a keyboard shortcut for controlling this feature), they can have tighter control over the process. But it is also configurable for full search and item type. An early design offered search results by rank, but wasn't used very much because, apparently because it was too hard to understand what it meant. The current version instead offers results sorted by name or date.

Speed and Feedback: Notable was built to be very fast, giving nearly instant feedback as you type. There is no need to press go, enter, or even finish words in order to see search results. This is a kind of a dynamic query [1] that minimizes the typing one has to do to find a note. If, for example, a user is looking for notes about 'notable', but the result list is filtered down to just 3 notes after typing 'nota', then the user can immediately select the relevant note without finishing the search.

Use: Notable is under active development, but it is far enough along so that it has been used extensively for about two months by four people, including this paper's author. It has quickly become a standard application that is always on for each of us. Perhaps the most interesting observation is that it has changed the cost structure of information access. It used to be that finding notes was relatively slow, but editing notes was relatively fast, and so I would create fewer longer notes. Now, finding notes has become relatively fast, and I have started creating more shorter notes. In fact, I even edited several long notes and broke them up into shorter pieces so I could access each individually. We have not yet performed any real studies, including examination of relative subjective duration.

CONCLUSION

Flow is admittedly a hard concept to pin down. Yet it is a basic and important part of human experience. As such, anything we can do to increase the opportunities that computer users might have to improve their experience seems worthwhile. This paper attempts to learn from the existing literature on flow to give a language to interface designers working to improve user experience.

In summary, interfaces that are targeted at improving user's ability to stay in the flow shouldn't underestimate the importance of speed in supporting creativity, quality, and enjoyment. Every time there is an interruption, literal or conceptual that gets in the way of users concentrating on their tasks, flow is lost. Slow interfaces, which I define as any that get in the way of users acting on their work as quickly as they can think about it, are problematic.

Similarly, as has often been said before, users have extremely limited short term memory. Any interface elements that strain user's memory are problematic because, again, the user's flow will be interrupted.

Balanced with the many details of interface design is the constant need to consider the trade-offs between novice and expert users. Many interface designers find themselves feeling trapped with no opportunities to support expert users – but this is a trap that must be avoided, and can be. The new Mozilla Firebird web browser, for example, has a hidden “incremental search” feature that allows users to search within a page and follow links, all from the keyboard. This is an advanced and “scary” feature to some – but many of us that have put the energy into learning it have found that it has dramatically improved our web browsing efficiency.

My vision of computer interfaces is that they become tools in the best sense – that they become an invisible extension to our body, so we can apply them to our work with just barely being aware that we are doing so. Computers should be able to help us concentrate on our work, without concentrating on the computer.

This opens up major questions of evaluation. How do we know when we have built interfaces that succeed according

to this vision? How do we know when interfaces encourage flow? Czerwinski et al have offered a start by introducing Relative Subjective Duration (RSD) [8]. Novak et al have begun to explore user models to measure customer experience [18]. But this largely remains an open question. The question of how flow might apply to collaborative settings and team work also remains open.

These are hard issues, and not ones that are likely to be easily answered by simple quantitative user studies. But they are important nevertheless, and I hope our community does not shy away from the qualitative and non-traditional approaches that will likely be needed to increase our understanding.

ACKNOWLEDGMENTS

This work is a result of many discussions with a range of people. I particularly appreciate my interaction with Lance Good and Mark Stefik at PARC who got me started thinking about Flow. I then continued discussions with my local colleagues François Guimbretière, Ben Shneiderman, Allison Druin, Hilary Hutchinson, Harry Hochheiser and others at the HCIL, Peggy Storey at the University of Victoria, Andy Cockburn at the University of Canterbury, and John SanGiovanni at Microsoft Research. I also am thankful to Jesse Grosjean, Aaron Clamage and John Haller who have been early users of Notable.

This work was supported in part by a generous gift from Microsoft Research.

REFERENCES

- [1] Ahlberg, C., & Shneiderman, B. (1994). Visual Information Seeking: Tight Coupling of Dynamic Query Filters With Starfield Displays. *In Proceedings of Human Factors in Computing Systems (CHI 94)* ACM Press, pp. 313-317.
- [2] Anderson, J. R. (1995). *Learning and Memory*. New York: John Wiley.
- [3] Bederson, B. B., & Boltman, A. (1999). Does Animation Help Users Build Mental Maps of Spatial Information? *In Proceedings of Information Visualization Symposium (InfoVis 99)* New York: IEEE, pp. 28-35.
- [4] Bederson, B. B., Clamage, A., Czerwinski, M. P., & Robertson, G. G. DateLens: A Fisheye Calendar Interface for PDAs. *ACM Transactions on Computer-Human Interaction*, (in press).
- [5] Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988). An Empirical Comparison of Pie Vs. Linear Menus. *In Proceedings of Human Factors in Computing Systems (CHI 88)* ACM Press, pp. 95-100.
- [6] Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991). The Information Visualizer, an Information Workspace. *In Proceedings of Human Factors in*

- Computing Systems (CHI 91)* ACM Press, pp. 181-188.
- [7] Csikszentmihalyi, M. (1991). *Flow: The Psychology of Optimal Experience*. HarperCollins.
- [8] Czerwinski, M., Horvitz, E., & Cutrell, E. (2001). Subjective Duration Assessment: An Implicit Probe for Software Usability. (*Human-Computer Interaction (IHM-HCI 2001)*) pp. 167-170.
- [9] Dahley, A., Wisneski, C., & Ishii, H. (1998). Water Lamp and Pinwheels: Ambient Projection of Digital Information into Architectural Space. In *Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 98)* ACM Press, pp. 269-270.
- [10] Dumais, S. T., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., & Robbins, D. C. (2003). Stuff I've Seen: A System for Personal Information Retrieval and Re-Use. In *Proceedings of 26th Annual International Conference on Research and Development in Information Retrieval (SIGIR 2003)* New York: ACM, pp. 72-79.
- [11] Guimbretière, F. (2000). FlowMenu: Combining Command, Text, and Data Entry. *UIST 2000, ACM Symposium on User Interface Software and Technology, CHI Letters*, 2(2), pp. 213-216.
- [12] Horvitz, E. (1999). Principles of Mixed-Initiative User Interfaces. In *Proceedings of Human Factors in Computing Systems (CHI 99)* ACM Press, pp. 159-166.
- [13] Ishii, H., Mazalek, A., & Lee, J. (2001). Bottles As a Minimal Interface to Access Digital Information. In *Proceedings of Extended Abstracts of Human Factors in Computing Systems (CHI 2001)* ACM Press, pp. 1887-1888.
- [14] Kaptelinin, V. (1993). Item Recognition in Menu Selection: The Effect of Practice. In *Proceedings of Human Factors in Computing Systems (InterCHI 93)* ACM Press, pp. 183-184.
- [15] Kuzuoka, H., & Greenberg, S. (1999). Mediating Awareness and Communication Through Digital but Physical Surrogates. In *Proceedings of SIG-HI of Information Processing Society of Japan (SIG-HI)*
- [16] Lee, B., & Bederson, B. B. (2003). *Favorite Folders: A Configurable, Scalable File Browser*. Tech Report HCIL-2003-12, CS-TR-4468, UMIACS-TR-2003-38, Computer Science Department, University of Maryland, College Park, MD.
- [17] Norman, D. A. (1988). *The Psychology of Everyday Things*. Basic Books.
- [18] Novak, T. P., Hoffman, D. L., & Yung, Y.-F. (2000). Measuring the Customer Experience in Online Environments: A Structural Modeling Approach. *Marketing Science*, 19(1), pp. 22-42.
- [19] Perlin, K. (1998). Quikwriting: Continuous Stylus-Based Text Entry. In *Proceedings of User Interface and Software Technology (UIST 98)* ACM Press, pp. 215-216.
- [20] Robertson, G. G., Mackinlay, J. D., & Card, S. K. (1991). Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of Human Factors in Computing Systems (CHI 91)* ACM Press, pp. 189-194.
- [21] Weybrew, B. B. (1984). The Zeigarnik Phenomenon Revisited: Implications for Enhancement of Morale. *Perceptual and Motor Skills*, 58, pp. 223-226.