A Retrospective and Prospective View of Information Technology Professionals' Use of Tools: Maturing the User Experience

Candace Soderston

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052
csoders@microsoft.com

Abstract. The Information Technology (IT) professional's user experience has evolved both in terms of software tool attributes and the process workflow within which the IT tasks exist. Based on user research with IT professionals over the past two decades, including ethnographic observational field studies, focus group discussions, and hands-on user tests with analyses of errors and perceived complexities, this paper presents a set of hypotheses about user interface design aspects that across the industry most impede the workflow of typical IT professionals. It discusses how this relates to typical information workers' behavior, as well as what steps we as tools and systems designers can take to better support the IT professionals' user interface needs.

Keywords: IT professional, enterprise software, user interface, user experience, UI design, command line interface, CLI, graphical user interface, GUI.

1 Introduction

This paper started with an intent to re-examine and report on the state of the art in understanding and designing software to enable the most productive and satisfying workflow and habits of IT professional (hereafter referred to as "IT pro") audiences. Specifically, my objective was to challenge our typical answers to these 2 questions:

1. How far have we come in the last decades in extending IT management capabilities beyond the realm of the dedicated scientists who were deeply rooted in creating the technology to meet their own needs, to a more generalist though highly skilled IT pro end user audience?
2. Are IT pro audiences fundamentally different from the larger set of end users of information productivity software and, if yes, in what ways do these differences require fundamentally different UI designs, principles, and paradigms?

Although I started with the intent to answer these questions, my review did not end with this. In this paper I posit an answer to the first question, while decomposing the 2nd question into parts. Evidence leads me to claim that IT pro audiences are NOT *fundamentally* different from other end user sets, but rather, that they sit in different places on a continuum of user experience, and they carry their learning forward from their information worker role to their IT pro role. This understanding could help us

M.J. Smith and G. Salvendy (Eds.): Human Interface, Part I, HCII 2009, LNCS 5617, pp. 306–315, 2009.

change UI design approaches from the continuing typical "either/or" propositions (e.g., whether to optimize for novice or for expert) to a more synergistic approach that enables both, in concert. Training wheels don't have to be removed if they're not in the way, aren't heavy and are needed from time to time.

Admitting this common "humanity", as well as identifying those aspects that evolve with further learning, puts us in good position to more insightfully merge the well-proven benefits of the common graphical user interface (GUI) -- where users can explore and recognize landmarks – with the also well-proven benefit that expert fast paths provide, such as those offered by command lines and scripts.

1.1 How Far Have We Come in Extending IT Management Capabilities beyond the Realm of the Dedicated Scientists Who Created the Technology?

It is a commonplace understanding today that the computer was initially a tool for select scientists, to run their own analyses, to support their scientific research and lines of inquiry through sharing the time/cycles offered by a mainframe computer. At this stage in history, the people creating the technology and the interfaces to it, and the people using them, were essentially the same individuals.

As computing capabilities grew beyond the "glass house" (those working inside the raised-floor, water-cooled IT center), to support other audiences (e.g., word processing for professional authors, financial accounting for professional accountants, searchable databases for librarians, and all these for everyone), so did the number of IT producers grow and diverge from the targeted end users. In the United States alone, even during today's economic slow-down, there are over 5000 postings for "Information Technology management" jobs in online databases[1]). These skills are in high demand, even though they represent a large cost category for enterprises in managing their businesses (along with capital and energy).

Up through the 1970s, computer science and software engineering degree programs didn't typically exist as major areas of study, but were an adjunct to the mainline scientific degrees that used them (math, physics, chemistry, biology, and all the "engineerings": e.g., electrical, civil, mechanical.). Early generalists, hired to support the scientists, were trained on-the-job to write in binary, "machine language"

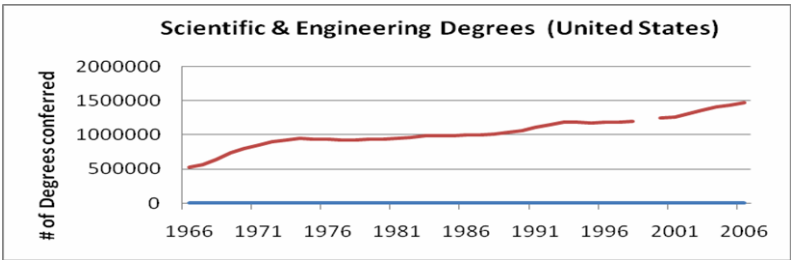

Fig. 1. National Science Foundation Statistics see http://www.nsf.gov/statistics/nsf08321/content.cfm?pub_id=3785&id=2

[1] For example, www.monster.com (on 2/9/2009)

or other low-level code to automate some of the more common aspects of programming, and came from other main-line disciplines, most notably from both music and math majors. Today, degrees in software engineering and computer science are offered by most universities and colleges, and the numbers are large (see Fig.1).

Graduates coming out of the computer science and software engineering programs today and working in the IT management jobs differ in many keys ways from their earlier counterparts. These graduates are highly technical, but they are also not dedicated to one particular scientific field, research focus, program, task, or language. They are much more horizontally distributed in focus than were yesterday's technocrats and they are not typically representative of their own end users.

1.2 In What Ways Do IT Professionals Require Fundamentally Different UI Designs, Principles, and Paradigms from Other End Users?

Today's IT pros' lives are filled with multi-tasking, with learning and using multiple programs, with diagnosing and fixing problems that occur at multiple layers in the stack. The wider heterogeneity in the end users being supported, the greater span in the applications being supported, and the more rapid rate of progress in technology development all contribute to the challenge an IT pro has in juggling knowledge … what gets left in short-term memory, constantly being wiped out and refilled, and what gets stored in long-term memory for continued, repetitive access.

1.3 The UI from "Yesterday" – Blank Screens, Programmed Function Keys, Keyboard Templates, "Run Books"

Command line sessions, blank screens. The command line, blank screen user interface was the only user interface in the early days of computing, enabling scientists to run their few dedicated programs and get their resulting data in "real-time" or to "batch" the programs up for running later during the slowest times, typically overnight. Command line access depends on ability to recall and produce the appropriate syntax.

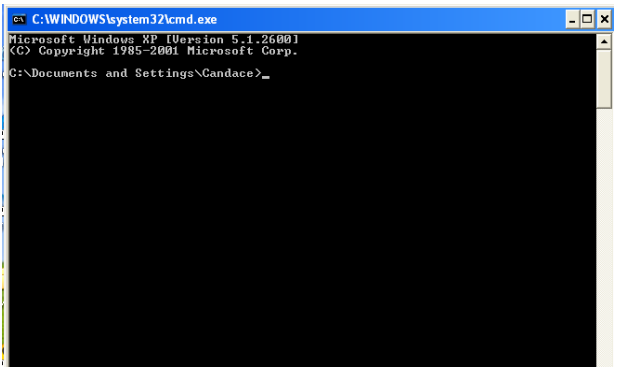

Fig. 2. Command Prompt

Fig. 3. Function keys

Programmed function keys. PF keys, shortened to F keys in most keyboards today (Fig 3), were created to enable IT staff and other users to tie their most frequently used sequences of commands (scripts) to particular function keys, so that a single key press could do what formerly required more typing. Remembering a single key rather than a string with exact sequence/syntax was much easier and cut down on errors and time.

Keyboard template overlays and paper "run books". Both of these user interface aspects -- dedicating a key to a frequent-use command, macro/script, or application, and making this visible via a keyboard template overlay – were created to help with the recall problem that people face when they are not dedicated to and frequently using the functions. Today many applications and most users don't rely on these.

Through cognitive science we know that recognition is more robust than recall, and this applies to all humans, including IT pros and other end users. The UI paradigm shifted over the last decades from blank screen command prompt UIs to graphical UIs providing explorable landscapes, aimed to enable users to find features when they are multi-tasking rather than performing a single task repetitively.

1.4 Today – GUIs, Multi-tasking, Multiple Programs, Scripts, Commands

In the late 1980's - early 1990s, I led a small team at IBM in benchmarking early GUI shells that were developing, before there was a shake-out of the alternatives into a set of common principles and standards. Users were crying for consistency and predictability in common mechanics across applications and environments, so they could transfer their learning from task to task. (As a backdrop, the average number of software packages they reported using was 3; the packages listed most often included Lotus 1-2-3, WordPerfect, MacWrite, and DOS.)

In this work, we tested 95 information workers who used productivity software at home and work, in different GUI environments, and logged their behavior:

- 34 people did tasks across a full day, half of the people in Apple's MAC System 6 and half in Microsoft's Windows 3.0 environment.
- 31 people did the same tasks as above and other similar tasks across 2 full days, about half of the people in DOS and half in IBM's OS/2 environment.
- 30 people did the same tasks as above and other similar tasks across 3 full days, one-third of the people in Apple's MAC System 6, one-third in Tandy's DeskMate (a DOS shell), and one-third in IBM's PS/1 (another DOS shell).

We codified the errors people made within these environments and studied this across time to understand the learning curve characteristics: which errors persisted over time in a longitudinal study and which others dropped off as people became

more experienced and repeated tasks. Out of this lengthy and rigorous benchmarking effort (see Fig.4), we codified the top 10 user error types that occurred across ALL of the GUI environments and that persisted across days. (See Table 1.) We developed a faster heuristic evaluation method based on these aspects and used this to compare HP'sNewWave 3.0, Sun's OpenWindows 2.0 OPENLOOK, NeXT's NeXTStep 2.1, Metaphor, Microsoft's Windows 3.0, and OS/2. [1]

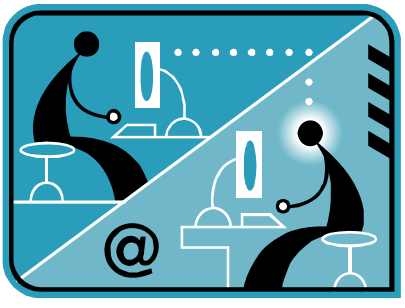

Fig. 4. Benchmark test representation

Table 1. Top 10 User Errors Associated with Graphical User Interfaces

1. Ambiguous menus and icons – Users were unable to tell from cryptic word labels what categories of tasks they represented. Object/action and noun/verb ambiguities contributed. For example, a user might be looking for a way to view a file, and would look under "view", but had to instead look under "file".
2. Single direction language – People chose actions without first selecting objects. This resulted in "no-operation": no action and the user was given no feedback. In communication between human beings, people can talk in either active or passive voice ("the ball was hit by the boy" = "the boy hit the ball").
3. Complex linkage between and within applications – Extraneous data conversions and mode switches were required to complete tasks.
4. Unclear step sequences -- People chose wrong actions, omitted required actions, and performed unnecessary actions – all contributing to losing their way in the sequence of steps to complete their overall task goal.
5. More steps to manage interface than to do common tasks (in users' model) -- For example, accomplishing "simple" cut-copy-paste goals required 12-15 steps in the GUI platforms evaluated, whereas participants tried all kinds of shortcuts that didn't work. This is highly related to the previous category but this is a subset that relates only to the common GUI tasks, not to the end-to-end step sequencing to achieve the users' overall task goal.
6. Inadequate feedback and confirmation – People were not confident that their requests were acted upon by the system so, for example, sent notes redundantly and saved data, then exited and re-entered files in order to make sure the changes had been made. This type of user operation resulted in no feedback from the system.

Table 1. (*Continued*)

7. Lack of system anticipation and intelligence – Users were forced out of their task by lack of anticipation on the part of the software. E.g., when trying to save a file to an unformatted diskette, most systems simply issued a dead-end error message. On the other hand, some systems (the MAC, notably) issued an error *dialog* listing the alternatives that could be behind the error condition (could be unformatted, formatted for another system, or damaged) and asked the user if he/she wanted to try formatting the diskette. If user chose 'yes', the system formatted the diskette and saved the user's file.
8. Input and direct manipulation limits – People had difficulty with operations related to positioning the mouse-cursor correctly: to pick up small targets, such as window borders; to place an insert cursor within text, etc., and related to timing: double-click speed. User statements indicated frustration with these miscellaneous recoverable errors and requested support for alternatives to direct manipulation.
9. Highlighting/selecting limitations – People tried selecting multiple discontiguous text strings and spreadsheet cells to operate on and were frustrated when their 2nd selection caused the 1st to be de-selected. E.g., this happened when trying to make bold all proper nouns in a document and when trying to add a specific value into several spreadsheet cells. People also tried to refine selected areas, for example, to stretch one of the end-points out to cover more text or to shrink it to cover less.
10. Inadequate error messages, help, tutorials, documentation – Users referred to these support items and were unable to locate helpful information. Problems lay in retrieval aids and in information content that was descriptive rather than procedural.

Yes, these problems were identified by studying typical information workers rather than IT pros, and the GUI paradigm has improved since the time these data were collected. However, from observing many IT pros over the past decade doing their tasks in the field and in usability tests in the laboratory, there is reason to believe that these same UI design aspects remain top contributors to user errors – all users.

2 Do IT Professionals Require Fundamentally Different UI Designs, Principles, and Paradigms Than Information Workers?

Over the past 4 years I and other user experience professionals in the teams I've worked with here at Microsoft collaborated in conducting ethnographic field research at customer sites[2]. This research incorporated typical "follow me to work / day in the life of" observational study procedures, where we spent a minimum of a day and an average of 2 days at each site. User experience researchers "shadowed" selected participants over several hours, intruding as little as possible so as to observe the typical work day/work flow, and concluded with a structured interview at the end.

Included in these studies were IT pros and particular types of information workers (data analyzed separately), including Exchange messaging administrators, SQL

[2] Lin, A., Huang, D., Engelbeck, G., Mueller, L., Stempski, M., Gunderson, N., Mings, S., Soderston, C., (2006/2007)

Server database administrators, database developers, PBX / voice messaging administrators, Compliance / security administrators, business analysts, financial analysts, administrative assistants, and other highly mobile information workers. Following are three of the hypotheses developed from these studies that have already been incorporated into Microsoft solutions for these audiences in the field.

2.1 Hypothesis 1: IT Pros Spend Most Time in Information Worker Tools

These IT pros spend the lion's share of the time that they are at their computer in their email program, their internet browser, and their spreadsheet program, e.g., Outlook, Internet Explorer, and Excel. They multi-task, they have many programs open at the same time and they shift their attention back and forth across windows as needed to support the tasks that pre-empt, interrupt and integrate with their on-going work flow. They like having multi-windowing capability, they like being able to do another task while one is running in the background. They collaborate, they delegate, and there is workflow that they support outside of their typical IT tools.

This mirrors what we observe in the typical office information workers' behavior. The IT pro end users are very familiar with and are spending the lion's share of their computer time in the same UI constructs as the information workers. This supports the proposition that UIs don't have to be *fundamentally* different for IT pros from what they are for information workers. IT pros will be using both "classes" of tools side-by-side and transfer learning at the UI "primitives" level from one to another if they are consistent at this level. For a "caveat" on this, see the next items.

2.2 Hypothesis 2: IT Pros Expect Full GUI & Scripting Support for All Operations

Most messaging and database administrators do not write scripts in their daily jobs today. They either write scripts infrequently or use existing scripts that were written by others for infrequent tasks (re-installing), and for frequent bulk, repetitive operations. Like information workers, the IT pros expect to be able to do all tasks through a GUI. Unlike information workers, however, they also expect all tasks to be "scriptable".

This finding was addressed in a novel way in a "version 1" sense in Microsoft's Exchange Server 2007, in the design of the Exchange Management Console, and in Windows Powershell (an extensible command line shell and scripting capability – see http://en.wikipedia.org/wiki/Windows_PowerShell)[3]. The management console in Exchange Server 2007 was the first Microsoft administrator console written entirely in Powershell. All GUI operations done by the user are executed by the GUI "under the covers" in Powershell.

Early usability testing showed that Exchange administrators would have a learning hurdle to overcome in scripting in Powershell, though they loved the value proposition. To help in the initial learning curve as well as in continuing memory refresh, the team designed a way for the GUI to show what Powershell scripts were being run to do the tasks (see Fig.5). People can ignore this or attend to it in order to "learn by doing". In addition, they can copy/paste to create and modify their own scripts.

[3] Lin, A., Langowski, A., Clark, B., Frijlink, N., Mings, S., Sharma, V., Soderston, C. (2006).

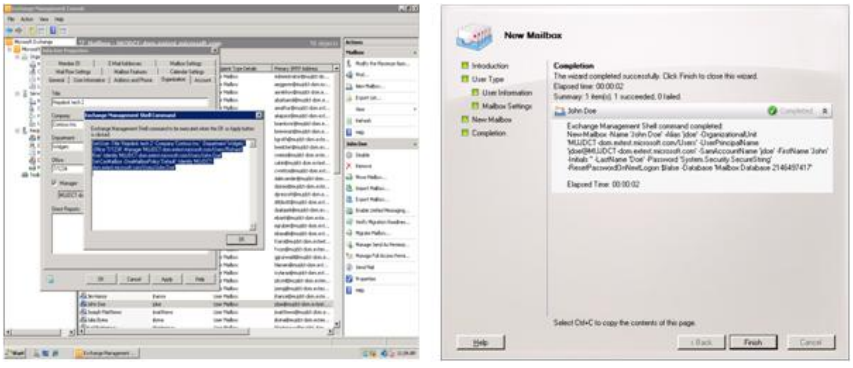

Fig. 5. Script display in dialog confirmation and in wizard

Information workers don't typically encapsulate their GUI tasks into macros, but IT pros appreciate the capability to do so.

2.3 Hypothesis 3: IT Pros Need Task Orientation in UIs

The last hypothesis on the commonality between IT pros and average information workers has to do with transfer of learning and navigation challenges. IT pro roles today are separated across individuals, but their roles are constantly evolving. The way responsibilities are allocated across IT pro team members shifts continually, the task responsibilities transfer across individuals, and the trend is that the pace of change in this regard is accelerating.

Being able to transfer learning from one tool to another and to find function (if authorized) from a common launchpoint, is highly valued by both IT pros and their companies and is still a significant challenge today. They are stressed to find function as quickly as possible, and many times this is in front of their customer/end user.

IT pros, like other users, think in task terms, in verb/action language, and have similar challenges fitting this aspect of their interaction approach into the object-oriented, single direction language environment. They're challenged to find where the first step for tasks lies in deeply nested object hierarchies, and what order to perform steps when their task crosses objects and hierarchies without any sequencing built into the software. In a "version 1 and 2" sense, this proposition was addressed in the UI design of Microsoft's BizTalk Server 2006 and Exchange Server 2007.

Microsoft administrator consoles, like others, are object-oriented, laid out with a navigable tree structure in a left pane. IT pros navigate to the objects of their choice and then perform their chosen tasks applied to those objects. In BizTalk Server 2006, the team applied a task overlay to the console, enabling the IT pros to locate their task start points through navigating a task hierarchy – *in addition to locating through the object tree* (see Fig.6). This console "root hub" of the tree hierarchy provides a display in the right pane of the major task categories, with procedural information on how to complete them. This was a user experience-led initiative, which quickly

garnered great support and collaboration across the technical teams and user education staff [4].

In Exchange Server 2007, the team took this concept even further, to support learning the OO structure "by doing". In Exchange Server 2007, the tree structure was significantly simplified, an actions pane was added on the right that is context-sensitive and "bubbles up" the tasks that can be applied to the selected objects, and a console root hub page similar to BizTalk Server's was added to integrate all configuration tasks (see Fig. 7).

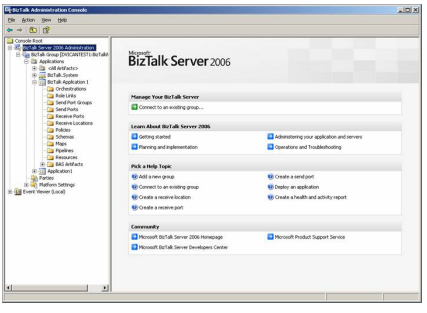

Fig. 6. BizTalk Server 2006

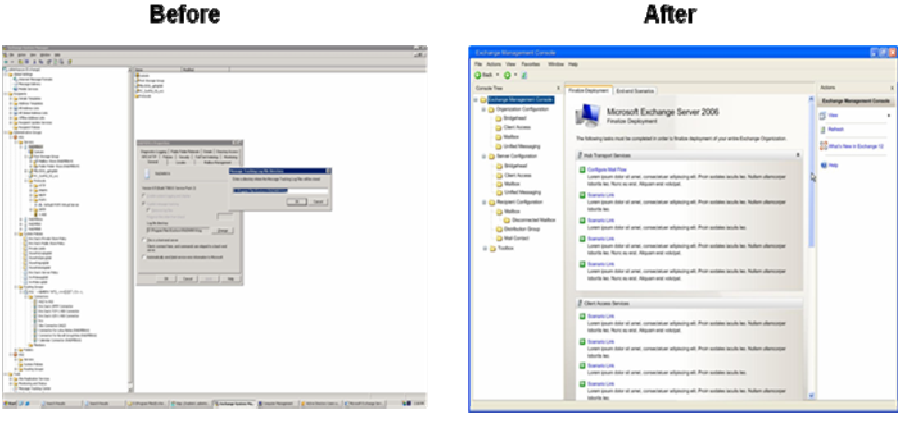

Fig. 7. Exchange Server 2003 vs 2007

This "start page" took the design to the next level, creating a tab structure to make it extensible to other task categories, and adding what we light-heartedly code-named a "post-its" construct to provide procedural task information and links to the appropriate places in the object hierarchy. These persist and float on top, even when the task flow replaces the middle pane start page information with the particular object's contents. These start page structures and the floating task instructions/links enable IT

[4] Vican, D., Scott, P., Christman, C., Hanswadkar, K., Perry, N., McElroy, P., Casey, T., Soderston, C., (2005).

pros to learn the object hierarchy while they're doing tasks. This was a relatively lightweight way to superimpose task structure on top of an existing object hierarchy, without having to re-architect the object hierarchy in a more substantial way.

3 Conclusion

Observational studies of IT pros and information workers in their environment show the many ways their behavior is similar [See also 2,3,4]. IT pros typically spend the lion's share of their working day in email, browser, and spreadsheet programs (e.g., Outlook, IE, Excel). Being able to transfer learning from one tool to another and to find function (if authorized) from a common launchpoint, is highly valued by both IT pros and information workers. Both IT pros and information workers think in task terms, in verb/action language. They have similar challenges fitting this aspect of their interaction approach into a single-direction UI, one in which objects must always first be chosen. Ideally they want to be able to traverse through software in both object/action and action/object sequences, depending on which better fits the immediate goal.

Like information workers, the IT pros we studied report that they expect to be able to do all tasks through a GUI. Unlike information workers, they also expect all tasks to be "scriptable". When we moved the lion's share of user interface interactions from blank screen command line interfaces, first to page/panel driven text UIs and then to full GUIs, we didn't consider how to build in command line and query capability seamlessly in the GUI paradigm. Instead, these are in separate windows and work doesn't typically flow seamlessly between the two. This lost opportunity is recoverable: it is time to more insightfully craft the best conventions for converging command line access into the standard GUI paradigm and for enabling task navigation/structure in object-oriented frameworks.

References

1. Soderston, C., Lanzetta, T., Scanlon, J.: Problem-Guided Heuristic User Interface Evaluation Method: Identifying Nuggets of Opportunity & Fool's Gold in User Interface Design. IBM Technical Report 36.0017 (1992)[5]
2. Intel Ease of Use/PC Quality Roundtable white paper: Improving PC Ease-of-Use for IT Professionals (2001), http://www.eouroundtable.com/files/ITFiles/ITWhitePaper.pdf
3. Wichansky, A.: Customers Reveal Top Enterprise Software Usability Issues to Oracle (2008), http://usableapps.oracle.com/design/pages/OUAB.html
4. Haber, E.M., Bailey, J.: Design Guidelines for System Administration Tools Developed through Ethnographic Field Studies. Paper presented at the 1st symposium on Computer Human Interaction for Management of Information Technology (CHIMIT) Cambridge, MA (2007), http://chimit.cs.tufts.edu/ap-papers.html

[5] This work was first cited externally by Theo Mandel in his 1995 book entitled "The GUI-OOUI War: Windows vs. Os/2: The Designer's Guide to Human-Computer Interfaces". The full report is not available externally; is now dated, not sensitive. The results of this work helped to inform what became IBM's Common User Access architecture, which was licensed in early versions of Microsoft Windows, and other environments. See http://en.wikipedia.org/wiki/Common_User_Access