HEWLETT PACKARD

Creating Presentation Slides: A Study of Task-Specific vs. Generic Application Software

Jeff A. Johnson, Bonnie A. Nardi
Software Technology Laboratory
HPL-93-83
September, 1993

application software,
task-specific,
user-studies
ethnography,
slidemaking

A study was conducted to investigate the use of task-generic vs. task-specific application software by people who create and maintain presentation slides. Sixteen people who prepare and maintain presentations as part of their jobs were interviewed to determine: what is involved in that task, what software they use for it, how well the software they use supports the task. The informants varied in how central slide-preparation was to their jobs. The hypothesis driving the study was that: 1) some software applications are intended for use in a wide-variety of tasks, while others are intended to support very specific tasks; 2) software that is specific to the task at hand is preferable, but is often not used because of cost, learning effort, or lack of availability; and 3) people who infrequently perform a task tend to use generic tools, while people who often perform it tend to use task-specific tools. Our findings suggest that the truth is more complex: 1) task-specificity/genericness is not a single dimension: there are different kinds of task-specificity; 2) how much the user knows about the task is at least as important as how frequently s/he performs it in determining what type of software is most suitable; and 3) most informants use several software products of varying degree -- and type -- of task specificity in combination to produce and maintain slides.

Internal Accession Date Only

1 Introduction

It is widely-acknowledged that personal computing has failed to achieve the degree of ubiquity predicted in the late Seventies, at the dawn of the personal computer age. One reason for this failure is that most people are not interested in using computers *per se*, but only in performing tasks, and computers provide poor support for most tasks. Computers force users to master unfamiliar concepts (e.g., files, directories, commands and arguments, control characters, cursors, modes, text strings, selection) that are about computation, rather than about the users' task or problem domain. Furthermore, computers -- even ones with graphical, WYSIWYG, menu-driven user interfaces -- force users to provide the mapping between the objects and operations provided by the computer and the goals, objects, and operations of the users' task-domain. It is, however, well-established that users cannot easily provide this mapping: they cannot easily decompose their tasks into pieces that match the capabilities of today's computers, and they cannot easily combine the computer's capabilities so as to produce a solution to their problem (Hutchins, Hollan, and Norman, 1986; Lewis and Olson, 1987). The failure of personal computers to become as ubiquitous as, say, washing machines, should thus come as no surprise: the predictions of the Seventies were overoptimistic about human facility at performing cognitive mappings between real-world problems and generic computational concepts.

What does it mean for a computer application to provide a high degree of support for a task? In what ways do computer-based applications vary in their degree of support for tasks? Under what circumstances is a high degree of task-support necessary or unnecessary? Because of our interest in fostering development of task-specific applications in a variety of domains (Nardi and Zarmer, 1991; Zarmer *et al*, 1992; Johnson *et al.*, 1993), we conducted a study to gain some insight into how preference for task-specific *vs.* generic software is related to users' job-responsibilities, the precise nature of their task, and other factors. The task domain we studied is the creation and editing of slides for visual presentations. We began with a working hypothesis, but some background is necessary before we state it and describe the study. We first give some examples of software applications and discuss their varying levels of task-specificity. Next, we compare the support that various types of programs provide for slidemaking, and state our initial working hypothesis. Finally, we describe our study and findings, and offer some conclusions.

2 Background: Task-Specific vs. Generic Tools

Most kitchens contain a large variety of tools. Some tools are used in many different tasks, e.g., knives, bowls, spoons, stoves, pots. Others are used for a relatively small set of tasks, e.g., blenders, peelers, tongs, basters, graters, cutting boards, "butter" knives. Still others are used for one task only, e.g., fish scalers, cheese slicers, nutmeg grinders, apple corers, cookie cutters, coffee makers.

For many tasks in which computers are used, a similarly large variety of software tools are in use, ranging from extremely generic to extremely specific. Some people use calculators for preparing income tax returns, some use spreadsheets, and some use income-tax programs designed for a particular year. Some companies do their accounting on calculators, some use spreadsheets, some use general accounting packages, some use accounting packages designed for a particular type of business (e.g., restaurants), and some use custom accounting software developed exclusively for them. For preparing organization charts, some people use painting programs, some use structured drawing programs, some use general tree-graph editors, and some use organization-chart editors. A similar series can be seen for the production of family trees.

The application programs in each of these series provide successively more built-in semantic support for the task. They increase not what results can be produced, but rather the ease with which they are produced. A more formal analysis of what it means for an application or tool to

be task-specific will be deferred to a subsequent paper (Johnson and Miller, in preparation). Our purpose at here is to describe our interview study.

2.1 Creating Presentation Slides

Slide preparation is a task for which a wide variety of computer-based tools are used. People prepare slides using text editors, desktop publishing systems, painting programs, drawing programs, spreadsheets, statistical-analysis programs, business-graphics programs, animation programs, and, recently, presentation-making programs.

Many people use structured drawing programs for making presentation slides. With drawing programs, users place manipulable graphical objects on a canvas. Text and graphics, once placed, can be edited, moved, or copied. However, drawing programs' degree of support for slidemaking is limited. They offer, for example, no notion of a presentation *set* of slides. Users can compensate by putting slides into separate files and grouping them in folders or directories, or by placing all the drawings of a presentation together on one canvas, but such workarounds are inconvenient and inefficient. Additional drawbacks of drawing programs as tools for slidemaking are:

- They provide no support for consistency of format, font, and layout in a presentation. To have the same margins on every slide, users must painstakingly arrange things that way, separately for each slide. If the formatting requirements change, users must change every slide.
- Standard content, such as logos, headers and footers, must be explicitly placed on every slide, and changes require editing every slide.
- They provide little help in changing the structure of a presentation's content. Splitting one slide into two requires much explicit copying, moving, and deleting of graphic objects.

Put succinctly, drawing programs lack the concepts of slides, relationships between slides, and presentations. Most of the other types of software used for preparing slides share with drawing programs a basic lack of support for the process of creating and editing presentation slides. Nonetheless, they are commonly used for that task.

In recent years, software designed specifically for creating presentations has become available (e.g., Charisma, Persuasion, PowerPoint). Since such programs are intended only for slidemaking, they can provide much more support for that task. Slides comprising a presentation are contained in one file. The format and common content of slides are specified once for the presentation, rather than separately for each slide. Typical slide editing actions, like removing or adding a level of detail, are explicitly supported.

Note that given enough time, skill, and talent, any slide or presentation that can be produced with a presentation program can also be produced with a drawing program. Presentation programs facilitate the *process* of creating, editing, and maintaining presentations by providing built-in domain-knowledge. Supporting the process of performing a task may be more important for the usability of a software application than getting the user interface right – at least as the term "user interface" has conventionally been used. We illustrate this with a story from the experience of the first author.

2.2 An Illuminating Anecdote

At one company, employees made presentation-slides using a text editor in conjunction with a textual slide-formatting program. To create a set of slides, users created a text file containing the textual slide content and embedded formatting commands. The text file was then "compiled" using the slide formatter, producing a set of graphics files containing images of the slides. Users had many complaints about this process. The formatting commands were hard to learn; it was

hard to tell how a particular slide would look from its source file; one had to "compile" the entire set to check how a single slide looked.

When interactive painting and drawing programs became available, users switched to them immediately. The new programs were easier to learn and provided much better feedback than did the text editor/formatter combination. However, users soon learned that the new programs, for all their user-friendliness, did not support slidemaking very well. With the new programs, it was hard to obtain consistent formatting, hard to manage sets of slides, and hard to edit slides. After learning to use the new programs, most users soon switched back to the old ones, occasionally using the new programs to enhance slides generated by the slide-formatter.

At the time, this mass retreat to the old tools was difficult to understand. After all, the new applications were more representative of the menu-and-direct-manipulation controlled, WYSIWYG approach to application design than the text editor/formatter combination was. The present analysis makes the reason for the retreat clear: the new programs were not slidemaking programs, they were generic painting and drawing programs. Though they made some aspects of the slidemaking task easier, they made others harder. The aspects of the task that became easier were those in the domain of the "user interface"; they had more to do with controlling the computer than with making slides. The aspects of slidemaking that the new tools made more difficult were the deeper, more task-related aspects. While users could, with effort and talent, make nicer-looking presentations with the new tools, more users could produce acceptable presentations much more quickly with the old tools. Having a "task-friendly" application was more important to users than was having a "user-friendly" interface.

2.3 Working Hypothesis and Motivation for Study

Our experience with interactive computer-based applications, the gist of which is captured in the preceding anecdote, led us to the following set of beliefs:

1. Software applications can be considered to occupy a position along a continuum, from completely generic to completely task-specific.
2. For performing a task, task-specific tools are always preferable, i.e., it is always better to have a tool designed specifically for the task one is performing. E.g., for creating or editing an electronic schematic drawing, a schematic editor would be preferable to a drawing editor.
3. Good support for a task is more important for overall usability and productivity than is a good "user interface" in the traditional, limited sense of the term.
4. The more task-specific a software application, the smaller its potential market, requiring the developer to either charge a higher price or be satisfied with less revenue.
5. People resort to using generic applications when more task-specific ones: are unavailable, cost too much, or require too much (incremental) learning effort. In today's software market, and given the state of application development technology (Myers, 1989; Zarmer and Johnson, 1990), these conditions usually hold, so most of the software most people use for most tasks is generic. That is, most users "get by" with generic tools because their level of need does not justify the cost of obtaining and learning to use task-specific ones.
6. People who perform a task frequently -- e.g., it is their main job -- prefer task-specific tools. Their level of use justifies the overhead of acquiring and learning to use the tool.

Based on these beliefs, we (with other colleagues) set about developing an Application Construction Environment (ACE) designed to facilitate the development of task-specific software applications (Nardi and Zarmer, 1991; Zarmer *et al*, 1992; Johnson *et al.*, 1993). One of our goals was to change the economics of software development: to make application development easy enough and cheap enough that people could develop applications whose target

task-domain was very narrow and whose intended use-period was a matter of months or weeks, much as people do with spreadsheets (Nardi and Miller, 1990, 1991). A related goal of ACE was to move the center of the development process much closer to the users, who best understand the task, than it is in traditional software development.

As we developed ACE, we were aware that the beliefs upon which it was based should be regarded as a composite hypothesis to be empirically tested. Over time, we began to have reason to suspect that our initial working hypothesis wasn't quite right. For example, someone we knew who had worked as a graphic artist producing presentation slides for others indicated that important parts of it were false. She claimed that experienced professional slidemakers prefer generic drawing and painting software for creating slides because it doesn't restrict them from doing what they want, while dedicated slidemaking programs often impose over-simplified views of the task and restrict the results that can be produced. Based on this counter-claim and on our own further analysis of the nature of task-specificity, we decided to conduct an empirical study as a first step in evaluating and correcting our working hypothesis. We chose the domain of slidemaking because of the large variety of software tools used for that task and because of the relative accessibility of users as informants.

3 Method

3.1 Informants

The informants[1] were sixteen people whose jobs involved creating, editing, and maintaining slide presentations. All were college educated with several years experience making slides. They worked for a variety of companies, ranging from single-person independent consultantships to large multinational corporations. Six of the sixteen informants worked in research or marketing, and made slides for their own use in presentations, with slidemaking being only one of many of their job-responsibilities. The other ten informants had as a significant (for some, dominant) part of their job the creation of presentation slides for others, in a variety of business areas: legal, advertising, research, and general business.

3.2 Procedure

A set of questions were developed that covered the issues of interest in this study (see Appendix B). Most of the interviews were conducted at the informant's workplace, often with a computer slidemaking system ready-at-hand.

The interviewer began each interview by explaining to the informant that the purpose of the study was to learn what is involved in making slide presentations, what sorts of software people use for the task, and what people's reasons are for using or not using various software tools. The interviewer then conducted a semi-structured interview, beginning by asking the informant to describe the entire slidemaking process, from start to finish. The interviewer allowed the conversation to flow more-or-less naturally rather than strictly following the list of questions, but made sure that answers to each of the predetermined questions were captured on tape. The interviewer did not explain the distinction between task-specific vs. generic software, or our initial working hypothesis.

Interviews were audio-taped, then transcribed onto computer text files. Over 160 pages of transcripts resulted from the audiotaped interviews.

---

[1] In an ethnographic study, study participants are called *informants* as their role is to actively inform the investigator. This is in contrast to the use of the term *subjects* for experimental studies, where participants are subjected to experimental conditions and observed.

3.3 Data Analysis

The authors read transcripts of each interview, in some cases referring to the audiotape to clarify transcription problems or informant intent. A summary was made of each interview that included: the informant's job role and involvement in slidemaking, the context in which slides were being produced, a summary of the slide-production process as described by the informant, the software the informant uses or has used for slidemaking, the informant's reasons for using it, software features that the informant considered useful or a hindrance in slidemaking, and a few informant remarks (if any) that seemed especially germane to the study.

4 Results and Discussion

We found that our informants were quite happy to talk about their slidemaking software. Several warned when making the appointment that their busy schedule could accommodate only a brief interview, but in the interview seemed willing to talk for as long as the interviewer would listen. People apparently have strong opinions, both positive and negative, about the software they use. See Appendix A for a summary of our informants' role in slidemaking and the type of software they use.

The main finding of the study was that our original hypothesis was right in some respects and wrong in others, but the truth is more complex than either we or our graphic artist friend predicted. As we predicted, people who make slides only infrequently typically use whatever general-purpose software they use for other work, e.g., word-processing. It is uncommon for them to use dedicated slidemaking software. For them, it just doesn't pay. However, contrary to our hypothesis (but in agreement with the claims of our graphic artist friend), many graphic artists who produce presentations for a living use generic software such as drawing, painting, and word-processing programs extensively rather than sticking to task-specific slidemaking programs. However, this does not mean that dedicated slidemakers don't use task-specific software. It depends upon what one means by task-specific. It also depends on several other factors. In the following sub-sections, we revise our distinction between task-specific and generic applications and describe the factors affecting software choice that emerged from our interviews.

4.1 Taxonomy of Task-Specificity

Our analysis of task-specificity, it turns out, was too simple. We had originally conceived of task-specificity in software applications as being simply a matter of degree, with completely generic applications at one extreme and totally task-specific ones at the other. It is now clear that there are at least two distinct kinds of task-specificity:

- Subtask-specificity: specialization for a specific subtask that occurs in a variety of higher-level tasks.
- End-to-end task-specificity: specialization a particular top-level task, i.e., supporting all subtasks in the process, from start to finish.

An example of this distinction in kitchen tools is that between a dough-kneader and a breadmaking machine. A dough-kneader is very task-specific in the sense of what sort of material it works on, but is used in service of a variety of high-level goals (e.g., making bread, making coffeecake, making pizza, making rolls). The knowledge for the high-level task must be provided by the user. A bread-making machine is very task-specific in the sense of automating a particular high-level task (i.e., ingredients in; bread out), but includes a variety of subtasks (i.e., mixing, kneading, rising, baking). It embodies a great deal of breadmaking knowledge, in fact substituting for breadmaking knowledge and skill on the part of the user.

In a business organization, an organization-chart editor specifically designed for that organization would be an example of a tool that was highly sub-task specific: it would be very

useful for creating and updating charts of that organization, independent of why one wanted such a chart. On the other hand, the organization might have a program that automated much of the production of quarterly status reports, including personnel information (illustrated using an organization chart) as well as other information. Such a program would be end-to-end task specific: data in, report out.

The ultimate end-to-end task-specific slidemaking tool would be one that took brief verbal descriptions of content and produced well-designed slides in company-standard format. In many companies, graphic artists act as end-to-end task-specific slidemaking tools for other employees, using subtask-specific tools to do their work.

Skilled graphic artists often use slidemaking programs, ironically, not to make slides, but rather to contain and organize them. The programs that we had regarded as task-specific tools that supported, end to end, the entire slidemaking process, were in many cases being used as tools to support a specific subtask of slidemaking, namely, containing and organizing slides for presentations. Many other slidemaking capabilities of these programs were often ignored because they weren't good enough (see below). Said one graphic artist informant: "SLIDEMAKER is like one of these ... software packages that try to incorporate everything, yet no particular area is very strong."

Not only are some software programs used differently than they were designed to be used, some software is designed to be specializable for tasks, and as such does not have a fixed degree of task-specificity. For example, one of our graphic-artist informants said that at his company, the preferred tool for slidemaking is a nominally generic document editor, which through the use of stylesheets and template files is specialized to provide good support for creating and editing presentations, even very high quality ones.

4.2 Factors Affecting Choice of Software

Turnaround Time and Presentation Quality

One factor that influences the choice of software is the desired turnaround time for producing a presentation. Presentation slides are often produced on very short schedules, with turnaround time being more important than illustration quality. For example, one informant said: "...usually speed is an issue here rather than quality... it's always like down to the last minute." Another said: "...they always wanted everything yesterday. They will come to me with very little time to turn around slides." Obviously, however, not all presentations are last-minute; some are anticipated and prepared long in advance.

Turnaround time trades off with a second important factor: the desired quality of a presentation. Our data make it clear that slidemaking tasks vary a great deal in requirements. Some presentations are for company employees and some are for external customers. Some presentations are relatively unimportant, while others have millions of dollars riding on the impression they make.

Many slidemaking organizations use different software, processes, and even personnel for producing "ordinary" slides and presentations, which account for the vast majority, and "fancy" or "very important" ones, of which there are relatively few. To produce relatively simple presentations quickly, professional slidemakers sometimes use dedicated slidemaking programs (though factors such as familiarity and availability limit this tendency). For fancier presentations, they usually use generic drawing, painting, desktop publishing, or animation software. Some firms employ somewhat less-skilled graphics personnel to produce the more straightforward slide presentations, and more highly-trained graphic designers and artists to produce fancy graphics and presentations. For example, one firm has one specialist who creates high-quality color presentations and fancy graphics using generic illustration software, and

"everyone else just does straight charts and graphs" and "word-slides," using a presentation program.

Drawing Support

Highly-trained graphic artists use generic illustration software because the drawing capabilities of slidemaking programs are insufficient and limiting from their point of view. Some illustrative quotations[2]:

- "SLIDEMAKER's not the best drawing tool."
- "...the graphic tools in SLIDEMAKER are kind of low-end, not very powerful." "SLIDEMAKER's drawing tools are too weak."
- "[With DRAWTOOL], you have more control."
- "You can do it in SLIDEMAKER, but depending on the art, sometimes its faster ... to do it in DRAWTOOL and paste it in."
- "I think the main point about why we use DRAWTOOL is because, yeah ... SLIDEMAKER would be better for a lot of word slides... But nobody's willing to simplify their graphs that much. You know? It's like they would have to ... work at such a simple level to make a presentation, that nobody, they can't, the [clients] can't cut down on the complexity of their slides, to be able to fit in with the limitations of a program like that."
- "SLIDEMAKER's drawing capability sucks."

Because of this, professional graphic artists who create illustrations for slides tend to use generic drawing and painting tools, which give them the freedom to use their skills and to produce the illustrations they want. Some illustration programs provide image-enhancing features such as anti-aliasing, 3-dimensional effects, and highlighting, which are absent in dedicated slidemaking programs.

Skill Level

A third important factor affecting the choice of tools, therefore, is the skill-level of the slidemakers. Our initial working hypothesis distinguished only between users for whom slidemaking is central to their job and those for whom it is only peripheral, but it is now clear that there are different kinds of "dedicated" slidemakers: those having a lot of talent, skill, and knowledge in the task-domain (i.e., presentation style, graphic art) and those having less domain talent, skill, and knowledge. If, for simplicity, we divide the dedicated-slidemaker skill-level continuum into high vs. low-skill categories, we have three categories instead of our original two: high-skilled dedicated slidemakers, low-skilled dedicated slidemakers, and casual or infrequent slidemakers. Dedicated slidemakers having more skills, talent, and job-autonomy tend to use tools that provide users with more freedom to exercise their own domain knowledge and creativity. Heavy slidemakers who are less knowledgeable about graphic arts and wish to remain so, or who are in jobs allowing less autonomy and creativity, use tools that provide the bulk of the task-knowledge built-in. Thus, our original hypothesis, though incorrect for highly-skilled dedicated slidemakers as our graphic artist friend had argued, was essentially correct for less-skilled ones.

---

[2] Note: all slidemaking programs are referred to herein as SLIDEMAKER, generic drawing programs as DRAWTOOL, etc.

Support for Teamwork

A fourth factor is whether the degree of teamwork supported or allowed by the software matches the work-practice of the users' organization (see Nardi, 1993 for a discussion of collaborative application-development practices). Most dedicated slidemaking programs are designed to support an individual who produces slides alone. However, in many slidemaking organizations, people work in teams to produce and maintain slides. Existing dedicated slidemaking programs make it difficult to do this. According to one informant:

> "I looked at SLIDEMAKER, because everybody was saying that SLIDEMAKER was great. And I think a bunch of the secretaries ... used it as well. And I think the programs that are that specific are very well designed for a person who is going to sit down and think up a presentation and create the presentation right there. But the way we work is that, you know, there are dozens of people out there thinking up things, and we integrate presentations for all of them. And so for us to be able to distribute that work amongst enough people to get it done, we need to break it down into smaller units. ... For each job here, if we used SLIDEMAKER, each job ... would have its own ... file with all of its slides in it. But slides get used from one job to another... And so I think because of that it wouldn't work. The outlining, you know is wonderful. But it's really designed for a different type of work atmosphere. It's is designed for the guy who's sitting down and going to do his own presentation."

Though more generic tools don't provide real support for team-production, they at least don't interfere with it in the sense that they impose little structure on the process at all.

Earlier, we stated our finding that professional graphic artist usually use slidemaking software to contain and organize slides rather than to create them. Not surprisingly therefore, interoperability is very important to them: professional slidemakers don't think much of dedicated slidemaking tools that cannot easily accept text and graphics from a variety of sources.

5 Conclusions

We went into the study thinking that software applications vary along a continuum of task-specificity, and found that it isn't that simple:

- There are different kinds of task specificity:
  a) End-to-end. An end-to-end slidemaking tool would be one that supported the entire process of making and maintaining presentations.
  b) Subtask. Some tools are best for working with specific kinds of materials, or producing specific kinds of results, e.g., data-charting programs. Examples of subtasks in the slidemaking domain: some programs (e.g., data-charting programs), are good for producing certain kinds of results; some programs are good for holding and organizing slides, but not so good for producing the slides themselves.
- Some tools are specializable, i.e., nominally generic, but transformable via the addition of encoded task-knowledge (e.g., spreadsheet formulae) to make them task-specific, e.g., LaTeX and Framemaker can be specialized for slidemaking.

In our original hypothesis, we had claimed that the primary hindrance to the use of task-specific software was acquisition and learning costs, i.e., how much does it cost to get the software, and how much of an incremental learning "hump" does using the tool require? Does the need justify these costs? Companies don't like to spend money. Users really avoid learning new things. Thus, tools they already have and know have a strong advantage.

But cost isn't the whole story. We found that several additional criteria are important in deciding what tools will be used to make presentations:

- Power: Can the required slides be produced with the tool?
- Support for Teamwork: Does the tool support people working together on a presentation, if that is how presentations are produced at the worksite in question? Currently, end-to-end slidemaking tools assume a single user working alone, but many presentations are created by teams, with different people contributing different parts.
- Interoperability: Can a particular tool easily take input from other sub-task-specific tools, e.g., can a drawing made with one program be put into a slide or presentation made using another? Programs that provide good interoperability are preferred.
- Speed vs. Quality Tradeoff: How important is it that presentations be fancy vs. done quickly? It is common for there to be two separate production processes, using different tools: one for most presentations using good end-to-end task-specific tools and users with lower domain expertise, and one for the few fancy or unusual presentations, using generic or sub-task-specific tools, operated by task-domain experts.
- Skill Level of Users: How much domain-knowledge about slidemaking do users have? How much do they want to learn about the task? The less they know (or want to know), the more domain knowledge must be provided by the tools. E.g., breadmaking machines are not for experts; they are for people who make a lot of bread but don't have the skill or time to do so.
- Company policy: company determines tools; workers use what they're given.

We therefore revise our original working hypothesis as follows:

1. Software applications can be considered to occupy a position in a two-dimensional space, with one axis being degree of sub-task specificity and the other being degree of end-to-end task specificity. Applications that are low on both axes are task-generic.
2. For performing a task, end-to-end task-specific tools are always preferable, i.e., it is always better to have a tool designed specifically for the task one is performing: goals in, results out.
3. Good support for a task is more important for overall usability and productivity than is a good "user interface" in the traditional, limited sense of the term.
4. The more task-specific a software application is, the smaller its potential market, requiring the developer to either charge a higher price or be satisfied with less revenue.

4'. The more end-to-end task-specific a software application is, the less flexible it is likely to be in terms of what sorts of results it can produce and in terms of what sort of work-practice it supports. If an end-to-end task-specific application conflicts in these two regards with customer needs, it will not be used.

5. People resort to using generic applications when more task-specific ones: are unavailable, cost too much, or require too much (incremental) learning effort. In today's software market, and given the state of application development technology, these conditions usually hold, so most of the software most people use for most tasks is generic. That is, most users "get by" with generic tools because their level of need does not justify the cost of obtaining and learning to use task-specific ones.
6. People who perform a task frequently -- e.g., it is their main job -- and who possess a high degree of skill and wish to exercise that skill prefer using collections of interoperable subtask-specific tools. Their level of use justifies the overhead of acquiring and learning to use the tool. People who perform a task frequently but who either possess little domain skill or do not wish to exercise their skill use end-to-end task specific tools.

7. Some tools are nominally generic, but specializable (e.g., a power beater has different attachments, which make it into a batter mixer, a blender, a dough kneader, etc.)

Though the present study focused on a particular task-domain, our interest was in exploring the costs and benefits -- and users perceptions of them as manifested in their choice of tools -- of task-specific vs. task-generic application software. We feel that our findings do speak to the more general issues raised in our working hypothesis; if nothing else they forced us to modify it based upon our findings. Of course, the generality of the findings reported here should be validiated through comparable studies in other task-domains. Furthermore, within the slidemaking task-domain, more-structured studies would help to shed additional light on issues such as the relationship between task-specificity and user productivity, the tradeoff between task-specificity and learning cost, and the distinction between types of task-specificity. If nothing else, we are hopeful that this paper will make more software developers aware of task-specificity as an issue to be considered in application sofware design.

6 References

Johnson, J.A., Nardi, B.A., Zarmer, C.L., and Miller, J.R. (1993) "ACE: Building Interactive Graphical Applications," *Communications of the ACM*, April.

Myers, B.A. (1989) "User Interface Tools: Introduction and Survey," *IEEE Software*, 6(1).

Nardi, B.A. (1993) *A Small Matter of Programming: Perspectives on End User Computing*, Cambridge, Mass: MIT Press.

Nardi, B.A. and Miller, J.R. (1990) "The Spreadsheet Interface: A Basis for End-User Programming," In D. Diaper, D. Gilmore, G. Cockton, and B. Shackel (eds.), *Proceedings of Interact'90*, New York: Elesevier Science Publishing Company.

Nardi, B.A. and Miller, J.R. (1991) "Twinkling Lights and Nested Loops: Distributed Problem-Solving and Spreadsheet Development," *International Journal of Man-Machine Studies*, 34, pages 161-184.

Nardi, B.A. and Zarmer, C.L. (1991) "Visual Formalisms as Application Frameworks." *Proceedings of HICSS-24*, January.

Zarmer, C.L. and Johnson, J.A. (1990) "User-Interface Tools: Past, Present, and Future Trends," *Hewlett-Packard Laboratories Technical Report HPL-90-20.*

Zarmer, C.L., Nardi, B.A., Johnson, J.A., and Miller, J.R. (1992) "ACE: Zen and the Art of Application Building," *Proceedings of HICSS-25*, January.

7 Acknowledgments

The authors thank Michelle Gantt, the anthropology student intern and graphic artist who served as the interviewer for the study and also -- by questioning our initial hypothesis about who uses task-specific vs. generic software -- as the primary catalyst for our conducting it.

Appendix A: Who uses what?

- Informant is a manager in an advertising firm. Heads department that creates slides for employees to present to external customers. Informant uses Macromind Director™ to produce complex or important presentations. Department includes 80 clerical employees who use Persuasion™ to create the majority of presentations.
- Informant works for management consulting firm. Heads department that creates slides and graphs for presentations by employees to external customers. Informant uses Persuasion™ to organize presentations containing graphics created by the seven graphic artists in the department. Prepares some presentations himself.
- Informant worked for a slidemaking service as a production artist and production manager. Used Persuasion™ for organizing "normal" presentations, and Macromind Director™ where special effects were required and budgets were higher. Created graphics using MacDraw™. Used Photoshop™ for large-font text because of anti-aliasing.
- Informant is graphics manager at a management consulting firm. Department includes one specialist who creates high-quality presentations and fancy graphics, and seventeen others who "just do straight charts and graphs." All use MacDraw Pro™ and DeltaGraph™.
- Informant is artist in graphics department of an electronics company. Department includes four artists who produce graphics for internal and external presentations by employees and for company publications. Informant uses Frame Maker™ with custom template files for most slide content. Other content is produced using an assortment of drawing and image editing programs.
- Informant was one of two graphic artists in a large company. Creating presentations and trade-show exhibits was half of the job, but most of this was creating presentations for legal or business purposes. Uses Persuasion™ for most slides, but creates fancy graphics in PixelPaint™ and imports into Persuasion™. Uses Macromind Director™ for very fancy presentations.
- Informant is a freelance graphic artist. Prepares presentations and publications for corporate clients, using their facilities. About one-fourth of time is spent on presentations. Software used depends on what client company has and on graphic content, but prefers MacDraw™.
- Informant was product marketing specialist at a large company, then became marketing director for a startup. Presentations were for informant's own use. At the large company, informant designed presentations and gave to staff to make; at the startup, informant did everything himself, using Persuasion™ because they had it and it worked. Created fancy graphics using Canvas™ and data-driven graphics using Excel™.
- Informant is VP of Networking Services at a large company. Manages 500 engineers nationwide. Creates presentations for his own use; spends about 5% of time preparing presentations. Uses Microsoft Word™ for most slides, occasionally importing graphics from MacDraw™ or Excel™. Doesn't like existing slidemaking programs.
- Informant is researcher in a corporate research lab. Creates presentations for own use, as very small part of job. Formerly used LaTeX; now uses Frame Maker™. Slides are mostly words.
- Informant is a project manager in a corporate research lab. Gives presentations sporadically but often, both internal and external. Makes presentations for own use. Informant formerly used document editors, painting, and drawing programs, but now uses slide-presentation software. Recently switched from PowerPoint™ to Persuasion™ for compatibility with corporate graphic arts department.

- Informant is researcher in a corporate research lab. Makes slides mainly for presentation of research results. Spends very little (~5%) of time preparing presentations. Uses EMACS and LaTeX only. Rarely has graphics content; if needed, photocopies it and physically pastes it in.
- Informants (husband and wife team) are freelance graphic artists who work in-house at management consulting companies, producing presentations and reports. Tools used depend on company. At one company, informants use PowerPoint™, importing graphics from MacDraw™, Illustrator™, and CricketGraph™. At another company they use PageMaker™, importing from different Illustrator™ and Freehand™.
- Informant is consultant-in-training at a market consulting firm. Makes presentations to client companies to inform them about markets. Usually sketches slides and gives to graphics dept. Sometimes makes data-driven charts for himself, using Paradox™ and Lotus 123™. Occasionally makes his own graphic slides using MacDraw™ or Freelance™.
- Informant is a desktop-publishing specialist for a large international law firm. Produces presentations, publications, and exhibits. About half-time is spent on presentations. Uses mainly Ventura Publisher™ as container, importing from Corell Draw™, Word Perfect™, Harvard Graphics™, Lotus 123™, and a scanner.
- Informant is a graphic artist and word-processor in an international management consulting firm. Department has fifteen people: nine on day shift, three on night shift. Informant is on night shift. Eighty percent of job is preparing slides and graphs (day shift does more word processing). Uses mainly Word™ for text and as a container, importing graphics from MacDraw™ and Excel™. Occasionally uses Persuasion™.

Appendix B: Questions Addressed in Interviews

1. What is your role in producing presentation slides?
   - 1.1 Do you produce slides yourself or do you supervise others who do it?
     - 1.1.1 What sort of training or experience is required to do the job you do?
     - 1.1.2 [If supervises others] What is the skill level of your employees?
   - 1.2 How much of your total job involves producing presentation slides?
   - 1.3 For whom do you produce these presentation slides?
     - 1.3.1 Who is the customer (i.e., who approves the slides)?
     - 1.3.2 Who is the audience for the presentations?
   - 1.4 What sort of quality level is required for the slides?
     - 1.4.1 How important are elaborate special effects (e.g., animation, dissolve)?
     - 1.4.2 Who decides on appearance and quality, you or the customer?
     - 1.4.3 Are there different kinds of presentations with different quality requirements?
   - 1.5 Do you (your department) follow slide formatting standards?
     - 1.5.1 How do you assure that slides adhere to those standards?
     - 1.5.2 Does your slide-making software help with standardization of presentations?
2. What software do you use to create presentation slides?
   - 2.1 Who decides what software you use for this?
   - 2.2 Do you use one program or a collection of them?
     - 2.2.1 [If many] What are the different programs used for?
   - 2.3 Do you use general-purpose drawing software or slide-making software?
     - 2.3.1 Why?
   - 2.4 What do you like about each of the programs you use?
   - 2.5 What do you dislike about each one; what would you like to see changed?
     - 2.5.1 Describe some of the things you do to "work around" limitations of the software.
   - 2.6 How easy is the software for new users to learn?
     - 2.6.1 How do they learn the software (classes, manuals, using, asking)?
     - 2.6.2 How did you learn it?
   - 2.7 What other software have you used, tried, or considered for making slides, either here on in previous jobs?
     - 2.7.1 Why don't you use it now?
3. What is involved in making slides?
   - 3.1 Describe the complete process of producing a presentation, from when you take the assignment to when to deliver it to the customer.
     - 3.1.1 How much revision is usually required before a presentation is considered done?
   - 3.2 Do you usually create new presentation slides?
     - 3.2.1 What is hard and what is easy about creating new material, i.e., what goes quickly and what takes time and work?
   - 3.3 Do you re-use old slides in new presentations?

3.3.1 What is hard and what is easy about reusing old material, i.e., what goes quickly and what takes time and work?

3.4 How do you (your department) organize and keep track of slides and presentations?

3.4.1 Is each slide a separate file, or are all the slides in a presentation together in one file?

3.4.2 Do you use directories (folders) and subdirectories (subfolders) to organize your material?

3.4.3 How do you name your slide (or presentation) files?

3.4.4 Do you ever fail to find a slide you know you have?

3.4.5 How does your software hinder you in reusing material?

3.4.6 How easily can you include a single slide in several different presentations?

3.5 What kinds of revisions are often required in the process of preparing a presentation?

3.5.1 Which are easy and which are hard?

3.5.2 Are the same revisions easy and hard for each of the slidemaking programs you use?

3.5.3 Some specific cases we'd like to know about:

3.5.3.1 A slide used in multiple presentations is changed.

3.5.3.2 A company logo or standard border must be added to every slide in a presentation.

3.5.3.3 The order of slides in a presentation must be changed.

3.5.3.4 The round bullets throughout a presentation must be changed to square bullets.

3.5.3.5 The font used throughout a presentation must be changed.

3.5.3.6 Each of the points on a particular slide must be expanded into a separate slide.

Appendix C: Trademarks

MacroMind Director is a trademark of MacroMedia, Inc.

Persuasion and PageMaker are trademarks of Aldus Corporation.

MacDraw and MacDraw Pro are trademarks of Claris, Inc.

DeltaGraph is a trademark of DeltaPoint, Inc.

Frame Maker is a trademark of Frame Technology Corp.

Excel, PowerPoint, and Word are trademarks of Microsoft Corp.

Canvas is a trademark of Deneva, Inc.

Illustrator and Photoshop are trademarks of Adobe, Inc.

CricketGraph is a trademark of Computer Associates.

123 is a trademark of Lotus Development Corp.

Harvard Graphics is a trademark of Software Publishers Corp.

Ventura Publisher is a trademark of Xerox Corporation.

Corell Draw is a trademark of Corell, Inc.

Word Perfect is a trademark of Word Perfect, Inc.