

Practitioner Tools and Workstyles for User-Interface Design

Pedro Campos and Nuno Jardim Nunes, *University of Madeira*

A survey asked 370 practitioners about how they worked and used tools. The findings offer insight into how developers can build human-centered tools for practitioners interested in user-interface design.

Both software engineers and designers often complain that their tools are unsupportive and unusable.¹ Although there's evidence that technology reasonably improves product quality and consistency,^{2,3} the relationship between practitioners and their tools has always been love-hate. The *ACM Queue's* What's on Your Hard Drive? column, for example, testifies to the fact that tools influence practitioners' work to the point where they really do hate or love particular tools.⁴ Some

practitioners even love and hate the same tool. And, as *IEEE Software's* Tools of the Trade column recently reported, practitioners have problems finding the most appropriate tools for a given task.⁵

In our research, we observed that we could ease the user interface (UI) aspects of software development by giving both designers and developers tools that transparently adapt to particular styles of work (or *workstyles*). We've found—through empirical observation in small software development settings,⁶ informal⁷ and formal usability studies, and a survey—that both designers and software engineers often use several different workstyles and move between them frequently.⁸ We call this movement *workstyle transition*.

Many studies have analyzed general software development practices, but qualitative studies of UI-related work practices in software develop-

ment are relatively rare (see the sidebar, "Related Work in Usability Research").

Here, we add to this small but growing body of knowledge by arguing for the importance of supporting workstyle transitions in UI practices. To this end, we surveyed 370 practitioners, who answered questions about their current tool usage and workstyle transitions. On the basis of the survey results and previous work, we've built two design tools to support key transitions.

A survey of UI design tools and workstyles

We focused our survey on interaction designers, but other types of software development professionals are also involved in UI activities. We therefore gathered responses from programmers, system analysts, and project managers as well. We planned, designed, and

Related Work in Usability Research

Don Dillman (www.sesrc.wsu.edu/dillman) is one of survey-based research's best-known researchers. In 1978, he developed the Total Design Method, a general implementation method that's known to achieve high response rates. Dillman has since expanded the design and renamed it *The Tailored Design Method*.¹ The method is highly useful, and we attempted to follow its explicit recommendations, including designing a respondent-friendly questionnaire, shortening the questionnaire (fewer questions reduces respondent burden), and creating interesting questions. However, because surveys are one-sided,² we also ran formal and informal usability studies on workstyle transitions (which are beyond the main article's scope) and conducted follow-up surveys with some respondents to gather ideas and corroborate some conclusions.

Related studies

Although many studies have analyzed and tried to better support general software development practices,^{3–6} literature that qualitatively studies UI-related work practices in software development is relatively rare. Ahmed Seffah and Rex Kline showed a gap between how tools represent and manipulate programs and the software developers' actual experiences.⁶ Their work quantitatively measured the developers' experiences using heuristic and psychometric evaluation. However, they didn't specifically study UI-related issues, which is our work's main concern. Our work also relates to the problem of integrating usability and software engineering.

James Wu and Nicholas Graham described a novel model for recording people's working style while using an interactive system.⁵ They also described a study aimed specifically at collaborative software development (which actually inspired their model's development). Workstyle modeling complements task modeling by providing information on how people communicate and coordinate their activities and by showing the artifact style they produce. Wu and Graham developed their workstyle model as part of the Software Design Board project, which aims to provide better software design tools. They validated the model by evaluating existing design tools and were also motivated to design a new software design tool.

Workstyle analysis

The workstyle model for collaborative software design is simple to apply and clearly shows where a tool can fail to match the intended work context. However, it's insufficient for

capturing UI-specific activities. In previous research, we described a structured model specifically aimed at capturing UI-related workstyles and workstyle transitions.⁷ We described how designers engage and transition in terms of

- perspective (from code to design or from requirements to final solution),
- detail (from highly detailed descriptions to few details and back),
- formality (meaningless sketches or semantically sound models),
- modifiability (some designers prefer tools for frequent tasks such as repositioning and renaming),
- traceability,
- functionality,
- asynchrony, and
- distribution.

This was the starting point for our investigation of interaction design aspects of software development tools and workstyle transitions. Our survey of the tools and workstyles of practitioners performing UI-related activities is informed by both our previous research findings^{7,8} and Dillman's book.¹

References

1. D.A. Dillman, *Mail and Internet Surveys: The Tailored Design Method*, John Wiley & Sons, 1999.
2. X. Ferré et al., "Usability Basics for Software Developers," *IEEE Software*, vol. 18, no. 1, 2001, pp. 22–29.
3. S. Jarzabek and R. Huang, "The Case for User-Centered CASE Tools," *Comm. ACM*, vol. 41, no. 8, 2004, pp. 93–99.
4. J. Ilvari, "Why Are CASE Tools Not Used?" *Comm. ACM*, vol. 39, no. 10, 1996, pp. 94–103.
5. J. Wu and T.C.N. Graham, "The Software Design Board: A Tool Supporting Work-Style Transitions in Collaborative Software Design," *Proc. Int'l Conf. Eng. Human-Computer Interaction/Int'l Workshop Design, Specification and Verification of Interactive Systems (EHCI/DSV-IS 04)*, LNCS 3425, Springer, 2004, pp. 363–382.
6. A. Seffah, J. Gulliksen, and M. Desmarais, eds., *Human-Centered Software Engineering*, Springer, 2005.
7. P. Campos and N.J. Nunes, *Galactic Dimensions: A Unifying Workstyle Model for User-Centered Design*, LNCS 3585, Springer, 2005, pp. 158–169.
8. P. Campos and N.J. Nunes, "CanonSketch: A User-Centered Tool for Canonical Abstract Prototyping," *Proc. Int'l Conf. Eng. Human-Computer Interaction/Int'l Workshop Design, Specification and Verification of Interactive Systems (EHCI/DSV-IS 04)*, LNCS 3425, Springer, 2004, pp. 146–163.

executed our survey on the basis of Don Dillman's book *Mail and Internet Surveys*.⁹ We also conducted follow-up interviews to corroborate some conclusions.

We disseminated our questionnaire to two major interaction-design mailing lists: CHI-WEB

and the IxDA (Interaction Design Association). We also sent it to many industry contacts. In less than two weeks, we collected more than 225 valid responses. We then surveyed two software development mailing lists: IS-World and Flex-Coders. In this second phase, we gathered

Table 1**Respondent demographics**

What is your primary role in your organization?	Percentage of responses	Number of responses
Usability expert, interaction designer	34.0	126
Systems analyst and designer	24.3	90
Programmer	21.6	80
CIO, project manager	10.5	39
Other	9.5	35
Total	100.0	370
How many years of professional experience do you have in that role?		
< 2 years	20.3	75
3–6 years	40.8	151
7–10 years	22.7	84
> 10 years	16.2	60
Total	100.0	370
How many information systems employees does your organization have?		
Fewer than 5	15.2	55
5–14	16.3	59
15–49	15.1	55
50–100	11.0	40
More than 100	42.2	154
Total	100.0	363
How would you classify your organization's software development process?		
Just do it	14.0	61
Waterfall	16.7	54
Spiral	6.3	38
Evolutionary development (prototyping)	25.3	84
Exploratory development	3.6	17
Formal methods specifications	14.5	43
Composition through reusable components	6.3	18
Other	13.1	48
Total	100.0	363

another 145 responses (more than 90 percent from system analysts and programmers), achieving a total of 370 responses. Results were similar for interaction designers and system analysts and programmers, which wasn't surprising because they all were evaluating UI-related activities.

In the questionnaire's initial part, we asked participants about their organizational role, professional experience, organization size, and development process. Table 1 shows the percentage and total number of responses to "What is your primary role in your organization?" Respondents included 126 in-

teraction designers and usability specialists, and 170 programmers and systems analyst and designers.

Regarding professional experience, 40.8 percent of the respondents declared three to six years of experience in their current roles. Most respondents (42.2 percent) worked at large organizations with more than 100 information systems employees. Evolutionary prototyping was the most common development process. This result isn't surprising; it clearly reinforces the idea that both interaction design and modern software development are inherently iterative and incremental processes.

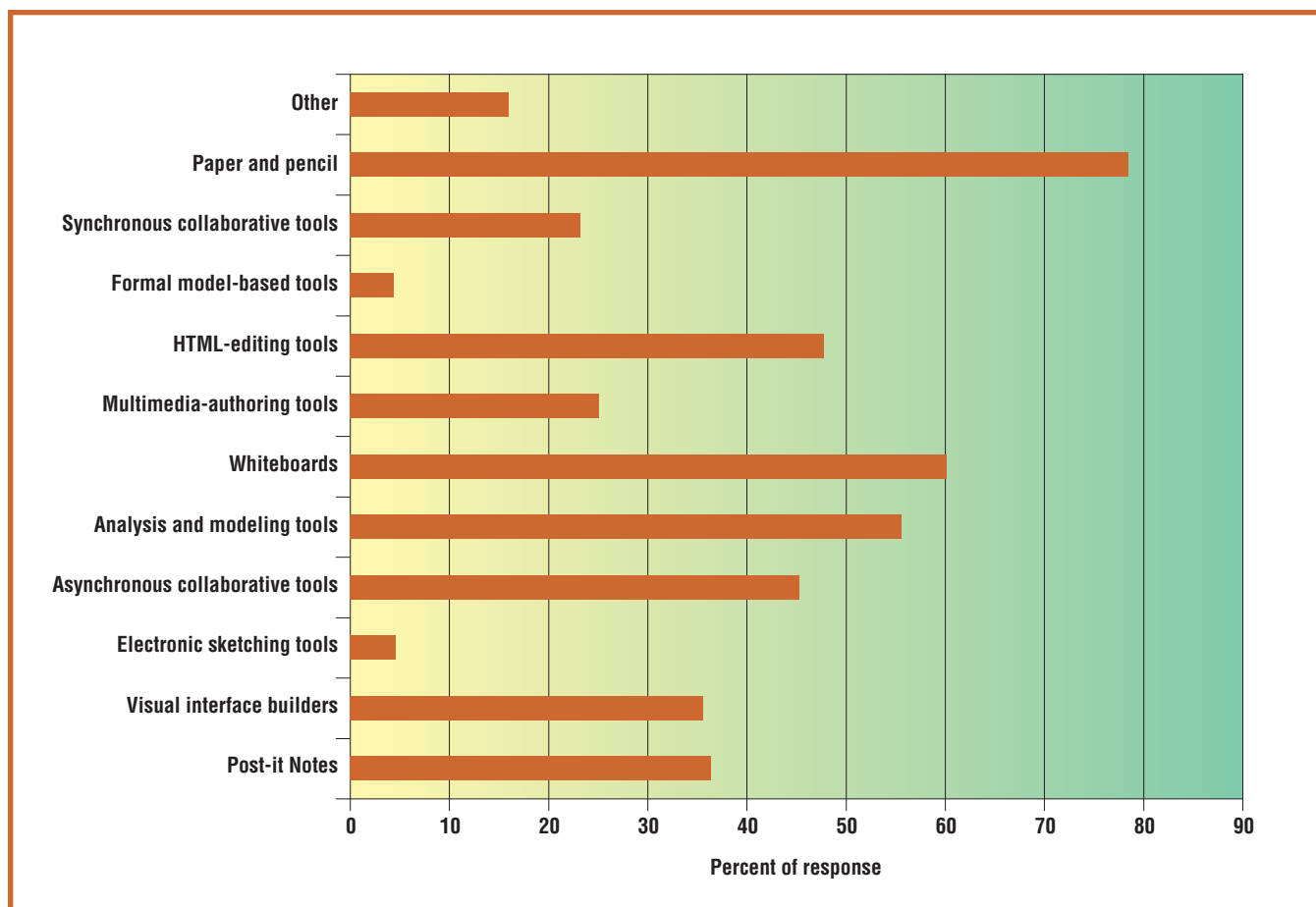


Figure 1. The tools practitioners use. Although model-based tools are popular in general software development, study participants didn't regard them as critical to user-interface-related activities.

Tools and tool use patterns

To assess tool use, we offered a list of tools and asked interaction designers and software developers, "Which tool(s) do you currently use to perform user interface design? (Check all that apply to any user-interface-related activity.)" Figure 1 summarizes our results.

Perhaps not surprisingly, paper and pencil was the most referenced tool, followed by whiteboards, analysis and modeling tools, HTML editors, asynchronous collaborative tools, Post-it Notes, and visual-interface builders. Although formal model-based tools are popular in general software development, our practitioners didn't regard them as critical to UI-related activities. In terms of synchronous workstyles, designers typically work at the same time and place (using whiteboards) rather than at the same time but different locations (using messengers or video-conferencing, for example). For asynchronous work, almost one-half of the respondents reported using tools such as CVS (Concurrent Ver-

sions System) and email to perform UI-related activities.

Visual interface builders were the seventh most popular tool class (out of 12). This is surprising, because our survey specifically asked participants what tools they used for UI design. This suggests that—despite current interface builders' success—there's a trend toward using widespread (and informal) tools for interaction design. We believe this reflects the nature of interaction design, which is highly creative, inherently interdisciplinary, and communication intensive. So, to succeed, designers should mesh formal, digital tools with informal, flexible, collaboration tools to support their work.

We also studied the relationship between the development process and the tools used. Table 2 and figure 2 show a subset of the results for various tools. Whiteboards and paper and pencil are the two most commonly used tools for all process styles, except formal methods. In those cases, developers used asyn-

Table 2**Practitioner tool choices for popular development processes**

Development process	Type of tool				
	Interface builders	Asynchronous tools	Whiteboards	Synchronous tools	Paper and pencil
Just do it	39	36	54	16	59
Spiral	34	29	61	11	82
Prototyping	37	45	55	23	80
Formal methods	30	56	47	26	81
Waterfall	46	39	69	28	80

chronous tools more often than whiteboards. The “just do it” approach is clearly the least tool intensive. We believe this reflects coding activities that rely mainly on tools the survey didn’t include, such as editors, compilers, and debuggers.

As these results show, more research is needed to understand how software and interaction designers perform tasks. For a deeper analysis of the survey results, see this article’s companion Web site (<http://dme.uma.pt/pcampos/workstyle>).

Workstyle transitions: Frequency and cost

When your work speaks for itself, don’t interrupt. —Henry J. Kaiser (1882–1967), American industrialist

Interaction and software designers’ UI-related activities clearly suffer numerous interruptions that can cause cognitive breakdowns. Among the constant interruptions designers face are colleagues, emails, switching from individual to collaborative workstyles many times a day, and shifting from low-tech card sorting to high-tech modeling with digital tools—something that also happens in general software development.¹⁰

We wanted to know what professionals thought about this, so we gave respondents several concrete scenarios of workstyle transitions. One scenario, for example, was “moving from high-level descriptions of the user interface (site maps, navigation maps, and so on) to detailed screens (with concrete widgets, buttons, and so on).” For each such transition, we asked them to rate

- the *frequency*; that is, “how many times [the respondent] engages and transitions in those workstyles,” and

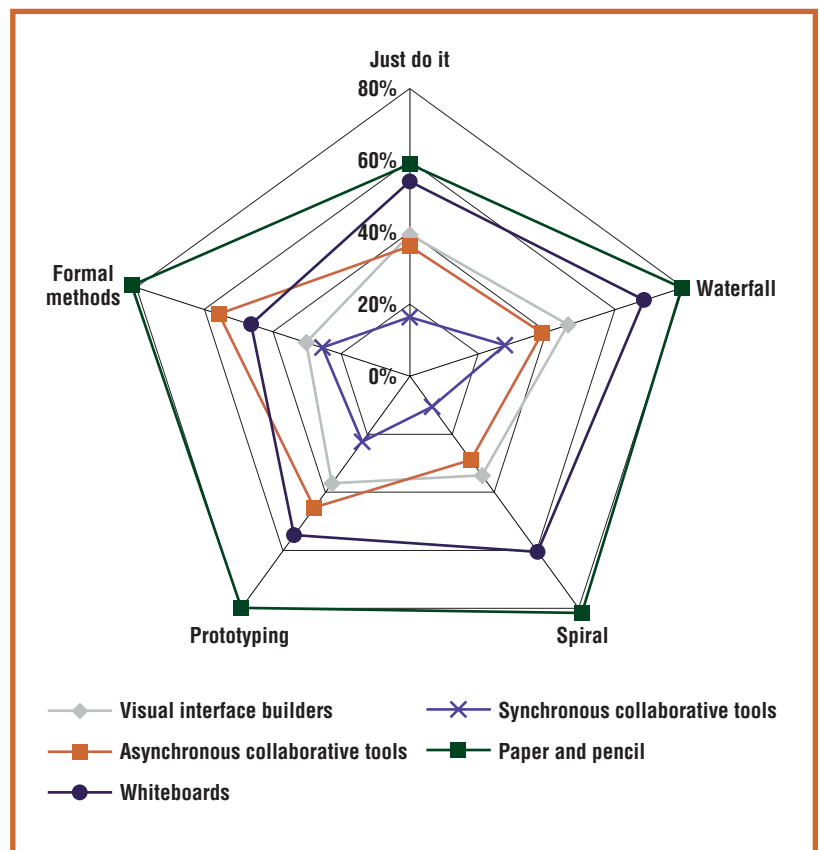


Figure 2. Tool use in the most popular development processes.

- the cost; that is, “how difficult [the respondent] finds it to perform that transition.”

Participants gave a qualitative rating by selecting a value from a seven-point Likert scale. Figure 3 plots the average over the total number of responses. As the figure shows, the second most frequent transition is moving from high-level descriptions to detailed screens. Interestingly enough, this is also one of the least costly transitions (exceeding only “moving from a

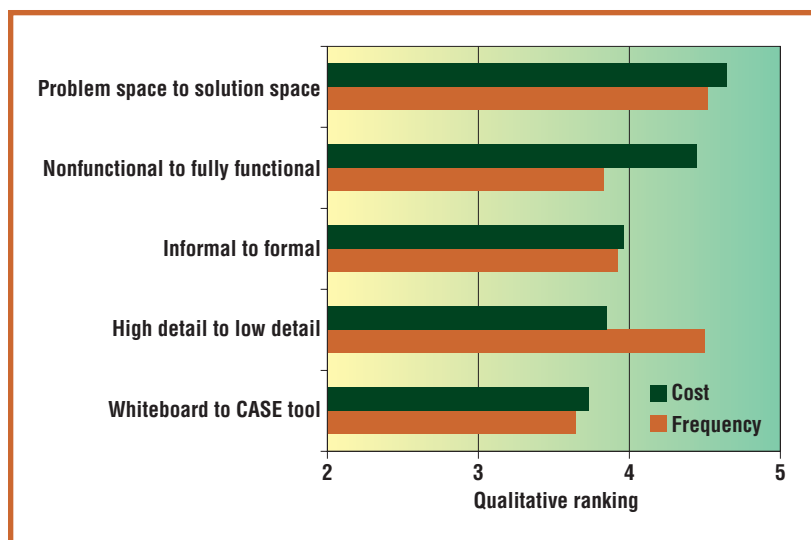


Figure 3. Workstyle transitions and their frequencies and cost.

whiteboard to a CASE tool”). The second most costly transition was “moving from nonfunctional to fully functional prototypes.”

The transition that ranked highest for both frequency and cost was “moving from business rules, use cases, and problem space concepts into final solution design, and back.” Another costly but less frequent transition was, “moving from nonfunctional prototypes into fully functional prototypes.” This suggests that designers add functionality to prototypes in clearly defined stages of evolution—after they’ve already committed to some design decisions.

In one follow-up interview, a designer stated that his answers would have been different if he were being questioned about issues such as debugging or code maintenance. This response was similar to those of other interviewees, who noted, for example, “the importance of building better tools for UI-related activities” and “how transitions were more difficult in UI-related issues.” Such responses help reinforce our claim about the importance of supporting workstyle transitions in UI activities.

Example tools for workstyle transitions

This study’s results have clear implications for developers of UI-related software design tools. As we mentioned before, the most difficult and frequent transition was moving from problem space concepts into final solution design and back. As an example of how to exploit such information, we developed the TaskSketch

development environment. TaskSketch supports changes in perspective (that is, in the problem or solution space) and embodies concepts that Larry Constantine has long supported;¹¹ he has given us many ideas that we’re trying to realize in testable tools.⁷

Another example is the CanonSketch tool, which supports changes in perspective, formality (adding informal sketches, adding formal semantics to models), functionality (the user can create functional, partially functional or nonfunctional prototypes), and detail.

TaskSketch

To better help designers move from problem space concepts to a final solution design and back, TaskSketch not only offers a set of constructs and views for use-case modeling but also provides drag-and-drop mechanisms for extracting an initial conceptual architecture for an interactive system. Designers can smoothly switch from a use-case perspective—described as a UML activity diagram—to a system architecture perspective. When designers use drag-and-drop between views, TaskSketch converts each model element in one view to the corresponding semantic equivalent in the other. TaskSketch thereby helps the designer (or system analyst) concentrate on what’s really important: architecting the system.

In our experience with small software development companies, traceability support is important in reducing the cognitive load of both designers and engineers. It’s also important in tasks such as prioritizing and negotiating development tasks. To support traceability, TaskSketch color-codes elements in relation to their use cases. Designers can thus immediately identify the most-colored elements as those that support the most use cases. Whether these elements are a user description or a concrete UI widget, they will probably be more difficult to implement.

TaskSketch also supports workstyle transitions by letting users edit task flows at three different—but synchronized—views:

- the participatory view, which typically results from a participatory session in which end users manipulate sticky notes;
- usage-centered design’s use-case narratives, which can be printed on index cards for different stakeholders to stack and order; and

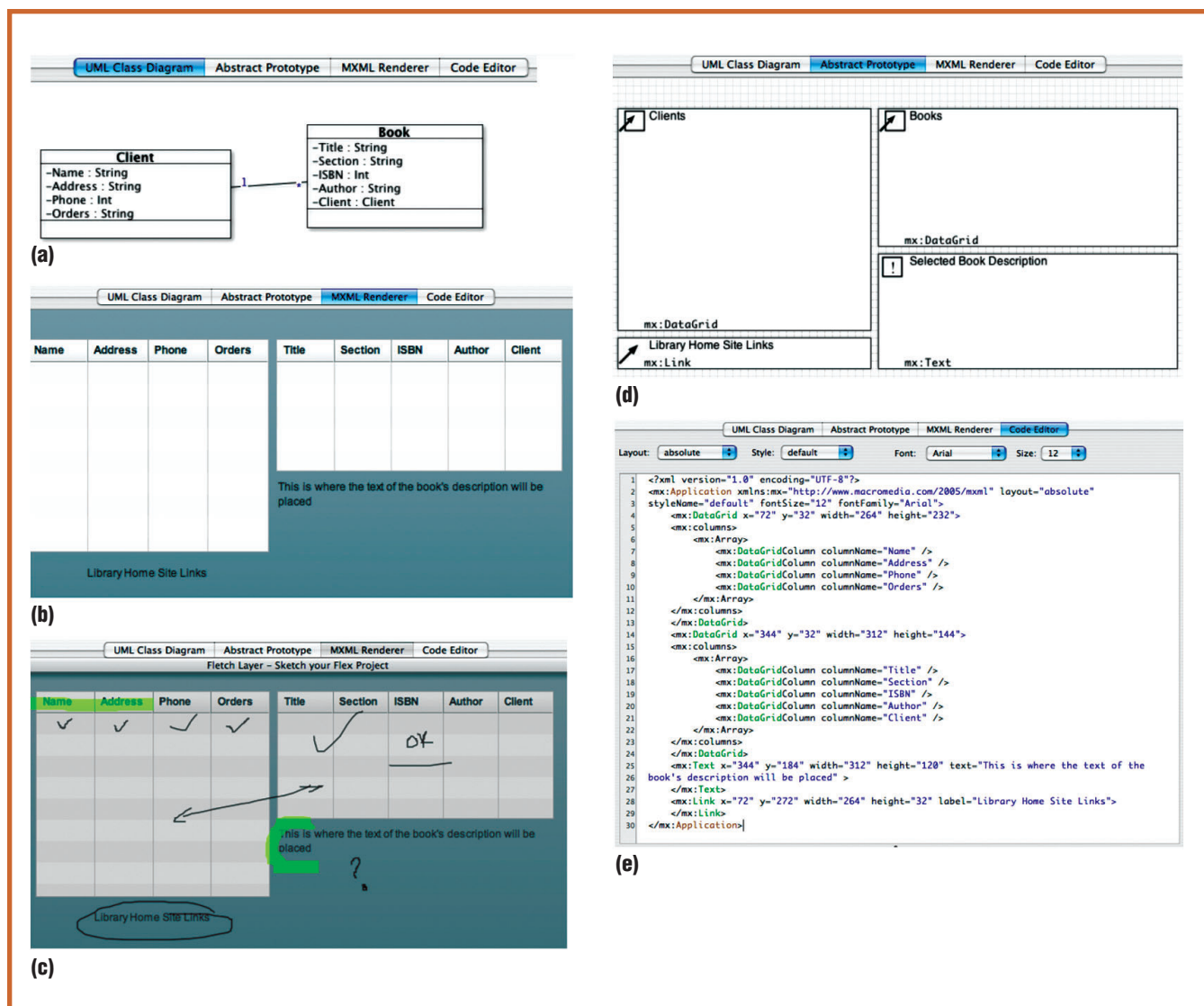


Figure 4. The CanonSketch tool supports workstyle transitions through four views: (a) the Unified Modeling Language class view for domain modeling, (b) the concrete UI view for testing the fully functional system, (c) a layer over the concrete view that lets users add informal comments and annotations, (d) the abstract prototype view for laying out the user interface and specifying UI elements and connecting them to UML classes, and (e) the code editor view for adding extra functionality.

- activity diagrams, which might include additional, relevant details that other views don't depict.

CanonSketch

This tool is an example of how workstyle transition support inspired new tool design ideas. CanonSketch offers four views and maintains the models synchronized between views, smoothing the designers' workstyle transitions. As a simple example, the user might start by creating a domain model class (**Client**) using

CanonSketch's UML class view. Using the abstract-prototype view, the user then creates a rough, abstract UI sketch using a smartboard or a tablet, for example. Next, the user applies the sketch recognition function to automatically sound abstract UI model (based on the UML's semantic model). Finally, the user selects concrete elements for each abstract element and switches to the code editor view to freely add functionality to the final UI. Figure 4 illustrates the CanonSketch tool; figure 4c shows how

About the Authors



Pedro Campos is a teaching assistant and researcher in the University of Madeira's Mathematics and Engineering Department, where he's a member of the Laboratory for Usage-Centered Software Engineering. His research interests are in the intersection of software engineering tools and usability engineering, as well as in user-centered development, software engineering processes, and UI design. Campos received his MSc in software engineering from the Technical University of Lisbon and is a member of the ACM and the IFIP Working Group on Human-Work Interaction Design. Contact him at LabUSE, Univ. of Madeira, Campus da Penteada, 9000-390 Funchal, Portugal; pcampos@uma.pt; dme.uma.pt/pcampos.

Nuno Jardim Nunes is an associate professor of computer science in the University of Madeira's Mathematics and Engineering Department. He's a founding member of the Laboratory for Usage-Centered Software Engineering, an R&D initiative dedicated to making technology more useful, usable, and accessible. His research interests are in bridging the gap between software engineering and human-computer interaction, in particular developing new techniques, methods, languages, and tools for software engineering approaches centered on human needs. He received his PhD in computer science from the University of Madeira and is a member of the IEEE Computer Society, the ACM, Eurographics, and the IFIP Working Group on User Interface Engineering. Contact him at LabUSE, Univ. of Madeira, Campus da Penteada, 9000-390 Funchal, Portugal; njn@uma.pt; dme.uma.pt/njn.



users can add a “layer” to freely sketch informal annotations over any of the views.

Tool availability

A beta version of TaskSketch is available at <http://dme.uma.pt/tasksketch>. Although the tool runs only on Mac OS X, in just eight months it registered more than 3,000 downloads (more than a third were from the US and less than 10 percent from Portugal). CanonSketch, which is freely available at <http://dme.uma.pt/canonsketch>, registered more than 2,000 downloads over a one-year period. Video demonstrations of both tools are available at <http://dme.uma.pt/pcampos/workstyle>.

With the probable dissemination of technology such as smartboards, as well as new developments in paperlike display technologies, we believe that tools such as CanonSketch will become more usable and will eventually replace paper and pencil and whiteboards as the tools of choice for UI development. Also, going from problem space to solution space is clearly a designer's hardest workstyle transition. To support this, researchers must develop additional integrated modeling tools like TaskSketch.

Empirical research that asks professionals to qualitatively classify the cost and frequency of various development scenarios is highly useful for understanding their daily work practices. Such an understanding offers a solid

foundation for building a better world of tools—and ultimately for removing the “hate” from the love-hate relationship between practitioners and their tools. ☺

Acknowledgments

We thank all the professionals who answered our survey as well as the mailing-list moderators for facilitating the survey's dissemination. A special thanks goes to Larry Constantine for feedback on CanonSketch and TaskSketch, and to Grady Booch, whose blogs about our work also helped disseminate the study. Thanks also to the anonymous reviewers for their clear, constructive comments and suggestions.

References

1. B. Myers, S. Hudson, and R. Pausch, “Past, Present and Future of User Interface Software Tools,” *ACM Trans. Computer-Human Interaction*, vol. 7, no. 1, 2000, pp. 3–28.
2. S. Jarzabek and R. Huang, “The Case for User-Centered CASE Tools,” *Comm. ACM*, vol. 41, no. 8, 2004, pp. 93–99.
3. J. Ilvari, “Why Are CASE Tools Not Used?” *Comm. ACM*, vol. 39, no. 10, 1996, pp. 94–103.
4. C. O'Hanlon, “What's on Your Hard Drive?” *ACM Queue*, vol. 4, no. 8, 2006, p. 9.
5. D. Spinellis, “The Tools at Hand,” *IEEE Software*, vol. 22, no. 1, 2005, pp. 10–12.
6. N.J. Nunes and J.F. Cunha, “Wisdom: A Software Engineering Method for Small Software Development Companies,” *IEEE Software*, vol. 17, no. 5, 2000, pp. 113–119.
7. P. Campos and N.J. Nunes, “CanonSketch: A User-Centered Tool for Canonical Abstract Prototyping,” *Proc. Int'l Conf. Eng. Human-Computer Interaction/Int'l Workshop Design, Specification and Verification of Interactive Systems (EHCI/DSV-IS 04)*, LNCS 3425, Springer, 2004, pp. 146–163.
8. P. Campos and N.J. Nunes, *Galactic Dimensions: A Unifying Workstyle Model for User-Centered Design*, LNCS 3585, Springer, 2005, pp. 158–169.
9. D.A. Dillman, *Mail and Internet Surveys: The Tailored Design Method*, John Wiley & Sons, 1999.
10. J. Wu and T.C.N. Graham, “The Software Design Board: A Tool Supporting Work-Style Transitions in Collaborative Software Design,” *Proc. Int'l Conf. Eng. Human-Computer Interaction/Int'l Workshop Design, Specification and Verification of Interactive Systems (EHCI/DSV-IS 04)*, LNCS 3425, Springer, 2004, pp. 363–382.
11. L. Constantine and L. Lockwood, *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley, 1999, pp. 194–195.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.