Multipurpose Prototypes for Assessing User Interfaces in Pervasive Computing Systems

In pervasive computing systems, prototypes can communicate design ideas, support user testing, and predict skilled performance. Designers can use CogTool to build HTML storyboards as prototypes for these purposes.

Bonnie E. John
Carnegie Mellon University

Dario D. Salvucci
Drexel University

In pervasive computing systems, prototypes serve several uses and have different requirements related to those uses. Some prototypes test a pervasive computer system's physical form—for example, will this wearable computer be light enough for eight hours of maintenance work or small enough to let a maintenance worker crawl through a narrow passage inside a machine? The essential features of a prototype for this purpose are form factor and weight, but it need not be functional in any interactive way. Other prototypes demonstrate whether a proposed technical solution can function at all—for instance, can a signal get from A to B in sufficient time and with sufficient strength to fulfill the envisioned system's functional needs? These prototypes must be narrowly functional (perhaps only sending a signal in a full round-trip) but not broadly functional, and they don't need to simulate the system's expected user interface.

Prototypes for pervasive system UIs often serve one of three purposes: they communicate design ideas to the development team, management, or customers; they provide a basis for user studies, often with novice users, to identify the design's usability problems; and they gather information about skilled users' projected performance. In deciding when to build and test prototypes, the balance between development costs and benefits tilts in favor of easy-to-build prototypes that fulfill multiple purposes.

We've developed CogTool to enable low-cost, rapid construction of interactive prototypes that serve all three UI purposes.[1] CogTool's core prototyping technique is storyboarding—specifically, interactive storyboarding using HTML. Rather than covering all possible pervasive systems, CogTool focuses on systems involving deliberate commands that the user invokes by some motor action. Such systems include PDAs, cell phones, handheld terminals (such as those used by rental car return personnel), in-vehicle driver information systems, and certain wearable computers that run desktop- or PDA-like applications.

Multiple uses for HTML prototypes

Designers can use storyboard-like UI prototypes to communicate design ideas to team members from software development, management, marketing, sales, and technical writing. Such prototypes can

- create a shared understanding and get various team members' buy-in for the design's direction;
- provide formative information about the UI—that is, identify usability problems early in the development process so team members can

TABLE 1
Requirements and products of prototypes for different purposes.

	Purpose of the prototype		
Requirements	**Communicating design ideas**	**User testing**	**Predictive human-performance modeling**
Information navigation	Yes	Yes	Yes
Correct procedures	Yes	It depends	Yes
Best-guess labels and commands	It depends	Yes	No
Best-guess visuals	It depends	Yes	No
Size and layout of controls	It depends	Yes	Yes
Best-guess response time	No	No	Yes
Interactivity	No	It helps a lot	It depends
Products	Team understanding	Usability problems of novices	Performance time of skilled users
	Team buy-in	Ideas for revised design	Bottlenecks in performance

make improvements before investing substantial coding time; and

- provide summative information—that is, show how much better the new system will be than the current system. For example, some pervasive computing systems replace paper-based systems, such as a wearable computer replacing a paper checklist or an in-vehicle navigation system replacing maps. Often, the new system will be more expensive per unit than paper. A return on investment analysis that estimates metrics such as task performance time for skilled users can help convince management or consumers to invest in the product.

Each purpose delivers different information to the development team and puts different requirements on the prototype, as table 1 shows.

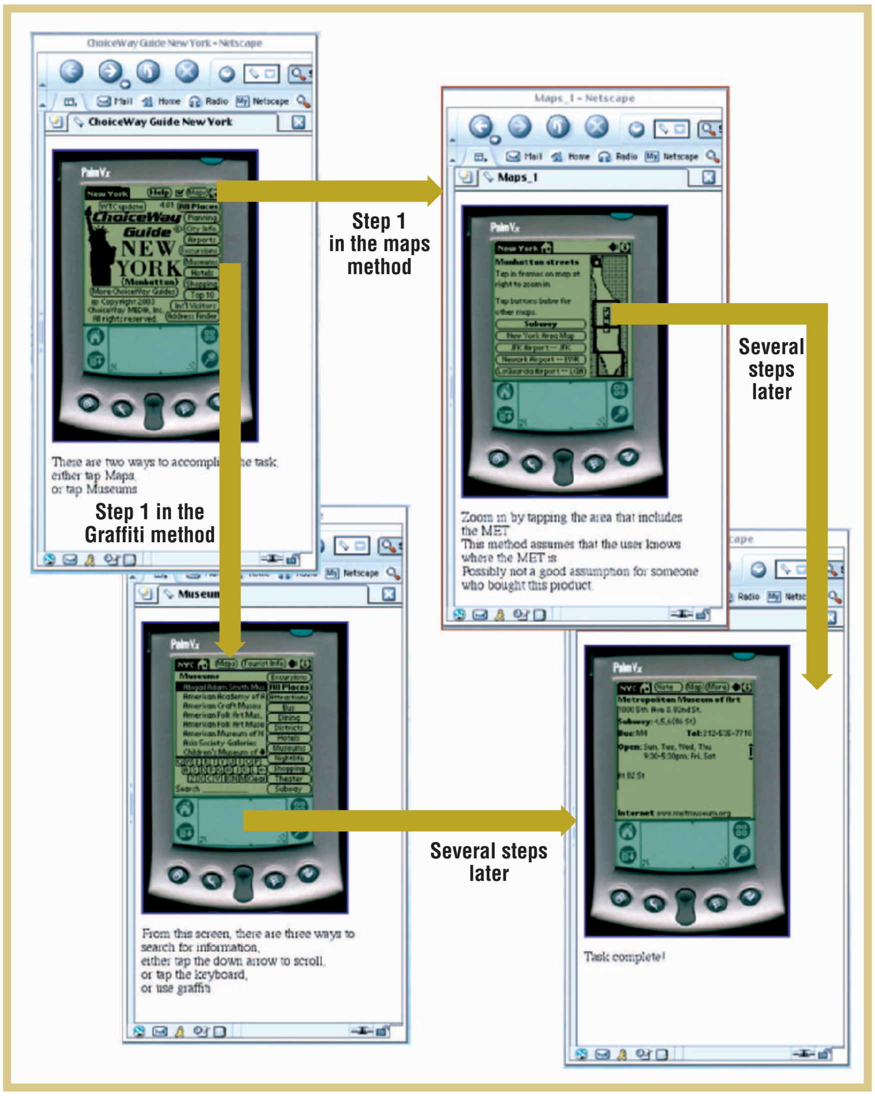
Communicating design ideas

To communicate design ideas to people not directly involved in the design process, a prototype must convey the planned interface's general feeling. The prototype might be as simple as a series of drawings of the different system states (such as screens) that a designer presents to the rest of the group. The prototype must convey how users will access information and perform tasks on the system so the team can see the envisioned interaction techniques and accessibility of information, as well as what performing some tasks would feel like. Because designers often present their work in person, they don't need to make the storyboard interactive or fill in all button and link labels; designers can describe interaction and command responses rather than fully specify them. Typically, they don't need to provide information about projected system response time for user actions because such lower-level information doesn't communicate well during a presentation. However, the prototype's look-and-feel is critical: the closer the storyboard's visuals are to the final system, the easier it will be for the audience to imagine a user's interaction with the system.

With the advent of what-you-see-is-what-you-get HTML editing tools, many designers already develop and present their prototypes in HTML. Because designers can incorporate arbitrary images into HTML, the technology is expressive enough to create storyboard prototypes that communicate design ideas at whatever level of detail they desire (for example, hand-drawn sketches or photographs of real devices or components). HTML also allows for simple interactivity with link transitions and basic components (pull-down menus and text boxes, for example) and annotations explaining visuals, labels, procedures, and design rationale, such as those used in the city guide in figure 1. Such prototypes can also serve as a form of interface requirements from which designers can write code. HTML storyboards are easy to share over the Web (which sometimes unintentionally gives them "legs" as they are passed around and appear in places the original designers didn't intend). Overall, HTML storyboards are efficient representations for producing shared understanding and buy-in in an organization.

User testing

User testing places different requirements on prototypes. As before, designers must build the anticipated information navigation into the prototype because navigation schemes are easily and usefully tested by asking users to think aloud as they perform tasks. The design team doesn't have to specify correct or optimal procedures for each task because user tests often aim to determine whether a novice user can discover a correct path to accomplish each task. (However, if the user test includes assessing training material effectiveness, correct procedures are usually written into the training documents and are thus a requirement for the test.) The designers must put into the prototype their best guesses at button or link labels, command names, and icons because user tests can uncover ambiguities in such items. Although user testing can be done with noninteractive systems such as paper prototypes, tests are much easier to administer and the user experience is


There are two ways to accompli[illegible]he task
either tap Maps
or tap Museums

Zoom in by tapping the area that includes
the MET
This method assumes that the user knows
where the MET is
Possibly not a good assumption for someone
who bought this product

From this screen, there are three ways to
search for information,
either tap the down arrow to scroll,
or tap the keyboard,
or use graffiti

Task complete!

ChoiceWay Guide New York - Netscape
Maps_1 - Netscape
Step 1 in the maps method
Step 1 in the Graffiti method
Several steps later
Several steps later

Figure 1. Parts of an HTML storyboard of an information-finding task on a Palm PDA, annotated with user procedures and design rationale.

much closer to actual use if the system is at least minimally interactive. However, because thinking aloud during a task slows performance, timing measures are unreliable in user tests and typically aren't recorded. Thus, the prototype doesn't have to provide a realistic system-response time.

HTML prototypes are well suited for user testing. They can include realistic information navigation, labels, commands, buttons, controls, and visuals. Links that take users to another page in response to their action can simulate interactivity. Because timing measures aren't recorded, unrealistic system response time incurred by the browser isn't a problem. User tests uncover many pervasive computing system usability problems even if the prototype is displayed on a desktop browser and operated with a mouse. In some cases, designers can even approximate the pervasive system's form factor using the HTML prototype. For example, they can design the HTML to display well on Pocket Internet Explorer and thereby deliver a prototype on a PDA. In our lab, we've positioned a touch screen on the dashboard of a driving simulator and used this to prototype in-vehicle navigation systems in HTML.

Predictive human performance modeling

Prototyping is also useful for obtaining information about skilled users' performance. Doing this empirically is expensive because measuring skilled performance requires users to interact with a robust prototype for hours, days, or even weeks. However, we can obtain skilled users' performance time analytically using predictive models of human behavior. Such models have been a centerpiece of applied cognitive psychology in human-computer interaction (HCI) for two decades. In the early 1980s, researchers introduced and validated the model human processor (MHP); keystroke-level models (KLMs); and goals, operators, methods, and selection rules (GOMS).[2] Since that time, other cognitive architectures—unified psychological theories and computational frameworks—have been developed to model human performance at many levels. For example, researchers have used ACT-R[3] and similar cognitive architectures to simulate behavior in several complex dynamic domains, including air-traffic control[4] and driving.[5]

Historically, predictive human behavior modeling has been done

- manually by a human analyst,
- computationally via faked interaction with a skeletal prototype, or
- computationally via real interaction with an implemented prototype.

In the most basic frameworks (such as MHP and KLM), a skilled analyst lists all of the actions the human must perform and the estimated system response time to those actions. The analyst then adds up the time to perform each listed operator. When using a computational cognitive architecture (for example, ACT-R and some versions of GOMS), the analyst can fake the interaction with a prototype simply by programming the architecture to wait a certain amount of time for each manual or perceptual

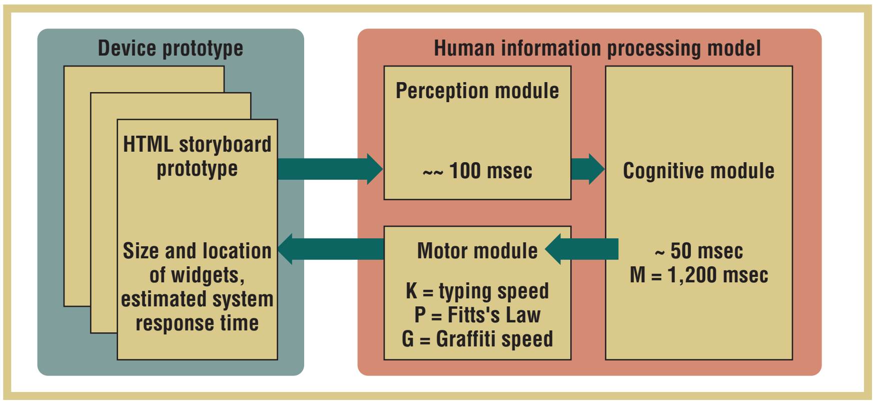


Figure 2. A human information processing model interacting with a device prototype.

action. The wait duration would correspond to established estimates from the psychology literature. Finally, the analyst might choose to exploit the computational cognitive architecture by connecting the model to some functional prototype so that the human simulation and the system simulation (prototype) can work in concert to generate predictions of skilled human performance.

In all three types of modeling, designers must specify the information navigation for the prototype that the human analyst works from (in the first and second types) or that the cognitive architecture connects to (in the third type), or the analysis can't proceed. The analyst must know the task's procedures, which must be programmed into the cognitive architecture as task knowledge to simulate a user who knows how to use the system. The designer need not work out labels and visuals because skilled users would have learned even the most unintuitive terms or icons. Also, cognitive architectures typically function as well with a generic label ("Label 1," for example) as they do with a semantic label for a button or link. (In contrast, some computational cognitive models, such as Marilyn Blackmon, Muneo Kitajima, and Peter Polson's comprehension-based linked model of deliberate search, require semantic labels for buttons, menu items, or links to predict the exploration behavior of users.[6]) However, skilled performance time depends on the controls' size and position, so the prototype must specify these. All three types of models require an estimate of system response time to make accurate predictions, but an interactive prototype is needed only when it's connected to a cognitive architecture.

HTML storyboards fill the requirements of prototypes for predicting human performance. They depict the interface sufficiently to give information navigation, correct procedures, and the controls' size and layout. They usually depict best-guess labels, commands, and visuals, which aren't needed by human performance models but also don't get in the models' way. As we describe next, we adapted ACT-R to interact with an HTML prototype through a Web browser and created a mechanism for specifying estimated system response time that's independent of the browser's actual response time or the cognitive model's speed.

Thus, HTML storyboards can fulfill the requirements to predict skilled human performance as well as those related to communicating design ideas and user testing. Building HTML storyboards as prototypes therefore fulfills multiple purposes for sharing and evaluating proposed user interfaces for a large subset of interactive pervasive systems.

CogTool

CogTool aims to make human performance modeling more accessible and affordable for computer system designers.[1] The psychological theory underlying CogTool draws from scores of results in the psychological literature and has been validated through HCI research and real-world use over the last 20 years.[7] In recent years, we've undertaken user-centered design of CogTool to understand UI designers' knowledge, skills, workflow, and workplace influences; uncover common error patterns; and user-test iterations to improve the design. The result is a tool that produces keystroke-level predictions of human performance from a demonstration of tasks on an HTML prototype. These predictions are as accurate as can be expected from the underlying theory (typically regarded as producing skilled performance time estimates within 20 percent of the actual performance) with an order of magnitude less effort than when done by hand.

Underlying cognitive theory

We based CogTool on the KLM version of a human information processing (HIP) theory of human behavior that Stuart Card, Thomas Moran, and Allen Newell proposed in the early 1980s.[2] Figure 2 shows a HIP model, which consists of three modules:

- The *perception* module takes information from the world (that is, the prototype) and passes it to a cognitive module.
- The *cognitive* module processes the information from the perception module, possibly combining it with information from other sources (such as a long-term memory), and sends commands to a motor module.
- The *motor* module manipulates the prototype to take action in the world.

The HIP theory underlying CogTool is specific to skilled human behavior with computer-based tasks with certain time parameters. Perception takes hundreds of milliseconds (ms) because signals in computer-based tasks are assumed to be relatively easy to perceive. (For example, perception would take longer if the tasks took place in extremely low light for visual signals or in high noise for auditory signals.) The motor module in KLMs for desktop applications includes operators for pressing keys (K), pointing (P), and homing between the keyboard and the mouse (not relevant to many pervasive computing systems, so not shown). To extend this module to PDAs, we added a Graffiti motor operator (G). We set the motor operators' durations based on psychology literature with K dependent on a person's typing speed, P related by Fitts's Law to the distance moved and the target's size, and G empirically determined in an experiment using a Palm PDA.[8] On a desktop, we'd model point-and-click by a K following each P to a target; for pervasive systems, the K is unnecessary because the original formulation of Fitts's Law includes physical contact with the target,[9] analogous to pressing a physical button or tapping with a stylus.

KLM has one mental operator (M) that represents a composite of all unobservable activities, including cognitive and perceptual activities such as comprehending perceptual input and recognizing the situation, command recall, and decisions. Card, Moran, and Newell empirically estimated M to be 1,350 ms. In CogTool, we estimate M to be 1,200 ms to accommodate the ACT-R architecture, which normally requires a 150-ms *look-at* operator to accompany each typical mental operation.

KLM makes approximate predictions of human behavior by assuming that a skilled person will execute each operator necessary to complete a given task sequentially. It adds all times together for a total task time estimate. Predicting the sequence of motor operators is straightforward, requiring only that an analyst know the task and system sufficiently well to list the operators needed to perform the task successfully. Including mental operators is more difficult. Card, Moran, and Newell, working with command-based desktop interfaces, defined five rules for placing Ms in a KLM. These rules depend on distinctions such as commands, arguments, and cognitive units. CogTool has updated and automated the placement of M operators by tying them to specific types of widgets found in modern interfaces. If the modeled system's response time makes a user wait (which in practice means that it exceeds the duration of an M), the KLM includes the waiting time in the total task time.

CogTool has updated and automated the placement of mental operators by tying them to specific types of widgets found in modern interfaces.

In the last two decades, while developers were using KLM pragmatically in HCI, cognitive psychologists were developing computational cognitive architectures with more sophisticated explanations of human performance. ACT-R incorporates significantly more detailed theoretical accounts of human cognition, perception, and performance than a typical KLM. For instance, the ACT-R computational engine includes theories of eye movement and visual attention, motor preparation and execution, and a theory of which operations can proceed in parallel with other operations. In 2003, researchers implemented a KLM-like modeling language, ACT-Simple, that compiles into ACT-R commands.[10] We built CogTool on top of ACT-Simple to marry KLM's simplicity with the ACT-R computational engine's predictive power. We added the ability for ACT-R to interact directly with HTML storyboard prototypes, "seeing" the widgets on the HTML page and operating those widgets with ACT-R's "hands."

Most work with KLM assumes that the user is currently engaged in only one task. Our research combines separately developed models of single tasks to predict the multitasking behavior that results from performing both tasks together. Building on experience developing multitasking driving and driver distraction models solely in the ACT-R architecture,[5] we've incorporated an ACT-R model of driver behavior into CogTool.[11] We can use CogTool to build prototypes of secondary tasks, such as dialing a cell phone or finding the nearest gas station with an in-vehicle navigation system, and integrate them with the driving model. Although theories of how people switch between tasks are still evolving, the CogTool driving model approximates human behavior by switching to the secondary task when driving is safe (that is, the vehicle is stable and near the lane center). The analyst must tell CogTool when to switch from the secondary task back to driving—for example, after a few button presses, at the end of a subtask, or at the end of the entire task. Task-switching strategies can have a large impact on driving performance. The integration with driving extends CogTool conceptually into a much broader array of interfaces and devices that are used in the context of real-world multitasking like many of today's pervasive systems.

How designers use CogTool

CogTool's most important aspect is that designers don't need to have any experience with or knowledge of human performance modeling—in fact, the theory's details and models are hidden from

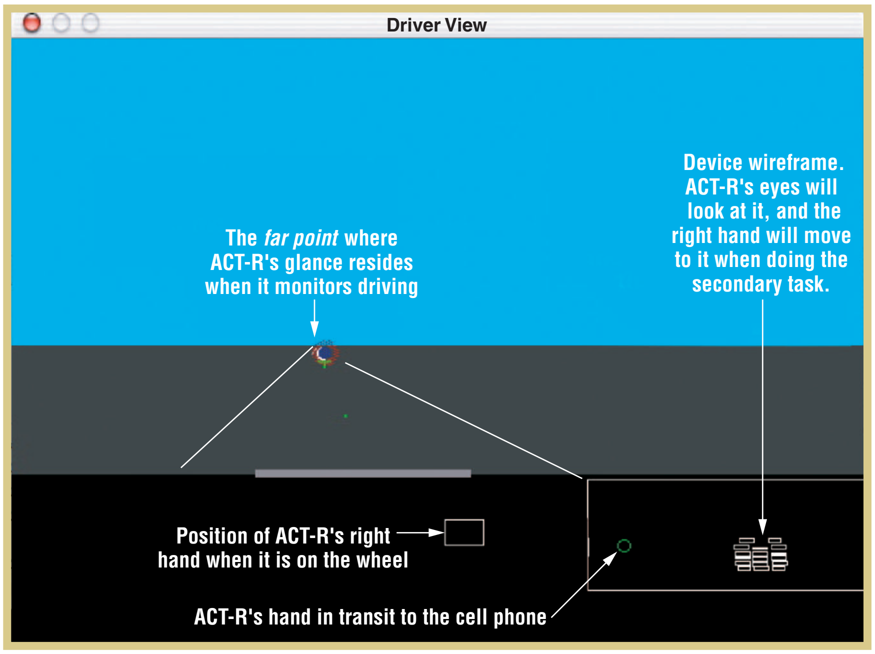

Driver View
The *far point* where ACT-R's glance resides when it monitors driving
Device wireframe. ACT-R's eyes will look at it, and the right hand will move to it when doing the secondary task.
Position of ACT-R's right hand when it is on the wheel
ACT-R's hand in transit to the cell phone

Figure 3. Driver view of the CogTool interface showing a user performing a secondary task while driving (the primary task).

the designer. Instead, CogTool provides a simple interface for building interfaces, specifying common tasks on the interface, and automatically predicting and analyzing the behavioral results of skilled users, with models in the background acting as simulated users.

A designer begins by building an HTML storyboard prototype of the proposed interface using CogTool. CogTool adds a palette to Dreamweaver that produces HTML and JavaScript that can communicate with CogTool while the analyst demonstrates a task and with ACT-R when a model is run. This palette contains many familiar UI widgets for desktop applications—such as buttons, checkboxes, links, and textboxes. It also contains tools more suited for prototyping pervasive computing systems, such as images, hotspots to place on the images, and a Graffiti area. Other palette tools let designers express nonmanual actions in a task. A tool that looks like a steering wheel tells the model to switch back to driving from a secondary task. The eye tool tells the model to visually gather information from a hotspot but not touch it. The stopwatch inserts a system response-time delay before the HTML storyboard page loads. The microphone and speaker tools let the model speak and listen to the storyboard, thereby generating a prototype for auditory interfaces.

After creating the HTML prototype, the designer demonstrates the desired tasks by clicking on widgets in the storyboard. Behind the scenes, CogTool's Behavior Recorder records and stores all interactions with the HTML prototype.[12] (The Behavior Recorder doesn't time the designer's actions during these demonstrations; the designer could click on one widget, have a cup of coffee, then click on the next widget, and the resulting model of skilled performance would be the same as if the designer didn't take a coffee break.) The designer then tells CogTool whether the prototype is of a mouse-based desktop system or a system that uses a stylus (or finger) as its primary interaction mode.

For standard desktop systems, CogTool translates the demonstrated clicks to mouse and keyboard actions, but for more general pervasive systems, it interprets clicks as taps with a stylus, Graffiti strokes, or physical depressions of a button or similar component (such as on a phone or navigation device). CogTool also places mental operators according to rules derived from Card, Moran, and Newell's results but updated to apply to the widgets in the CogTool palette. The demonstration's end result is a model expressed in ACT-R. When run, ACT-R interacts directly with the HTML prototype, "seeing" the widgets on the page and "pressing" them, thereby operating the prototype and producing a trace of skilled user behavior with quantitative predictions of performance time.

If the designer wishes to evaluate the interface while driving, he or she selects the Driving + Task Model option in the Model Launcher. A window opens, showing a road from a driver's viewpoint and a wire frame of the interface on the dashboard (figure 3). The model drives the car down the road and performs the secondary task, switching tasks as described previously. The resulting performance predictions are typical of empirical driver-distraction studies—for example, vehicle lateral deviation from lane center and brake-response delay from the onset of an external event (such as lead-car braking). Obtaining such results from CogTool requires no additional effort from the designer, except that it is sometimes useful to designate different reasonable points at which the driver switches between interface use and driving to explore the boundaries of safe and reckless attention to the road.

Experiences in pervasive computing

We've used CogTool as a prototyping and evaluation tool in various pervasive computing tasks. In fall 2004, 27 undergraduate and graduate students in Carnegie Mellon's HCI Methods class used CogTool to model different methods of entering a new appointment into a PDA calendar. Graduate students in a spring 2004 cognitive modeling class used CogTool to model common tasks on a Palm PDA and an in-vehicle navigation device. One student created more than 300 models to explore strategies for error

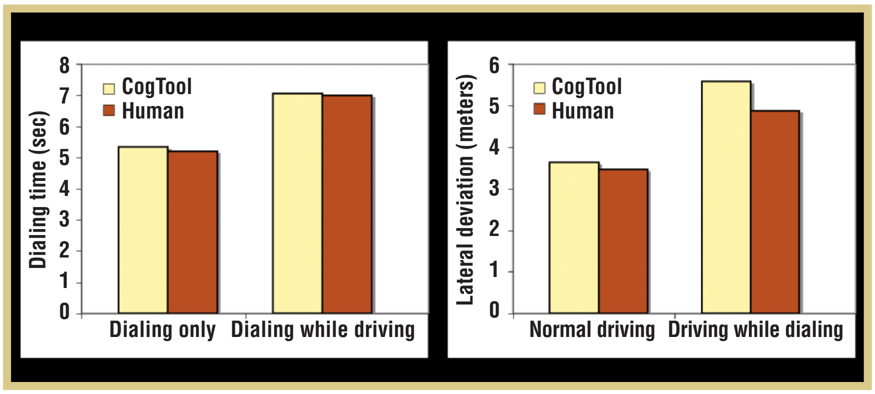


Figure 4. Comparison of CogTool and human user predictions of the effects of (a) driving on the time spent dialing a cell phone and (b) dialing a cell phone on a vehicle's deviation from the lane center.

detection and correction in cell phone short-message-service-like text entry using both multitap and T9 (www.T9.com). One software engineering researcher used CogTool to predict the skilled use of a Palm application to find information about the Metropolitan Museum of Art (see figure 1) and checked her predictions with a user study.[13] In her study, people accessed the requested information using one of four methods: map navigation, query by a soft keyboard, query by Graffiti, and locating the museum in a scrollable list. CogTool quickly generated a prototype of the interface and demonstrated the four tasks to produce predicted performance results. CogTool's performance corresponded extremely well with human user results, with an average absolute error of less than 8 percent.

Finally, we used CogTool to replicate earlier results that predicted driver distraction with cell phone dialing models written by expert ACT-R modelers.[10] We created mock-ups of a simple cell phone and demonstrated a 7-digit dialing task, assuming that drivers switched back to driving between the 3- and 4-digit blocks of the full number. Simply by asking CogTool to run this as a secondary task while driving, we could simulate this behavior and gather performance measures of the interactive effects of dialing and driving. Figure 4 shows that the tool's predictions match well with the human data. Driving slightly increases the total time needed to complete dialing, and dialing significantly affects the vehicle's lateral deviation (the distance from lane center as a measure of steering accuracy). CogTool thus produced the earlier expert findings with only minutes of prototyping and simulation, hiding the complex models' details from the designer and simply presenting the final measures for analysis.

Because CogTool simplifies the modeling process from a programming task, which involves specifying human behavior at the eye- and finger-movement level, to a demonstration task, it makes human performance modeling more accessible to pervasive and interactive system designers. Because ACT-R already includes several modalities (such as vision and audition or manual and speech acts) and is relatively easy to extend, CogTool will grow as a modeling tool for pervasive computing systems as well as traditional desktop systems. You can find CogTool documentation and downloads at www.cs.cmu.edu/~bej/cogtool. We look forward to hearing about your experiences and improving CogTool to meet your needs.

ACKNOWLEDGMENTS

We thank Peter Centgraf, Alex Eiser, Mike Horowitz, and Gus Prevas for their work on CogTool; Lu Luo for her Palm PDA model and data; and the students in HCI Methods and Cognitive Models for HCI at Carnegie Mellon University for their efforts using CogTool for modeling assignments.

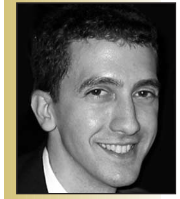
Bonnie John was supported by US Office of Naval Research grant N00014-03-1-0086. Dario Salvucci was supported by US Office of Naval Research grant N00014-03-1-0036. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Office of Naval Research or the US government.

REFERENCES

1. B.E. John et al., "Predictive Human Performance Modeling Made Easy," *Proc. Human Factors in Computing Systems: CHI 2004 Conf.*, ACM Press, 2004, pp. 455–462.
2. S. Card, T. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, 1983.
3. J.R. Anderson et al., "An Integrated Theory of the Mind," *Psychological Rev.*, vol. 111, no. 4, 2004, pp. 1036–1060.
4. N.A. Taatgen and F.J. Lee, "Production Compilation: A Simple Mechanism to Model Complex Skill Acquisition," *Human Factors*, vol. 45, no. 1, 2003, pp. 61–76.
5. D.D. Salvucci, "Predicting the Effects of In-Car Interface Use on Driver Performance: An Integrated Model Approach," *Int'l J. Human-Computer Studies*, vol. 55, no. 1, 2001, pp. 85–107.
6. M.H. Blackmon, M. Kitajima, and P.G. Polson, "Tool for Accurately Predicting Website Navigation Problems, Nonproblems, Problem Severity, and Effectiveness of Repairs," *Proc. Human Factors in Computing Systems: CHI 2005 Conf.*, ACM Press, 2005, pp. 31–40.
7. B.E. John and D.E. Kieras, "Using GOMS for User Interface Design and Evaluation: Which Technique?" *ACM Trans. Computer-Human Interaction*, vol. 3, no. 4, 1996, pp. 287–319.
8. M.D. Fleetwood et al., "An Evaluation of Text Entry in Palm OS-Graffiti and the Virtual Keyboard," *Proc. Human Factors and Ergonomics Soc., 46th Ann. Meeting*, CD-ROM, Human Factors and Ergonomics Soc. 2002.
9. P.M. Fitts, "The Information Capacity of the Human Motor System in Controlling Amplitude of Movement," *J. Experimental Psychology*, vol. 47, 1954, pp. 381–391.
10. D.D. Salvucci and F.J. Lee, "Simple Cogni-

the AUTHORS

Bonnie E. John is a professor in the Human-Computer Interaction Institute at Carnegie Mellon University. Her research focuses on modeling human performance to produce quantitative predictions of performance with less effort than prototyping and user testing. She also bridges the gap between HCI and software engineering by including usability concerns in software architecture design. She received her PhD in psychology from Carnegie Mellon University and is a member of the CHI Academy. Contact her at the Human-Computer Interaction Inst., Carnegie Mellon Univ., 5000 Forbes Ave., Pittsburgh, PA 15213; bej@cs.cmu.edu; www.cs.cmu.edu/~bej.



Dario D. Salvucci is an assistant professor of computer science at Drexel University. His research explores how computers can represent, predict, and respond to human behavior with cognitive models, particularly as applied to driving and driver distraction. He received his PhD in computer science from Carnegie Mellon University. Contact him at the Dept. of Computer Science, Drexel Univ., Philadelphia, PA 19104; salvucci@cs.drexel.edu; www.cs.drexel.edu/~salvucci.

tive Modeling in a Complex Cognitive Architecture," *Proc. Human Factors in Computing Systems: CHI 2003 Conf.*, ACM Press, 2003, pp. 265–272.

11. B.E. John et al., "Integrating Models and Tools in the Context of Driving and In-Vehicle Devices," *Proc. 6th Int'l Conf. Cognitive Modeling*, Lawrence Erlbaum, 2004, pp. 130–135.

12. K.R. Koedinger, V. Aleven, and N. Heffernan, "Toward a Rapid Development Environment for Cognitive Tutors. Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies, " *Proc. Conf. Artificial Intelligence in Education*, IOS Press, 2003, pp. 455–457.

13. L. Luo and B.E. John, "Predicting Task Execution Time on Handheld Devices Using the Keystroke-Level Model, *Proc. Human Factors in Computing Systems: CHI 2005 Conf.*, ACM Press, 2005, pp. 1605–1608.

ADVERTISER / PRODUCT INDEX

OCTOBER-DECEMBER 2005

Advertiser / Product	Page Number
Advanced Three-Dimensional Television System Technologies	5
Charles River Media	**14**
Filter UK	7
IBM Research	5
IO2 Technology	5
Newbury Networks	4
Realm Systems	6
Sharp	6

Boldface denotes advertisements in this issue.

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 212 419 7766
Fax: +1 212 419 7589
Email: md.ieeemedia@ieee.org

Marian Anderson
Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
IEEE Computer Society,
Business Development Manager
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

Advertising Sales Representatives

Product Display
John Restchack
Phone: +1 212 419 7578
Fax: +1 212 419 7589
Email: j.restchack@ieee.org

Recruitment Display
Mid Atlantic (recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0161
Email: db.ieeemedia@ieee.org

New England (recruitment)
Robert Zwick
Phone: +1 212 419 7765
Fax: +1 212 419 7570
Email: r.zwick@ieee.org

Southeast (recruitment)
Thomas M. Flynn
Phone: +1 770 645 2944
Fax: +1 770 993 4423
Email: flynntom@mindspring.com

Midwest/Southwest (recruitment)
Darcy Giovingo
Phone: +1 847 498-4520
Fax: +1 847 498-5911
Email: dg.ieeemedia@ieee.org

Northwest/Southern CA (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Japan (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Europe (recruitment)
Hilary Turnbull
Phone: +44 1875 825700
Fax: +44 1875 825701
Email:
impress@impressmedia.com

IEEE Pervasive Computing
10662 Los Vaqueros Circle
Los Alamitos, California 90720-1314
Phone: +1 714 821 8380; Fax: +1 714 821 4010
http://www.computer.org
advertising@computer.org