Learning When to Switch Tasks in a Dynamic Multitasking Environment

Dario D. Salvucci (salvucci@cs.drexel.edu) [1]
Niels A. Taatgen (taatgen@cmu.edu) [2]
Yelena Kushleyeva (yk45@drexel.edu) [1]

[1] Department of Computer Science, Drexel University
3141 Chestnut St., Philadelphia, PA 19104, USA

[2] Department of Psychology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA

Abstract

When performing concurrent tasks of non-trivial durations, people balance task processing by interleaving segments of one task with another. In this paper we describe a cognitive model of multitasking and interleaving for such extended concurrent tasks, specifically focusing on learning when to switch from one task to another. To explore this issue, we performed an empirical study of "discrete driving" in which participants used a keyboard to steer a vehicle while entering navigation information as a secondary task. We then compare and contrast two models of this task that learn task-switching behavior based on the (changing) characteristics of the discrete driving task.

Introduction

Human multitasking arises in many flavors and contexts. Much of the experimentation and modeling for multiple task performance has focused on short-duration tasks in paradigms such as psychological refractory period (PRP) tasks and alternating choice tasks. Scaling such tasks up to common scenarios in the real world, many real domains involve complex multitasking where multiple tasks are executed for extended periods of time. This type of multitasking is ubiquitous in our everyday lives, from a receptionist managing walk-in clients while answering phones, to a cashier totaling up purchases while bagging goods, to an air-traffic controller monitoring air traffic while communicating with multiple aircraft.

Cognitive models, especially those developed using today's larger-scale cognitive architectures, have accounted for this type of extended multitasking in at least two ways. Most commonly, models of complex tasks such as game playing (e.g., Laird & Duchi, 2000) and air-traffic control (e.g., Taatgen & Lee, 2003) have incorporated task-interleaving and scheduling mechanisms to manage component subtasks in their respective domains; as such, these models have specialized mechanisms, also called "customized executives" (Kieras et al., 2000), fine-tuned for the particular task. Other modeling efforts have focused on characteristics of domain-independent multitasking, such as a computational "general executive," for integration of component task models into larger multitasking models. Kieras et al. (2000) discuss issues of customized versus general executives at length in their treatment of many types of multitasking. Also, Salvucci (2005) has posited a queue-based general executive for the ACT-R architecture (Anderson et al., 2004), and Taatgen (2005) has described a general way in which this architecture can account for optimized performance of multiple concurrent tasks.

This paper explores a specific issue that arises in extended multitasking: how people determine when to switch tasks and how task interleaving may evolve in changing task conditions. To observe people's task interleaving behavior, we use a novel "discrete driving" task in which people steer a simulated vehicle with simple keystrokes while, at the same time, entering navigation information with the mouse as a secondary task. The motivation for this task arose from earlier studies of real driving and driver distraction: while realistic driving (in a driving simulator or instrumented vehicle) provides data clearly relevant to real-world scenarios, the points at which drivers switch between driving and the secondary task are obscured by the continuous steering input signal. The discrete driving task, by requiring separate discrete inputs for steering (i.e., the individual keystrokes), more clearly elucidates interleaving exhibited in the task. In addition, the task allows us to manipulate the difficulty of the steering task (by perturbing the vehicle more or less often, as described shortly), thus providing data for whether and how drivers adapt to changing conditions in the primary task.

Our modeling approach for this task is guided by the idea that time — or more accurately, a person's perception of time — has a significant influence on task interleaving. To this end, we demonstrate how the ACT-R cognitive architecture along with a new temporal module for time estimation (Taatgen et al., 2005) and general executive for goal interleaving (Salvucci, 2005) account well for certain aspects of human performance in the discrete driving task. In particular, we compare and contrast two models based on the same component task models but utilizing different methods for interleaving tasks, one that emphasizes bottom-up control and another that emphasizes top-down control. These models aim to demonstrate how, for an interesting set of complex task domains, cognitive models with a perceived sense of time can sense and adapt to the temporal characteristics of the component tasks.

Discrete Driving: Task and Empirical Study

The discrete driving task was inspired by earlier studies of driver behavior, especially those involving driver distraction with a secondary task such as cell-phone dialing or radio tuning (e.g., Salvucci, 2001, 2005; Salvucci & Macuga, 2002). While these studies were reasonably faithful to the real-world task with data collected in a driving simulator, the analysis of task switching was somewhat difficult because of the continuous nature of the steering input: the steering wheel is generally moving in a steady continuous fashion (although the movement may be updated by discrete signals that change its characteristics), and thus determining when people were cognitively processing the driving task versus the secondary task was a difficult inference. To address this issue, the discrete driving task allows for discrete driver "steering" by means of keystrokes on a desktop keyboard. The secondary task, a navigation entry task, requires mouse movements and clicks that mimic pointing on an in-vehicle touch-screen. Because both tasks provide discrete inputs (keystrokes and mouse clicks), the data can be readily analyzed to determine when people are performing actions in one task versus the other.

Task Overview

A sample screen of the driving task is shown in Figure 1. As the vehicle moves down the road, it is perturbed by discrete movements of 10 pixels to the left or right at semi-random intervals. In the *fast* condition, the vehicle moves after 0.5, 0.75, or 1.0 seconds with equal probability; in the *slow* condition, the vehicle moves after 2.0, 2.5, or 3.0 seconds. To "steer" the vehicle back to center, the driver uses two keys ('a' and 'd') to move left or right by 10 pixels per keystroke, and any keystroke resets the perturbation timer. When the vehicle sits in the road center, it moves left or right randomly with equal probability. When the vehicle sits to the right of center, it moves farther right with a probability of 2/3, and to the left of center, farther left with the same probability; this heuristic makes it more likely that the vehicle will drift away from center as opposed to toward the center, as is the typical case for real driving.

The secondary task performed while driving, the navigation entry task, asks drivers to enter street, city, and state information into a mock-up navigation device. Figure 2 shows one screen of the navigation interface used in our study (top), as well as the real navigation device (bottom) on which our interface is based (Garmin StreetPilot 2620).

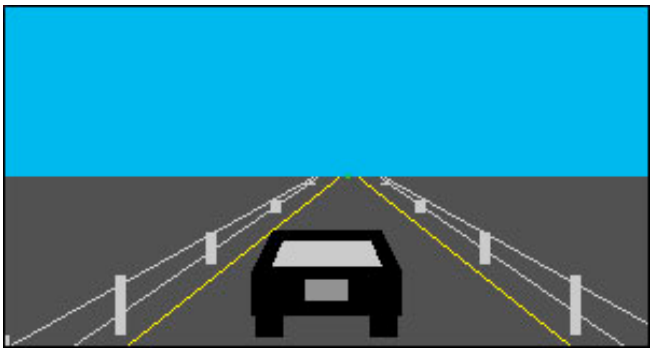

Figure 1: Sample screen of the discrete driving task.

Navigation entry begins with a screen in which the user selects one of four information items to enter: street number, street name, city, and state. When one of these is clicked, the interface screen changes to a keyboard layout — the alphabetic keyboard shown in Figure 2 for name/city/state, or a numeric/alphabetic keyboard (not shown but similar to Figure 2) for street number. The user clicks the individual letters and clicks "OK" when done. This process continues until all four items are entered, followed by a final "Done" button click. The information to be entered is displayed as simple text fields below the navigation interface.

Six stimuli for the navigation-entry task were created and standardized as follows. All street numbers contained three digits (all digits randomized over the stimuli). All street names contained six letters (e.g., "Sunset", "Walnut"), and all city names contained either nine or twelve letters (e.g., "Baltimore", "Philadelphia"). The state names contained two letters and correctly corresponded to the associated city name with no repeated states.

The discrete driving task including navigation entry task requires extended interleaving of two tasks, given that the entry task requires approximately 30-90 seconds for completion. While the task mimics real-world tasks, a much more realistic version would put drivers in a real vehicle or driving simulator while entering information on a real touch-screen GPS device. (Although navigation entry should never be performed during real driving, it has been estimated that over 2100 crashes per year may result from entry into these types of systems: Green, 1998.) However, the simplicity and discrete inputs of this task represents a balance of realism and simplicity as required to examine the detailed nature of multitasking in such a complex task.

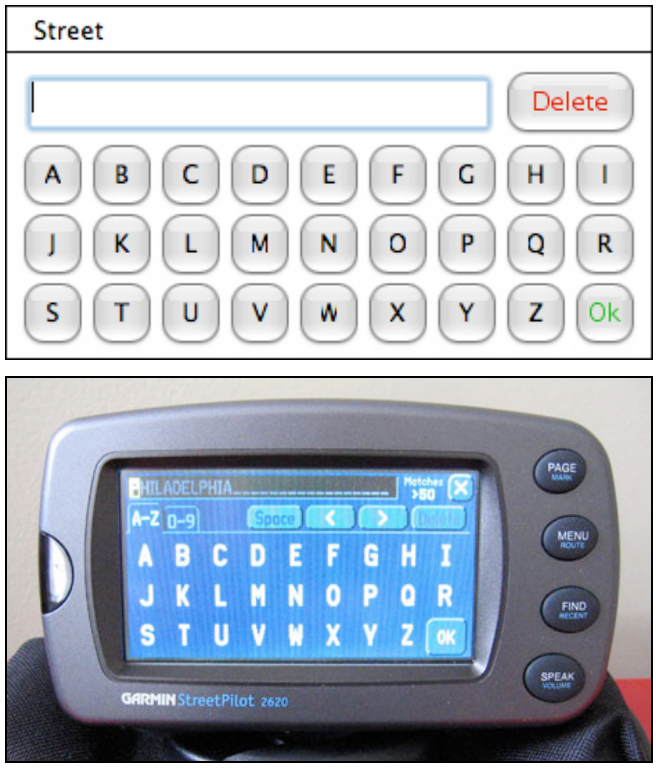

Figure 2: Sample screen of the navigation entry task, and the real navigation device on which it is based.

Experiment Overview

Ten university students (2 women, 8 men) participated in the experiment. Each participant was first instructed on the basic structure of the task; participants were asked to treat the discrete driving task with high priority, the way they treat normal driving, and to keep the vehicle as centered as possible on the roadway. Then, participants were allowed to practice each task separately: first navigation and then driving. The two practice blocks were followed by three experimental segments: navigation alone, then driving + navigation, then again navigation alone. In the remainder of this paper we discuss only on the central segment of trials with both tasks. This segment began with 20 seconds of driving, followed by onset of the first navigation-entry stimulus. The completion of the entry task was followed by 10 seconds of driving alone and then the next entry stimulus. In all, participants completed 18 trials of the entry task — three blocks of the six different stimuli in randomized, counterbalanced order.

One critical aspect of the experiment is the changing nature of the driving task. For the first two blocks of 6 trials (12 trials), the driving task remained in the initial *fast* condition. For the final block of 6 trials, the driving task changed to the *slow* condition. Participants were not told of this change, and thus we can examine their implicit adaptation to task demands, initially in learning the demands of the fast condition and later in learning the demands of the slow condition. The results of the empirical study are reported later in the next section along with the results of the cognitive model simulations.

Models of the Discrete Driving Task

We are interested in modeling several aspects of behavior in the discrete driving task; most importantly, we wish to explore the adaptation to the changing timing characteristics of the driving task with respect to interleaving the two tasks. As the core framework for this modeling effort, we use the ACT-R cognitive architecture (Anderson et al., 2004) with two previously validated modules: the temporal module for time perception (Taatgen et al. 2005), and the general executive for task interleaving (Salvucci, 2005). We describe two possible models for the task: one assumes an integrated goal representation with "bottom-up" shifts of control initiated by the individual tasks; the other assumes separate goal representations for each task with "top-down" shifts of control initiated by a general executive mechanism.

Modeling Architecture

The ACT-R cognitive architecture represents cognition as a system of production rules; the system receives input from a group of special modules, each corresponding to a semi-independent processor of a specific modality of information, as shown in Figure 3. The core architecture does not include a module that allows for temporal perception, and contains a simple goal module that includes just a buffer with the current goal. These two current limitations led us to use two recently validated architectural extensions: the temporal module and the general executive.

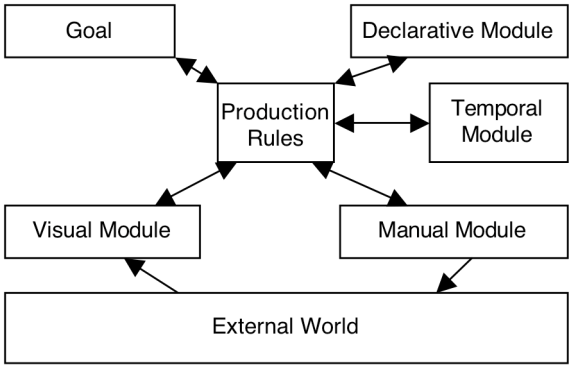

Figure 3: Outline of the ACT-R architecture.

Temporal Module. The temporal module (Taatgen et al., 2005) allows ACT-R to estimate and reproduce time intervals up to approximately 30 seconds. It acts like a metronome, but one that starts ticking slower and slower as time progresses. The interval estimate is based on the number of ticks the metronome has produced. More precisely, the duration of the first tick t_0 is set to some start value *start-tick*. Then, each subsequent tick is separated from the previous tick by an interval that is a times the interval between the previous two ticks. Each interval also has noise drawn from a logistic distribution added to it. The distribution of this noise is determined by the current tick-length. More concisely, the tick duration t_n is computed as:

$$t_{n+1} = at_n + \text{noise}(mean = 0, sd = b \cdot at_n)$$

In ACT-R, as suggested by Figure 3, production rules must start and read the temporal module timer. This means that errors in time estimates are due not only to noise in the temporal module itself, but also to the production system not initiating or reading the timer at the right moment.

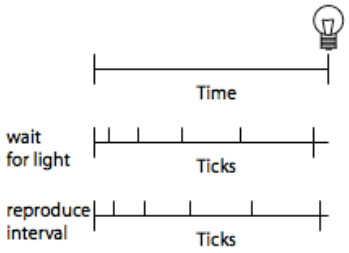

Figure 4: Illustration of the temporal module.

For example, suppose we want to reproduce a time interval before a light turns on, as represented in the first horizontal bar in Figure 4. The timer is initiated at the start of the trial. When the light comes on, the value of the timer (5 ticks in the example) is read and stored. When the interval has to be reproduced, the value of the timer perceived earlier is used. As the figure illustrates, the reproduced interval is not entirely accurate due to the noise in the calculations. Taatgen et al. (2005) have estimated values for the parameters in these equations to obtain an optimal fit to the Rakitin et al. (1998) experiment of interval estimation: 11 ms for *start-tick*, 1.1 for a, and 0.015 for b. These values

also provided excellent fits to the other experiments discussed in that paper, and are also the values we have used in the models described in this paper.

General Executive. The general executive (Salvucci, 2005) allows ACT-R to manage a set of current goals and execute them on a first-in, first-out (FIFO) basis. In the Figure 3 diagram, the general executive resides in the goal module, taking requests for new goals and placing them onto the queue. When any changes are made to the goal set — that is, when a new goal is requested or the active goal is terminated — the general executive sets the foremost goal on the queue as the current goal. It is important to note that "goals" in this case are typically small subgoals of a continuous or extended task; for example, a driving goal might be to look at the road and center the road, or a navigation goal might be to type one character. Thus, the general executive relies on the goal representation and production rules to define where interleaving may take place: essentially, interleaving can happen between (sub)goals but not during a particular (sub)goal.

The most recent formulation of the general executive (Salvucci, 2005) incorporated its own timing mechanism that could alter the order of goals in the queue based on requested start times. In the present work, we do not use this timing mechanism, and instead rely on the temporal module to perform the timing within the production rules. This decision was motivated by the desire to use a single timing mechanism for both models below, and to show how the more rigorously validated temporal module described above meshes well with the mechanisms of the general executive, providing more validation for both mechanisms.

Component Models and Integration

We developed two models of the full task, both based on the same models of the two component tasks, driving and navigation entry. The driving model followed the basic structure of a fully developed model of real driving (see Salvucci, 2005) but with modifications to perform discrete driving. Specifically, the model centers the vehicle by attending to the center point of the road, determining whether it is left or right of center, pressing the appropriate key for moving the vehicle one step, and repeating this cycle until the vehicle is centered on the roadway.

The model of navigation entry performs the basic steps of the task in the expected order. On the first category screen, the model selects the first unspecified category (i.e., street number, street name, city, or state) and clicks this button to begin entry. After encoding the respective information provided below the navigation interface, it types the individual letters/digits by clicking on their respective buttons. The model is provided with locations of the buttons and thus not required to search for the correct button, thus assuming that initial visual-location learning has already occurred in the practice trials. When each item is completely typed, the model clicks "OK" and moves to the next unspecified item until all information has been entered, ending with a final click on the "Done" button.

These component models were integrated in two ways to create distinct models which we characterize as "bottom-up" and "top-down" control models. In truth, neither model is purely one or the other: the bottom-up model includes top-down characteristics and vice-versa. However, we thus term the models for the sake of simplicity and to recognize that one model is *more* bottom-up and the other more top-down, as opposed to completely one or the other.

Bottom-Up Control Model. The first of our two models uses "bottom-up control" in that events determine what it will do, including switching between the two tasks. Events in this context are defined in terms of changes in ACT-R's perceptual buffers (which for this purpose includes the temporal buffer). The model uses three control states to keep track of what it is doing: driving, determining which address item (number, street, city, or state) has to be entered, and typing.

The main control aspect of the model is the switching between the two tasks. The model switches from driving to typing whenever the car is in the center of the road. At that moment it also gives the start signal to the temporal module, which will start keeping track of time. Initially the model does not know how much time can be spent on typing without endangering the driving, but it tries to determine this on the basis of experience. For this we use an experience-based method that Taatgen et al. (2005) designed for another time perception task in which a time interval had to be determined. While the model is typing, it also tries to retrieve from memory a previous experience of looking away from the road. Specifically, it tries to find an experience that matches the current time (as represented in the temporal buffer) or the near future (within 10 "ticks"). Retrieving experiences and typing can be interleaved reasonably well because typing mainly involves perceptual and motor actions. If the model fails to retrieve a past experience (which is guaranteed to happen the first time), it immediately switches back to driving. If the model looks back at the road and the car is still in the middle of the road, it evaluates the experience as *early*; if car is slightly off the middle of the road, it will evaluate it as a *success*; and if the car is far from the center (we define "far" in an upcoming section), it will evaluate it as *late*.

If the model does retrieve a prior experience, it will act on how that experience was evaluated. When the model later retrieves an *early* experience, it will continue typing and retrieve a new experience; if the model retrieves a *success* experience, it will continue typing but will switch back to driving as soon as the temporal buffer reaches the time in the retrieved experience; if the model retrieves a *late* experience, it will switch back to driving right away. The bottom-up model can switch back to driving at any moment during the typing, and will primarily do so during mouse movements, since these movements take up the majority of the entry time.

Top-Down Control Model. The second model uses "top-down control" in that it allows the general executive to perform the interleaving of the two tasks. The rules for the component models were first modified such that any rule that modified the existing goal chunk instead created a new goal chunk, except for rules that made requests to the visual module (to avoid interrupting visual processing). For the

driving model, this meant that the top-down model allows task interruption after each centering movement; for the navigation-entry task, the model allows interruption between each mouse click. The rules for each task make no mention of the other task, as if they had been learned independently; this facilitates running each task by itself or, if desired, running it with other additional tasks.

The timing in the top-down model is contained in the rules for the driving task. The model maintains a desired time delay to wait before each new centering movement, quantified in terms of number of ticks, as provided by the temporal module. With an initial delay of 0, the model can either effectively increment or decrement the delay based on the safety margins of the driving task. Specifically, when the model returns to the task after the desired delay, it classifies the delay as *early*, *late*, or *success* (analogous to the bottom-up model) and adjusts the delay. When the vehicle is still centered, the delay is *early* and is reset to the elapsed time since the last task (which is at least 50 ms greater than the current delay). When the vehicle is farther from center than the desired safety margin, the delay is *late* and is reset to 0. When the vehicle is an acceptable distance away from center, the delay is *success* and remains the same. The model also allows for some probability (preset to 0.5) of checking the roadway before the desired time delay; if the vehicle is outside of the safety margins, the delay resets to 0, otherwise it remains the same.

Model Summary. While the models differ in their specifics of how timing can occur in "bottom-up" and "top-down" models, both models essentially take a satisficing approach to learning appropriate time intervals for switching: they determine whether recent switches were early, late, or successful as defined by the task, and adapt the time interval accordingly. 'Early' can easily be defined in this context, namely when the vehicle is still in the lane center. 'Late' is somewhat more difficult, in that it represents a person's tolerance for how far the vehicle can stray from the lane center. (It turned out that this parameter had little impact on the model results for the bottom-up model, though slightly more impact on the top-down model.)

Results

We ran simulations for each model and analyzed the results in conjunction with those of the human participants in the experiment. The bottom-up model was constructed by the second author without him having access to the data, so it can be considered as a true prediction of the outcome. The top-down model is a modification of the bottom-up model to include the use of the central executive.

We begin with the results of primary interest in this paper, namely how the changing characteristics of the driving task affected task interleaving of the two tasks. Figure 5 shows the average time spent on the secondary task (navigation entry) before switching back to driving — that is, the time between driving keystrokes when interrupted by button clicks for the navigation task. The human and model data are split into three stages, namely the first, second, and third set of 6 trials in the experiment, where the first and second represent the *fast* driving condition and the third

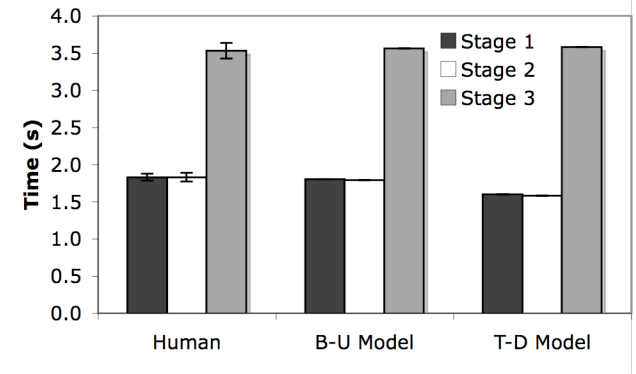

Figure 5: Average time between driving actions.

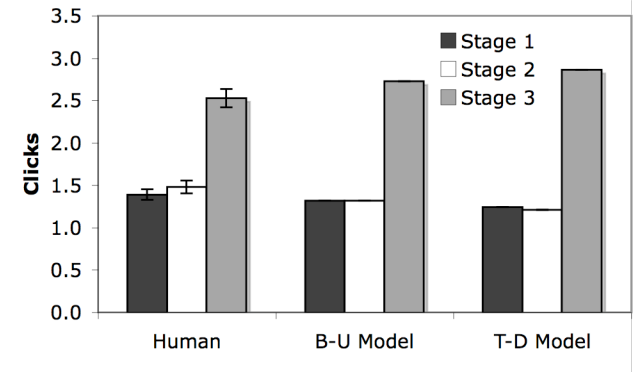

Figure 6: Average number of navigation-task mouse clicks between driving actions.

represents the *slow* condition. The human participants show an average time of approximately 1.8 s, and the times for Stages 1 and 2 are not significantly different, $t(9)=.73$, $p>.48$. However, there is a clearly significant jump from Stage 2 to 3 when the driving condition changes from *fast* to *slow*, $t(9)=16.6$, $p<.01$, to a value of approximately 3.5 s. Both models reproduce these values very closely, $R^2>.99$. Clearly the models are correctly learning appropriate switch times based on the characteristics of the driving task, both initially at the onset of the experiment for the *fast* condition, and later for Stage 3 after the change to the *slow* condition.

Another way to analyze the amount of processing on the secondary task is to examine the number of mouse clicks instead of total time. Figure 6 shows the average number of clicks between interrupted driving keystrokes. Again the human participants show a large effect from Stage 2 to 3, $t(9)=14.1$, $p<.01$. For this measure, however, they also exhibit a significant albeit very small effect between Stages 1 and 2, $t(9)=2.7$, $p<.05$. The model results closely correspond with the human results, $R^2>.99$, and in particular predict the large effect for Stage 3. The models do not predict any effect between the first two stages; we believe the human data exhibit the small effect due to final visual learning of the button locations on the interface, and not surprisingly the models do not predict this effect due to their preset visual locations. Interestingly, this does not lead to a significant difference in time for the human data in Stages 1

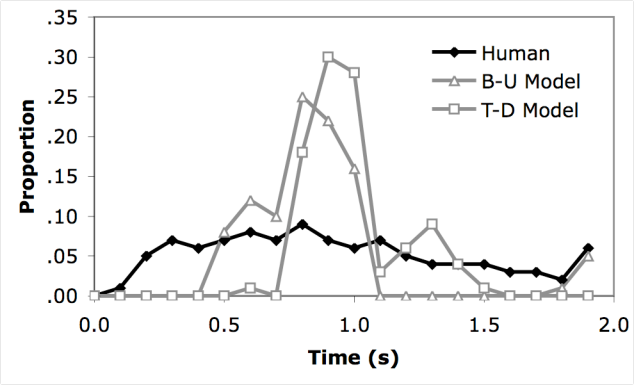

Figure 7: Histogram of times between a driving action and the subsequent navigation-task mouse click.

and 2 (as shown in Figure 5); apparently the participants are keeping time constant while squeezing slightly more clicks into this time in the second stage.

While the behavior of both models corresponds well to human behavior for these measures, it is interesting to investigate where they differ. One key difference between the models is that the top-down model only allows switching back to driving at natural subtask boundaries, while the bottom-up model allows switching back any time. The bottom-up model therefore allows switching back to driving between the mouse-movement towards a key and the mouse-click on the key, contrary to the top-down model. If the bottom-up model is right there should be fast mouse-clicks directly after switching back from driving. Figure 7 shows histograms for the time delay between a driving keystroke and the subsequent navigation mouse click (including Stage 2 only). The human data show a wide spread of times between 0.25 and 2.0 s. In contrast, both models exhibit a tighter distribution: the bottom-up model tends to click sooner than the top-down model because it sometimes processes the driving task during a mouse movement, and thus has less work to complete upon returning to the entry task; the top-down model always moves and clicks in sequence with no interruption, with the navigation model using vision and thus not allowing the general executive to switch tasks during the movement. The longest human times could be due to visual searching early in the experiment, whereas the shortest times could be attributed to driving during mouse movement, consistent with the behavior of the bottom-up model.

General Discussion

The key result presented here is that models developed in a cognitive architecture such as ACT-R can learn appropriate task-switching intervals based on the changing demands of the component tasks. Some previous work (e.g., Salvucci, 2005) utilized models with task-switching time intervals, but the actual values of the time intervals were estimated as a free model parameter and fixed as a constant throughout a model run. Here, we have demonstrated that not only can the model learn appropriate intervals, but can also adapt them by taking a satisficing view of task demands that evaluate task scenarios as early, late, or successful.

There are many aspects of the discrete driving data, both human and model, which cannot be included here due to space limitations. We are now performing a fuller analysis of these results to understand the larger scope of the models' behavior and how well they account for other task measures, such as effects of distraction on the primary driving task and effects of representation (e.g., time intervals for each letter in the six-letter street name) on both the driving and navigation entry tasks.

Acknowledgments

This work was supported by ONR grant #N00014-03-1-0036 to the first author, ONR grant #N00014-06-1-0055 to the second author, and an NSF Graduate Fellowship to the third author.

References

Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, *111*, 1036-1060

Green, P. (1999). Visual and task demands of driver information systems. Tech. Rep. #UMTRI-98-16, University of Michigan, Transportation Research Institute, Human Factors Group / US Society of Automotive Engineers.

Kieras, D.E., Meyer, D.E., Ballas, J.A., & Lauber, E.J. (2000). Modern computational perspectives on executive mental processes and cognitive control: Where to from here?. In S. Monsell & J. Driver (Eds.), *Control of Cognitive Processes* (pp. 681-712). Cambridge, MA: MIT Press.

Rakitin, B.C., Gibbon, J., Penney, T.B., Malapani, C., Hinton, S.C. and Meck, W.H. (1998). Scalar Expectancy Theory and Peak-Interval Timing in Humans. *Journal of Experimental Psychology: Animal Behavior Proces*ses, *24*, 15-33.

Salvucci, D.D. (2001). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies*, *55*, 85-107.

Salvucci, D.D. (2005). A multitasking general executive for compound continuous tasks. *Cognitive Science*, *29*, 457-492.

Salvucci, D.D., & Macuga, K.L. (2002). Predicting the effects of cellular-phone dialing on driver performance. *Cognitive Systems Research*, *3*, 95-102.

Taatgen, N.A. (2005). Modeling parallelization and flexibility improvements in skill acquisition: From dual tasks to complex dynamic skills. *Cognitive Science*, *29*, 421-455.

Taatgen, N.A., Anderson, J.R., Dickison, D. & van Rijn, H. (2005). Time interval estimation: Internal clock or attentional mechanism? In *Proceedings of the Twenty-Seventh Annual Meeting of the Cognitive Science Society* (pp. 2122-2127). Mahwah, NJ: Erlbaum.

Taatgen, N.A., & Lee, F.J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*, 61-76.