

Developing a Cognitive Model of Expert Performance for Ship Navigation Maneuvers in an Intelligent Tutoring System

Jason H. Wong¹, Susan S. Kirschenbaum¹, Stanley Peters²

¹Naval Undersea Warfare Center, Newport, RI

²Stanford University, Stanford, CA

jason.h.wong@navy.mil, susan.kirschenbaum@navy.mil, peters@csli.stanford.edu

Keywords:

expert performance, ship navigation, perceptual heuristics, intelligent tutor, cognitive task analysis

ABSTRACT: *The goal of this project is to develop a cognitive model of expert ship-handling performance. This model was integrated with an intelligent tutoring system and an immersive visual simulation used by the U.S. Navy. This intelligent tutor and expert cognitive model (written in a Java-based version of ACT-R) provides feedback to the student based on the student actions in order to reduce workload on the instructors. The nature of ship navigation and the requirements for the intelligent tutor presented unique challenges for development. This paper describes how the resulting cognitive model balances a need for expert performance while compensating for student error, uses perceptual heuristics when the ACT-R vision module is not feasible, and how these and other issues affected model development. Future plans for system test and evaluation are also discussed in the context of improving training.*

1. Project Overview

The Conning Officer Virtual Environment (COVE) is a ship-handling simulation system used by the U.S. Navy to train officers in how to complete ship navigation maneuvers (known in the U.S. Navy as ship-handling “evolutions”). These can include docking a ship, getting a ship underway, or twisting a ship about its axis. This training occurs after students undergo classroom instruction, so this simulation provides a hands-on practice environment for novices. COVE, which is based on the Virtual Ship software (Computer Sciences Corporation, 2009), is used to provide students with ship-handling training without the cost or risk to equipment of at-sea exercises. One downside to this system is that an expert instructor is required to constantly monitor progress and provide feedback, no matter how basic the exercise.

In order to reduce the overall workload on instructors, the goal of this project was to develop a system consisting of a set of new components that interact with each other. One component is an intelligent tutor (Bratt, Schultz & Peters, 2007) that monitors student progress and provides appropriate feedback. The second component, and the subject of this paper, is a cognitive model (developed using a Java-based implementation of ACT-R; Harrison, 2009) of expert performance. This model is designed to represent expert performance in various ship navigation evolutions to provide a point of comparison against the actions taken by the student.

The requirement that the model represent human performance led to the selection of ACT-R (Anderson, et al., 2004; Anderson, 2007) as choice of cognitive architecture to implement the specific cognitive and perceptual operations used in completing an evolution. The use of a cognitive architecture guides the creation of a system that represents human cognition (and its limits) instead of a computer-based algorithmic solution that ignores the constraints of cognition.

The expert model was designed to provide the tutor with a sense of how an expert would perform the navigation evolution, including the actions taken, rules followed, and perceptual cues that are used. The entire system would then be able to give feedback to the student based on the actions taken and visual cues examined. While some cognitive models have been developed to operate with other components, few have been developed to support an intelligent tutoring system, and this presents a unique set of challenges.

2. Description of System Components

The task environment that the cognitive model operates in consists of multiple pieces. The primary component is the COVE simulation software itself. The simulation strives for realism in many important areas (Smallman & St. John, 2005), including elements in the visual environment such as hydrodynamics, weather, currents, piers, buoys, and ships. Some ships are also modeled in high-fidelity; that is, the physics of the engine and rudder are accurately modeled instead of the ship following a

simple speed and course. All of these elements are rendered in an immersive environment that can be displayed on a single monitor, in a more complex multiple monitor setup, or using a head-mounted display. A screenshot of the rendered scenario can be seen in Figure 1 (top). The multi-monitor setup is complete with head-tracking, control through voice recognition, text-to-speech capability, and a separate instructor console for monitoring performance.

COVE scenarios are created using specially designed software that includes detailed real-world ports and realistic ships that are placed in the environment. Ships can be given a set of waypoints to follow, new physical objects can be added, and the weather can be changed using the scenario creator (Figure 1, bottom).

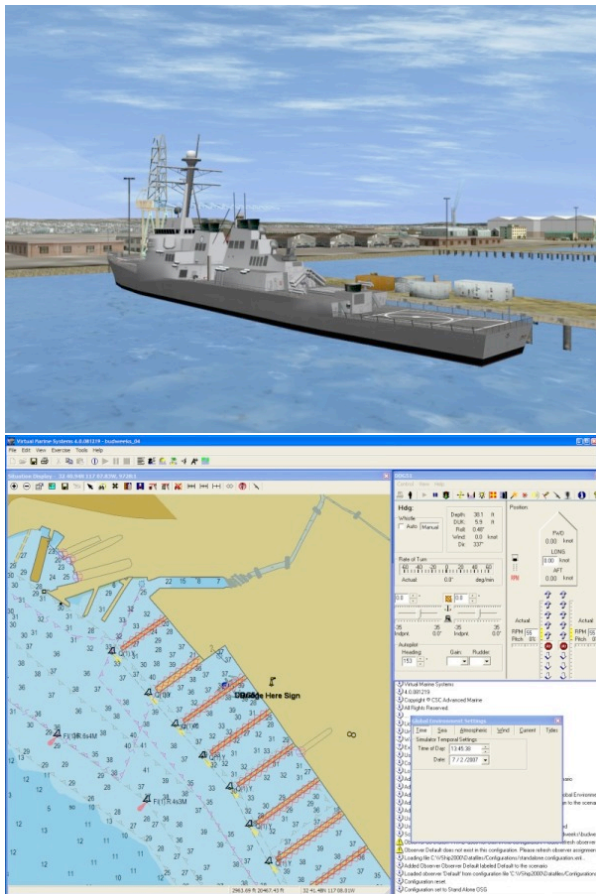


Figure 1.

The student interacts directly with the COVE simulation, issuing verbal commands, listening to responses and status reports, and viewing the environment and the ship under their command (known as “ownship”). The intelligent tutoring system adds two components into this dynamic – an intelligent tutor and expert model (Figure 2). The tutor

monitors student progress and compare the student’s actions with those of the expert model in order to provide feedback. The expert model needs to accomplish various navigation evolutions and inform the tutor as to what actions were taken and why.

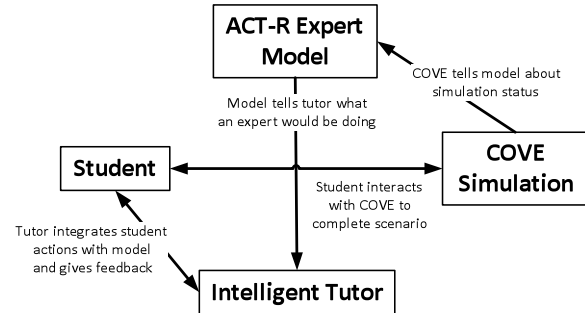


Figure 2.

This intelligent tutor/cognitive model system is designed to be implemented in the complex multi-monitor COVE simulators, which presented many challenges, including how vision is accomplished. The ACT-R vision component can only handle a single display, so a software solution was created to compensate for this shortfall and will be described in further detail later in the paper.

3. Ship Navigation Maneuvers

Several different ship navigation tasks varying in complexity were modeled. One basic evolution is intersecting a range, where a ship is transiting and must make a turn to intersect a new heading. While this may seem trivial, there is much skill in knowing when to begin the turn, how hard to take the turn, and when to ease off the engines and rudder. Another basic evolution is twisting a ship in a box, which involves rotating the ship on its pivot point without moving the ship forwards or backwards. This is difficult because students often do not have previous experience performing this kind of maneuver, and managing the engines and rudder so that the ship does not move laterally is a challenge.

Advanced navigation evolutions are also going to be modeled. To a great extent, these use more basic evolutions as building blocks (Rigeluth, 2007). For example, getting underway from the dock involves twisting the ship away from the pier, transiting forward and then making a turn to go out to sea. There is more to keep track of with these complex tasks, but they still use basic maneuvers at their core.

The intelligent tutoring system was not designed to replace the current ship-handling curriculum already in place. Instead, the system would augment training by supporting the scaffolding approach already taken by the course: begin by mastering simple maneuvers, then grow those into more complicated ones over the length of the training. The tutoring system supports both simple and complex maneuvers, so students can use the system throughout the course.

All of these tasks are difficult for students to grasp due to a number of factors. The hydrodynamics of maneuvering a ship can be difficult to understand, since students rarely have prior experience with ship-handling. Additionally, the tools available to affect the ship's speed and heading (rudder, port and starboard engines, and a tugboat in some cases) work differently when used in different combinations. Finally, there is often a lag between issuing a command (e.g., "All engines ahead full") and observing the effect of that command (e.g., increased speed), so a comprehension of cause-and-effect can take time to develop.

Another factor is that there are many paths to accomplish the same goal. One expert may attempt to increase the rate of ownship turn by increasing engine speed while another may instead decide to set the rudder farther over. Both options are correct, and it was important to capture all the possibilities. Also, different experts may teach their preferred method of accomplishing a task, increasing the necessity of the cognitive model and tutor to accommodate all the action paths available to the student. Finally, if the student deviates from a given parameter, the expert model must still function even if the action was not one of an "expert."

Implementing these ship navigation maneuvers in a cognitive model was difficult due to the perceptual nature of these maneuvers. Intersecting a range requires starting a turn, assessing speed through visual cues such as motion parallax (the apparent displacement of objects caused by a change in observer position), and lining up two separate range markers to ensure that the ship is in the ideal position in a harbor channel. These perceptual judgments often occur in the form of heuristics. An example heuristic used by baseball outfielders is that they will keep a constant visual angle between themselves and the ball instead of performing complex calculations (McBeath, Shaffer & Kaiser, 1995). These heuristics also apply to ship navigation and were implemented into a cognitive model. Determining how these strategies are

used was derived from a combination of expert interviews and observing ship-handling performance.

4. Expert Model Development

ACT-R was a natural choice of cognitive architecture due to the requirement of cognitive plausibility of the expert ship-handling model. Due to the necessity for the cognitive model to communicate with COVE and the intelligent tutor, the model was developed in Java-based jACT-R (Harrison, 2009) instead of Lisp-based ACT-R. Using Java increased compatibility with other system components and was more easily modified by those unfamiliar with Lisp, since Java is a more accessible language. jACT-R was designed to be as similar to ACT-R as possible, especially in terms of retaining the aspects of cognitive plausibility in the architecture.

The primary focus of developing a cognitive model for this project is the accurate modeling of several factors. One factor involves the possible actions that an expert could perform in order to maneuver the ship, and another is the perceptual monitoring and scanning behaviors that takes place to ensure successful completion of a navigation task.

4.1 Task Analysis Foundations for the Model

In order to develop a cognitive model of expert ship navigation, subject matter experts from the Naval Surface Warfare Officers School were consulted over multiple sessions. One phase of information collection involved watching students practicing using COVE and examining the feedback that instructors provided them. By analyzing the tone (positive or negative) and content of the feedback (pre-action advise or post-action critique), an understanding was developed of what aspects of ship-handling were emphasized and evaluated by human instructors. This influenced the development of the intelligent tutor as well as the expert model. For example, it became quickly apparent that the use of perceptual cues was critical to success. Also, a majority of the feedback came after an action was taken, so the student had to be allowed to make a mistake first.

Another phase of information collection centered around how course instructors performed various ship-handling maneuvers. Experts were interviewed and observed performing these tasks in the COVE simulator. These sessions were analyzed and distilled into cognitive task analyses. These took the form of a

traditional task analysis (which lists a sequence of observable tasks). Additionally, internal cognitive processes were also taken into account (Zachary, Ryder & Hicinbothom, 1999) and included.

The task analysis framework known as GOMS (goal, operator, method, selection; Card, Moran & Newell, 1983; John & Kieras, 1996) was selected for the task analysis because of the hierarchical nature of the tasks. There is a specific order as to which events happen, so representing tasks as a series of goals and sub-goals provided a great deal of benefit when translating these task analyses into cognitive models. For some navigation evolutions, GOMS-like task analyses were already completed (Grassi, 2000), so they were integrated into this project.

However, navigation maneuvers do not lend themselves perfectly to GOMS modeling. GOMS does not take into account the perceptual cues that are used in ship navigation. As an example, Figure 3 shows two range markers (the orange and white boards) that serve as a visual cue for ownship heading when they are lined up with the bow jackstaff. While GOMS is able to support a goal such as “Monitor speed until desired heading achieved,” there are a number of perceptual cues that indicate heading (such as range markers) that may be used in various combinations, and GOMS does not include a method for incorporating these visual cues.



Figure 3.

Due to this shortfall, a Critical Cue Inventory (CCI) was created to support a list of perceptual cue descriptions that could be used to accomplish a goal. The CCI could also include heuristics as to when a particular visual cue is more likely to be used, which

aided in building the expert cognitive model. An example truncated CCI used for determining the rate of swing of the bow can be found in Table 1.

Table 1.

Critical Cue Inventory for: Determine Rate of Swing of Bow	
CUE	DESCRIPTION
Jackstaff	Examine the rate of swing of the jackstaff compared to a fixed environmental object. Used when there is a physical landmark present.
Rate of Turn indicator	Interpret the Rate of Turn visual indicator in the COVE instrument cluster.
Change in heading	Determine how quickly the heading is changing over time using the various heading indicators. Used when there is a lack of landmarks.

4.2 Goal Stack Component

The cognitive model built in jACT-R used many standard components in ACT-R models, including the goal and retrieval buffers. Nonetheless, there are several noteworthy characteristics of the model that arose from project requirements. The first is the implementation of a goal stack that drives the entire execution of the cognitive model. Due to the hierarchical nature of navigation evolutions, it is only natural to create a chunk that can hold multiple goal levels in the goal buffer. Various productions push and pop goals from the stack, and the state of the goal stack is checked during the conflict resolution process to determine which production to fire next.

Certain steps in a navigation evolution must occur at a specific time, and properly utilizing the goal stack assisted with this need. Knowing when a turn is complete, for example, requires monitoring ownship heading or lining up the jackstaff with an environmental object. The production to stop the turn (i.e., “approaching-heading-NOW-stop-turn”) needs to fire when the goal stack matches specific conditions. The bottom of the goal-stack needs to match the basic goal of “make-turn,” and a goal at the top needs to match the goal of “monitor-until-desired-heading-reached.” Once these conditions are met, the production “pops,” or removes, the old goal from the stack and “pushes” on a new goal which, presumably, would stop the turn by shifting the rudder or slowing

the engines.

This implementation aids in creating generic productions that are needed by many different higher-level goals and may occur multiple times throughout an evolution (e.g., “activate-rudder”). The entire goal stack does not need verification – instead, the production only needs to check the top goal. None of the other goals below need to be checked – for example, it does not matter whether a lower-level goal is “make-turn” or “twist-ship.” Either way, the rudder must be activated. Another advantage from being able to check specific goals within the goal stack is that multiple possible action paths to complete a task are easily implemented.

Figure 4 contains a jACT-R pseudocode example that checks the goal stack. The top production is generic and may be called multiple times in an evolution. This production also does not need to verify the entire goal stack. The bottom production must occur at a specific time and checks each level of the goal stack.

```

<!-- This generic production only needs to
ensure the top goal is to issue the engine order -->
<production name="issue-starboard-engine-order">
  <conditions>
    <match buffer="goal" >
      <slot name="goal-2" equals="issue-starboard-
engine-order"/>
  ...

<!-- This specific production ensures the top goal
matches the desired goal and the other goals also
match (or are clear) -->
<production name="monitor-speed-heading-until-
turn-time">
  <conditions>
    <match buffer="goal" >
      <slot name="goal-1" equals="ownship-ahead"/>
      <slot name="goal-2" equals="monitor-until-turn-
time"/>
      <slot name="goal-3" equals="clear"/>
      <slot name="goal-4" equals="clear"/>
  ...

```

Figure 4.

4.3 Use of Perceptual Cues

There are many visual cues in the environment that an expert uses to properly execute ship maneuvers. It was necessary to pull this information from the COVE simulation directly instead of going through the jACT-R vision module, but it was critical to maintain cognitive plausibility for vision in the model, so the software pulling information from the COVE simulation must act “behind the scenes” to fill a

jACT-R buffer that is accessible to the model. Even a basic subgoal, such as visually scanning to assess ship status (speed, heading, etc.) required accurate cognitive modeling. Experts will often alternate between paying attention to the environment and to the ship status indicators while executing a navigation evolution, and cycling between these objects takes place frequently. This behavior was assessed in experts through interviews and head-tracking within the COVE system.

This scanning behavior was inserted into the model so the expert model’s current awareness of the situation correctly reflects the experience of a human expert. From this, the intelligent tutor can detect if the student is exhibiting similar scanning behavior. If this was not the case, the tutor can issue prompts to the student to check speed, heading, rudder status, and other important parameters.

Another example that demonstrates the criticality of the vision system is the monitoring of specific perceptual thresholds (e.g., to know when to begin and end turns, when ownship is far enough away or close enough to the dock, etc.). Experts do not intently stare at one location in the environment waiting for this threshold to be passed, nor are they able to focus on more than one area at once. Instead, scanning behavior is used (again derived from interviews and head-tracking), and the expert model needed to accurately capture this behavior.

The standard vision module within jACT-R is able to gather visual information from a display using attentional and imagery constructs, and locations are represented by their x- and y-positional screen coordinates. This vision scheme has been implemented successfully in heavily perceptual tasks such as driving (Salvucci, Boer & Liu, 2001). However, the COVE simulation is too complex to use the relatively basic jACT-R vision module. This is because the visual scene is distributed amongst multiple monitors and computers, which the vision module cannot handle. Instead, information must be passed directly from the simulation to the Java core of jACT-R using a client-server model. This information is then inserted into a buffer created for this model.

The solution to this problem was to implement “vision” through external software. COVE generates the rendered environment and keeps track of some environmental objects (e.g., piers), the environmental conditions (e.g., current and wind speed), and the

status of all the ships (e.g., speed and course of ownship, tugs, etc.). The simulation does not keep track of objects such as buoys, and the locations of those objects had to be measured manually and inserted into a separate database. Together, these components possessed the information that the expert cognitive model would otherwise try and obtain through more traditional means of vision. It was more efficient and easier to retrieve the necessary information about the environment directly from COVE instead of attempting to adapt the jACT-R vision module.

A technical necessity for the entire tutoring system was the separation of various system components. The COVE simulation needed to remain a separate entity. While the expert model is an important piece of the intelligent tutor, the tutor itself also needed the option to run as a standalone component. Due to the need for separation between each system component, software bridges were built to interface between the COVE software, the Java core of jACT-R, and the intelligent tutor. The bridge between COVE and jACT-R requests information from COVE and then fills a custom jACT-R buffer that is accessible by the cognitive model. This buffer was programmed as an Eclipse IDE plug-in and imported into jACT-R.

A simple example will help illustrate this process. In the case of monitoring ownship speed, a human expert would look down at a console (on a monitor separate from the rendered environment) and read off the speed. For the cognitive model to do this, it would request the speed from the COVE-ship-state buffer (similar to requesting a particular chunk from the retrieval buffer) first. This buffer is refreshed by the COVE/Java bridge, which periodically queries the COVE software as to the state of the simulation, which includes ownship speed information.

This software bridge allows for the simulation of many of the visual cues utilized by human experts but is pulled directly from the simulation. Therefore, the perceptual cues and heuristics used by human experts are still present in the cognitive model because the software bridge is abstracted away from the model. This abstraction allows for the ability of the model to accomplish something akin to traditional vision in a cognitively plausible manner.

4.4 Cognitive Plausibility

Defining cognitive plausibility for this expert

cognitive model was different from many other models. The purpose of the expert model within the tutoring system is to act as an “answer key” to compare against student actions. Therefore, the expert is not supposed to commit errors. For example, there is no need to learn new actions, nor is there need for millisecond accuracy in cognitive function. Also, memory decay was not implemented. Instead, the expert model implemented visual scanning behavior between the environment and ownship status indicators to refresh memory. This reflects student behavior because they are often told not to trust their own memory.

A plausible expert, especially one that is used as a yardstick to measure human students against, should always perform an evolution as flawlessly as possible. However, an expert model that is part of a tutoring system must also be able to adapt to student behaviors, which are not always optimal. The first iteration in creating a model for any ship-handling evolution represented optimal performance of a maneuver. Once this was achieved, multiple action paths were built out from this single path. For example, the expert model knows the optimal distance from a range in which to make a turn, and this behavior is the model default. If the student overshoots this range, the expert model was designed to compensate for the error. This action path may result in using a greater amount of rudder than is typically called for. While suboptimal, it was important that the model possess these behaviors both for some degree of cognitive plausibility and to be a useful components of the intelligent tutor.

One area where it was especially important to maintain cognitive plausibility was in visual scanning behavior. If a computer program was written that did not take into account the limits of human cognition, then a student would be compared to a computer instead of a simulated human expert. While a computer could monitor multiple information streams at once, this would not reflect human cognition. Instead, a reflection of expert human behavior would require scanning multiple sources of data in a serial manner.

5. Empirical Validation

Traditional validation of cognitive models seeks to match human performance with that of the model, typically on a temporal scale. For example, an accurate cognitive model of visual search should generate target detection times that are similar to

human reaction times. Here, we are attempting to match the perceptual actions of an expert instead. The actions taken by the model do need to occur with some degree of temporal accuracy, but the millisecond modeling accuracy of jACT-R is not necessary for this application.

An important first step of validation involves demonstrating a complete system to the instructors who will be using the system for instruction. This has already been done with the standalone intelligent tutor system, which only contained rudimentary knowledge about ship-handling (e.g., maximum limits on speed). The system was well-received by instructors, who noted that feedback on perceptual components of the task would add to the utility of the final product.

Further validation steps have not occurred but are currently being planned. One important step in this model validation plan will examine how the entire tutoring system performs when a novice student uses the simulation. This will be tested on actual students taking a ship-handling course, but can also be tested on non-student novices. The critical data to collect from these experiments is the performance of the expert model. This is to ensure that the model was able to traverse the multiple action paths in response to student performance. For example, if a novice stops a turn too late, the model should react by shifting the rudder in the opposite direction. If the model had direct control of the ship, this mistake should not have happened in the first place. However, the model does not have control and must compensate for many errors that a novice can make. This will require many novices and hours of testing, but will serve to make a more robust model.

A critical final test of the intelligent tutor and expert model system is to determine how training is improved with use of this system. Improvement will be measured along several factors, including performance in ship maneuvering (measured across several variables such as time to completion and deviation from optimal channel position), amount of training retention, and number of human instructor hours required during training. The hope is that the tutor can increase the number of students that a single instructor can supervise while maintaining the same level of (or improving) training effectiveness.

6. Conclusions

While there have been other projects that have

integrated a cognitive model into a larger framework, these have mostly focused on training applications by creating a simulated teammate to work with other humans (Scolaro & Santarelli, 2002; Ball, et al., 2009). The project described here also works within a larger framework, but not in a team context. Instead, the model represents a single expert that changes its behavior in direct response to student actions.

This project presents a unique application of the jACT-R cognitive architecture in many ways. The requirements for the project necessitated the development of an expert cognitive model that needed to balance cognitive plausibility with near-flawless expert performance, perceptual heuristics without actual vision, and multiple action paths with an emphasis on tutoring. As intelligent tutoring systems are becoming increasingly popular, it is important to understand how cognitive modeling can add to these systems in a useful way.

While an expert model “answer key” cannot make mistakes such as memory retrieval failures, the model must compensate for student errors in order to remain useful. This required a far more extensive knowledge gathering period in order to explore task performance more fully, and also requires a greater testing period to ensure that many practical possibilities for behaviors that deviate from the optimum are accounted for.

Intelligent tutoring systems are often used in complex environments, which requires ACT-R models to perceive information that cannot be retrieved through the current, primitive vision module. Instead, software bridges must interface between the simulation and ACT-R itself, but the simulation environment must remain abstracted away from the model to maintain cognitive plausibility. Overall, cognitive modeling has a great deal to offer intelligent tutoring systems, and an optimal methodology to create these models is currently being shaped.

7. References

- Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* New York: Oxford University Press.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4), 1036-1060.
- Ball, J., Myers, C., Heiberg, A., Cooke, N., Matessa,

- M., & Freiman, M. (2009). The Synthetic Teammate Project. *18th Conference on Behavior Representation in Modeling and Simulation*, Sundance, UT.
- Bratt, E. O., Schultz, K., & Peters, S. (2007). Challenges in Interpreting Spoken Military Commands and Tutoring Session Responses. *Grammar Engineering Across Frameworks 2007*, Stanford, CA.
- Card, S., Moran, T., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Computer Sciences Corporation. (2009). Virtual Ship [Computer Software]. Retrieved December 1, 2009.
- Grassi, C. R. (2000). A task analysis of pier-side ship-handling for virtual environment ship-handling simulator scenario development. Masters Thesis at the Naval Postgraduate School.
- Harrison, A. M. (2009). jACT-R [Computer Software]. Retrieved December 1, 2009, from <http://jactr.org/>.
- John, B. E., & Kieras, D. E. (1996). The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transactions on Computer-Human Interaction*, 3(4), 320-351.
- McBeath, M. K., Shaffer, D. M., & Kaiser, M. K. (1995). How baseball outfielders determine where to run to catch fly balls. *Science*, 268(5210), 569-573.
- Reigeluth, C.M. (2007). Order, first step to mastery: An introduction to sequencing in instructional design. In F. Ritter, J. Nerb, E. Lehtinen, & T. O'Shea (Eds.), *In Order to Learn: How the Sequence of Topics Influences Learning*. New York, NY: Oxford University Press.
- Salvucci, D. D., Boer, E. R., & Liu, A. (2001). Toward an Integrated Model of Driver Behavior in a Cognitive Architecture. *Transportation Research Record*, 1779, 9-16.
- Scolaro, J., & Santarelli, T. (2002). Cognitive Modeling Teamwork, Taskwork, and Instructional Behavior in Synthetic Teammates. *11th Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL.
- Smallman, H. S., & St. John, M. (2005). Naive Realism: Misplaced Faith in Realistic Displays. *Ergonomics in Design*, 13(3), 6-13.
- Zachary, W. W., Ryder, J. M., & Hicinbothom, J. H. (1999). Building Cognitive Task Analyses and Models of a Decision-making Team in a Complex Real-time Environment. In J. M. Schraagen, S. F. Chipman, & V. J. Shalin (Eds.), *Cognitive Task Analysis* (pp. 365-383). Mahwah, NJ: Lawrence Erlbaum Press.

Acknowledgments

The Office of Naval Research program of Training, Education, and Human Performance sponsored this research. Thanks to Alden Daley, Jia Huang, and Anthony Harrison for jACT-R and coding support.

Author Biographies

DR. JASON H. WONG is a Human Factors Scientist with the Naval Undersea Warfare Center. He received his Ph.D. in 2009 from George Mason University and applies his research to projects that integrate cognitive theory and system design. He examines the human-computer interaction aspects of complex systems, creates improved submariner training methodologies, and develops cognitive models to simulate human performance.

DR. SUSAN S. KIRSCHENBAUM is an Engineering Psychologist in the Combat Systems Department of the Naval Undersea Warfare Center Division, Newport, Rhode Island and the Human Systems Integration (HSI) Lead for NUWCDIVNPT. She has continuing interests in expertise and in the effects of uncertainty and other information variables on decision making. Applications for her research are in the design of systems for decision support, human-computer interaction, and display design.