33 Cognitive Modeling for Cognitive Engineering

Matthew L. Bolton and Wayne D. Gray

33.1 Introduction

Cognitive engineering is the application of cognitive science to human factors and systems engineering. When cognitive models are used for this purpose, the predictive or explanatory power of the model is used to improve engineering system performance. Cognitive models can be used at any stage of the engineering life cycle (Figure 33.1). They can be part of the *analysis* of an existing system to identify when human cognition is contributing to problems and establish requirements for new systems. Cognitive models can be used in *design* to produce system elements (human–machine interfaces, system behaviors, or training requirements) that are compatible with human cognition. Cognitive models can be incorporated into an actual system's *implementation* to provide humans with training and/or decision support. Cognitive models can inform system *testing and evaluation* by identifying cognitive conditions worthy of deeper analysis. Furthermore, model-based generation can create tests to ensure that all cognitively relevant system conditions are observed. Finally, cognitive models that were part of implementation can be used during a system's *operation and maintenance*.

Thus, while the cognitive models and architectures commonly associated with cognitive engineering [ACT-R (Anderson, 1993), EPIC (Kieras & Meyer, 1997), Soar (Newell, 1990), and QN-MHP (Liu, Feyen, & Tsimhoni, 2006)] were created to understand human behavior, their use and development in engineering has been done with the purpose of realizing better systems. This means cognitive engineering models are not strictly concerned with understanding the cognitive mechanisms underlying behavior (the emphasis of cognitive science) unless that understanding has utility for engineering goals. Fortuitously, the explainability of cognitive models does have value because it enables systems, analysts, and users to understand why behavior is occurring and use this to inform response and design.

Gray (2008) identified five key differences between cognitive science and engineering: (1) Cognitive engineers pick the problems they address (system performance, safety, workload, usability, financial impact, trust in automation, etc.) because there is an operational need, not because they are necessarily scientifically interesting. (2) Because of operational need, cognitive engineers often work in emerging domains or those where there has been little prior study

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

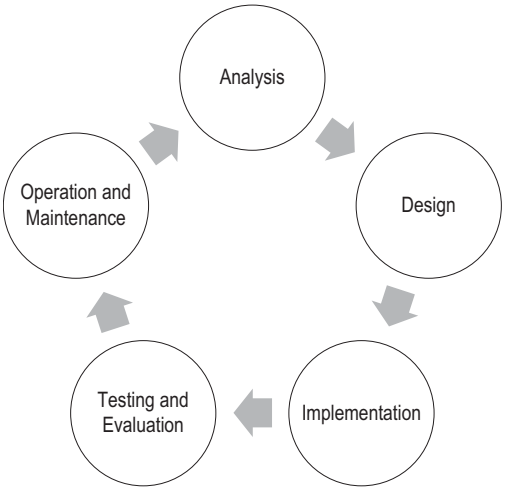

Figure 33.1 *The engineering life cycle.*

(like autonomous driving or wearable computing); not the well-trodden applications common to cognitive science. (3) This means that cognitive engineering modelers must rely on domain experts (i.e., subject matter experts or practitioners) to supply information when there is a lack of historical data or cognitive theory. (4) For cognitive engineers, model utility is prioritized over its ability to provide a depth of insight into the represented phenomenon. Finally, (5) cognitive engineers are typically responsible for predicting human performance as part of a complex system. As such, a significant amount of cognitive engineering is focused on capturing the control of integrated cognitive systems in their models (Gray, 2007). In this situation, any lower level cognitive constructs (like memory or categorization) are included in service of accomplishing and/or explaining the control.

These distinctions produce an environment where model fidelity varies based on application goals. This chapter provides readers with a history of cognitive modeling in cognitive engineering and its diverse contributions. It first reviews the seminal work of Card, Moran, and Newell (1983), which laid the foundations for many developments. Then, to give readers a sense of the issues facing contemporary cognitive engineering, the chapter examines the use of cognitive models in complex systems. The chapter concludes with a summary and a discussion of potential threats and future advances.

33.2 Initial Approaches to Cognitive Modeling for Cognitive Engineering

Attempts to apply computational and mathematical modeling techniques in human factors and systems engineering have a history (see Byrne, 2007; Kieras, 2007; Pew, 2007) that is beyond the scope of this chapter. This

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

section focuses on the seminal work of Card, Moran, and Newell on GOMS (goals, operators, methods, and selection rules). GOMS is a task-analytic framework for modeling human information-processing, behavior, and performance. GOMS is based on the human's (a) goals, the (b) operators (low-level perceptual, motor, or cognitive acts) needed to accomplish the goals, sequences of operators and sub-goals that constitute (c) methods for accomplishing a goal, and (d) selection rules for choosing methods.

Most cognitive science researchers were trained in experimental psychology. This tradition focuses on discovering truths about the natural world with large, controlled studies. People with this background often cannot conceive of how someone could model something as complex as driving or unmanned aerial vehicle (UAV) operation.

Such developments are possible because most human behavior can be modeled as a hierarchy of tasks and subtasks (Kirwan & Ainsworth, 1992; Simon, 1996, chapter 8). The structure of this hierarchy is generally determined by the task environment, rather than the human operator. As such, cognitive engineers can break behavior down to the level required by analysis goals. This *task analysis* works well for designing complex industrial operations and procedures for human tasks (Kirwan & Ainsworth, 1992; Shepherd, 1998, 2001). For those interested in interactive systems, task analyses can be straightforward because most human behavior is produced in direct response to changes in the environment. Although interactive behavior is complex, the complexity lies in (a) evaluating the current state of the environment; (b) deciding what can be done to advance user goals; (c) evaluating strategies for accomplishing these goals; and (d) executing the strategy. The key to this cycle is the *unit task*.

33.2.1 The Unit Task as a Control Construct for Human Interactive Behavior

Card, Moran, and Newell's conceptual breakthrough was that most tasks were composed from smaller "*unit tasks* within which behavior is highly integrated and between which dependencies are minimal. This quasi-independence of unit tasks means that their effects are approximately additive" (Card et al., 1983, p. 313). Thus, the "unit task is fundamentally a control construct, not a task construct" (Card et al., 1983, p. 386). The unit task is not given by the task environment, but results from the interaction of the task structure with the control problems faced by the user.

The prototypical unit task example (Card et al., 1983, chapter 11) is the structure imposed on a typist during transcription. The transcription task environment consists of dictated speech, a word processor, and a foot pedal that controls recording playback. As speech is typically faster than skilled typing, the basic problem faced by typists is how much of the recording to listen to before pausing. Efficient typists listen while typing. The longer typists listen, the greater the lag between what is heard and what is typed. At some point, typists will pause the recording and type until they cannot confidently

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

remember more of the recording. With experience, a skilled typist will minimize the amount of rewind and replay while maximizing the amount typed per unit task. This "chopping up" of the task environment into unit tasks reflects a control process that adjusts performance to task characteristics (dictation speed and speech clarity), the typist's skill (words per minute), and to the typist's cognitive, perceptual, and motor limits.

33.2.2 The Path from Unit Tasks, Through Interactive Routines, to Embodiment

Table 33.1 shows a typical GOMS unit task using Natural GOMS Language (NGOMSL). This is one of approximately twenty needed to model Lovett's and Anderson's (1996) building sticks task: a game whose objective is to match the length of a stick by building a new one from pieces of various sizes. This unit task would be invoked to subtract length from the built stick when it was larger than the target. This example (Table 33.1) shows that each line/statement has an execution overhead (statement time; Stmt Time) of 0.1 seconds (s). There are three operator types used: a point operator (P) that is assumed to have a time of 1.1 s; a button click (BB; up and down) with duration 0.2 s; and a mental operator (M) with duration 1.2 s. The entire method for accomplishing this unit task lasts 5.8 s.

As the table suggests, NGOMSL (Kieras, 1997) reduces all operators to one of a small set. The duration of each operator is based on empirical data or mathematical models (such as Fitts' Law or Hick's Law). Much of what goes into an NGOMSL analysis comes from the second chapter of Card et al. (1983), which casts many regularities gleaned from experimental psychology into a form that has utility for engineers.

GOMS was intended as a tool for cognitive engineering. Hence, each line of the NGOMSL analysis could be more precise and tailored based on factors

Table 33.1 *Example unit task for the "building sticks task" using natural GOMS language (NGOMSL; Kieras, 1997)*

Step	Description	Stmt time (s)	Op	# Ops	Op time	Total time (s)
Method for goal: Subtract stick<position>		0.1				0.1
Step 1	Point to stick<position>	0.1	P	1	1.1	1.2
Step 2	Mouse click stick<position>	0.1	BB	1	0.2	0.3
Step 3	Confirm: Stick is now black	0.1	M	1	1.2	1.3
Step 4	Point to inside of "your stick"	0.1	P	1	1.1	1.2
Step 6	Click mouse	0.1	BB	1	0.2	0.3
Step 7	Confirm: Change in stick size	0.1	M	1	1.2	1.3
Step 8	Return with goal accomplished	0.1				0.1
					Overall Time (s):	5.8

Abbreviations: Stmt time = statement time; Op = operator; P = point operator; BB = button click; M = mental operator.

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

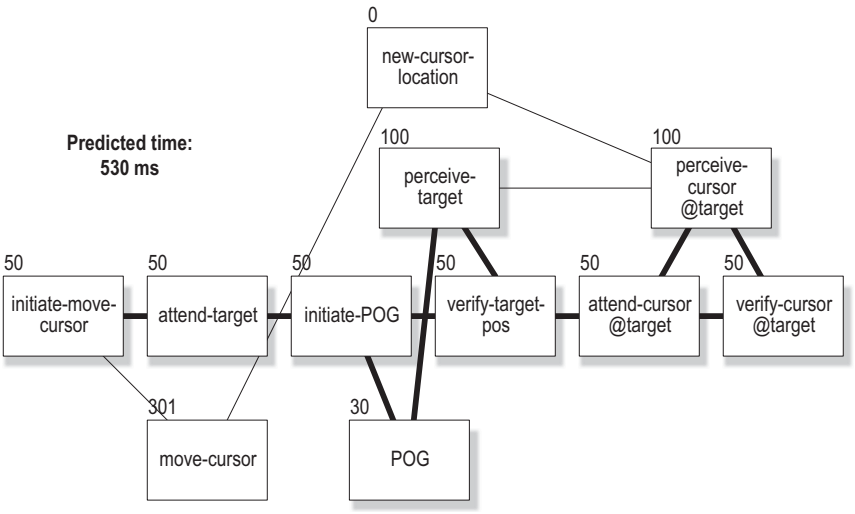

Figure 33.2 *A CPM-GOMS model of an interactive routine (Gray & Boehm-Davis, 2000), which could be instantiated as Steps 1 and 4 from Table 33.1. It shows the cognitive, perceptual, and motor operations required to move a mouse to a predetermined computer screen location. Total predicted time is 530 milliseconds (ms). The middle row shows cognitive operators with a default execution time of 50 ms each. Above that are the perceptual operators. Below it are the motor operators. Operators flow from left-to-right with lines indicating dependencies. Within an operator type, dependencies are sequential. However, between operator types, dependencies may be parallel. Numbers above each operator indicate its execution time in ms. Time accumulates from left-to-right along the critical path (bold lines connecting shadowed boxes).*

such as the exact distance moved. However, the granularity of GOMS analyses in Table 33.1 is too gross for some purposes. Indeed, to model transcription typing, John (1996) developed a version of GOMS that went to a lower level. John (1988) represented the dependencies between cognitive, perceptual, and motor operations during task performance (see Figure 33.2) in an activity network formalism (Schweickert, Fisher, & Proctor, 2003) that allowed for the computation of critical paths. This variant is called CPM-GOMS, where CPM has a double meaning as both critical path method and cognitive, perceptual, and motor operations.

The power of this representation was demonstrated through its ability to predict performance times for telephone Toll and Assistance Operators (TAOs; Gray, John, & Atwood, 1993). CPM-GOMS models predict the counterintuitive finding that TAOs using a proposed new workstation would perform more slowly than those who used the older workstations. After a field trial confirmed this prediction, the models provided a diagnosis, in terms of the procedures imposed by workstations on the TAO, as to why newer, faster technologies could perform more slowly than older ones.

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

33.2.3 The Legacy of Card, Moran, and Newell

GOMS and CPM-GOMS made several things obvious. First is the basic insight offered by the unit task; namely, that functional units of behavior resulted from an interaction between: the task being performed; detailed elements of the task environment's design; and limits of human cognitive, perceptual, and motor operations. Second, the notation of CPM-GOMS made it very clear that all human behavior was embodied behavior. Indeed, the mechanistic representations of CPM-GOMS were very compatible with the views of embodiment expressed by modelers such as Ballard (Ballard & Sprague, 2007) and Kieras (Kieras & Meyer, 1997). Third, whereas GOMS and NGOMSL (Kieras, 1997) emphasized control of cognition, CPM-GOMS provided a representation that showed that this control was far from linear, but entailed a complex interleaving of various parallel activities.

Since the nineties, many of the insights of CPM-GOMS have become standard among modelers and accelerated cognitive engineering progress. Researchers built GOMS-inspired hierarchical task modeling formalisms with increased expressive power for capturing nondeterminism in human behavior, representing different elements of cognition, and supporting different engineering efforts (see for example ConcurTaskTrees (CTT; Paternò et al., 1997), Enhanced Operator Function Model (EOFM; Bolton et al., 2011), HAMSTERS (Fahssi, Martinie, & Palanque, 2015), Work Models that Compute (Pritchett et al., 2014), and GOMS-HRA (Boring & Rasmussen, 2016)). Kieras and Myers built the EPIC cognitive architecture (Kieras & Meyer, 1997), by expanding Kieras' parsimonious production system (Bovair, Kieras, & Polson, 1990; Kieras & Bovair, 1986) to include separate modules for motor movement, eye movements, and so on. ACT-R (Anderson, 1993) has added a module for visual attention (Anderson, Matessa, & Lebiere, 1997), experimented with EPIC's modules (Byrne & Anderson, 1998), and completely restructured itself so that all cognitive activity (not simply that which required interactive behavior) entailed puts and calls to a modular mind (Anderson et al., 2004). During the same period, Ballard's notions of embodiment took literal form in Walter – a virtual human who could follow a path while avoiding obstacles, picking up trash, and stopping to check traffic before he crossed the street (Ballard & Sprague, 2007).

33.3 Computational Cognitive Modeling for Engineering Complex Systems

Cognitive modeling has shown significant utility in engineering, particularly for complex systems. A system is complex if it is composed of multiple interacting components (including human operators) that must work together to achieve system goals. In such systems, so called "human error," where a human diverges from a normative plan of action (Hollnagel, 1993), is regularly cited as a source of failure or system instability (Reason, 1990; Sheridan &

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

Parasuraman, 2005). Human error in medicine contributes to 251,000 deaths a year (Makary & Daniel, 2016); approximately 50 percent of commercial aviation and 75 percent of general aviation accidents (Kebabjian, 2016; Kenny, 2015); a third of UAV incidents (Manning et al., 2004); roughly 90 percent of automobile crashes (NHTSA, 2008); and high profile disasters like the catastrophe at Three Mile Island (Le Bot, 2004), often due to poorly designed human interaction (Bainbridge, 1983). This is an extremely topical area because engineered systems continue to become more complex and automated, often with little regard for the capabilities, cognitive limitations, or well-practiced experience of human operators (Strauch, 2017).

With this perspective, cognitive engineers attempt to analyze, design, and evaluate systems from a human-centered perspective: giving humans the information and controls they need to fulfill their role in the system safely and effectively. For engineers in the cognitive modeling space, this means using cognitive models to understand the demands on human cognitive, perceptual, and action resources during system operations, discover potentially dangerous operating conditions, and inform designs that will address or avoid problems and facilitate human performance. In fact, model-based analyses offered by cognitive models are particularly advantageous in complex systems for several reasons. First, many complex domains are safety critical, where it can be dangerous to evaluate human behavior in actual operational environments. Cognitive-model-based analyses can provide deep insights into human performance without the need for running the system in dangerous situations. Second, system failures are relatively uncommon and may be difficult or impossible to anticipate. Cognitive-model-based analyses can help engineers reduce the likelihood of human source of variability. They can also explore a system's state-space to discover previously unforeseen operating conditions. Finally, human subject experiments and testing are expensive, time consuming, and incomplete. Cognitive-model-based analyses can be performed without human participants or identify specific areas where human testing is necessary. This can lead to faster, more cost effective, and more complete engineering efforts.

The following discuss contemporary developments in cognitive models in complex systems engineering. It starts with a description of cognitive-architecture-based simulation advances before looking at the more applied applications of cognitive models in "formal methods" analyses.

33.3.1 Cognitive Architectures

Cognitive architectures offer frameworks around which to model human cognition and behavior computationally. In cognitive engineering, this is typically implemented based on the way that humans learn, store, and execute "if-then" production rules. These are typically used in simulation-based analyses where the model represents simulated humans in a simulated or real operational environment. The performance of the simulation is used in engineering analyses and evaluations. The cognitive portion of the model serves to explain human

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

behavior and/or a cognitive dimension of the behavior. There is a long history of cognitive-architecture-based models. Recent developments have focused on incorporating elements of visual and auditory perception (Kieras, Wakefield, Thompson, Iyer, & Simpson, 2016). The following sections describe several complex system areas where cognitive-architecture-based models have been advancing both cognitive science and cognitive engineering.

33.3.1.1 Unmanned Air Vehicles

An important challenge for cognitive engineering is the design of new systems, especially those that create new human operator roles. One such system is the UAV. UAVs are increasingly used by the defense, intelligence, and civilian organizations in contexts from piloting to package delivery.

Remotely piloting a slow-moving aircraft while searching for ground targets is difficult for even experienced pilots. A complete model that could take off, perform missions, interact with teammates, and return safely would entail the detailed integration of most, if not all, functional subsystems studied by cognitive scientists and raise challenges in the control of integrated cognitive systems. Such a complete system is beyond the current state-of-the-art. However, partial systems can be useful in determining limits of human performance and identifying strategies that work. This is the approach proposed by Gluck et al. (2005) and Ball et al. (2010), who outlined how a "synthetic teammate" for UAV ground control training could be realized using ACT-R. Since its proposal, this effort has produced what might be the largest and most complicated cognitive model ever created. Gluck, Ball, & Krusmark, (2007) advanced this approach by building partial models to study the challenges of human UAV pilots. These researchers modeled two alternative strategies, one based on a simple control strategy and the other based on what is taught to pilots. They showed that the simple one would not meet UAV performance demands and that actual human performance data suggested that the best pilots used the strategies from the best performing model. More recently, Rodgers, Myers, Ball, & Freiman (2011) have been exploring how to account for situation awareness in the synthetic teammate by integrating linguistic inputs, the context of discourse, the task process, and the model's knowledge in a new situation component. Demir et al. (2015, 2016) also advanced this approach by accounting for human–human communication and coordination. In this, language comprehension, language generation, dialog modelling, situation modelling, and agent-environment interaction components are ultimately used to communicate (textually) using common patterns from the work domain.

33.3.1.2 Driving Under Different Levels of Autonomy

Driving inherently occurs as part of a complex system that involves a dynamic environment, multiple vehicles, and multiple drivers. It is also cognitively demanding in that it requires the integration and interleaving of basic tasks

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

related to control for stable driving, tactical behavior for interacting with the dynamic environment, and strategic processes for planning (Salvucci, 2006).

Salvucci and colleagues (Salvucci, 2006; Salvucci & Gray, 2004; Salvucci & Macuga, 2002) did foundational work modeling human cognition (in ACT-R) while driving. Ultimately, Salvucci (2006) compared the models with human behavior on several dependent variables related to lane keeping, curve negotiation, and lane changing using simulations. The dependent variables included performance-based measures such as steering angle, lateral position, and eye data related to visual attention.

In recent years, the driving domain has been made more complex with varying levels of automobile autonomy being introduced or planned for near-term deployment. This creates many new potentially cognitively demanding situations for human drivers who must now monitor the environment and the automation and be prepared to take control at any time. Not surprisingly, driver modeling has been the subject of many contemporary advances. For example, Rehman, Cao, and MacGregor (2019) determined how to model driver situation awareness into the Queueing Network variant of ACT-R using dynamic visual sampling to simulate realistic patterns of driver attention allocation. Rhie et al. (2018) used the queueing-network-based architecture to account for oculomotor behaviors that include things like reaction time and movement patterns to understand the level of human information processing. Similarly, Jeong and Liu (2017) used queueing-based models to predict eye glances and workload for secondary stimulus response tasks (related to auditory-manual, auditory-speech, visual-manual, and visual-speech modalities) humans perform while driving. In all cases, simulated model behavior was validated against actual human data. Finally, Mirman, Curry, and Mirman (2019) used computational cognitive modeling to show that population changes in driver crash rates (post licensing) are consistent with sudden, nonincremental decreases in individual crash risks. Mirman (2019) used these findings to formulate a new theory of driver behavior based on dynamical systems principles, the so-called phase transition framework, to explain and do research on this and similar phenomena.

33.3.2 Formal Methods for Human Interaction with Complex Systems

The cognitive-architecture-based analyses discussed above all use simulation for their analyses. These can have very high-fidelity, predictive models. However, they can miss critical conditions that could be the source of system failures. Recent developments have shown that these limitations can be overcome by using cognitive models with formal methods. Specifically, the complexity of many modern systems can make it extremely difficult for designers to determine how humans will interact with system elements, how erroneous behavior can occur, how these can cause failures, and how to design-away problems. Formal methods are tools and techniques that have grown out of computer science for mathematically modeling, specifying, and proving properties about (formally

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

verifying) systems (Wing, 1990). The formal models mathematically describe the behavior of the target system. Specification properties describe conditions that should always be true in the system. Formal verification is the process of mathematically proving if the formal model satisfies the specifications. There are many different ways of using formal methods. These run the gamut from pen and paper proofs to automated processes. For example, model checking is a fully automated, computer-software-based approach (Clarke et al., 1999). In this, the target system is formally modeled as a state machine: variables whose values indicate state and transition between states occur based on inputs and/or the current state. Specification properties logically assert desirable system conditions (such as the lack of an unsafe condition) using modal logic (such as temporal logic; Emerson, 1990). During formal verification, the model checker exhaustively searches the formal model's statespace. If no violation is found, the model checker has proven that the model satisfies the specification. If a violation is found, the model checker returns a counterexample, a trace through the model's statespace that explicitly proves why the specification is not true.

Formal methods are mostly used in computer hardware and software engineering (Wing, 1990). Because they are adept at finding problems that arise from interactions between components in complex systems, researchers have been exploring how they can be used for human interactive systems (Bolton, 2017a; Bolton, Bass, & Siminiceanu, 2013; Degani, 2004; Weyers, Bowen, Dix, & Palanque, 2017; Wu, Rothrock, & Bolton, 2019). Most topical is the work that has integrated models based on human task behavior and cognitive architectures with larger system models to use formal methods to improve system reliability and safety.

33.3.2.1 Task-Model-Based Approaches

Many of the GOMS-inspired task models are composed of hierarchies of goal-based activities that decompose down to atomic actions. These can be represented using discrete, tree-like graphs and thus readily interpreted as state machines or process algebras, enabling their use in larger system models and formal methods analyses of safety. For analyses focused on normative behavior, formal proofs can determine whether a system will always perform safely and enable humans to complete their goals based on how people actually behave (as determined by a task analysis) or are expected to behave (based on training or manuals) (Abbate & Bass, 2015; Aït-Ameur & Baron, 2006; Basnyat, Palanque, Bernhaupt, & Poupart, 2008; Basnyat, Palanque, Schupp, & Wright, 2007; Bolton & Bass, 2010; Bolton, Siminiceanu, & Bass, 2011; Degani, Heymann, & Shafto, 1999; Paternò & Santoro, 2001). These techniques are powerful, but can miss the impact of erroneous acts. Other researchers have determined how to allow experts to manually include specific human errors into normative tasks using mutation patterns (Bastide & Basnyat, 2007; Fields, 2001). Finally, researchers can automatically generate human errors using systematic deviations from normative tasks based on human error genotypes

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

(errors are classified based on cognitive causes) and/or phenotypes (errors are classified based on observable deviations from a normative plan) (Barbosa, Paiva, & Campos, 2011; Bolton, 2015; Bolton & Bass, 2013; Bolton, Bass, & Siminiceanu, 2012; Pan & Bolton, 2018).

33.3.2.1.1 An Illustrative Example

To show how formal methods and task models can be used to determine how human behavior (including unanticipated human error) can assess system safety, consider a radiation therapy machine example (originally from Bolton et al. 2012, 2019). This machine is a room-sized, computer-controlled, medical linear accelerator. Its important feature is that it has two treatment modes: electron beam mode for treating shallow tissue and X-ray mode (which uses a beam one hundred times stronger than electron beam mode) for deeper treatments. To account for the increased power, the X-ray mode uses a spreader (not used in the other mode) to attenuate the radiation beam. The mode and other treatment information are controlled by a practitioner who must select options and administer treatment using a computer console. Clearly, this is a complex machine whose proper function relies on human interaction that could have profound implications for patient health and safety. The following describes a formal model of this machine along with the human task used to interact with it. Formal verification model checking analyses for assessing system safety with both normative and potentially unanticipated human errors is presented afterwards.

The human–machine interface formal model (top of Figure 33.3) takes five keyboard keys as input ("X," "E," "Enter," "↑," and "B") and information presented to a practitioner who is administering treatment on a computer monitor. The interface state (*InterfaceState*) starts in *Edit* where the human operator can press "X" or "E" (*PressX* or *PressE*) to select the X-ray or electron beam mode and, thus, transition to the *ConfirmXrayData* or *ConfirmEBeamData* respectively. When in a confirmation state, the corresponding treatment data are displayed (*DisplayedData*). The practitioner can confirm the displayed treatment by pressing enter (advancing to *PrepareToFireXray* or *PrepareToFireEBeam*) or go back to *Edit* by pressing "↑" (*PressUp*). In *PrepareToFireXray* or *PrepareToFireEBeam*, the human operator waits for the beam to become ready (*BeamState*), at which point a press of "B" (*PressB*) will fire the beam. This transitions the interface to *TreatmentAdministered*. Alternatively, the operator presses "↑" to return to the previous state.

The device automation model (bottom of Figure 33.3) controls beam power level, spreader position, and beam firing. The beam power level (*BeamLevel*) is initially not set (*NotSet*). When the human selects the mode, the power level transitions to the appropriate setting (*XrayLevel* or *EBeamLevel*). However, if the human selects a new mode, there is a transition delay to the correct level, where power remains in an intermediary state (*XtoE* or *EtoX*) at the old level before automatically transitioning to the new one. The spreader position (*Spreader*) starts either in- or out-of-place (*InPlace* or *OutOfPlace*). When the

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

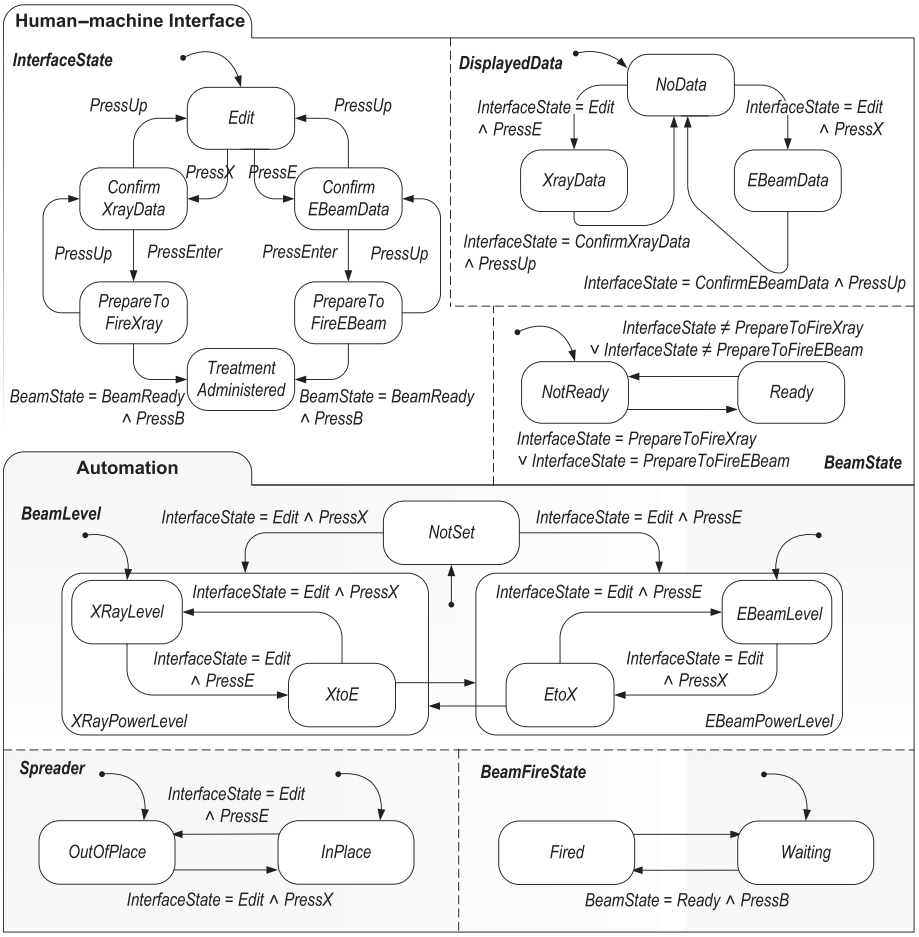

Figure 33.3 *Concurrent state machine representation of the formal human–machine interface (top) and automation (bottom) models for the radiation therapy application (Bolton et al., 2012). Rounded rectangles represent states. Arrows between states are transitions. Dotted arrows indicate initial states.*

human selects X-ray or electron beam treatment, the spreader transitions to the correct configuration (*InPlace* or *OutOfPlace* respectively). The beam firing state (*BeamFireState*) is initially waiting (*Waiting*). When the human fires the beam (presses "B" when the beam is ready), the beam fires (*Fired*) and returns to waiting.

The normative task for interacting with this machine was represented using EOFM (Bolton et al., 2011) using three tasks (Figure 33.4): (a) selecting the treatment mode; (b) confirming treatment data; and (c) firing the beam. These tasks access variables from other parts of the model such as the human–machine interface, displayed treatment data (*DisplayedData*), and the ready status of the beam (*BeamState*). It also has a variable (*TreatmentType*) that nondeterministically specifies which treatment is prescribed (*Xray* or *EBeam*).

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

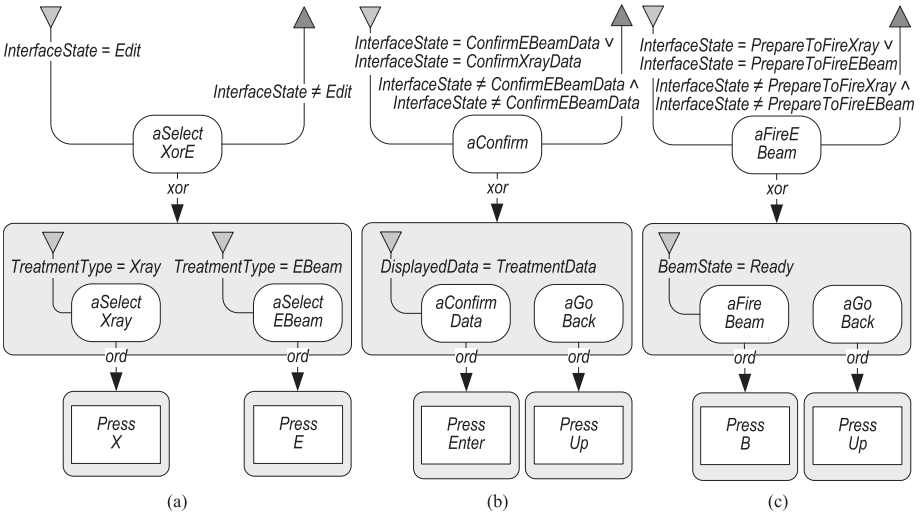

Figure 33.4 *Visualization of the EOFM tasks for interacting with the radiation therapy machine (Bolton et al., 2012): (a) selecting the treatment mode; (b) confirming treatment data; and (c) firing the beam. Atomic actions are rectangles and goal-directed activities are rounded rectangles. An activity decomposes into sub-activities or actions via an arrow labeled with a decomposition operator. This operator logically describes how many and in what order decomposed acts are executed (i.e.* xor *for only one sub-act and* ord *for all executing in order from left to right). Strategic knowledge (environmental conditions that influence task performance) conditions are connected to associated activities. These are labeled with the Boolean logic of the condition. A* Precondition *(what must be true for an activity to begin) is a yellow, downward triangle. A* CompletionCondition *(what must be true for an activity to complete) is a magenta, upward triangle.*

When the interface is in the edit state (*aSelectXorE*), the practitioner selects the appropriate treatment mode by performing the actions for pressing the X or E keys. When the interface is in either of the two data confirmation states (*aConfirm*), the practitioner can choose to confirm the displayed data (if the data correspond to the prescribed treatment) by pressing enter. Alternatively, he or she can return to *Edit* by pressing up ("↑"). When the interface is in either state for preparing to fire (*aFire*), the practitioner can fire if the beam is ready (by pressing "B") or press "↑" to return to the previous state.

A compelling contribution of EOFM is its formal semantics (Figure 33.5a–b). These provide unambiguous, mathematical interpretations of the task's behavior (Bolton et al., 2011). Every activity and action is treated as a state machine that transitions between three states: *Ready* (the initial state), *Executing*, and *Done*. An activity transitions between these states based on whether the Boolean conditions on the labeled transitions are true. These are defined using activity strategic knowledge conditions (*Precondition*s, *RepeatCondition*s (not shown in Figure 33.5), and *CompletionCondition*s) and

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

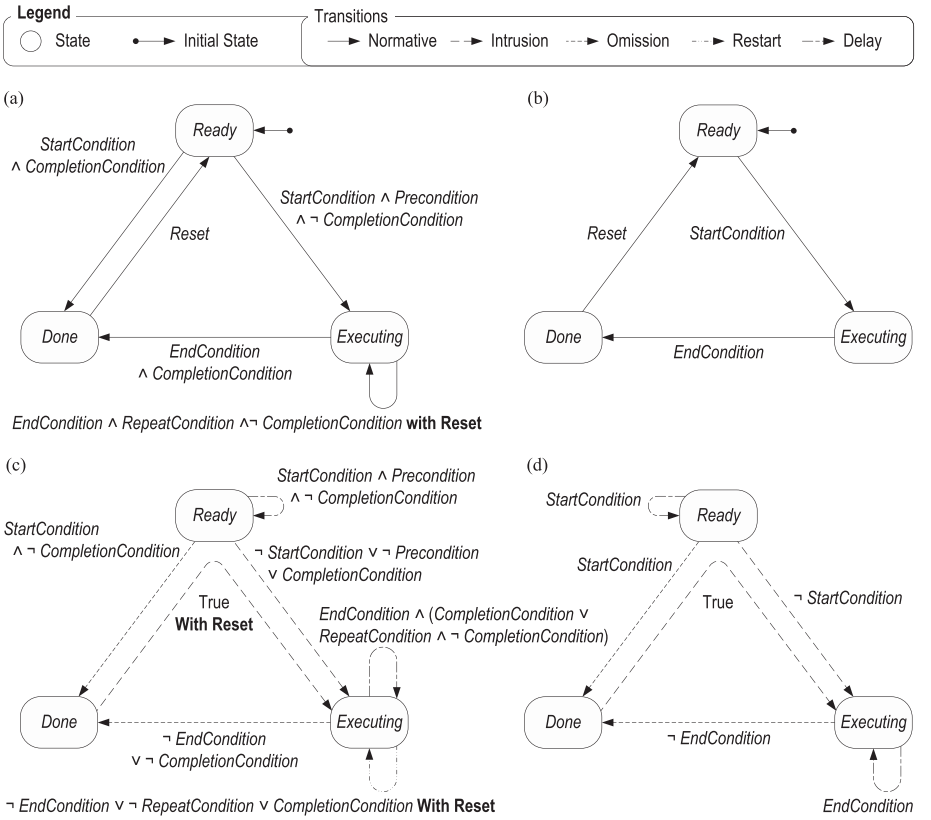

Figure 33.5 *The formal semantics used to interpret EOFMs (like the one from Figure 33.4) as a formal model. (a) and (b) are the normative semantics for task activities and actions respectively (Bolton et al., 2011). (c) and (d) are additional erroneous transitions (for activities and actions respectively) used for generating human errors (Bolton et al., 2019) using the task-based taxonomy (Bolton, 2017b).*

three additional, implicit conditions. These assert whether an activity or action can start, end, or reset based on its position in the task and other relevant activity and action states (Bolton et al., 2011, 2017). These formal semantics are the basis for automated translator software that converts EOFMs into a formal representation for inclusion in a larger formal model.

For the radiation therapy example, the task from Figure 33.4 was translated into a formal model and paired with the elements from Figure 33.3. Model checking was used to check a linear temporal logic specification:

$$\mathbf{G}\neg\begin{pmatrix} BeamFireState = Fired \\ \wedge BeamLevel = XRayPowerLevel \\ \wedge Spreader = OutOfPlace \end{pmatrix}. \tag{33.1}$$

This asserts that the machine should globally (**G**) never (¬) irradiate a patient by administering an unshielded X-ray treatment when the spreader is out of place.

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

This verified to true, proving that the radiation therapy machine will never irradiate a patient if the human operator behaves normatively.

While the ability to prove that a model is safe with normative behavior is powerful, this provides no insights into human error (especially that which is unanticipated). Another contribution of EOFM can address this. Specifically, EOFM was used in the formulation of the task-based taxonomy of erroneous human behavior (Bolton, 2017b). This classifies where a deviation occurs based on a violation of task formal semantics and thus indicates the observable manifestation of the error (its phenotype (Hollnagel, 1993)) and its associated failure of attention (the genotype of the slip (Reason, 1990)). While there are multiple levels of classification in this taxonomy, this discussion focuses on error modes: erroneous transitions that can occur between execution states (Figure 33.5c–d). An intrusion occurs when an act (an activity or action) executes when it should not. An omission occurs when an act transitions to done when it should not. A restart occurs when an act's execution restarts when it should not. Finally, a delay occurs when an act does not transition when it should.

These erroneous semantics were incorporated into the translator (along with the original, normative transitions) to enable formal verification to consider all of the possible human errors encompassed by the taxonomy (Bolton et al., 2019). This enables modeling checking to determine if normative or potentially unanticipated erroneous human behavior can ever cause problems.

When the erroneous transitions were enabled for the radiation machine, the verification of (1) failed. This produced a counterexample showing how the patient could be irradiated. First, the practitioner accidentally selected the wrong mode for the machine (an activity *Ready*-to-*Executing* intrusion (Figure 33.5c) of *aSelectXray* (Figure 33.4a) when the human improperly attended to the precondition of the activity). This set the *BeamLevel* to the *XRayLevel* and moved the *Spreader InPlace*. The human noticed the mistake because the treatment data was incorrect. He/she then pressed "↑," corrected the error by selecting electron beam mode, thus moving the *Spreader OutOfPlace* and setting the *BeamLevel* to *XtoE*. The practitioner confirmed treatment data and, when the beam became ready, fired it. Because the beam was fired before the *BeamLevel* transitioned away from *XtoE*, an *XRayPowerLevel* was delivered without the *Spreader* being *InPlace*.

Bolton et al. (2019) went on to explore interventions that could address this discovered problem (by ensuring that *BeamState* does not become ready until the *BeamLevel* matches the entered treatment mode) and evaluated the resulting design with additional verifications.

33.3.2.1.2 Additional Capabilities of Formal, Task-Analytic Methods

The example presented above gives an illustration of both the capabilities of using task-models with formal methods and an example of how the engineering developments in this area can lead to new ways of using cognitive science. There are many other applications of formal task analytic methods. Researchers (España, Pederiva, & Panach, 2007; Li, Wei, Zheng, & Bolton, 2017; Luyten,

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

Clerckx, Coninx, & Vanderdonckt, 2003; Santoro, 2005) have explored methods for automatically designing human–machine interfaces directly from task models so that the interfaces are guaranteed to always support the human's tasks. Additionally, researchers have explored how cognitive models of human reliability can be integrated with tasks to determine the likelihood of human errors causing failures (Fahssi, Martinie, & Palanque, 2015; Zheng, Bolton, Daly, & Biltekoff, 2020). Finally, formal task models have been used for automated test case generation (Barbosa et al., 2011; Campos et al., 2016; Li & Bolton, 2019; Vieira, Leduc, Hasling, Subramanyan, & Kazmeier, 2006): a method where tests are created from formal models to guarantee that analyst-specified criteria are satisfied in tests. Tests can be executed automatically (to validate that the system conforms to the model) or with human subjects (to gain insights about things like usability and workload not manifest in the model).

33.3.2.2 Cognitive-Architecture-Based Approaches

Practical and cognitive insights can be made for formal analyses based on task models. However, without a deeper model of cognition, analyses will be limited. To address this, multiple researchers have explored how cognitive architectures can be formalized so that sophisticated cognitive models can be used to understand how human cognition contributes to system problems.

The most significant research in this area was the *generic user modeling* (Curzon & Rukšėnas, 2017). This approach built off of preceding work on Programmable User Models (PUMs) (Young et al., 1989), where the human has goals to achieve with an application and actions they can perform. A rule set (beliefs or knowledge) defines when the human may attempt to pursue a specific goal based on the state of the human–automation interface, the environment, or other goals currently being considered. An action can be performed when a human commits to applying it according to production rules. A separate action execution occurs after the human commits to performing that action.

Generic user modeling has been used in many formal verification analyses. Curzon and Blandford (2004) identified how cognitively plausible human errors could manifest in their models. These included performance/coordination errors associated with the phenotypes of erroneous action (Hollnagel, 1993) and mechanisms for identifying post-completion errors, special omissions where the human forgets to perform actions that occur after a primary goal has been achieved (Byrne & Bovair, 1997). Problems discovered with formal verification can be addressed with design rules (Curzon & Blandford, 2004). Work has investigated how to use these types of formal cognitive models to determine when different operators (expert vs. novice) may commit errors (Curzon, Rukšėnas, & Blandford, 2007). Later contributions incorporated additional cognitive mechanisms to account for salience, cognitive load, interpretation of spatial cues, and timing in analyses (Rukšėnas, Back, Curzon, & Blandford, 2008, 2009; Rukšėnas et al., 2007; Rukšėnas, Curzon, Back, & Blandford, 2007). Similarly, Basuki, Cerone, Griesmayer, and Schlatte (2009) used

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

heuristics for modeling human habituation, impatience, and carefulness within the architecture.

An illustrative example was reported by Curzon et al. (2007), who evaluated the human–machine interaction of an automated teller machine (ATM) with a cognitive architecture and automated theorem prover to determine if a human could ever leave the machine without completing all intended goals. The verification discovered the presence of a post-completion error where the human could receive his or her cash (the primary goal) and leave without retrieving the ATM card. Curzon et al. also explored improved machine designs that had the human retrieve the card before cash was dispensed, which was verified to prevent the error.

33.4 Conclusions

This chapter has described the area of cognitive engineering and explored the ways that cognitive modeling is used within this area to accomplish engineering goals throughout the engineering life cycle. In particular, the chapter showed how cognitive engineering differs from cognitive science in that: (1) cognitive engineering addresses problems based on practical need more than academic interest; (2) cognitive engineers tend to work in emerging technological domains rather than well-studied fields; (3) cognitive engineering modelers rely heavily on domain experts to acquire information and data needed for modeling; (4) the utility of models to engineering goals is paramount, and insights into human cognition are only interesting if they serve these engineering goals; and (5) cognitive engineers are typically dealing with human performance in a complex system and must capture the control of integrated cognitive systems in their models.

As such, cognitive models are often used by engineers to help ensure that complex systems are human-centric. This means enabling systems to allow humans to accomplish their goals while avoiding system performance and safety problems. This domain was used to explore the different ways that cognitive models have been used in engineering. To provide context, the chapter delved into the foundational work Card, Moran, and Newell did for GOMS. It then covered simulation analyses and showed how cognitive architectures can be used as the basis for models that provide engineers with insights into human performance in emerging areas such as UAV piloting and autonomous driving. The chapter also explored how the requirements of applying cognitive models to these environments expanded the canon of both cognitive science and modeling. While the simulation-based cognitive architecture models are definitely at the forefront of cognitive science developments, they can miss dangerous operating conditions. The use of cognitive models in formal-methods-based verification addresses this shortcoming. In this context, the cognitive models may be simple tasks (in the spirit of GOMS) or based on cognitive architectures. It is important to note that, compared

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

with simulation, formal verifications, as a consequence of being exhaustive, scale very badly (something colloquially called the state explosion problem; Clarke et al., 1999). Thus, it is not surprising that the formal methods are much simpler than those used in simulation. The innovation in this domain comes from determining how to include cognitive concepts in formal models so that the power of verification can account for them. As such, the formal methods research is heavily dependent on the advances made on the more conventional cognitive architecture front.

As systems become more complex and integrated into everybody's lives, it is more important than ever that these systems be human-centered and aligned with fulfilling humanistic goals. As such, cognitive-model-based engineering should remain topical and relevant far into the future, especially as the capabilities and validity of the methods improve. To this end, ACT-R (and its variants) remains the premier architecture for cognitive modeling advances. This is largely because ACT-R is easy to extend, has kept pace with advances in cognitive science, and is capable of interacting with the same software as human users (Gray, 2008).

Despite this, current research trends actually run counter to those traditionally upheld by cognitive engineering modelers. Specifically, advances in data science, machine learning (ML), and artificial intelligence (AI) tend to favor algorithms that can (sometimes) do a remarkably good job of predicting performance or exerting control. There is thus a serious push to use these approaches everywhere. While it is true that cognitive modeling is a form of ML or AI, the new methods are fundamentally different in that they are not based on any specific theory of human cognition and are often incapable of "explaining" their predictions. In fact, "explainable AI" is a hot topic within cognitive engineering, with ACT-R even being used as a potential tool for this (Gunning & Aha, 2019). Such developments have the potential to be extremely useful from an engineering perspective because they could provide systems with an unprecedented ability to recognize human behavior, respond to humans, or simulate human behavior. However, these developments also have potential pitfalls because, when used in place of cognitive models, the AI will likely not provide the same explanations, reduce insight, and limit their import to cognitive science. Whether or not this is a critical flaw for an engineering effort will largely depend on whether explainability is important. As the examples above demonstrate, significant insights into cognitive science can be gained through cognitive engineering advances. Thus, it should be a priority for researchers and engineers moving forward to maintain the synergistic relationship between cognitive science and engineering as this will allow both fields to advance.

Historically, the introduction of advanced and unexplained automation has caused complex system problems in ways that could be exacerbated by ML and AI (Bainbridge, 1983; Strauch, 2017): automation can be brittle and fail in situations unanticipated during design and/or model fitting; the human may not be able to track the state of the system, leading to mode confusion, disorienting

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

automation surprise, and human errors; and humans can have their roles change to ones (such as monitoring) incompatible with their abilities and competencies. Thus, cognitive engineers should be very careful moving forward not to abandon cognitive models in their efforts, as joint developments of cognitive science and engineering will help ensure that engineering projects will be human centered.

References

Abbate, A. J., & Bass, E. J. (2015). Using computational tree logic methods to analyze reachability in user documentation. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 59, pp. 1481–1485).

Aït-Ameur, Y., & Baron, M. (2006). Formal and experimental validation approaches in HCI systems design based on a shared event B model. *International Journal on Software Tools for Technology Transfer*, *8(6)*, 547–563.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglas, S., Lebiere, C., & Quin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111(4)*, 1036–1060.

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: a theory of higher-level cognition and its relation to visual attention. *Human-Computer Interaction, 12(4)*, 439–462.

Bainbridge, L. (1983). Ironies of automation. *Automatica, 19(6)*, 775–780.

Ball, J., Myers, C., Heiberg, A., et al. (2010). The synthetic teammate project. *Computational and Mathematical Organization Theory, 16(3)*, 271–299.

Ballard, D. H., & Sprague, N. (2007). On the role of embodiment in modeling natural behaviors. In W. D. Gray (Ed.), *Integrated Models of Cognitive Systems*. New York, NY: Oxford University Press.

Barbosa, A., Paiva, A. C., & Campos, J. C. (2011). Test case generation from mutated task models. In *Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 175–184).

Basnyat, S., Palanque, P. A., Bernhaupt, R., & Poupart, E. (2008). Formal modelling of incidents and accidents as a means for enriching training material for satellite control operations. In *Proceedings of the Joint ESREL 2008 and 17th SRA-Europe Conference* (CD-ROM). London: Taylor & Francis.

Basnyat, S., Palanque, P., Schupp, B., & Wright, P. (2007). Formal socio-technical barrier modelling for safety-critical interactive systems design. *Safety Science*, *45(5)*, 545–565.

Bastide, R., & Basnyat, S. (2007). Error patterns: systematic investigation of deviations in task models. In *Task Models and Diagrams for Users Interface Design 5th International Workshop* (pp. 109–121). Berlin: Springer.

Basuki, T. A., Cerone, A., Griesmayer, A., & Schlatte, R. (2009). Model-checking user behaviour using interacting components. *Formal Aspects of Computing*, 1–18.

Bolton, M. L. (2015). Model checking human–human communication protocols using task models and miscommunication generation. *Journal of Aerospace Information Systems*, *12(7)*, 476–489.

https://doi.org/10.1017/9781108755610.038 Published online by Cambridge University Press

Bolton, M. L. (2017a). Novel developments in formal methods for human factors engineering. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 715–717).

Bolton, M. L. (2017b). A task-based taxonomy of erroneous human behavior. *International Journal of Human-Computer Studies*, *108*, 105–121.

Bolton, M. L., & Bass, E. J. (2010). Formally verifying human-automation interaction as part of a system model: limitations and tradeoffs. *Innovations in Systems and Software Engineering: A NASA Journal*, *6(3)*, 219–231.

Bolton, M. L., & Bass, E. J. (2013). Generating erroneous human behavior from strategic knowledge in task models and evaluating its impact on system safety with model checking. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, *43(6)*, 1314–1327.

Bolton, M. L., & Bass, E. J. (2017). Enhanced operator function model (EOFM): a task analytic modeling formalism for including human behavior in the verification of complex systems. In B. Weyers, J. Bowen, A. Dix, & P. Palanque (Eds.), *The Handbook of Formal Methods in Human-Computer Interaction*. Berlin: Springer.

Bolton, M. L., Bass, E. J., & Siminiceanu, R. I. (2012). Generating phenotypical erroneous human behavior to evaluate human–automation interaction using model checking. *International Journal of Human-Computer Studies*, *70(11)*, 888–906.

Bolton, M. L., Bass, E. J., & Siminiceanu, R. I. (2013). Using formal verification to evaluate human-automation interaction in safety critical systems, a review. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, *43(3)*, 488–503.

Bolton, M. L., Molinaro, K. A., & Houser, A. M. (2019). A formal method for assessing the impact of task-based erroneous human behavior on system safety. *Reliability Engineering & System Safety*, *188*, 168–180.

Bolton, M. L., Siminiceanu, R. I., & Bass, E. J. (2011). A systematic approach to model checking human-automation interaction using task-analytic models. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, *41(5)*, 961–976.

Boring, R. L., & Rasmussen, M. (2016). GOMS-HRA: a method for treating subtasks in dynamic human reliability analysis. In *Proceedings of the 2016 European Safety and Reliability Conference* (pp. 956–963).

Bovair, S., Kieras, D. E., & Polson, P. G. (1990). The acquisition and performance of text-editing skill: a cognitive complexity analysis. *Human-Computer Interaction, 5(1)*, 1–48.

Byrne, M. D. (2007). Cognitive architecture. In A. Sears & J. A. Jacko (Eds.), *The Human-Computer Interaction Handbook* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Byrne, M. D., & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebière (Eds.), *The Atomic Components of Thought* (pp. 167–200). Hillsdale, NJ: Lawrence Erlbaum Associates.

Byrne, M. D., & Bovair, S. (1997). A working memory model of a common procedural error. *Cognitive Science*, *21(1)*, 31–61.

Campos, J. C., Fayollas, C., Martinie, C., Navarre, D., Palanque, P., & Pinto, M. (2016). Systematic automation of scenario-based testing of user interfaces. In *Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 138–148).

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Clarke, E. M., Grumberg, O., & Peled, D. A. (1999). *Model Checking*. Cambridge, MA: MIT Press.

Curzon, P., & Blandford, A. (2004). Formally justifying user-centered design rules: a case study on post-completion errors. In *Proceedings of the 4th International Conference on Integrated Formal Methods* (pp. 461–480). Berlin: Springer.

Curzon, P., & Rukšėnas, R. (2017). Modelling the user. In B. Weyers, J. Bowen, A. Dix, & P. Palanque (Eds.), *The Handbook of Formal Methods in Human-Computer Interaction*. Berlin: Springer.

Curzon, P., Rukšėnas, R., & Blandford, A. (2007). An approach to formal verification of human–computer interaction. *Formal Aspects of Computing*, *19(4)*, 513–550.

Degani, A. (2004).*Taming HAL: Designing Interfaces Beyond 2001*. New York, NY: Macmillan.

Degani, A., Heymann, M., & Shafto, M. (1999). Formal aspects of procedures: the problem of sequential correctness. In *Proceedings of the 43rd Annual Meeting of the Human Factors and Ergonomics Society* (pp. 1113–1117). Los Angeles, CA: SAGE.

Demir, M., McNeese, N. J., Cooke, N. J., Ball, J. T., Myers, C., & Frieman, M. (2015). Synthetic teammate communication and coordination with humans. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 951–955). Los Angeles, CA: SAGE.

Demir, M., McNeese, N. J., & Cooke, N. J. (2016). Team communication behaviors of the human-automation teaming. In *2016 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)* (pp. 28–34). New York, NY: IEEE.

Emerson, E. A. (1990). Temporal and modal logic. In *Formal Models and Semantics* (pp. 995–1072). Oxford: Elsevier.

España, S., Pederiva, I., & Panach, J. I. (2007). Integrating model-based and task-based approaches to user interface generation. In *Computer-Aided Design of User Interfaces V* (pp. 253–260). Amsterdam: Springer.

Fahssi, R., Martinie, C., & Palanque, P. (2015). Enhanced task modelling for systematic identification and explicit representation of human errors. In *Human-Computer Interaction – Interact 2015* (pp. 192–212). Cham: Springer International Publishing.

Fields, R. E. (2001). Analysis of erroneous actions in the design of critical systems. Unpublished doctoral dissertation, University of York, York.

Gluck, K. A., Ball, J. T., Gunzelmann, G., Krusmark, M., Lyon, D., & Cooke, N. (2005). A prospective look at a synthetic teammate for UAV applications. In *Infotech@ Aerospace*. Reston: American Institute of Aeronautics and Astronautics.

Gluck, K. A., Ball, J. T., & Krusmark, M. A. (2007). Cognitive control in a computational model of the predator pilot. In W. D. Gray (Ed.), *Integrated Models of Cognitive Systems* (pp. 13–28). New York, NY: Oxford University Press.

Gray, W. D. (2008). Cognitive modeling for cognitive engineering. In R. Sun (Ed.), *The Cambrdge Handbook of Computational Psychology* (pp. 565–588). Cambridge: Cambridge University Press.

Gray, W. D. (Ed.). (2007). *Integrated Models of Cognitive Systems*. New York, NY: Oxford University Press.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: an introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied, 6(4)*, 322–335.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: validating a GOMS analysis for predicting and explaining real-world performance. *Human-Computer Interaction, 8(3)*, 237–309.

Gunning, D., & Aha, D. W. (2019). DARPA's explainable artificial intelligence program. *AI Magazine, 40(2)*, 44–58.

Hollnagel, E. (1993). The phenotype of erroneous actions. *International Journal of Man-Machine Studies*, *39(1)*, 1–32.

Jeong, H., & Liu, Y. (2017). Modeling of stimulus-response secondary tasks with different modalities while driving in a computational cognitive architecture. In *Proceedings of the 9th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design* (pp. 193–199). Iowa, IA: University of Iowa.

John, B. E. (1988). Contributions to engineering models of human-computer interaction. Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

John, B. E. (1996). TYPIST: a theory of performance in skilled typing. *Human-Computer Interaction, 11(4)*, 321–355.

Kebabjian, R. (2016). *Accident statistics*. planecrashinfo.com. Retrieved from www.planecrashinfo.com/cause.htm [last accessed July 30, 2022].

Kenny, D. J. (2015). 26th Joseph T. Nall Report: General Aviation Accidents in 2014. Technical Report. Frederick, MD: AOPA Foundation.

Kieras, D. E. (1997). A guide to GOMS model usability evaluation using NGOMSL. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), *Handbook of Human-Computer Interaction* (2nd ed., pp. 733–766). New York, NY: Elsevier.

Kieras, D. E. (2007). Model-based evaluation. In A. Sears & J. A. Jacko (Eds.), *The Human-Computer Interaction Handbook* (2nd ed.). Mahwah, NJ: Lawrence Erlbaum Associates.

Kieras, D. E., & Bovair, S. (1986). The acquisition of procedures from text: a production-system analysis of transfer of training. *Journal of Memory and Language, 25*, 507–524.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction, 12(4)*, 391–438.

Kieras, D. E., Wakefield, G. H., Thompson, E. R., Iyer, N., & Simpson, B. D. (2016). Modeling two-channel speech processing with the EPIC cognitive architecture. *Topics in Cognitive Science, 8(1)*, 291–304.

Kirwan, B., & Ainsworth, L. K. (Eds.). (1992). *A Guide to Task Analysis*. Washington, DC: Taylor & Francis.

Le Bot, P. (2004). Human reliability data, human error and accident models – illustration through the Three Mile Island accident analysis. *Reliability Engineering & System Safety, 83(2)*, 153–167.

Li, M., & Bolton, M. L. (2019). Task-based automated test case generation for human-machine interaction. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 63, pp. 807–811).

Li, M., Wei, J., Zheng, X., & Bolton, M. L. (2017). A formal machine learning approach to generating human-machine interfaces from task models. *IEEE Transactions of Human Machine Systems, 47(6),* 822–833.

Liu, Y., Feyen, R., & Tsimhoni, O. (2006). Queueing Network-Model Human Processor (QN-MHP): a computational architecture for multitask performance in human-machine systems. *ACM Transactions on Computer-Human Interaction (TOCHI), 13(1)*, 37–70.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: combined influences on operator selection. *Cognitive Psychology, 31*, 168–217.

Luyten, K., Clerckx, T., Coninx, K., & Vanderdonckt, J. (2003). Derivation of a dialog model from a task model by activity chain extraction. In *Proceedings of the 10th International Workshop on Interactive Systems. Design, Specification, and Verification* (pp. 203–217). Berlin: Springer.

Manning, S. D., Rash, C. E., LeDuc, P. A., Noback, R. K., & McKeon, J. (2004). The Role of human Causal Factors in US Army Unmanned Aerial Vehicle Accidents. Technical Report No. 2004-11. Adelphi, MD: USA Army Research Laboratory.

Makary, M. A., & Daniel, M. (2016). Medical error – the third leading cause of death in the US. *BMJ*, *353*, 5.

Mirman, J. H. (2019). A dynamical systems perspective on driver behavior. *Transportation Research Part F: Traffic Psychology and Behaviour*, *63*, 193–203.

Mirman, J. H., Curry, A. E., & Mirman, D. (2019). Learning to drive: a reconceptualization. *Transportation Research Part F: Traffic Psychology and Behaviour*, *62*, 316–326.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Card, S. K. (1985). The prospects for psychological science in human-computer interaction. *Human-Computer Interaction, 1(3)*, 209–242.

NHTSA. (2008). National Motor Vehicle Crash Causation Survey: Report to Congress. Technical Report No. DOT HS 811 059. Springfield: National Highway Traffic Safety Administration.

Pan, D., & Bolton, M. L. (2018). Properties for formally assessing the performance level of human-human collaborative procedures with miscommunications and erroneous human behavior. *International Journal of Industrial Ergonomics*, *63*, 75–88.

Paternò, F., & Santoro, C. (2001). Integrating model checking and HCI tools to help designers verify user interface properties. In *Proceedings of the 7th International Workshop on the Design, Specification, and Verification of Interactive Systems* (pp. 135–150). Berlin: Springer.

Paternò, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: a diagrammatic notation for specifying task models. In *Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction* (pp. 362–369). London: Chapman & Hall.

Pew, R. W. (2007). Some history of human performance modeling. In W. D. Gray (Ed.), *Integrated Models of Cognitive Systems*. New York, NY: Oxford University Press.

Pritchett, A. R., Feigh, K. M., Kim, S. Y., & Kannan, S. K. (2014). Work models that compute to describe multiagent concepts of operation: part 1. *Journal of Aerospace Information Systems, 11(10)*, 610–622.

Reason, J. (1990). *Human Error*. New York, NY: Cambridge University Press.

Rehman, U., Cao, S., & MacGregor, C. (2019). Using an integrated cognitive architecture to model the effect of environmental complexity on drivers' situation awareness. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (pp. 812–816).

Rhie, Y. L., Lim, J. H., & Yun, M. H. (2018). Queueing network based driver model for varying levels of information processing. *IEEE Transactions on Human-Machine Systems*, *49(6)*, 508–517.

Rodgers, S., Myers, C., Ball, J., & Freiman, M. (2011). The situation model in the synthetic teammate project. In *Proceedings of the 20th Annual Conference on Behavior Representation in Modeling and Simulation* (pp. 66–73).

Rukšėnas, R., Back, J., Curzon, P., & Blandford, A. (2008). Formal modelling of salience and cognitive load. In *Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems* (pp. 57–75). Amsterdam: Elsevier Science Publishers.

Rukšėnas, R., Back, J., Curzon, P., & Blandford, A. (2009). Verification-guided modelling of salience and cognitive load. *Formal Aspects of Computing*, *21(6)*, 541–569.

Rukšėnas, R., Curzon, P., Back, J., & Blandford, A. (2007). Formal modelling of cognitive interpretation. In *Proceedings of the 13th International Workshop on the Design, Specification, and Verification of Interactive Systems* (pp. 123–136). London: Springer.

Rukšėnas, R., Curzon, P., Blandford, A., & Back, J. (2014). Combining human error verification and timing analysis: a case study on an infusion pump. In *Proceedings of the 13th International Workshop on the Design, Specification, and Verification of Interactive Systems* (pp. 123–136). London: Springer.

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(1)*, 85–107.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors, 48(2)*, 362–380.

Salvucci, D. D., & Gray, R. (2004). A two-point visual control model of steering. *Perception, 33(10)*, 1233–1248.

Salvucci, D. D., & Macuga, K. L. (2002). Predicting the effects of cellular-phone dialing on driver performance. *Cognitive Systems Research*, *3(1)*, 95–102.

Santoro, C. (2005). *A Task Model-Based Approach for Design and Evaluation of Innovative User Interfaces*. Belgium: Presses universitaires de Louvain.

Schweickert, R., Fisher, D. L., & Proctor, R. W. (2003). Steps toward building mathematical and computer models from cognitive task analyses. *Human Factors, 45(1)*, 77–103.

Shepherd, A. (1998). HTA as a framework for task analysis. *Ergonomics, 41(11)*, 1537–1552.

Shepherd, A. (2001). *Hierarchical Task Analysis*. New York, NY: Taylor & Francis.

Sheridan, T. B., & Parasuraman, R. (2005). Human-automation interaction. *Reviews of Human Factors and Ergonomics*, *1(1)*, 89–129.

Simon, H. A. (1996). *The Sciences of the Artificial* (3rd ed.). Cambridge, MA: MIT Press.

Strauch, B. (2017). Ironies of automation: still unresolved after all these years. *IEEE Transactions on Human-Machine Systems, 48(5)*, 419–433.

Thomas, M. (1994). The role of formal methods in achieving dependable software. *Reliability Engineering & System Safety*, *43(2)*, 129–134.

Vieira, M., Leduc, J., Hasling, B., Subramanyan, R., & Kazmeier, J. (2006). Automation of GUI testing using a model-driven approach. In *Proceedings of the 2006 International Workshop on Automation of Software Test* (pp. 9–14).

Weyers, B., Bowen, J., Dix, A., & Palanque, P. (Eds.). (2017). *The Handbook of Formal Methods in Human-Computer Interaction*. Berlin: Springer.

Wing, J. M. (1990). A specifier's introduction to formal methods. *Computer*, *23(9)*, 8–22.

Wu, C., Rothrock, L., & Bolton, M. (2019). Editorial special issue on computational human performance modeling. *IEEE Transactions on Human-Machine Systems, 49(6)*, 470–473.

Young, R. M., Green, T. R. G., & Simon, T. (1989). Programmable user models for predictive evaluation of interface designs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 15–19). New York: ACM.

Zheng, X., Bolton, M. L., Daly, C., & Biltekoff, E. (2020). The development of a next-generation human reliability analysis: systems analysis for formal pharmaceutical human reliability (SAFPHR). *Reliability Engineering & System Safety*, *20*. https://doi.org/10.1016/j.ress.2020.106927