Invited paper, presented at Safecomp 2001, Budapest, Hungary, September 2001.
Springer-Verlag LNCS Vol 2187, pp. 86–91. ©Springer-Verlag

Modeling the Human in Human Factors
Extended Abstract*

John Rushby

Computer Science Laboratory
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, USA
rushby@csl.sri.com

Abstract. Human operators use mental models to guide their interaction with automated systems. We can "model the human" by constructing explicit descriptions of plausible mental models. Using mechanized formal methods, we can then calculate divergences between the actual system behavior and that suggested by the mental model. These divergences indicate possible automation surprises and other human factors problems and suggest places where the design should be improved.

1 Introduction

Human error is implicated in many accidents and incidents involving computerized systems, with problems and design flaws in the human-computer interface often cited as a contributory factor. These issues are particularly well-documented in the cockpits of advanced commercial aircraft, where several fatal crashes and other incidents are attributed to problems in the "flightcrew-automation interface" [8, Appendix D].

There is much work, and voluminous literature, on topics related to these issues, including mode confusions [22] and other "automation surprises" [23], human error [18], human cognition [16], and human-centered design [1].

The human-centered approach to automation design explicitly recognizes the interaction between human and computer in complex systems, and the need for each side of the interaction to have a model of the other's current state and possible future behavior.

> "*To command effectively, the human operator must be involved and informed. Automated systems need to be predictable and capable of being monitored by human operators. Each element of the system must have knowledge of the other's intent*" [1, Chapter 3].

Computer scientists might recognize in this description something akin to the interaction of concurrent processes, and might then speculate that the combined behavior of human and computer could be analyzed and understood in ways that are similar to those

* This research was supported by the DARPA SEC program through USAF Rome Laboratory contract F33615-00-C-3043 and by NASA Langley Research Center through contract NAS1-00079.

used to reason about interacting processes. In the "assume-guarantee" approach [13], for example, each process records what it assumes about the other and specifies, in return, what it will guarantee if those assumptions are met. Now, the computer side of this interaction is, or can naturally be modeled as, a process in some formal system for reasoning about computational artifacts. But what about the human side: is it reasonable to model the human as a computational system?

It turns out that modern cognitive science holds the view that the mind is, precisely, a computational system (or, at least, an information processor) of some kind [15]. Thus, we can imagine constructing a computational model of some aspects of human cognition and behavior, confronting this with a similar model of the computerized system with which it is to interact, and using formal calculation to derive observations or conclusions about their joint behavior.

Explicit models of human performance have long been used in computer interface design: for example, GOMS (Goals, Operators, Methods, and Selections) analysis dates back to 1983 [3] and has spawned many variants that are used today [11]. More recently, cognitive models have been used to simulate human capabilities in systems for developing and evaluating user interfaces [19]. Deeper models such as ICS (Interacting Cognitive Subsystems) allow examination of the cognitive resources required to operate a particular interface [2, 6]. These approaches are useful in identifying error-prone features in interfaces to safety-critical systems (e.g., the complex process that must be followed to enter a new flight plan into a flight management system), but they do not seem to address the most worrying kinds of problems: those associated with mode confusions and other kinds of automation surprise.

Automation surprises occur when an automated system does not behave as its operator expects. Modern cognitive psychology has established the importance of *mental models* in guiding humans' interaction with the world [12]; in particular, operators and users of automated systems develop such models of their system's behavior and use these to guide their interaction [14, 17]. Seen from this perspective, an automation surprise occurs when the actual behavior of a system departs from that predicted by its operator's mental model.

Mental models of physical systems are three-dimensional kinematic structures that correspond to the structure of what they represent. They are akin to architects' models of buildings and to chemists' models of complex molecules. For logical systems, it is uncertain whether a mental model is a state transition system, or a more goal-oriented representation (e.g., chains of actions for satisfying specific goals). There is some experimental support for the latter view [7], but this may depend on how well the operator understands the real system (with deeper understanding corresponding to a more state-centered view). In any case, a mental model is an approximate representation of the real system—an analogy or imitation—that permits trial and evaluation of alternative courses of action, and prediction of outcomes. Being approximate, it is bound to break down occasionally by "showing properties not found in the process it imitates, or by not possessing properties of the process it imitates" [4]. In principle, we could attempt (by observations, questionnaires, or experiments) to discover the mental model of a particular computerized system held by a particular operator, and could then examine how it differs from the real system and thereby predict where automation surprises might occur for that combination of system and operator.

It is, of course, difficult and expensive to extract the individual mental models needed to perform this comparison. Fortunately, it is not necessary (although it might be interesting for experiment and demonstration): most automation surprises reported in the literature are not the result of an errant operator holding a specific and inaccurate mental model but are instead due to the design of the automation being so poor that *no* plausible mental model can represent it accurately. Quite generic mental models are adequate for the purpose of detecting such flawed designs. In the next section, I propose methods for constructing such mental models and for using them to guide development of systems that are less likely to provoke automation surprises.

2 Proposed Approach

Generic mental models can be constructed as state machines whose states and inputs are derived from information available to the operator (e.g., the position of certain switches and dials, the illumination of certain lamps, or the contents of certain displays), information in the operators' manual, and the expectation that there should be some reasonably simple and regular structure to the transitions. If a mental model is an accurate representation of the real system, there should be a simulation relationship between its state machine and that which describes the real system. Proposed simulation relations can be checked automatically using model checking or reachability analysis: these explore all possible behaviors by a brute force search and will report scenarios that cause the simulation relation to fail.

Colleagues and I have used this kind of analysis to explore automation surprises in the autopilots of the MD-88 [20], A320 [5], and 737 [21]. In each case, a plausible mental model exposed exactly the scenarios that have led to reported surprises and consequent "altitude busts," and pinpointed elements in the behavior of the actual system that preclude construction of an accurate mental model (because the behavior of the actual system depends on state transitions that are invisible at the user interface).

These experiments have convinced me of the basic efficacy of the approach, but the exciting opportunity is to move beyond detection of known flaws in existing systems to the development of a method that can be used to predict and eliminate such flaws during design. For this purpose, we need a systematic and repeatable method for constructing generic—yet credible—mental models. Work by Javaux suggests the general "shape" of such models and a process to create that shape [9].

Javaux proposes that training initially equips operators with fairly detailed and precise mental models. Experience then simplifies these initial models through two processes. The process of *frequential simplification* causes rarely taken transitions, or rarely encountered guards on transitions, to be forgotten. The process of *inferential simplification* causes transition rules that are "similar" to one another to be merged into a single prototypical rule that blurs their differences. We can imagine a computer program that applies these simplifications to turn the representation of an initial mental model into one for a more realistic "mature" one.

Given such a program that mechanizes Javaux' simplifications, I propose the following approach to development of automated systems.

- Construct a representation (i.e., a formal model, a simulation, or running code) of the actual automation design.
- Construct an initial mental model.
  This could be based on the instruction manual for the proposed design, or constructed by a process similar to that used to develop an instruction manual, or it could even be a taken directly from the actual system design.
- Check the initial mental model against the actual design.
  Using model checking techniques similar to those described previously [5, 20, 21], check whether the initial mental model is an adequate description of the actual system. If so, proceed to the next step, otherwise modify the design and its initial mental model and iterate this and the previous steps (there is no point in proceeding until we have *some* description that accurately reflects the actual system).
- Construct a simplified mental model.
  Use a mechanization of Javaux' two processes to simplify the initial mental model into a more realistic one.
- Check the simplified mental model against the actual design.
  Using model checking techniques, check whether the simplified mental model is an adequate description of the actual system. Terminate if it is, otherwise modify the design and iterate this and the previous steps.

The outcome of this process should be a system design whose visible behavior is sufficiently simple and regular that an operator, guided only by externally visible information, can accurately predict its behavior and thereby interact with it in an informed and safe manner. Furthermore, the simplified mental model produced in the process can provide the basis for an accurate and effective training manual.

It is important to note that the point of this process is not to construct a mental model that is claimed to be faithful to that of any particular operator, but to use what is known about the characteristics of mental models to coerce the design of the actual automation into a form that is capable of supporting an accurate mental model.

3 Conclusion

To predict the joint behavior of two interacting systems, we can construct formal models for each of them and calculate properties of their combination. If one of the systems concerned is a human, then we can extend this approach by modeling computational aspects of human cognitive functions. For the case of human operators of automated systems, it is known that they use simplified representations of the system as a mental model to guide their interaction with it.

The mode confusions and other automation surprises that are a source of concern in operator's interactions with many automated systems can be attributed to appallingly bad designs that admit no plausibly simple, yet accurate, mental models. By "modeling the human"—that is by explicitly constructing generic mental models, and by mechanizing plausible processes that simplify them in ways characteristic of real mental models—we can construct a touchstone that highlights cognitively complex aspects of proposed designs and guides their reformulation in ways that promotes simplicity and regularity and hence—it is hoped—reduces the number and severity of human factors problems that they provoke.

This approach suggests a number of interesting possibilities for modeling and analysis in addition to those already illustrated.

– We can examine the consequences of a faulty operator: simply endow the mental model with selected faulty behaviors and observe their consequences. The effectiveness of remedies such as lockins and lockouts, or improved displays, can be evaluated similarly.
– We can examine the cognitive load placed on an operator: if the simplest mental model that can adequately track the actual system requires many states, or a moderately complicated data structure such as a stack, then we may consider the system too complex for reliable human operation. We can use the same method to evaluate any improvement achieved by additional or modified output displays, or by redesigning the system behavior. This could provide a formal way to evaluate the methods proposed by Vakil and Hansman for mitigating the complexity of interfaces [24].
– We could take a mental model from one system (e.g., an A320) and check it against a different actual system (e.g., an A340). Discrepancies could highlight areas that should be given special attention in training programs to convert operators from one system to the other.
– We could extend the approach to multi-operator systems: for example, the air traffic control system, where the controller and the pilot may act according to different mental models of the same situation.

References

[1] Charles E. Billings. *Aviation Automation: The Search for a Human-Centered Approach*. Human Factors in Transportation. Lawrence Erlbaum Associates, Mahwah, NJ, 1997.

[2] Howard Bowman and Giorgio Faconti. Analysing cognitive behaviour using LOTOS and Mexitl. *Formal Aspects of Computing*, 11(2):132–159, 1999.

[3] S. K. Card, T. P. Moran, and A. Newell. *The psychology of human-computer interaction*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[4] K. Craik. *The Nature of Explanation*. Cambridge University Press, Cambridge, UK, 1943.

[5] Judith Crow, Denis Javaux, and John Rushby. Models and mechanized methods that integrate human factors into automation design. In Kathy Abbott, Jean-Jacques Speyer, and Guy Boy, editors, *International Conference on Human-Computer Interaction in Aeronautics: HCI-Aero 2000*, pages 163–168, Toulouse, France, September 2000. `http://www.csl.sri.com/reports/html/hci-aero00.html`.

[6] David Duke and David Duce. The formalization of a cognitive architecture and its application to reasoning about human computer interaction. *Formal Aspects of Computing*, 11(6):665–689, 1999.

[7] Edmund Eberleh. The structure of mental models: Goal directed or state centered? In F. Klix, N. A. Streitz, Y. Waern, and H. Wandke, editors, *MACINTER II: Man-Computer Interaction Research*. pages 89–103, Elsevier, 1989.

[8] The interfaces between flightcrews and modern flight deck systems. Report of the FAA human factors team, Federal Aviation Administration, 1995. Available at `http://www.faa.gov/avr/afs/interfac.pdf`.

[9] Denis Javaux. Explaining Sarter and Woods' classical results. In Nancy Leveson and Chris Johnson, editors, *Second Workshop on Human Error, Safety, and Software Design*, University of Washington, Seatle, WA, April 1998. Available at `http://www.cs.washington.edu/research/projects/safety/www/workshop/`.

[10] Denis Javaux and Véronique De Keyser, editors. *Proceedings of the 3rd Workshop on Human Error, Safety, and System Development (HESSD'99)*, University of Liege, Belgium, June 1999.

[11] Bonnie E. John and David E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, 3(4):320–351, December 1996.

[12] Philip N. Johnson-Laird. *The Computer and the Mind : An Introduction to Cognitive Science*. Harvard University Press, 1989.

[13] C. B. Jones. Tentative steps toward a development method for interfering programs. *ACM TOPLAS*, 5(4):596–619, 1983.

[14] Donald A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, NY, 1988. Also available in paperback under the title "The Design of Everyday Things".

[15] Steven Pinker. *How the Mind Works*. W. W. Norton, New York, NY, 1997.

[16] Jens Rasmussen. *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. Elsevier, New York, NY, 1986.

[17] Jens Rasmussen. Mental models and the control of action in complex environments. In D. Ackermann and M. J. Tauber, editors, *Mental Models and Human-Computer Interaction: 1*, pages 41–69. Elsevier, New York, NY, 1990.

[18] James Reason. *Human Error*. Cambridge University Press, Cambridge, UK, 1990.

[19] Frank E. Ritter, Gordon D. Baxter, Gary Jones, and Richard M. Young. Supporting cognitive models as users. *ACM Transactions on Computer-Human Interaction*, 7(2):141–173, June 2000.

[20] John Rushby. Using model checking to help discover mode confusions and other automation surprises. In Javaux and Keyser [10].

[21] John Rushby. Analyzing cockpit interfaces using formal methods. In H. Bowman, editor, *Proceedings of FM-Elsewhere*, Volume 43 of Elsevier *Electronic Notes in Theoretical Computer Science*, Pisa, Italy, October 2000. Available at `http://www.elsevier.nl/locate/entcs/volume43.html`.

[22] N. B. Sarter and D. D. Woods. How in the world did we ever get into that mode? Mode error and awareness in supervisory control. *Human Factors*, 37(1):5–19, 1995.

[23] N. B. Sarter, D. D. Woods, and C. E. Billings. Automation surprises. In Gavriel Salvendy, editor, *Handbook of Human Factors and Ergonomics*. John Wiley and Sons, second edition, 1997.

[24] Sanjay S. Vakil and R. John Hansman. Approaches to mitigating complexity-driven issues in commercial autoflight systems. In Javaux and Keyser [10].

Papers on formal methods and automated verification by SRI authors can generally be located by a search at `http://www.csl.sri.com`.