IDENTIFYING MODE CONFUSION POTENTIAL IN SOFTWARE DESIGN

Mario Rodriguez, Marc Zimmerman, Masafumi Katahira,
Maxime de Villepin, Benjamin Ingram, Nancy Leveson
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139

1. Abstract

While automation has eliminated many types of operator error, it has also created new types of technology−induced human errors. Many of these new errors are the result of what has been labeled *technology−centered automation*, where designers focus most of their attention on the mapping from software inputs to outputs, on mathematical models of required functionality, and on the technical details and problems internal to the computer: Little attention is given to evaluating software in terms of whether it provides transparent and consistent behavior that supports operators in their monitoring and control tasks. The goal of our research is to create and evaluate a methodology for integrated design of complex systems, including design of the automation and the human tasks, that minimizes human error through appropriate system and operator task design. The methodology is based on formal modeling, simulation, and analysis techniques for the software behavior, the user model of the system, and the operator tasks. This paper describes the human factors aspects of our approach using as an example the vertical flight control logic for a realistic aircraft flight management system FMS. Although the MD−11 FMS was used to derive the example for our case study, we made up much of the information due to our lack of knowledge about the design and the rationale of the real MD−11 design, and nothing in this paper should be taken as applying to that aircraft's actual automation. A companion paper at this conference describes the parts of the methodology focusing on the modeling and analysis of the automation itself.

2. Introduction

Advances in computing have had a significant impact on the aviation industry. Automation is assuming greater roles in commercial aircraft, adding flexibility and enhanced capabilities. At the same time, humans have had problems coping with this new automation, and breakdowns in the interaction of pilots and automated systems are becoming more common. Pilots are at times lost in the automation: They ask questions like "What is the computer doing?" "Why is it doing that?" "What will it do next?" "How do I stop it from doing that?" and "How do I get it to do what I want it to do?"[13] The resulting confusion, sometimes called mode confusion, has been cited as a contributing cause of accidents and serious incidents, highlighting the need to develop ways to prevent it.

Safety−critical systems, and particularly those with a high level of complexity, require a sophisticated design process. A methodology to support such a process will not only address unsafe and problematic system features, but will be able to do so early in the design process when changes can still be made relatively easily.

One of the long−term goals of the MIT Software Engineering Research Lab (SERL) is to create and evaluate such a methodology for integrated design of the automation and human tasks in complex systems. The methodology will be based on formal modeling, simulation, and analysis techniques starting with a user model of the system and generating appropriate and safe software and task models. The modeling and analysis tools should assist engineers and human factors experts in enhancing situation awareness, minimizing human errors such as those related to mode confusion, enhancing learnability, and simplifying the training of humans to interact with the automation.

Our first step in achieving these ambitious goals is to determine how to use modeling and analysis to detect or prevent automation features that can lead to operator mode confusion. To accomplish this, we use three different types of models:

1) a *user model* of the automation behavior: this model is necessarily a simplification or abstraction of the actual possible automation behavior (which is often not completely understood even by the designers),

2) an *operator task model* resulting from a task analysis that identifies the major tasks of the human controller and then breaks these down into subtasks, eventually specifying the tasks down to the level of the key presses, voice communications, display cues, etc. involved in performing the task, and

3) a detailed specification of the *blackbox automation behavior*, that is, the blackbox behavioral requirements of the automated system.

In a companion paper at this conference, we describe our approach to modeling the automation behavior [14]. In this paper, we describe the two other types of models (user and task) we have found helpful in detecting system features that can lead to mode confusion. We then describe a specific case study of the approach on a realistic vertical flight control system. The goals of the case study were to show scalability and efficacy of the approach for complex systems.

Background and Related Work

In our initial attempt to study mode confusion, Leveson et al. identified six categories of system design features that can lead to mode confusion errors: ambiguous interfaces, inconsistent system behavior, indirect mode transitions, lack of appropriate feedback, operator authority limits, and unintended side effects [8]. Leveson and Palmer made an initial attempt to learn more about how to identify such design flaws using a pilot error that has been the subject of many ASRS reports [9]. A small but relevant part of the flight control system's blackbox software behavior was formally modeled and the approach to identifying the cause of the error −− a software requirements flaw −− was demonstrated. One result of this case study was a recognition that such mode confusion errors could only be identified if the software (automation) model was augmented by a simple model of the controller's view of the software's behavior (a user model) −− the formal software specification was not enough.

In subsequent research, we experimented with task analysis and using an operator task model [1]. Most task analysis involves very expensive simulator studies. While traditional task analysis is very useful in learning about usability and human preferences, it is not a very efficient nor effective way to learn about potential human errors related to safety. Not only are such errors rare and situation dependent, but humans tend to use automation differently over time. Formal analysis techniques appear to be a useful adjunct to standard usability analysis, which is still important and necessary.

By integrating executable models of the automation and the human tasks, it should be possible to simulate the interaction between humans and computers in controlling the system. The analyst will be able to examine the ramifications of certain automation and human task design decisions on overall safety.

The closest approach to ours is that of Degani and of Harrison and Fields. Degani [3] provides a different classification of modes and of mode confusion detection criteria. His classification of mode confusion detection criteria is a subset of ours. Degani also developed a task modeling framework, known as OFAN, which is based on the Statecharts language. Our experience in using Statecharts on real systems found it to be inadequate for our goals. Therefore, we have designed a blackbox automation requirements specification and modeling language called SpecTRM−RL, which includes specification of modes and which we have found scales to large and complex systems [6]. Harrison and Fields use CSP models to attempt to achieve similar goals, but this language is very formal and probably not usable without extensive training in discrete mathematics.

Javaux takes a more psychological approach. He uses a finite state machine to describe a cognitive mental model, which he uses to identify

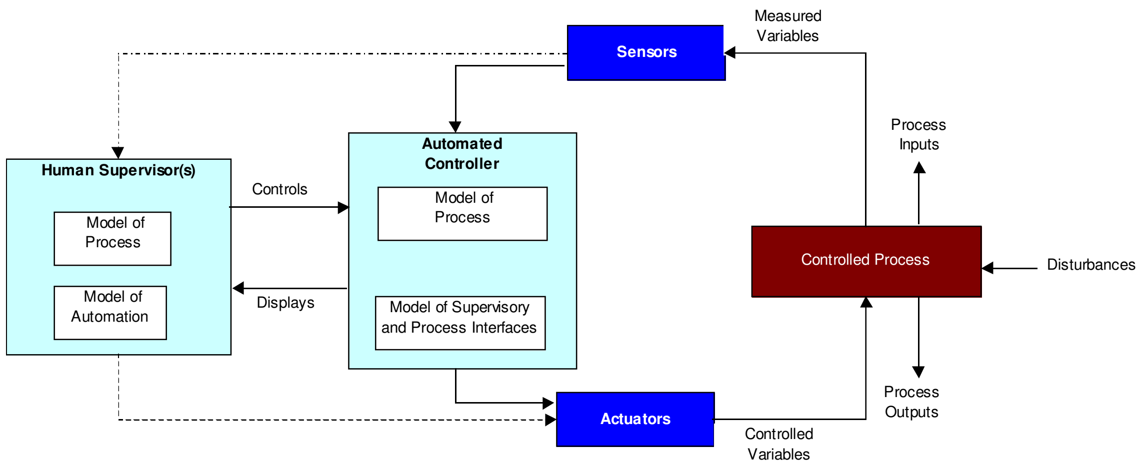

Figure 1. Basic Control Loop

potential instances of mode confusion [4,5]. In contrast, we do not try to model human cognition or human mental models but instead model the blackbox behavior of the automation that the user expects and depends upon and the required steps needed to complete a given task. Modeling the actions involved in an operator task potentially allows analyzing the interaction of the operator with a formal model of the rest of the system.

Vakil and Hansman [12] have proposed the use of predictability as a measure of system complexity, which they claim is directly related to mode confusion. Their work also suggests approaches to mitigating complexity–driven issues in commercial flight systems.

Other researchers have applied Leveson's safety criteria for mode confusion and tried to use formal methods techniques (e.g., model checking) to detect these error forms in system specifications [10,11].

3. Approach

To control a complex system, every controller must have a model of the general behavior and current state of the controlled process (see Figure 1). This model may be embedded in the control logic of an automated controller or in the mental model of a human controller or both. The model is updated and kept consistent with the actual system state through various forms of feedback from the system to the controller. In addition, any human controller (who is usually directly controlling the automation but only indirectly the controlled process) must also have a model (i.e., some understanding) of the expected automation behavior. When these models diverge from the actual state of the controlled process (or the actual automation behavior in the case of the human supervisor of the automation) erroneous control commands based on an incorrect model can lead to an accident [Safeware]. The situation becomes more complicated when there are multiple human and automated controllers because the models of the various controllers must also be kept consistent.

Errors result from inconsistent models of the controlled process. Models may diverge either because they were incorrect or incomplete to begin with (they do not adequately reflect the behavior of the controlled system) or because they are improperly updated due to incorrect feedback about the state of the modeled system. Note that there are several sources of inconsistency due to improper feedback.

We believe that explicitly specifying and validating these models during system design will allow engineers to identify and eliminate error–prone features of automation and interfaces and to assist in task analysis and development of operator training and reference materials. The models we use for specifying the automation behavior are described elsewhere. With respect to the human operator or supervisor, we use two models: the user model of the automation behavior and a task model of the operator's tasks in interacting with the automation and the system.

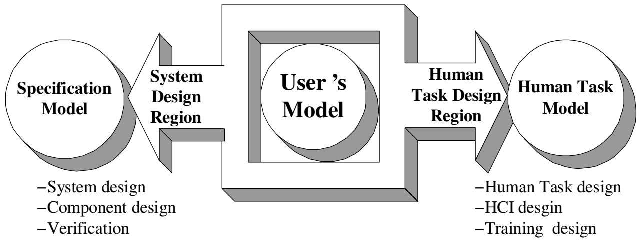

Figure 2. User−centered Approach to System Design.

If the system already exists, a human task model and user model (expected system behavior) can be extracted from the operator's manual and other operator documentation and training materials for the given system. Ideally, of course, these models would have been built first and the tasks and detailed automation specifications as well as training and operator materials will have been derived from the user model (see Figure 2). In this way, consistency can be ensured as the system is designed, eliminating the need to verify consistency at the end of the development process.

4. Flight Management System (FMS) Case Study

To evaluate the feasibility of this approach, we performed a case study on the vertical flight control system for a realistic FMS. In developing the user and task models for this work, we focused specifically on the descent phase of flight, including Early Descent, On−Path Descent, and Late Descent. We selected the descent phase for analysis because it has been documented as the flight phase during which most accidents or incidents occur. Information required for the pilot's task model and user model (expected system behavior) was created using the MD−11 Flight Management System's Pilot's Guide as an example. As noted earlier, the case study example does not match the real MD−11 FMC, but the documentation we had helped us understand the type of knowledge expected of the pilots and how this knowledge is used to operate the automation.

The components of the modeling language are shown in Figure 3. Steps required to complete a task are represented by s*tates*. A *transition* is defined as the process of changing from one state to the next. Conditions that trigger transitions are called *events* and *action* describes a result or output from the transition. A *communication point* links different models together. In our case, the human task model communicates its outputs (or actions) via a communication point to the system model. In this way, we are able to model human−computer interaction.

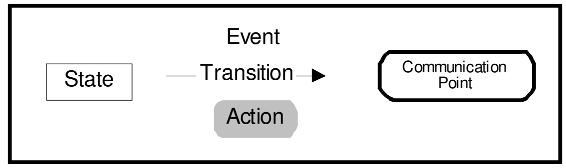

Figure 3. Components of Our Task Modeling Language

A *state* is represented by a square box. A *transition* is represented by an arrow from one state to the next, and *events* are shown in text above the transition; *actions* are represented by text with gray shade beneath the transition. Finally, c*ommunication points* are round boxes and refer to another component in the model.

Figure 4 shows the user and pilot task models we created. Relevant parts of a display model is also included that shows the feedback regarding the state of the automation provided to the pilot.

Our analysis of these models is broken into two parts. First, we examined the relationship between the operator task model and the user models in order to identify potential instances of mode confusion based on Leveson's categories. Then we compared the user model (created from information in a typical pilot's guide and other sources) with the actual automation design (obtained from our example FMS specification) to identify potential discrepancies between what the operator is told about the system behavior and the actual implemented system behavior.

Although we considered all six sources of mode confusion identified by Leveson, only two examples are presented here −− indirect mode changes and inconsistent system behavior. The next two subsections describe these features as well as instances of them found in our models. The third subsection describes our comparison of the user model (expected system behavior as described in the pilot manual) with the actual implemented system behavior.

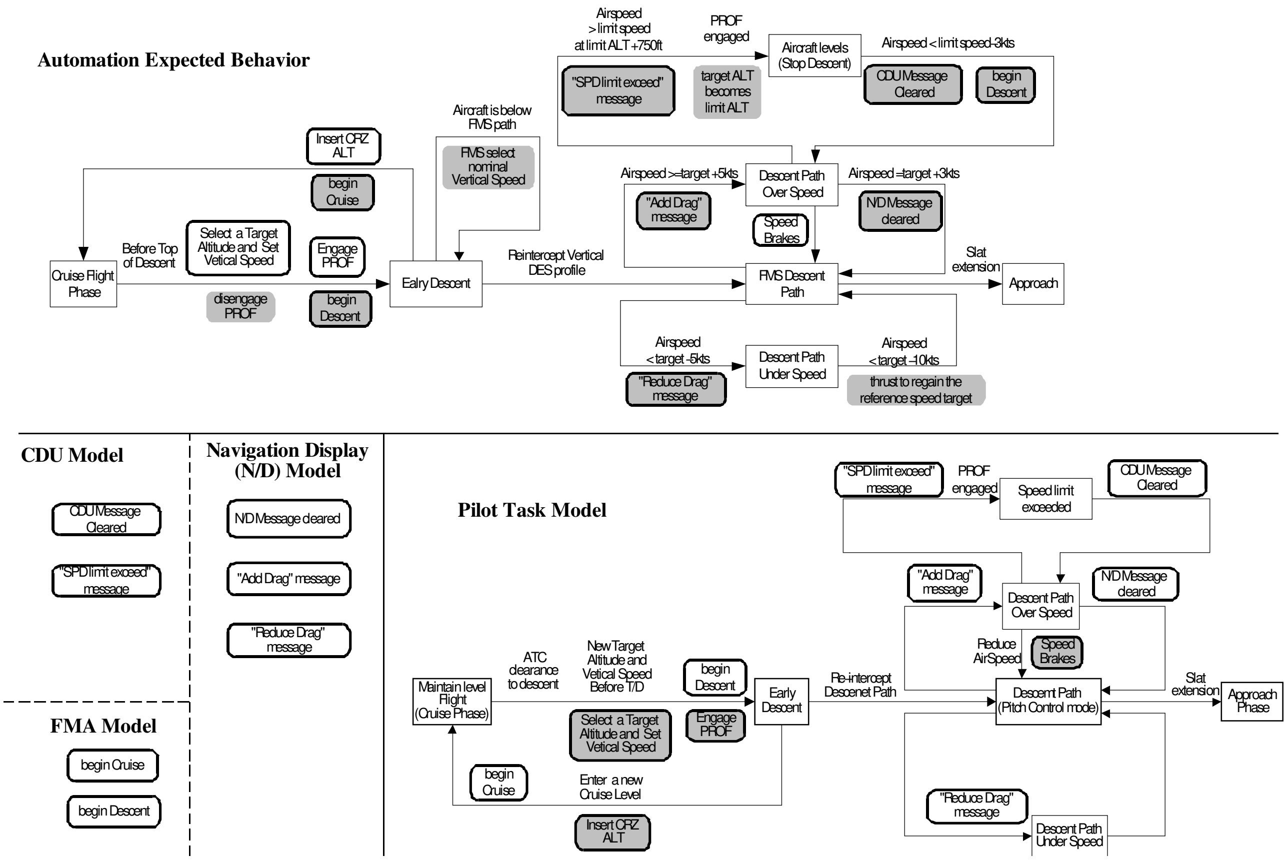

Figure 4. User and Task Models

4.1 Indirect Mode Changes

An indirect mode change occurs whenever there is a change in mode by the automation without any explicit command from the operator. We show how an indirect mode change can be detected with our models using an example from our FMS case study.

On a descent path, the automation maintains continuous altitude control along a predefined path. When the aircraft speed exceeds the target speed by more than five knots, the automation expects the pilot to manually apply the speed brakes. However, if the airspeed continues to increase and exceeds the speed limit at an altitude 750 feet higher than the speed limit altitude, the FMS will automatically take action. It first displays the "Speed limit exceed" label in the Control Display Unit. It then replaces the target altitude by the speed limit altitude and stops the descent. The aircraft then is considered to be in a descent path overspeed scenario. This triggers a mode change in the automation from the Descent Path mode to the Descent Path Overspeed mode. The automation accordingly levels the aircraft, temporarily ending the descent. Once the airspeed becomes three knots less than the speed limit, the automation clears the message in the control display unit (CDU) and begins the descent again. This scenario is shown in Figure 5.

Examination of the task models clearly shows that this is an instance of an indirect mode change. The automation transitions to the Descent Path Overspeed mode based solely on the aircraft’s altitude and speed – there is no direct interaction with the pilot. Further, by viewing the task and automation models simultaneously, we were able to identify lack of feedback as a contributing factor to a potential mode confusion error. According to the design of the automation, the information displayed by the Flight Mode Annunciation (FMA) does not change throughout the operation. According to the pilot’s model, the only event that the pilot observes is the appearance (and

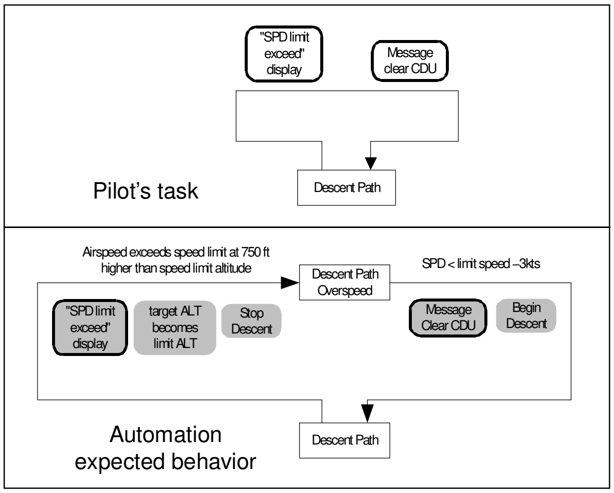

Figure 5. Indirect Mode Transition.

disappearance) of a display. The pilot has no way to know that the FMS has switched to the Descent Path Overspeed mode.

Our models not only help identify sources of mode confusion, but also help suggest improvements to account for the error−prone features identified. For example, Figure 6 shows how including *Stop Descent* and *Begin Descent* as output alerts or actions in the automation model might make the pilot aware that a mode transition has taken place. While this does not eliminate the indirect mode transition, it might lessen the likelihood that it would result in a mode confusion error. Human factors experts would need to determine what responses should be made to potentially hazardous indirect mode transitions that are detected during analysis.

4.2 Inconsistent System Behavior

A second example of a potential source of mode confusion is inconsistent system behavior. A

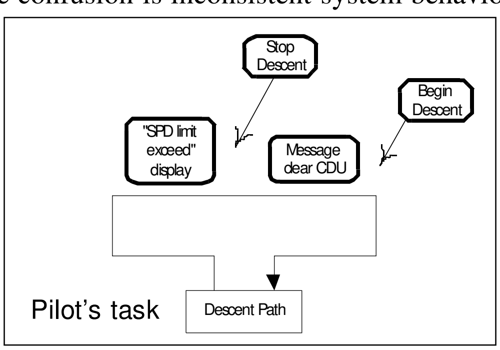

Figure 6. A Way to Improve Feedback.

fully consistent system allows similar tasks and goals to be associated with similar or identical actions. When the system is inconsistent by this definition, the operator will have more difficulty learning to operate the system and creating an appropriate mental model. The following is an example from our case study.

During an On−Path Descent, two events can occur. First, if the aircraft speed is more than the speed target plus five knots, then the "ADD DRAG" message is displayed in the navigation display. However, if the aircraft speed is below the speed target by less than five knots, then the "REDUCE DRAG" message is displayed in the navigation display.

In the first case, the aircraft speed is too high and an action from the pilot is required. In this case, the pilot is required to engage the speed brakes. On the other hand, the second case requires no action from the pilot. If the speed decreases to 10 knots below the target, then the FMS will apply thrust to regain speed.

Figure 7 shows a visual formalism of this scenario in our task modeling language. Our formalism allows us to identify inconsistent system behavior quickly. It is manifested clearly as a lack of symmetry in the pilot model diagram.

4.3 Inconsistent Documentation

Mode confusion and other types of errors can occur when the user's model of the automation's design differs from its actual design, as reflected by differences between the automation behavior implied by the pilot's guide and other operation and training documentation and the actual implemented automation behavior.

As an example of this, consider the transition from Cruise Phase to Early Descent in our models. In comparing the user model derived from the pilot's guide and the automation model generated from the FMS system specification, we found both an ambiguity in the definition of the cruise and early descent phases and differences in the conditions that are specified as triggering the flight phase transition. In addition, the models explicitly denote automation control modes that are never annunciated to the pilot, such as the Flight Control Computer Engaged mode, the vertical guidance type, and the vertical guidance control mode. It is clear that pilots should not have to know all the

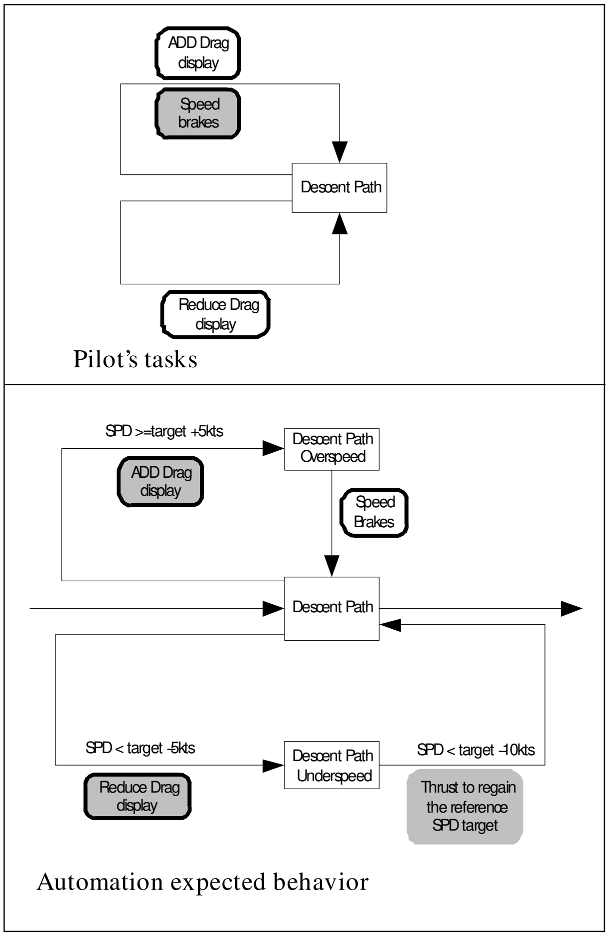

Figure 7. Visual Model of Inconsistent System Behavior.

potential automation control modes. However, there may be instances where lack of information can lead to pilot error and it would be useful to determine what information the pilot needs to successfully supervise the automation. For example, during the transition to Early Descent mode, the pilot must ensure that the current profile is engaged. Using our user model, we see that the system does indicate whether the current profile is engaged. However, it does not include information regarding factors used by the automation (other than the pilot command) that can indirectly affect the "engaged" status of the current profile. There is then a potential for an indirect mode transition, and thus mode confusion although a human factors expert would need to determine whether this is actually a problem or not.

5. Conclusions

By using a visual formalism consisting of a user model of the automation, the actual automation design, and an operator task model, we have been able to detect several potential cases where mode confusion could arise in a realistic aircraft FMS. Potential for mode confusion was identified using two different approaches: (1) comparing the task model to the user model of the automation and (2) comparing the user model of the automation to the actual automation behavior model. The resulting information can be used by human factors experts in evaluating the system design.

Our visual formalism serves additional purposes as well, such as providing a clear and concise means by which system designers can communicate about human−computer interaction issues and providing information that can be used to generate user documentation and training materials.

Our visual task modeling language can be translated into our formal requirements specification language, SpecTRM−RL. SpecTRM−RL offers a suite of analysis tools including execution of the models, completeness and consistency checking, and various safety analysis tools that can help assure total system safety. Future work will involve automating this translation, thereby making our task models more amenable to formal analysis.

In the long term, we would like to design a methodology and a set of automated tools to support it that starts with the user model of the system. The detailed automation behavior, operator tasks, and documentation and training materials would be generated starting from this user model. Analysis tools would be used to evaluate various properties, including safety, during early system development phases.

6. References

[1] Brown, M. and Leveson, N.G. (1998). "Modeling Controller Tasks for Safety Analysis." Presented at the Workshop on Human Error and System Development, Seattle, April 1998.

[2] Carroll, J.M. and Olson, J.R. "Mental Models in Human−Computer Interaction." In M.Helander

(Ed). Handbook of Human−Computer Interaction, Elsevier Science Publishers, pp.45−65, 1988.

[3] Degani, A., (1994). Modeling Human−Machine Errors: on Modes, Error and Patterns of Interaction. Doctoral Thesis. Atlanta, GA: Georgia Institute of Technology.

[4] Javaux, D. (1998). "An Algorithmic Method For Predicting Pilot−Mode Interaction Difficulties." 17th Digital Avionics and Systems Conference. October 31 − November 6, Bellevue, WA.

[5] Javaux, D. and De Keyser, V. (1998). "The cognitive complexity of Pilot−Mode Interaction. A possible explanation of Sarter and Wood's classical results. In G. Boy & C. Graeber (eds.), Proceedings of the International Conference on Human−Computer Interaction in Aeronautics, May 27−29, Montreal, Canada.

[6] Leveson, N. (2000) "Completeness in Formal Specification Language Design for Process Control Systems." To appear in the Proceedings of Formal Methods in Software Practice.

[7] Leveson, N. Safeware: System Safety and Computers. Addison−Wesley, New York, 1995.

[8] Leveson, N. et al. (1997). "Analyzing Software Specifications for Mode Confusion Potential." Presented at the Workshop on Human Error and System Development, Glasgow, March 1997.

[9] Leveson, N. and Palmer, E (NASA Ames Research Center). "Designing Automation to Reduce Operator Errors." In the Proceedings of Systems, Man, and Cybernetics Conference, Oct. 1997

[10] Luttgen, G. and Carreño,V. (1999). "Analyzing Mode Confusion via Model Checking." NASA/CR−1999−209332. ICASE Report.

[11] Miller, S.P., and Potts, J.N. (1999). "Detecting Mode Confusion Through formal modeling and Analysis." NASA/CR−1999−208971. [ICASE REPORT]

[12] Vakil, S., Hansman, R.J. (1999). "Approaches to Mitigating Complexity−Driven Issues in Commercial Autoflight Systems." 3rd Workshop on Human Error, Safety, and System development. Liege, Belgium, June 7−8.

[13] Wiener, E.L. and Curry, R.E. (1980) Flight−deck automation: Promises and problems. NASA Technical Memorandum 81206, NASA Ames Research Center.

[14] Zimmerman, M. et al.(2000) Making Formal Specifications Practical. To appear in the 19th Digital Avionics and Systems Conference. Philadelphia, PA, October 7−13.