# Demonstration of a Safety Analysis on a Complex System[*]

N. Leveson, L. Alfaro, C. Alvarado, M. Brown,
E.B. Hunt, M. Jaffe, S. Joslyn, D. Pinnel,
J. Reese, J. Samarziya, S. Sandys, A. Shaw, Z. Zabinsky

University of Washington
Seattle, WA 98195

For the past 17 years, Professor Leveson and her graduate students have been developing a theoretical foundation for safety in complex systems and building a methodology upon that foundation. The methodology (as described in her book *Safeware* [2]) includes special management structures and procedures, system hazard analyses, software hazard analysis, requirements modeling and analysis for completeness and safety, special software design techniques including the design of human-machine interaction, verification, operational feedback, and change analysis.

The *Safeware* methodology is based on system safety techniques that are extended to deal with software and human error. Automation is used to enhance our ability to cope with complex systems. Identification, classification, and evaluation of hazards is done using modeling and analysis. To be effective, the models and analysis tools must consider the hardware, software, and human components in these systems. They also need to include a variety of analysis techniques and orthogonal approaches: There exists no single safety analysis or evaluation technique that can handle all aspects of complex systems. Applying only one or two may make us feel satisfied, but will produce limited results.

We report here on a demonstration, performed as part of a contract with NASA Langley Research Center, of the Safeware methodology on the Center-TRACON Automation System (CTAS) portion of the air traffic control (ATC) system and procedures currently employed at the Dallas/Fort Worth (DFW) TRACON (Terminal Radar Approach CONtrol). CTAS is an automated system to assist controllers in handling arrival traffic in the DFW area.

Safety is a system property, not a component property, so our safety analysis considers the entire system and not simply the automated components. Because safety analysis of a complex system is an interdisciplinary effort, our team included system engineers, software engineers, human factors experts, and cognitive psychologists.

[*]This work was partially supported by grants from NASA Langley and NASA Ames.

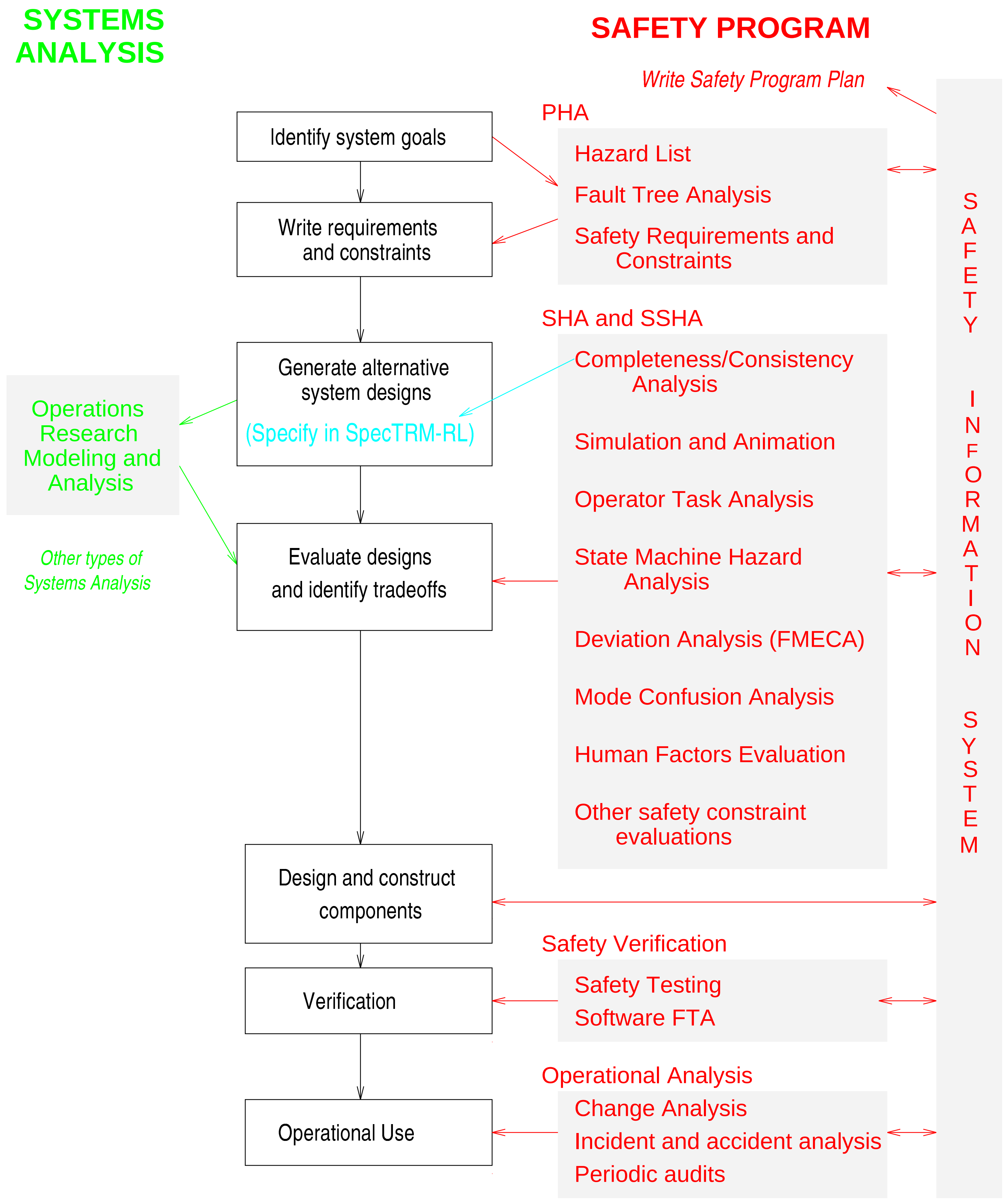

SYSTEMS ANALYSIS
SAFETY PROGRAM
Write Safety Program Plan
PHA
Identify system goals
Hazard List
Fault Tree Analysis
Safety Requirements and Constraints
Write requirements and constraints
SHA and SSHA
Generate alternative system designs
(Specify in SpecTRM-RL)
Operations Research Modeling and Analysis
Completeness/Consistency Analysis
Simulation and Animation
Operator Task Analysis
Evaluate designs and identify tradeoffs
Other types of Systems Analysis
State Machine Hazard Analysis
Deviation Analysis (FMECA)
Mode Confusion Analysis
Human Factors Evaluation
Other safety constraint evaluations
Design and construct components
Safety Verification
Verification
Safety Testing
Software FTA
Operational Analysis
Operational Use
Change Analysis
Incident and accident analysis
Periodic audits
SAFETY INFORMATION SYSTEM

Figure 1:

Figure 1 shows our design of a system safety program. Such a process is highly iterative and includes continual updating of what has been done previously as new information is gained through the system development process. In order to make the diagram less cluttered, the backward links are not shown, but note that the safety information system assists in the iteration process. An effective safety information has been found to rank second only to top management concern about safety in discriminating between safe and unsafe companies matched on other variables [1].

The center column of Figure 1 shows the standard system engineering tasks while the right column shows the special safety tasks and how they interact. We also performed some operations research modeling and analysis to demonstrate how information might be obtained and used to assist in making tradeoffs between alternative system designs.

We can only only provide an overview of the safety assessment and approach in this paper. Interested readers will find the final project report (containing many more details) at URL: http://www.cs.washington.edu/projects/safety/www/dfw. The assessment contained the following components:

**Preliminary Hazard Identification and Standard Hazard Analyses.** A safe system is one that is free from accidents or unacceptable losses. Accidents result from hazards, where a *hazard* is defined as a system state or set of conditions that can lead to an accident (given certain other, probably uncontrollable or unpredictable environmental conditions). In safety engineering, any safety assessment starts with identifying and analyzing the system for hazards. Once the hazards are identified, steps can be taken to eliminate them, reduce their likelihood, or mitigate their effects.

In addition, some hazard *causes* can be identified and eliminated or controlled. Although it is usually impossible to anticipate all potential causes of hazards, obtaining more information about them usually allows greater protection to be provided with fewer tradeoffs, especially if the hazards are identified early in the design phase. The hazards and the hazard causes can be used to write system safety requirements and constraints.

We performed a standard PHA using the DFW TRACON as an example, and wrote some preliminary safety requirements and constraints for CTAS and for air traffic control in general. As part of the PHA, we produced parts of a fault tree for TRACON operations that are related to the operation of CTAS. The information we generated was used in the demonstration of our analysis techniques.

**Modeling.** In order to do more than an evaluation of only the high-level ATC concept, a detailed specification or model of the behavior of the system components is required. A high-level design may appear to be safe while the detailed design contains hazardous component interactions. The hazards and design constraints identified in the first step must be traced to the system components, and assurance must be provided that the hazards have been eliminated or mitigated and the design constraints satisfied. Although theoretically this type of process could be performed on the detailed design of the system (including code if the component is a computer), the only

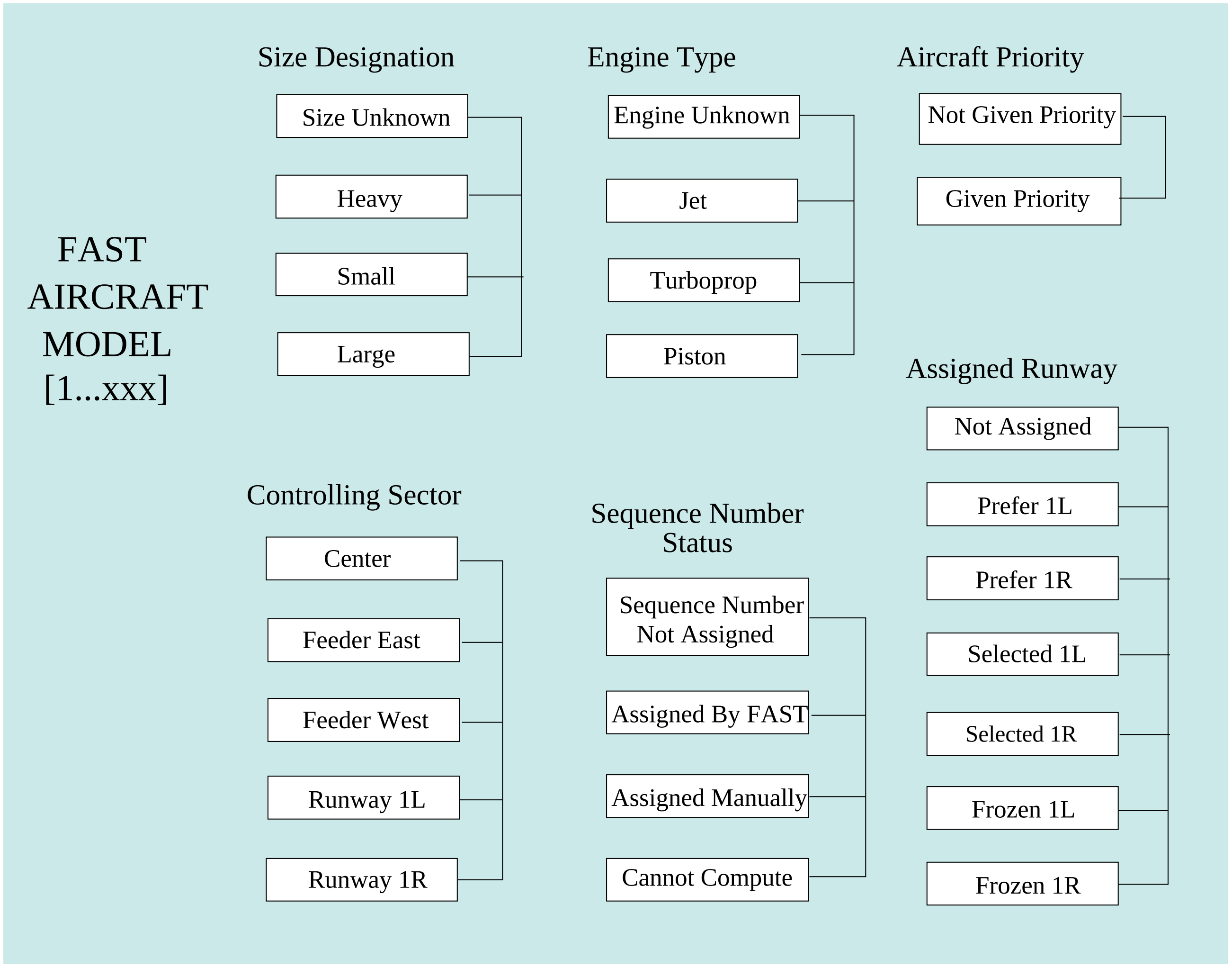


Figure 2:

practical approach is to provide hierarchical models and break the process up into steps. We built a state-based, blackbox model of the components of the DFW TRACON using a language called SpecTRM-RL that is readable and understandable with minimal training but has a formal foundation that allows analysis. Figure 2 shows a small piece of the model.

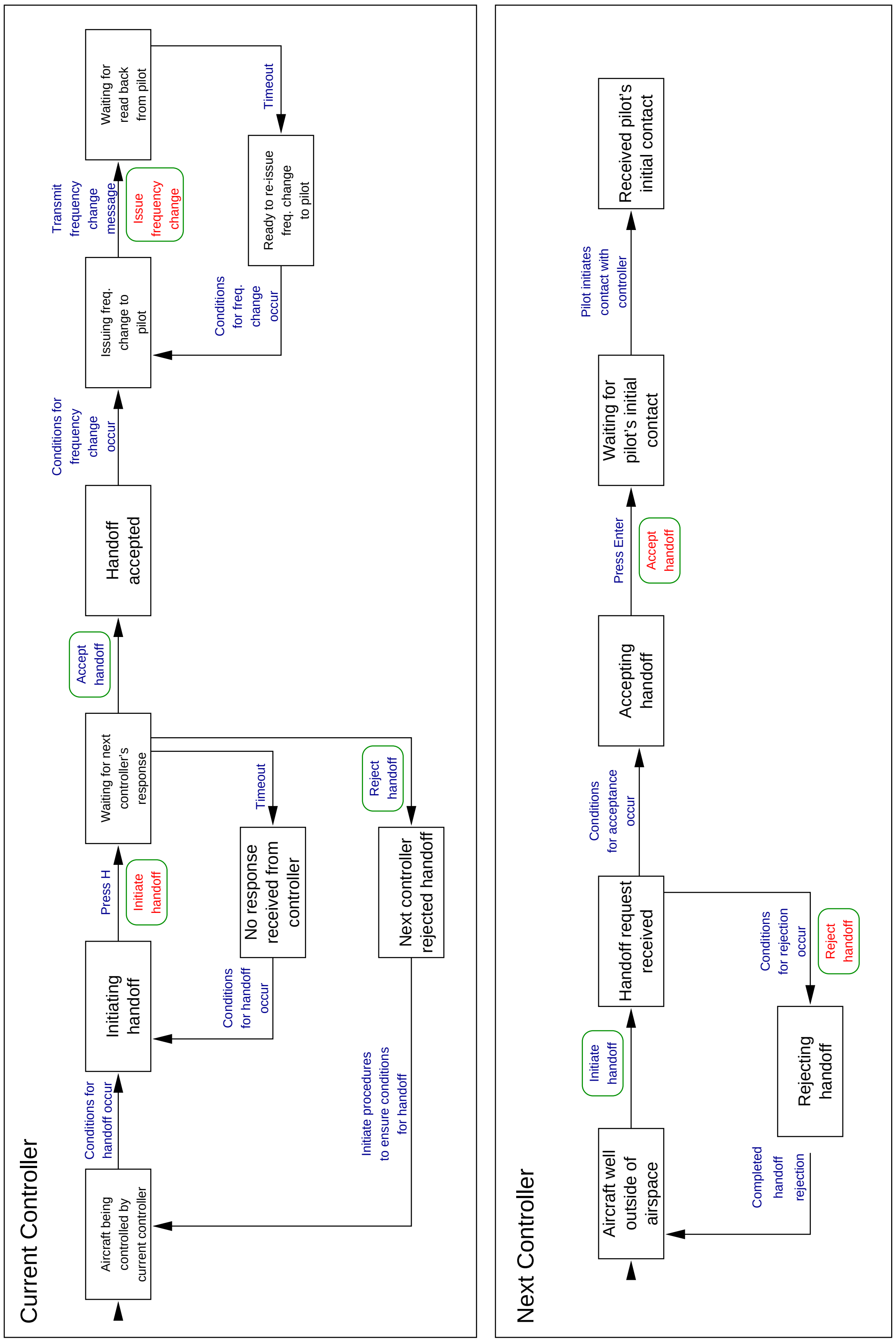
**Simulation and Animation.** Our models are executable and we have visualization tools to build animations appropriate to the model's domain (in this case, air traffic control). Our IB toolkit is an interface and visualization builder that allows users to build graphical user interfaces and animations of SpecTRM-RL models quickly and easily. Once the graphical design of the animation is completed, it can be attached to a SpecTRM-RL model to control the model's execution, display the execution outputs, or display internal states or actions of the model during execution.

As an example, we have created an animation that shows the behavior of aircraft within the TRACON area as the formal model is stepped through its states for a given set of inputs. This animation shows the controlled airspace and the aircraft in it, a timeline containing each aircraft's estimated time to landing, and an altitude indicator for the aircraft. As the model execution proceeds, the designer can see the aircraft moving along their projected flightpaths. During the simulated execution, the designer may click on parts of the animated display to get selected aircraft information such as speed, assigned runway, and assigned landing sequence number.

**Controller Task Analysis.** Humans form an important part of the ATC system, and they cannot be ignored in any safety analysis. We model operator procedures in the same language (SpecTRM-RL) as the other parts of the ATC system in order to allow executing and analyzing the ATC model as a whole. However, we use our visualization tools to display the information for human review in a more appropriate format. Figure 3 shows part of the task performed during a handoff procedure. This new language is used to display the nominal tasks that the controllers and pilots perform. We appreciate that humans do not necessarily perform tasks in the expected way. However, the first step in a safety analysis is to determine whether the nominal or expected behavior is safe. The implications of human error or deviations from nominal behavior is investigated in our other analyses. For the demonstration of the methodology, we built animations of the controller task models, including one that indicates through color coding the current cognitive and perceptual load on the pilot.

**Completeness and Consistency Analysis.** Accidents involving computers can usually be traced to incompleteness or other errors in the software requirements specification, not coding errors [3, 2]. Once a blackbox model of the required system behavior has been built, the model can be evaluated as to whether it satisfies design criteria that are known to minimize errors and accidents. We have developed such a set of criteria to identify missing, incorrect, and ambiguous requirements related to safety in process control systems. These criteria include much more than the mathematical completeness that is checkable on most formal models, although we can check this too.

**State Machine Hazard Analysis.** Hazard analysis techniques that use backward search start with a hazardous state and determine the events that could lead to this state. The analysis starts from hazards identified during the preliminary hazard analysis and identifies their precursors. The information derived about both normal and failure behavior can be used to redesign the system to prevent or minimize the probability of the hazard. We have found that the backward reachability graph explodes quickly for complex systems. Many of the branches are physically impossible or are less interesting than others, so we currently implement the process by having the analyst start the model in a hazardous state and work back one step at a time, using our backward simulation capability. At each step, the analyst prunes the tree of irrelevant branches and decides which branch to follow next.



Current Controller
Aircraft being controlled by current controller
Conditions for handoff occur
Initiating handoff
Press H
Initiate handoff
Waiting for next controller's response
Accept handoff
Handoff accepted
Conditions for frequency change occur
Issuing freq. change to pilot
Transmit frequency change message
Issue frequency change
Waiting for read back from pilot
Timeout
Ready to re-issue freq. change to pilot
Conditions for freq. change occur
Timeout
No response received from controller
Conditions for handoff occur
Reject handoff
Next controller rejected handoff
Initiate procedures to ensure conditions for handoff
Next Controller
Aircraft well outside of airspace
Initiate handoff
Handoff request received
Conditions for acceptance occur
Accepting handoff
Press Enter
Accept handoff
Waiting for pilot's initial contact
Pilot initiates contact with controller
Received pilot's initial contact
Conditions for rejection occur
Reject handoff
Rejecting handoff
Completed handoff rejection

Figure 3:

**Deviation Analysis.** Forward search techniques start with an initiating event and trace it forward in time. Applying a Failure Modes and Effects Criticality Analysis (FMECA), HAZOP, or any other forward analysis technique to software is complicated by the large number of ways that computers can contribute to system hazards. In addition, when a forward analysis traces a failure to a computer component, it may be difficult to determine what affect the failure will have on the software behavior and outputs, particularly before the software has been implemented. We solve this problem using a new forward analysis technique for software called Software Deviation Analysis (SDA).

SDA is based on the underlying assumption that many accidents are the result of deviations in system variables. A deviation is the difference between the actual and correct values. SDA can determine whether a hazardous software behavior (usually an output) can result from a class of input deviations, such as measured aircraft speed too low (the measured or assumed speed is less than the actual speed). SDA is a way to evaluate system components for *robustness* (in the security community this is often called *survivability*) or how they will behave in an imperfect environment.

**Human Error Analysis.** Humans are and will continue to be for quite some time an important part of any air traffic control system. Therefore, an effective safety program cannot just look at the automated parts of the system but must consider the impact of human error on the system and the effect of system design on human error. Increased automation in complex systems has led to changes in the human controller's role and to new types of technology-induced human error. We approach this problem in two ways.

The first is a method we are developing for using our formal system models to detect error-prone automation features early in the development process while significant changes can still be made. We have taken what has been learned by cognitive psychologists from past accidents, incidents and simulator studies, and identified a set of automation design flaws that are likely to induce errors in humans that interact with the automation. The information produced from this *mode confusion analysis* can be used to redesign the automation to take out the error-inducing features or to design better human-machine interfaces, operator procedures, and training programs.

Our second approach to safety analysis of human error is a more traditional form of human factors analysis. For this DFW CTAS study, we first looked at the types of human errors in the current ATC system and then performed a comparative analysis of the controller's job before and after CTAS. Potential safety issues were identified involving decreased situation awareness, increased vigilance requirements, and skills degradation. Normally this step would be followed by running experiments to determine the effect of the changes on human performance with respect to these identified safety issues. However, the time limitations of this study did not allow us to perform this final step. Instead, we described some relevant hypotheses and an experimental paradigm for evaluating these hypotheses.

**Operations Research Modeling and Analyses.** Safety is not the only goal of an air traffic control system. The systems engineering process involves making tradeoffs between various goals such as safety, throughput, and fuel efficiency. If a proposed upgrade turns out to degrade safety significantly while providing only minimal benefit in terms of throughput or fuel economy, then it may not be worthwhile to implement it or an alternative design may provide a better result. We used a discrete-event simulation to compare the total delay and fuel burn for five different scheduling algorithms that may be used to control aircraft from arrival at an enroute ATC center to their arrival at a feeder gate into the TRACON.

The scheduling algorithms ranged from a basic algorithm that does not allow any passing or altitude overtakes and simply delays aircraft, to a more sophisticated scheduling algorithm that allows passing and introduces a freeze horizon. The scheduling algorithms may be viewed as having different levels of safety. For example, two of the five scheduling algorithms seek the path for each aircraft that minimizes fuel burn, even though this path may result in two or more advisories from the controller. The other three scheduling algorithms minimize the number of advisories issued to reduce the number of communications between controller and pilot and thereby minimize an important causal factor in accidents.

The models can provide information such as the amount of delay, or the amount of fuel consumed for various air traffic profiles operating under different scheduling algorithms. We showed how these models can be used for tradeoff studies, in order to evaluate proposed scheduling algorithms in CTAS.

**Intent Specifications.** The types of formal modeling and hazard analysis described so far provide a comprehensive assessment methodology. The most effective way to create a safe system, however, is to build safety in from the beginning. The preliminary hazard analysis should start at the earliest concept formation stages of system development and the information should be used to guide the emerging design. Later, system and subsystem hazard analysis information is used to evaluate the designs and make tradeoff decisions.

*Intent specifications* support both (1) general system development and evolution and (2) system safety analysis. The design rationale and other information that is normally lost during development are preserved in a single, logically structured document whose design is based on fundamental principles of human problem solving. Safety-related requirements and design constraints are traced from the highest levels down through system design, component design, and into hardware schematics or software code. An important feature of intent specifications is that they integrate formal and informal specifications and enhance their interaction.

We did not have the information necessary to build a complete intent specification for CTAS. Instead, we built a sample intent specification for TCAS, an airborne collision avoidance system with similar aircraft tracking functions. The sample TCAS system specification (800 pages long) can be viewed at the following URL: http://www.cs.washington.edu/research/projects/safety/www/intent.ps

**Other Parts of a Complete Safety Program.** We did not perform any safety testing or evaluation of the actual code, but this would obviously be part of any complete safety program. During operational use of the system, incident and accident data would be collected and analyzed along with analysis of any changes or modifications. Change analysis uses the same procedures as those used during the original development. Our modular models along with the tracing of safety-related constraints to the design and code that is part of an intent specification should make it easier to perform the change analysis. In addition, periodic audits should be made to ensure that the assumptions underlying the safety analysis (which are recorded in the intent specification) have not been violated by the natural changes that occur in any system over time.

## Summary

How best to assure safety in complex systems is an open question. We have described one approach to achieving this goal that has been demonstrated on several real systems, including proposed ATC automation upgrades. Safety, however, is not something that is simply assessed after the fact but must be built into a system. By identifying safety-related requirements and design constraints early in the development process, special design and analysis techniques can be used throughout the system life cycle to guide safe software development and evolution.

## References

[1] Urban Kjellan. Deviations and the Feedback Control of Accidents. in J. Rasmussen, K. Duncan, and J. Leplat (eds.) *New Technology and Human Error*, pp. 143-156, John Wiley & Sons, 1987.

[2] Nancy G. Leveson. *Safeware: System Safety and Computers.* Addison-Wesley Publishing Co., 1995.

[3] Robin R. Lutz. Analyzing Software Requirements Errors in Safety-Critical, Embedded Systems. *International Symposium on Requirements Engineering*, San Diego, 1992.