A Safety and Human-Centered Approach to Developing New Air Traffic Management Tools[*†]

Nancy Leveson, Maxime de Villepin, Mirna Daouk,
John Bellingham, Jayakanth Srinivasan, Natasha Neogi, Ed Bachelder
Aeronautics and Astronautics Department
Massachusetts Institute of Technology

Nadine Pilon and Geraldine Flynn
Eurocontrol Experimental Centre

Abstract

This paper describes a safety-driven, human-centered process for designing and integrating new components into an airspace management system. The general design of a conflict detection function currently being evaluated by Eurocontrol is being used as the testbed for the methodology, although the details differ somewhat. The development and evaluation approach proposed is based on the principle that critical properties must be designed into a system from the start. As a result, our methodology integrates safety analysis, functional decomposition and allocation, and human factors from the very beginning of the system development process. It also emphasizes using both formal and informal modeling to accumulate the information needed to make tradeoff decisions and ensure that desired system qualities are satisfied early in the design process when changes are easier and less costly. The formal modeling language was designed with readability as a primary criterion and therefore the models can act as an unambiguous communication medium among the developers and implementers. The methodology is supported by a new specification structuring approach, called Intent Specifications, that supports traceability and documentation of design rationale as the development process proceeds.

Introduction

The current Air Traffic Control systems have proven over time to be very safe. This high safety level can be attributed to a variety of factors, all of which are important: a high level of professionalism in the ATC work force, designed-in redundancy and mutual checking, large error margins (e.g., in the separation criteria for aircraft), and loose coupling (so that errors are contained and do not propagate rapidly throughout the various components). The limits of such a conservative system, however, along with growing demand for increased system capacity are leading to the introduction of new automation.

[*] This paper will be presented at ATM 2001 in Albuquerque, December 2001.

[†] This work was partially supported by NSF ITR Grant CCR-0085829 and NASA Ames IS (Human-Centered Computing) Grant NCC2-1223.

Automation has the potential to overcome human perceptual and cognitive limits and to reduce or eliminate specific common human errors in the current system, such as those that arise in human voice communication. At the same time, computer automation and assistance has led to new types of human errors [SW95]. The problems inherent in upgrading a national or international airspace management system while maintaining high levels of safety are further exacerbated by the fact that a completely new system is not being designed at one time but changes will be introduced in stages. There needs to be a way to add tools and new technology and to evolve functionality over time without compromising the safety of the existing system and infrastructure.

The Eurocontrol Organization has an additional problem not faced by a country like the U.S. in that the sovereign countries of Europe will be upgrading and changing their facilities at different rates and in different ways. This type of asynchronous evolution of system components has been identified by Leplat [Lep87] as one of the primary contributors to risk and accidents.

No engineering projects of this level of complexity have ever been attempted and the successful completion will require new approaches and methodologies that stretch the limits of what we can currently achieve, particularly in terms of system safety and cognitive engineering.

Safety and human factors are often considered at too late a stage in system development to have adequate impact on the system design. It has been estimated that 70-90% of the decisions relevant to safety are made in the early conceptual design stages of a project [Joh80]. Relying on after-the-fact safety assessment emphasizes creating an assessment model that proves the completed design is safe rather than constructing a design that eliminates or mitigates hazards. Too often, after-the-fact safety assessment leads to adjusting the model until it provides the desired answer rather than to improving the design. In the same way, when the human role in the system is considered after the basic automation is designed, the choices to ensure usability and safety are limited to interface design, training, and human adaptation to the newly constructed tools. The latter approach has been labeled "technology-centered design" and has been accused of leading to "clumsy automation" [WCK91] and to new types of accidents in high-tech systems, such as new fly-by-wire aircraft [SW95]. Most of these accidents have been blamed on pilot error but more accurately can be described as the result of flaws in the overall system design.

While there have been calls in the literature for safety-driven design (e.g., [Lev95]) and human-centered design (e.g., [Bil97, HKC97] and a few attempts to define such a methodology (e.g., [KED97, Jac98, Jac99]), the two goals (safety and human-centered design) are usually separated. This paper describes an integrated safety and human-centered approach to developing new air traffic control tools and designs. We first outline the methodology and show how it can be applied to the development of a new ATC conflict detection function. Then we describe a new specification structuring technique called Intent Specifications that supports the methodology.

1 Safety and Human-Centered Design

We explain the methodology using an example medium term (0 to 60 minutes) conflict detection function (MTCD-X) that is similar but not identical to the MTCD function currently being developed and evaluated by Eurocontrol. We have changed the function slightly in order to demonstrate and experiment with features that are not part of MTCD.

Figure 1 shows the overall structure of the methodology. The steps in the middle column represent the general system engineering activities. The right column shows special safety engineering activities and those in the left column represent human factors engineering. This

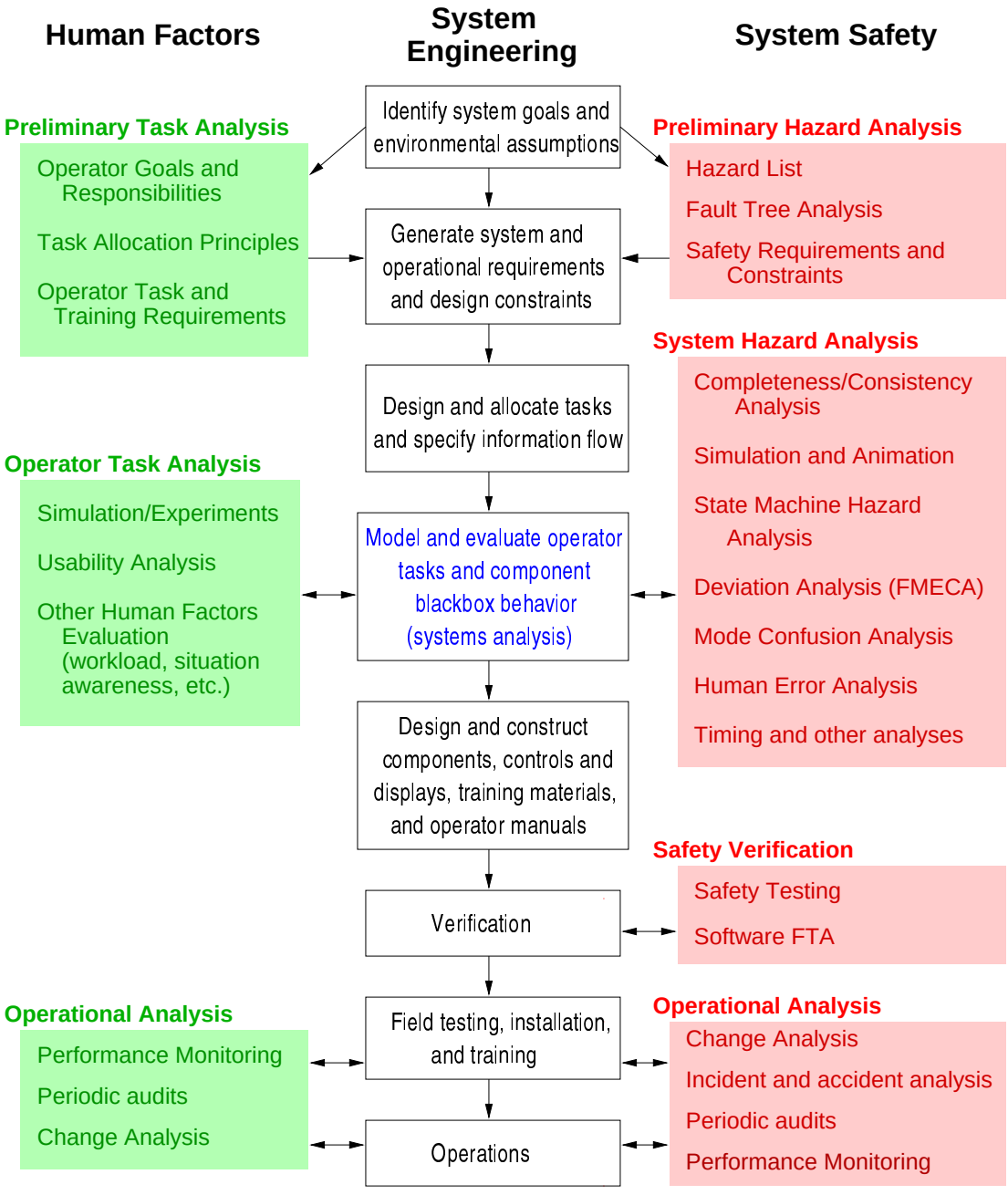


Figure 1: A Human-Centered, Safety-Driven Design Process

figure is notional only—the system engineering procedures (shown in the middle) integrate the human factors and safety analysis throughout development and operation and also involve more iteration and feedback than shown. In addition, some of the analysis procedures in the right column, such as mode confusion and human-error analyses, actually represent an overlap between safety and human factors engineering and their placement in the right column is arbitrary.

1.1 Identifying Goals and Assumptions

The process starts with identifying the high-level functional goals for the new system or component(s) and the assumptions and constraints on the new ATM function or component design arising from the environment. For example, two high-level goals for MTCD-X are:

G1: *To provide a conflict detection capability to air traffic controllers for all flights in the area of operation.*

G2: *To help keep the workload of the controllers within acceptable and safe limits despite an expected increase in traffic.*

Another early step is to describe the environment along with any assumptions and constraints about it that must be considered in the design of the new ATC function. Although some changes might be made to the existing environment when a new function or subsystem is developed, for the most part the environment will be fixed and the successful and safe development of the new tool or process will depend on how well it fits within the existing system. We consider any new or altered operator tasks related to the new function or component as "within" the system because such tasks must be designed together with the other new parts of the system.

For our example, the system consists of the conflict detection function itself (MTCD-X), the planning and tactical controllers (PC and TC, respectively) for the sector, and the human-machine interface (HMI) as a blackbox (i.e., the information flow through the HMI). The system being designed interacts directly with the real-time flight data processing system, the environment data processing system, and a recording function, and indirectly with various automated decision aids (such as an arrival sequencing manager and monitoring aids).

Two example assumptions about the interaction of MTCD-X with the real-time flight data processing system (FDPS) are:

Env-As-FDPS-01: *FDPS will provide MTCD-X with system trajectories for all eligible flights.*

Env-As-FDPS-03: *FDPS will inform MTDC-X when a flight leaves the area of operation.*

Because we believe the system design must consider human factors and safety from the very beginning in order to achieve high usability and system safety, the first steps in the methodology involve a preliminary system hazard analysis (PHA) and a preliminary controller task analysis (PTA).

1.2 Preliminary Hazard Analysis

The PHA starts from agreed upon system hazards, such as violation of minimum separation between aircraft or entry of an aircraft into a restricted area, and it identifies system behavior

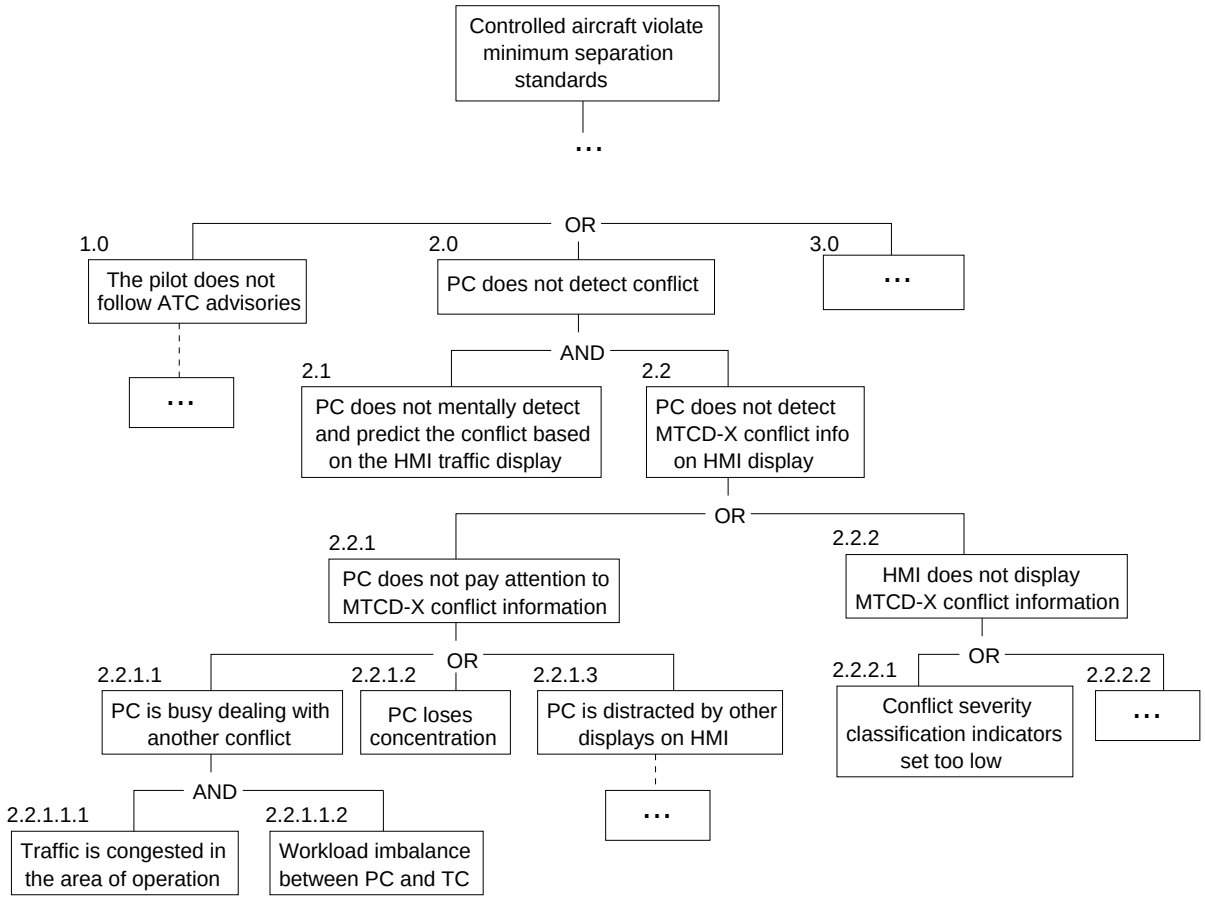


Figure 2: A Piece of a Fault Tree for Violating Minimum Separation

that could lead to those hazards. Emphasis is placed on identifying potential hazardous behavior related to the proposed new or changed functionality and new operational requirements and tasks.

Figure 2 shows a piece of the fault tree for the violation of minimum separation between controlled aircraft. The fault tree is used to derive requirements and design constraints related to safety. Each leaf node in the fault tree must either be traced to an operational or training requirement for the Controller tasks or to an MTCD-X requirement or design constraint (and thence to the design feature used to eliminate or mitigate it) or must be accepted as a necessary limitation of the system for those leaf nodes that cannot be eliminated or mitigated. Such limitations may in turn require changes in the operation or design of the overall ATM system. Information derived from the fault tree may also be used in the Preliminary Task Analysis (and vice versa).

The hazard analysis and system engineering processes are iterative and mutually reinforcing. In the beginning, when few system design decisions have been made, the hazard analysis may be very general. As the system design emerges, the hazard analysis will become more detailed and will impact additional design decisions. For example, the need for conflict severity categorization for MTCD-X was identified in another section of the PHA (not shown here) and leads to a requirement:

MTCD-X-08: *MTCD-X shall support conflict severity categorization.*

The box labelled 2.2.2.1 (*Conflict severity classification indicators are set too low*) will lead to requirements and design constraints related to conflict severity classification indicators, how they are set and how they can be changed, the conditions under which conflicts are displayed, and the need for feedback to the controller about the current value of the conflict severity categorization indicators. For example, the conflict detection function might include a requirement to allow the operators to change the conflict severity thresholds. At the same time, there may be a constraint on the controller tasks and interface design that requires permission before the conflict categorization indicators can be changed by the controller.

1.3 Preliminary Task Analysis

A Preliminary Task Analysis (PTA) is also performed at this early concept development stage and interacts closely with the concurrent PHA process. The PTA consists of cognitive engineers, human factors experts, and operators together specifying the goals and responsibilities of the users of a new tool or technology, the task allocation principles to be used, and operator task and training requirements.

For MTCD-X, we started by specifying all of the TC and PC responsibilities, not just those directly affected by MTCD-X. We included all responsibilities because any safety or usability analysis will require showing that MTCD-X does not negatively impact any of the controller activities. For instance, the PC (Planning Controller) is responsible for detecting sector entry or exit conflicts and formulating resolution plans with the PC of the adjacent sector and the TC (Tactical Controller) of the current sector. The TC, on the other hand, is responsible for in-sector tactical conflict detection and for implementing the plans formulated by the PC for entry or exit conflicts.

The next step in the PTA is to define the task allocation principles to be used in allocating tasks between the human controllers and the automation. This process uses the results of the Preliminary Hazard Analysis, previous accidents and incidents, human factors considerations, controller preferences and inputs, etc. For example, some task allocation principles for conflict resolution might be:

> *An automated conflict detection tool will assist the human controller in detecting conflicts that stretch the limits of human mental processing ability. The human controller should have final authority as far as the use of the prediction tool, the need for intervention, and the criticality of the situation. The controller will be responsible for devising solutions to the conflict.*

These principles, together with the PHA and high-level system goals, will be used to write requirements for the controller tasks, the automated function, and the human-machine interface. For example, the above PHA, the controller responsibilities, and the task allocation principles may lead to the following operator requirements:

> **MIT-OP-R02:** The PC shall plan traffic using MTCD-X output, and where a problem persists shall notify its existence and nature to the TC.
>
> **MIT-OP-R03:** If incorrect or inconvenient behavior (e.g. high rate of false alarms) is observed, the controller shall not use the MTCD-X function.
>
> **MIT-OP-R07:** The controller shall address conflicts detected by MTCD-X in a criticality-based order rather than time-based order.

The final step of the PTA is the generation of operator task and training requirements and constraints.

1.4 Generating Requirements, Constraints, and Preliminary Designs

The system goals and environmental assumptions and constraints along with the results from the PHA and PTA are then used to generate a complete set of system requirements (including functionality, maintenance, management, and interface requirements), operational requirements, and design constraints.

Using the system requirements and design constraints as well as the other information that has been generated to this point, a system design (or alternative system designs) is created and tasks are allocated to the system components (including the operators) to satisfy the requirements, task allocation principles, and operational goals. Note that this process will involve much iteration as the results of analysis, experimentation, review, etc. become available.

1.5 Evaluating the System Design

The next step involves validating the system design and requirements and performing any tradeoff and evaluation studies that may be required to select from among a set of design alternatives. Various types of operator task analyses and system hazard analyses play a part in this process.

The methodology includes using formal models in a language called SpecRL (Specification Requirements Language) to assist in this evaluation and validation process. Using SpecRL, designers construct formal, blackbox models of the required component behavior and operator tasks. The modeling language was designed with readability and reviewability of the models by various domain experts as a major goal. The models act as a communication medium among everyone involved and therefore must be easily understandable and unambiguous.

An earlier version of the current modeling language was used to specify the official requirements for TCAS II, and one goal of that project was to provide a specification language that could be read and reviewed by any interested parties with minimal training (less than an hour). The latest version of the formal modeling/specification language attempts to enhance readability and reviewability by reducing even further the semantic distance between the reviewer's mental model and the specification.

The blackbox component behavior models are built on an underlying state machine model. SpecRL blackbox models combine a graphical model of the system and its environment with tabular descriptions of the legal state changes. Figures 3 and 4 show pieces of our SpecRL model for MTCD-X. The MTCD-X model could be combined with a model of the airspace and models of the other system components and executed or analyzed together.

The graphical part of the model (as shown in Figure 3), is drawn in the form of a control loop showing the direct interactions of MTCD-X with other system components (the environment data processing system, the flight data processing system, and the controller working position). A future planned interface with a new arrival manager tool (AMAN) is shown, but no details provided because AMAN is still in the planning process.

A SpecRL model of a component itself (in this case MTCD-X) can have four parts:

- Display modes: the display mode will affect the information to be provided to the controller; a display mode specification is not needed for MTCD-X

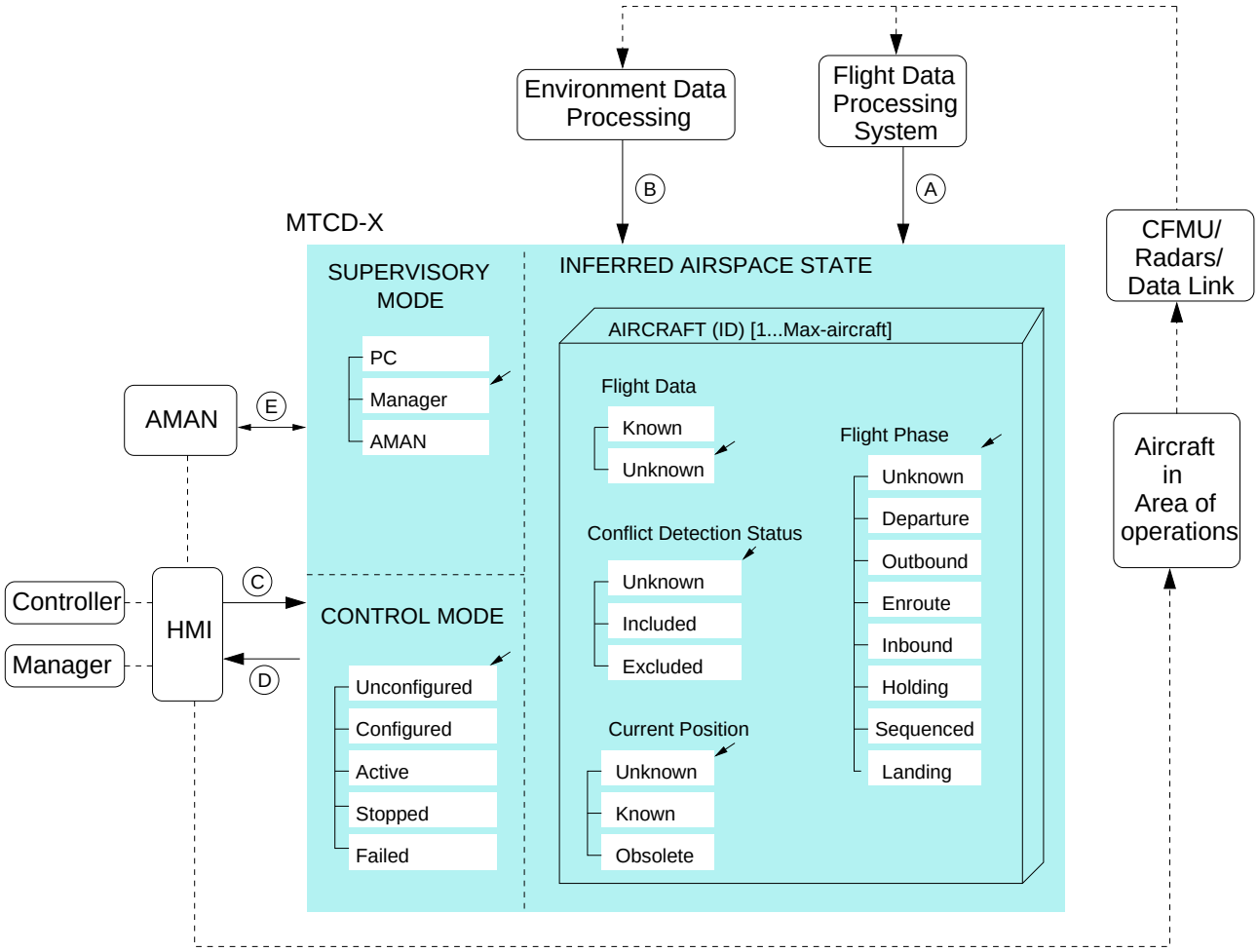
- Supervisory modes: the supervisory mode specifies who is using the component at any time, which affects which operations are legal; in this case the supervisors may be the PC, the operations manager, or AMAN
- Component control modes: the mode the automation is in, in the case of MTCD-X these include unconfigured, configured, active, stopped, and failed.
- Inferred airspace state: a model of the inferred state of the controlled system, in this case, the airspace in the area of operation.

The controlled system (airspace) state at any time is inferred from the inputs received and may be incorrect if those inputs are incorrect or not timely. The airspace model within MTCD-X consists of a model of the assumed state of each of the aircraft being evaluated for conflicts. State variables represent the information needed to perform conflict detection in addition to the actual inputs to MTCD-X. These values must be inferred from the information that MTCD-X gets from the controller working position or other devices. The model of MTCD-X shown has, for each aircraft, state variables representing the status of the flight data from that aircraft, the conflict detection status, the flight phase (needed because separation criteria will vary with flight phase), and the status of the current position information. Note that accidents occur when this inferred airspace state differs from the real state. The validation phase involves assuring that the model is correct and that the overall system is robust against errors in the information received about the current airspace state.

A complete model also needs to specify the conditions under which each of the MTCD-X control modes is used, the conditions under which the outputs are generated and their content, and how each of the inferred state variables is assigned a value. Figure 4 shows the logic for selecting the MTCD-X operating mode. The conditions under which each of the four values for operating mode become enabled are described using AND/OR tables. The operating mode takes a particular value when the *table* associated with that value evaluates to TRUE, which in turn happens when any *column* of the table evaluates to TRUE. A column is TRUE when each row satisfies the truth value shown (with a dot denoting "don't care"). In the example, the MTCD-X status becomes ACTIVE if either (1) the previous mode was CONFIGURED and an area of operation is received by MTCD-X *or* (2) the previous mode was STOPPED and a start command is received.

An executable human task modeling language has also been defined. In previous experimentation, we found that a different notation was more useful for modeling human tasks than that used for describing the automation behavior. Both generate the same type of underlying formal model, which allows integrated execution and analysis of the system as a whole, both the automation and the user tasks. An important aspect is the specification of the communication between the various controllers as well as between the controllers and the automation. We have shown how these task models can be used to find features of the combined automation and task design that can lead to mode confusion and other human errors [RZK00].

In addition to being reviewed by aviation and air traffic management experts, the formal models are executable and can be executed alone or integrated into an ATC simulation environment. Animation and visualization of the executing models assist in understanding and evaluating the proposed design of the automation and controller tasks. The executable specifications can also be used in experiments involving controllers to evaluate human factors. An advantage of executable specifications over prototypes or special simulation languages is that the specification can be changed as the evaluation proceeds. At the end of the evaluation



A FDPD → MTCD-X

For each aircraft:

Flight_ID
Aircraft_Type
Nav_Capabilities
Class_of_Flight
Current_Position (X,Y, Level)
Trajectory

E AMAN → MTCD-X

MTCD-X → AMAN

Undefined at this time

B EDPD → MTCD-X

For each airspace:

Airspace_ID
Upper_Level
Lower_Level
Boundary
Start_Time_of_Restriction
End_Time_of_Restriction
Type_of_Airspace
Separation_Parameters
Uncertainty_Parameters

For each parallel route:

Route_ID
Separation_Parameters

Area _of_Operation

C HMI → MTCD-X

Stop_MTCD-X
Start_MTCD-X
Configuration_Params
Include_Aircraft (ID)
Exclude_Aircraft (ID)

D MTCD-X → HMI

For each conflict::

Conflict_ID
Conflict_Type
Severity
Conflict_Data

Figure 3: Part of a SpecRL Model of MTCD-X

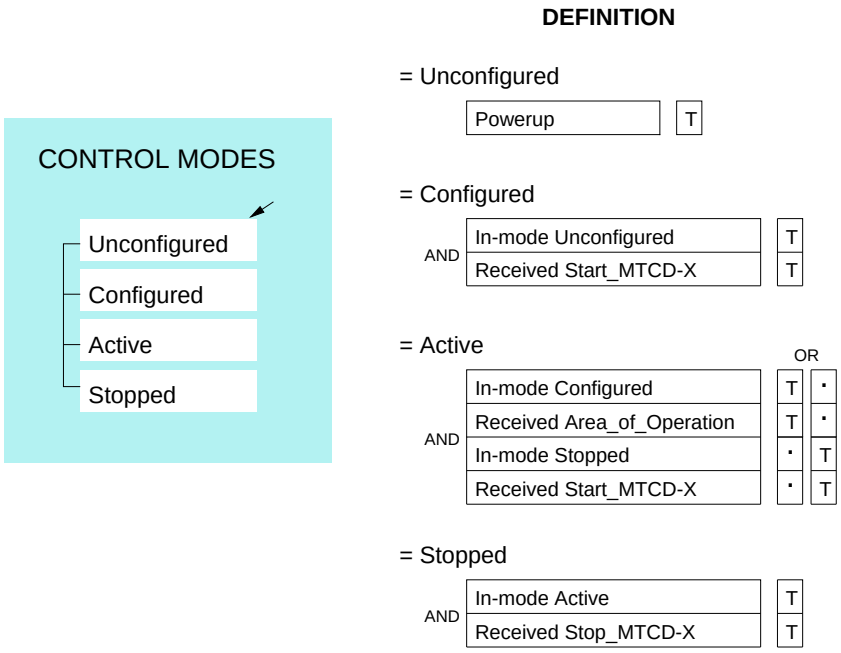


Figure 4: The Logic for Selecting the Current Operating Mode

stage, a final specification is ready for implementation without having to reverse engineer a specification from a prototype.

Because the modeling language is based on a formal mathematical model, various types of automated mathematical analysis can also be applied. We have developed techniques for analysis of consistency and completeness, robust operation in an imperfect environment, reachability of hazardous states, and potential mode confusion. A new hybrid (continuous and discrete) version of the basic modeling language (SpecRL-H) allows safety analysis of the conflict detection algorithms themselves.

Requirements errors and incompleteness account for most of the accidents in which digital automation has been involved. It is, therefore, particularly important that the requirements specification distinguish the desired behavior from that of any other, undesired behavior, that is, the specification must be precise (unambiguous), complete, and correct (consistent) with respect to the encompassing system requirements. We have built prototype tools to check our specifications for consistency and some aspects of mathematical completeness [HL96]. Other important completeness aspects are enforced by the design of SpecRL itself [Lev00b].

Robustness can be evaluated using an automated technique called Software Deviation Analysis [RL87]. SDA allows determining the effects of deviations of system parameters (inputs) on software in order to determine how the software will operate in an imperfect environment. The input to the SDA tool is an input deviation, for example, "*the altitude reported by the radar data processing function is lower than the actual altitude.*" The output is a list of scenarios, where a scenario is defined as a set of deviations in the software inputs plus constraints on the software execution states that are sufficient to lead to a deviation in an identified safety-critical output. The deviation analysis procedure can optionally add further deviations as it constrains the software state, allowing for the analysis of the effects of multiple, independent failures.

Other tools can be used to assist in system and subsystem hazard analysis [LS87]. Information from these analyses is useful in eliminating hazards from the design or in designing controls

and hazard mitigation. For example, one tool assists the designer in tracing back through the model from hazardous states to determine if and how they are reachable. Backward search can also reveal how the system can end up in a hazardous state if a failure occurs. Our backward reachability analysis on discrete state models has recently been augmented to include continuous states (a hybrid model) [Neo01]. Neogi has experimentally applied this approach to the conflict detection algorithm used in MTCD-X.

Finally, the specification/model of the blackbox automation behavior can be evaluated for its potential to lead to mode confusion [LRK97, LP97]. Six automation design categories have been identified as leading to mode confusion, based on accidents and simulator studies: ambiguous interface modes, inconsistent automation behavior, indirect mode changes, operator authority limits, unintended side effects, and lack of appropriate feedback. Analysis procedures are being developed to detect these features in SpecRL models.

Once the engineers are happy with the operator task and logical system design, detailed design and construction of the system components, controls and displays, training materials, and operator manuals can begin.

2 Intent Specifications

The methodology outlined above is supported by a new specification approach called Intent Specifications [Lev00a] and automated tools to assist with model construction, recording of design rationale, and traceability.

ATM systems are going to evolve and change continually as new technology and tools are added and new operational concepts are implemented. Maintaining safety in such a changing environment requires high-quality specifications that include detailed descriptions of the externally visible behavior of the existing components as well as the rationale for the system design choices. The design of any new system component must be based on the design and constraints of the surrounding environment and any changes to the current system must be analyzed for their effect on system requirements, operator tasks and responsibilities, safety constraints, and human factors.

When considering a requirements, design, or implementation change, it must be possible to determine any potential effect on system safety or usability. This need, in turn, calls for a level of traceability not normally found in system specifications. Although such traceability implies more planning and specification effort at the beginning of a project, the effort will allow changes to be made much more quickly and easily. It could be prohibitively expensive, for example, to generate a new hazard and safety assessment for each system change that is proposed. Being able to trace a particular design feature to the original hazard analysis will allow decisions to be made about whether and how that feature can be changed. The same is true for changes that affect operator activities and basic task allocation and usability principles that have been established for the ATM system. Pabel and Garron have noted the importance of specifying design rationale and the need for communication among experts in the HMI specification process [PGJ01]: Both are supported by Intent Specifications.

Intent specifications organize system specifications not only in terms of "what" and "how" (using refinement and part-whole abstractions) but also in terms of "why" (using intent abstraction) and integrate traceability and design rationale into the basic specification structure. They include both natural language and formal executable models, as described above. The design of intent specifications, using ideas from Rasmussen's means-ends abstraction hierarchy [Ras85], is based on fundamental knowledge about human problem solving and also on system

theory and basic system engineering principles.

There are six levels in an Intent Specification, each supporting a different type of reasoning about the system and representing a different model of the system. Each level also includes information about the verification and validation of the system model at that level. By organizing the specification in this way and linking the information at each level to the relevant information at the next higher and next lower level, higher-level purpose or intent, i.e., the rationale for design decisions, can be determined. In addition, by integrating and linking the system, software, human task, and interface design and development into one specification framework, intent specifications support an integrated rather than stovepiped approach to system design.

3 Conclusions

An experimental human-centered, safety-driven design process for ATM systems has been described. The process is supported by a specification methodology (intent specifications) and various modeling and analysis languages and tools. Parts of the process have been applied experimentally to CTAS [Lev97], TCAS II [LHH94], and, as described in this paper, ATC conflict detection.

The approach considers and builds safety and usability into the system design from the very beginning. Potential users of the automation, such as controllers and pilots, are an integral part of the process starting in the early concept formation stage and assist in setting goals, writing requirements, and establishing principles for allocating tasks between humans and automation. Formal specification methods allow analysis and detection of errors or poor design choices early in the design process before implementation of the software and other components. The specifications/models are executable and can also be subjected to various sophisticated types of automated analysis and used in human-in-the-loop simulations and experiments. At the same time, readability was stressed during the design of the formal specification language (SpecRM) so the models would be reviewable by engineering and human factors experts with a large variety of backgrounds and expertise. The successful achievement of this goal was demonstrated for an earlier version of the language during the development of the requirements specification for TCAS II for the U.S. FAA.

Evaluation of requirements often uses rapid prototyping. A software prototype is written quickly (without careful design and documentation) and then evaluated and changed until it exhibits acceptable behavior. But the software prototype has usually changed so much in that process that reverse engineering is required to determine exactly what it does so that a hardened and high-quality version of the final design can be constructed. This reverse engineering process is extremely expensive and difficult. The alternative, often chosen, is simply to deliver and use the prototype version, which has serious implications for safety and maintainability. By using executable specifications, the requirements specification itself becomes the prototype and when the process of analysis and design is completed, it can be used for the implementation and documentation process. This feature is particularly useful in a setting such as the Eurocontrol Organization where new ATM functions are developed and evaluated by Eurocontrol but the specifications are given to ATM providers to implement.

A set of commercial-quality tools are being developed to support this human-centered, safety-driven approach to ATM system design. Our next planned steps involve developing and improving further the parts of the approach through additional experimentation. We also want to experiment further with the use of the executable models and visualization to assist controllers in understanding and learning about the automation they are using.

References

[Bil97] Billings, C.E. *Aviation Automation: The Search for a Human-Centered Approach.* Lawrence Erlbaum Associates, 1997.

[HKC97] Hansman, R.J., Kuchar, J., Clarke, J.P., Vakil, S., Barhydt, R., and Pritchett, A. Integrated Human-Centered Systems Approach to the Development of Advanced Cockpit and Air Traffic Management Systems. *Digital Aviation Systems Conference*, 1997.

[HL96] Heimdahl, M.P.E. and Leveson, N.G. Completeness and Consistency in Hierarchical State-Based Requirements. *IEEE Transactions on Software Engineering*, SE-22, No. 6, June 1996.

[Jac98] Jackson, A. HF Integration within Concept Development, Design, and Pre-Operational Evaluation: A Pragmatic Approach. *Third EUROCONTROL Human Factors Workshop*, Luxembourg, 1998.

[Jac99] Jackson, A. HMI—Requirements to Implementation: learning from Experience. *EUROCONTROL/FAA TIM on Lessons Learned in HMI Design*, Toulouse, 1999.

[Joh80] Johnson, W.G. *MORT Safety Assurance Systems*, Marcel Dekker, Inc., 1980.

[KED97] Kirwan, B., Evans, A., Donohoe, L., Kilner, A, Lamoureux, T, Atkinson, T, and H. MacKendrick. Human Factors in the ATM System Design Life Cycle. *FAA/Eurocontrol ATM R&D Seminar*, Paris, June, 1997.

[Lep87] Leplat, J. Occupational Accident Research and Systems Approach. In Rasmussen, Duncan, and Leplat (Ed.), *New Technology and Human Error*, John Wiley & Sons, pp. 181–191, 1987.

[Lev95] Leveson, N.G. *Safeware: System Safety and Computers.* Addison-Wesley, 1995.

[Lev00a] Leveson, N.G. Intent Specifications. *IEEE Trans. on Software Engineering*, January 2000.

[Lev00b] Leveson, N.G. Completeness in Formal Specification Language Design for Process-Control Systems. *ACM Formal Methods in Software Practice*, Portland, August 2000

[Lev97] Leveson, N.G. Alfaro, L., Alvarado, C., Brown, M., Hunt, E.B., Jaffe, M., Joslyn, S., Pinnel, D., Reese, J., Samarziya, J, Sandys, S., Shaw, A., and Zabinsky, Z. Safety Analysis of Air Traffic Control Upgrades. Technical Report, University of Washington, 1997 (see http://sunnyday.mit.edu/papers.html).

[LH83] Leveson, N.G. and Harvey, P.R. Analyzing Software Safety. *IEEE Transactions on Software Engineering*, vol. SE-9, no. 5, 1983.

[LHH94] Leveson, N.G., Heimdahl, M.P.E., Hildreth, H., and Reese, J.D. Requirements Specification for Process-Control Systems. *IEEE Transactions on Software Engineering*, SE-20, No. 9, September, 1994.

[LP97] Leveson, N.G. and Palmer, E. "Designing Automation to Reduce Operator Errors, *International Conference on Systems, Man, and Cybernetics*, Florida, Oct. 1997.

[LRK97] Leveson, N.G., Reese, J.D., Koga, S., Pinnel, L.D., and Sandys, S.D. Analyzing Requirements Specifications for Mode Confusion Errors. *First International Workshop on Human Error and System Development*, Glasgow, March 1997.

[LS87] Leveson, N.G. and Stolzy, J.L. Safety Analysis Using Petri Nets. *IEEE Trans. on Software Engineering*, Vol. SE-13, No. 3, March 1987, pp. 386-397.

[Neo01] Neogi, N. Hybrid Modeling and Backwards Reachability. Ph.D. Dissertation, Aeronautics and Astronautics Dept., MIT, in preparation.

[PGJ01] Pabel, D. and Garron, J. Expression of Requirements for HMI Specifications for ATC/CWP. EEC Note No. 06/01.

[Ras85] Rasmussen, J. The Role of hierarchical knowledge representation in decision making and system management. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, March/April 1985.

[RL87] Reese, J.D. and Leveson, N.G. Software Deviation Analysis. *International Conference on Software Engineering*, Boston, May 1997.

[RZK00] Rodriguez, M., Zimmerman, M., Katahira, M., de Villepin, M., Ingram, B., and Leveson, N.G. Identifying Mode Confusion Potential in Software Design. *Digital Aviation Systems Conference*, Philadelphia, October 2000.

[SW95] Sarter, N.D. and Woods, D. "How in the World did I Ever Get into That Mode?": Mode Error and Awareness in Supervisory Control. *Human Factors 37*, 5–19.

[SW95] Sarter, N.D., Woods, D.D. and Billings, C.E. Automation Surprises. in G. Salvendy (Ed.) *Handbook of Human Factors/Ergonomics*, 2nd Edition, Wiley, New York, in press.

[WCK91] Wiener, E.L., Chidester, T.R., Kanki, B.G., Palmer E.A., Curry, R.E., and Gregorich, S.E. The Impact of Cockpit Automation on Crew Coordination and Communications. NASA Ames Research Center, 1991.