SOFTWARE SAFETY FOR AIR TRAFFIC MANAGEMENT SYSTEMS

Jeffrey J. Joyce, Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada V6T 1Z4 (http://www.ece.ubc.ca/~jeffj)

Introduction

The introduction of advanced Air Traffic Management (ATM) functionality is often heralded as a safety improvement. On the whole, the introduction of advanced ATM functionality will probably reduce the likelihood of an accident. However, its introduction may also entail new sources of safety risk that need to be carefully weighed against the safety benefits of advanced ATM functionality.

Many new sources of safety risk associated with the introduction of advanced automation are due to specific problems in the following categories:

- too much effort spent entering data
- too much information displayed
- increased semantic complexity
- not enough visibility into automated processes
- coarse-grained reuse of software
- conflicts between availability objectives and safety objectives.

It would be unreasonable to suggest that advanced ATM functionality is inherently dangerous or that its introduction should be resisted. Such technology is necessary for the increasing demands placed on air navigation systems.

Rather, the purpose of this paper is to argue that stakeholders must look beyond the "safety-net" functions to thoroughly understand the impact of advanced ATM functionality on the safety of an air navigation system. In particular, the provision of safety-net functions should never be viewed as a reason to be less concerned about behaviors of the system that could potentially mislead a controller into deciding that separation exists when it does not. In other words, displaying basic flight data used to make separation decisions in a timely and accurate manner must always take priority over safety-net functions.

This paper also urges stakeholders to distinguish safety from other qualities such as reliability and availability. An effective approach to the safety engineering of an advanced ATM system is not merely a matter of minimizing the likelihood of latent defects or meeting availability objectives.

A third purpose of this paper is to suggest some categories of problems that could be used as input to a "brainstorming session" during the preliminary hazard identification step of the safety program for an advanced ATM system.

Finally, the paper offers some remarks on the usefulness of safety-related standards and guidelines that are called upon in the development of advanced ATM systems.

Advanced ATM

For the purposes of this paper, we use the term "advanced ATM functionality" to refer broadly to a relatively new class of ATM systems that uses flight and environmental data to generate four-dimensional trajectories for flights.

The generation of flight trajectories underlies many advanced ATM functions including the coordination of flights between sectors and centers, the distribution of flight data and certain safety-net functions. The trajectories may also be used to generate extrapolated representations of aircraft position on the controller's situation display in combination with aircraft position information derived from radar or other sensors.

Advanced ATM functionality typically includes such safety-net functions as conflict prediction and conformance monitoring. Certain kinds of validation checks applied to clearance information entered by a controller might also be regarded as a form of safety-net functionality.

0-7803-7367-7/02/$17.00 © 2002 IEEE

Advanced ATM functionality may also include emerging technologies under development such as Controller Pilot Data Link Communications (CPDLC). However, this paper does not specifically consider these emerging technologies.

Definition of "Safety"

Safety may be defined as "freedom from accidents and losses". For the purposes of assessing the safety of an advanced ATM system, we focus specifically on the possibility that separation between an aircraft and another aircraft, terrain or a particular airspace may be "lost".

The primary responsibility of an air traffic controller is to ensure that flights under his jurisdiction remain separated from other aircraft, terrain and certain airspaces. There are specific rules called "separation rules" or "separation minima" that must be satisfied for a jurisdictional flight to be considered separated. For example, an aircraft may be considered separated from another aircraft under certain conditions if it is either 1,000 feet above or 1,000 feet below the altitude of the other aircraft. Alternate forms of separation include both lateral and longitudinal separation. Separation is said to "exist" when at least one applicable separation rule is satisfied. When none of the applicable separation rules are satisfied, separation is said to be "lost". It is reasonable to assume that the "loss of separation" is a necessary precondition for an accident to occur if we narrow the scope of the safety analysis to the enroute phase of flight.

It is important for the safety analyst and other stakeholders to have objective criteria for deciding when a particular behavior of an ATM system is unsafe. Elsewhere [1], we have identified several generic criteria that may be used as a "litmus test" for behaviors exhibited by an ATM system that could be unsafe. These criteria emphasize the importance of avoiding behaviors that could mislead a controller into deciding that separation exists in a particular situation when it does not.

The concept of ATM system safety engendered by these criteria has strongly influenced the general concerns highlighted in this paper.

Too Much Effort Spent Entering Data

One of the primary goals of introducing advanced ATM functionality is to reduce controller workload. It is true that advanced ATM functionality will relieve controllers of certain routine tasks, thus increasing their capacity to handle flights under their jurisdiction and to focus on non-routine tasks such as resolving conflicts. However, advanced ATM functionality requires controllers to enter clearances and other flight data for purposes such as the generation of flight trajectories. The entry of this information yields new tasks for controllers that partially reverse whatever reductions in controller workload may have been gained by the automation of routine tasks.

The entry of such clearances and other related information into a computer system is a new task for controllers, since such information is not typically required by legacy systems. For example, the task of changing the cleared altitude of a flight would only require the controller to annotate the flight strip and communicate by radio with the pilot in a typical operational environment prior to the introduction of advanced ATM functionality. But with the introduction of advanced ATM functionality, the controller must additionally interact with his workstation to update the clearance for the flight.

Hence, the reduction in controller workload resulting from the automation of routine tasks is partially offset by the introduction of new tasks, especially the need to update the computer with data derived from clearances and position reports.

The consequences of failing to "keep up" with the computer system's need for current and accurate flight data could be significant. For example, failing to enter the new cleared altitude for a flight could result in "false alarms" when the controller has cleared a flight to a different altitude to avoid a conflict. In this case, the false alarm would be a consequence of the system continuing to use the previous cleared altitude as a basis for predicting conflicts. Even worse, the computer system may

fail to predict a bona fide conflict at the new cleared altitude if the controller has not entered the new cleared altitude.

More generally, the failure to provide the ATM system with current and accurate flight data may have a variety of unforeseen consequences that could ultimately result in hazardous situations. For example, a change to the cleared route of a flight may affect the distribution of flight data to downstream sectors and centers. If such changes are not entered into the computer system, critical flight data may be not distributed as required. Depending on the procedures used for coordination between sectors and centers, there may be scenarios in which flight data distribution problems could result in hazardous situations. Similarly, stale clearance information may affect the accuracy of time estimates represented in the trajectory of a flight that, in turn, could yield hazardous situations including conflict prediction failures.

The design of the human computer interface for the entry of critical flight data involves some difficult choices. For example, the dialogue for changing the cleared altitude of a flight could require confirmation, i.e., the controller must click on "OK" after changing the cleared altitude. This confirmation step might be deemed necessary to reduce the likelihood of data entry errors. On the other hand, this extra step increases the number of interactions that must be performed by the controller.

The design of operational procedures is also an important consideration in tradeoffs between mitigating the risk of erroneous controller inputs and increasing controller workload. For example, operational procedures could be modified so that the controller first updates the computer with the cleared altitude and then reads back the cleared altitude from the updated display when communicating the new cleared altitude to the pilot. This constraint may reduce the risk that the cleared altitude stored in the computer for the flight differs from the cleared altitude communicated to the pilot. However, a controller is almost certainly going to give priority to voice communication with the pilot over updating the computer system in busy or urgent situations. Ironically, busy or urgent situations are when the failure to update the computer system with accurate and current information could be particularly hazardous.

When a controller is unable to keep the computer system fully updated in a busy or urgent situation, the computer system will become increasingly unsynchronized with reality. For example, position reports will be non-conformant with trajectories based on stale flight data. In addition to undermining the ability of the computer system to meet operational needs, the increasing loss of synchronization may increase the frequency of "false alarms" and other distractions.

For these reasons, it is overly simplistic to suggest that the introduction of advanced ATM functionality is inherently safer because it reduces controller workload. To properly assess the safety of introducing advanced ATM functionality into an operational environment, it is necessary to identify new tasks required of controllers. The impact of these new tasks on controller workload must then be weighed in balance with whatever reductions in controller workload may result from the automation of routine tasks.

Too Much Information Displayed

Not only do advanced ATM systems require a greater variety and volume of information from controllers as input, they also have the potential to produce a much greater variety and volume of information for display on controller workstations. Displaying too much information, or not providing a means by which a controller can regulate the variety and volume of the information generated by the advanced ATM system can be a source of safety risk.

In the early stages of developing an advanced ATM system, there is a temptation is to take full advantage of the flexibility of the software and computational power of the hardware. This may result in the implementation of additional functions that do not significantly increase the efficiency or safety of the air navigation system. In fact, they may even have the potential to interfere with the controller's ability to perform critical tasks.

One limiting factor is the fixed amount of "real estate" on the display of the controller's workstation. To allow more information to be displayed, characters fonts may be smaller than

desirable for readability. This could be very hazardous if, for example, it is difficult for controllers to distinguish between numerals such as "6", "8" and "3" in a number representing an altitude in a datablock on the situation display.

Another strategy is the use of panels (i.e., windows) that can be opened and then closed or minimized, thus providing the controller with a degree of control over what information is displayed at his workstation. However, care must be taken in the design of the human computer interface to ensure critical information is not hidden from the controller's view in a minimized window. This is particularly important for critical information not anticipated by the controller such as an alert.

The flexibility of software and computational power of modern hardware offers the potential of implementing a great variety of functions that yield various alerts and warnings for display to controllers. This includes alerts and warnings generated by safety-net functions, as well as other functions less directly related to safety objectives. Safety analysts should be concerned about the possibility that the system may generate too many "unhelpful" alerts or warnings. Excessive alerts or warnings may distract a controller from his responsibilities, especially if these alerts or warnings need to be acknowledged by the controller. Furthermore, the generation of too many alerts or warnings may desensitize a controller so that a truly "helpful" alert or warning fails to attract the notice of a controller.

A key part of the safety analysis of an advanced ATM system is making a list of all data elements displayed to controllers that are critical for decisions about separation. This list would include, for example, Mode C altitude (if available).

The list should also document the potential consequences of the data element being absence from the display as well as the potential consequences of the critical data element being corrupted (in a way that it may appear to be valid, but in fact is not valid). For instance, this step of the analysis is likely to conclude that the potential safety consequences of the Mode C altitude of a particular flight not being displayed are less serious than the potential consequences of displaying a corrupted Mode C altitude. If the controller cannot see the Mode C altitude, then he may have to do some extra work (e.g., ask the pilot to report his altitude or increase vertical separation). But a controller cannot be misled by a missing Mode C altitude in the same way that he might be misled by a corrupted Mode C altitude.

Next, the design of the human computer interface should be systematically reviewed using this list of critical data elements as a guide. For each element on the list, a series of generic questions should be asked such as "What conditions could cause this data element to be stale, i.e., no longer valid?". Answering such questions thoroughly will involve several iterations of investigation and analysis. The first iteration would likely focus on the functional requirements. This would be followed in the next iteration by an investigation of the design and perhaps an inspection of the implementation. Such questions may even be addressed by a dedicated safety testing activity in which the test engineer exploits his knowledge of the system to attempt to "trick" the system into displaying stale data.

Assuming that the computer human interface is thoroughly documented prior to implementation, it is possible to perform this part of the analysis at a relatively early stage in the development cycle when there is more opportunity to "build safety into the design".

Increased Semantic Complexity

As mentioned, a key characteristic of advanced ATM functionality is the generation of a four-dimensional trajectory from clearances and other information. The generation of trajectories from clearances depends on the correct interpretation of the information contained in the machine-readable representation of a clearance.

The phraselogy used by controllers to communicate with pilots is a constrained form of natural language. This phraselogy is informally defined in the "manual of operation" and related publications. It is learned by controllers and pilots largely through experience as trainee controllers and novice pilots.

To allow clearance information to be processed by a computer system, the clearance phraseology must be formalized by defining a

syntax that constrains the vocabulary of this language. The syntax rules must also constrain the way in which words of this vocabulary may be composed to make elements of a clearance such as "cross BLI at or above FL210".

Managing the syntactic complexity of formalized languages is a solved problem for software developers. Software developers can borrow techniques developed for the processing of other formal languages such as programming languages to manage the syntactic complexity of a formal language for expressing clearance information.

However, dealing with the semantic complexity of a formalized clearance language is not so easily addressed. The semantics of the phraselogy used by controllers to communicate clearances is very complex compared to the semantics of most other formal languages designed for interpretation by a computer system. As a result of this semantic complexity, there is a considerable potential for clearance elements to be ambiguous – that is, have more than one interpretation.

For example, the syntax rules of a formalized clearance language are not likely to exclude combinations of restrictions that are impossible to satisfy – for example, a restriction to cross a certain fix at or above a particular flight level in combination with another restriction to cross at below a lower flight level at the same fix. While such a combination may be obviously unsatisfiable and could be detected by software checking, there are other combinations that are problematic in less obvious ways.

Such ambiguities may also occur in the voice communications between controllers and pilots. However, they are often resolved when the pilot uses "common sense" to choose the interpretation intended by the controller. If the pilot is unsure about the intended interpretation, he can always query the controller for clarification.

In contrast, computer systems do not have the benefit of common sense to resolve ambiguities in clearance information. To a limited extent, it is possible to add various checks into the software to detect potentially ambiguous clearance elements. However, in the absence of a completely formalized definition of the semantics for the clearance language, such checks will be ad hoc. There will be no guarantee that every statement in the clearance language will be unambiguous, or that its interpretation will unfailingly agree with the interpretation intended by the controller.

As well, relying on extensive validation checks for clearances entered by the controller could have the undesirable consequence that "reasonable" clearance elements are too frequently rejected due to the validation checks. The system will be deemed unusable if clearance elements are frequently rejected even if they are unambiguous to the controllers and pilots, who have the advantage of common sense over the computer system.

Once again, it is important to consider the use of the system by a controller in a busy or urgent situation. In such a situation, the controller may be compelled to abandon his attempts to update the computer system with clearance information if these attempts are repeatedly rejected due to validation checking. As the computer system becomes increasingly unsynchronized with reality, it will become less capable of meeting operational needs. Also, as previously mentioned, this loss of synchronization is likely to yield false alarms and other anomalous behaviors.

A related problem may occur when clearance information is exchanged between different computer systems located, for example, at different centers. Even when the same software is used to process the clearance information and generate flight trajectories, there may be significant differences in the generated trajectories due to differences in adaptation data or environmental data. For example, a controller might locally enter a "spot observation" about the winds aloft that affects the time estimates for a particular fight. Even greater differences in the generation of trajectories may result when different computer systems with different software are used to interpret the same clearance information. In the case of ambiguous clearance information, one computer system may select one interpretation whereas the other computer system generates trajectories using a different interpretation.

The possibility that different computer systems may generate significantly different trajectories may or may not yield safety risk. Assuming that the trajectory generated by the computer system for

the controller with jurisdiction is correct (i.e., is consistent with the controller's intention), then it is not necessarily unsafe in the short-term for the downstream computer system to have generated a slightly different trajectory. The safety consequences would need to be investigated in the context of understanding how the trajectories are used to automate certain aspects of coordination between centers. Assessment of the associated safety risk would also need to take account of relevant operational procedures – particularly, with respect to coordination between the controller with jurisdiction and the downstream controller.

Safety concerns about differences in the machine interpretation of clearance information by different computer systems will also be important in the context of emerging technology. In particular, CPDLC involves the transmission of clearance information from ground systems to airborne systems. The initial single-site deployment of CPDLC known as Build-1 will not involve clearance messages that affect the trajectory of the aircraft. However, planned future deployments of CPDLC are expected to support clearance messages that would affect the trajectory of the flight.

There is no "silver bullet" to mitigate safety concerns related to the semantic complexity of clearance information, and its interpretation by a computer system. However, the combination of several measures would be prudent in the development of advanced ATM functionality, including:

- careful design of the formalized clearance language so that it is rich enough to meet operational needs, but without excessive variation
- a balanced approach to automated clearance validation that will detect obvious ambiguities and other semantic problems in clearances, but not result in the frequent rejection of "reasonable" clearances
- dedicated safety testing focused on the identification of anomalies in the machine interpretation of clearance information, especially situations in which the machine interpretation of a clearance may differ from a "common sense" interpretation by the controller and pilot.

In addition to the above measures, the challenge of developing software to consistently and accurately interpret clearance information would benefit from research initiatives to study how analogous problems for other formal languages have been addressed. For example, the ubiquity of Java and other interpreted programming languages owes much to portability of these languages across different platforms. Perhaps, the accurate and consistent rendering of trajectories from clearance information across different computer systems may be addressed by the development of a "bytecode" for clearance information.

Not Enough Visibility into Automated Processes

The automation of routine time-consuming tasks is one of the primary motivations for the introduction of advanced ATM functionality. Such automation is seen as a means to improve the efficiency of the air navigation system. It is also often suggested that such automation is a safety improvement because it reduces opportunities for operator error.

But as Leveson [2] observes, "the truth is that automation does not remove people from the systems -- it merely moves them to maintenance and to higher-level supervisory control and decision-making. The effects of human decisions and actions can then be extremely serious. At the same time, the increased system complexity makes the decision-making process more difficult."

One of the tasks most likely to be automated in an advanced ATM system is the distribution of flight data to downstream sectors and centers.

The distribution of flight data is mostly routine. In an operational environment without automated support, it represents a significant portion of a controller's workload even when some aspects of flight data distribution are delegated to other personnel. Typically, the flight data is distributed by a combination of data links and voice channels, e.g., "hotlines". The distribution of flight

data is governed by various rules and agreements established locally with adjacent jurisdictions.

While the automation of flight data distribution may indeed reduce safety risk due to human error, it is still necessary to provide controllers with visibility into flight data distribution so that they can in fact act in a supervisory manner and, when necessary, have sufficient information to intervene when the automated system fails to meet operational needs in a particular situation. It is not enough to simply notify controllers when transmission of a particular message is unsuccessful. In addition to such notifications, controllers require visibility into when, what and where a message has been sent. They also required sufficient visibility to determine when the system has decided not to send a message.

As with many other aspects of a controller's responsibility, the rules and agreements that govern the distribution of flight data have evolved over many years prior to the introduction of advanced ATM functionality. Just as the accurate and consistent interpretation of clearance information may rely on common sense, the manual distribution of flight data has also relied on common sense. Even when great care is taken to formalize these rules and agreements as algorithms for the automated distribution of flight data, there will always be a possibility of anomalous situations that result in critical flight data not being sent when or where it needs to be sent. Such problems would not be considered "failures" in the traditional sense since the system is performing exactly as required. Rather, they are the result of limitations on the ability to automate common sense in software.

For this reason, the initial premise of the safety analyst when analyzing the safety of automated flight data distribution should be that the required behavior will sometimes fail to meet the operational needs. The safety assessment then becomes mostly a matter of determining whether the controller will have sufficient visibility into the activity of flight plan distribution to recognize this problem so that he can intervene. This approach to the safety analysis of an ATM contrasts with a "reliability-oriented" concept of system safety that would likely focus on the possibility of defects in the implementation of the algorithm for the distribution of flight data.

Determining whether a controller has sufficient visibility into flight data distribution is not merely a question of whether it is possible for a controller to obtain sufficient information from the system. This determination must also take into account the ease with which the controller can answer specific questions as he monitors the distribution of flight data. For instance, it is not enough for the controller to have the ability to see a list of all messages sent to a downstream center in the past hour. Instead, the controller would require a means of answering specific queries – for example, to see all messages sent to a downstream center for a specific flight.

As previously mentioned, the distribution of flight data by an advanced ATM system is likely to be partially determined by the trajectories generated from clearances and other information for each flight. Thus, potential problems with the distribution of flight data due to limitations of an automated system to implement a common sense interpretation of the rules and agreements for flight data distribution would be exacerbated in situations where the trajectory generated by the computer system is not consistent with the common sense interpretation of the clearance information intended by the controller.

Although the discussion in this section has focused mostly on the distribution of flight data, this is just one example of a routine task that may be automated with the introduction of advanced ATM functionality. Another example would be the automation of a manual process for the acquisition and distribution of environmental data such as altimeter settings. Yet another example would the management of special use airspaces. For such cases where a routine task has been automated by the introduction of advanced ATM functionality, the safety analyst should start with the premise that the automation will sometimes fail to meet operational needs, and then consider whether the controller has sufficient visibility to recognize when a problem has occurred so that he can intervene.

Coarse-Grained Reuse of Software

The software implementation of advanced ATM functionality is inherently larger and more complex than the implementation of legacy systems. An advanced ATM system would

typically involve between one and two million source lines of code (SLOC), supplemented by substantial use of commercial off the shelf (COTS) components. Other commonly used measures of software complexity would similarly show that the software implementation of advanced ATM functionality involves significant complexity.

For obvious reasons, stakeholders in the procurement and development of an advanced ATM system are compelled to pursue strategies that reduce cost and schedule. One of the most common strategies is the reuse of previously developed software. Fine-grained forms of software reuse typically involve generic components which are instantiated for a variety of different purposes within a single implementation. Such forms of software reuse typically involve implementations of commonly used abstract data types, e.g., queues, stacks, lists. A coarser-grain form of software reuse involves the reuse of a subsystem from another product, or an earlier member of a specific product family.

Fine-grained forms of software reuse, such as the implementation of commonly used abstract data types, almost certainly makes a positive net contribution to the safety of an advanced ATM system. The only caution is that the use of generic components at this level can significantly increase the difficulty of performing safety verification using static methods, e.g., tracing the datapath for certain critical data element from input to output.

The course-grained reuse of software raises more serious concerns from a safety perspective. Leveson [2] suggests: "Although reuse of proven software components can increase reliability, reuse may actually decrease safety because of the complacency it engenders and because the specific hazards of the new system were not considered when the software was originally designed and constructed." Leveson and Clark [3] elaborate further on the assumption that reuse increases safety: "A naive assumption is often made that reusing software or using commercial off-the-shelf software increases safety because the software has been exercised extensively. Reusing software modules does not guarantee safety in the new system to which they are transferred and sometimes leads to awkward and dangerous designs. Safety is a quality of the system in which the software is used; it is not a quality of the software itself."

In addition to the above mentioned reasons, safety problems due to coarse-grained reuse of software are often the result of undocumented assumptions made during the original development of software. When the previously developed software is reused, these assumptions may not be fully appreciated.

A particular class of coarse-grained software is COTS. A distinction should be made between COTS software and the use of code that has been "lifted" from an earlier implementation. Typical examples of COTS include the operating system, databases and elements of the graphic user interface (GUI). These examples of COTS software are not domain-specific and for this reason, are less likely to be affected by problems caused by undocumented assumptions about the operational environment. A more likely source of safety problems with the use of COTS software is the need to compromise on certain aspects of the design to fit the limitations of the COTS software. For instance, the use of a particular COTS product in the implementation of the GUI may cause certain elements of the controller's display to be refreshed less frequently than desired from a safety perspective. Such a problem might, for example, be caused by an intrinsic limitation in the COTS product that makes it impossible to refresh certain elements of data more frequently.

Cost and schedule priorities are likely to overrule the total exclusion of previously developed code in the implementation of an advanced ATM system. But previously developed code should not be excluded from the testing regime that would have been applied to new code, especially at the system level. In particular, the argument that "this software has already been proven in the field" is not valid in the case of previously developed code that has been lifted from an earlier implementation. Whenever a non-trivial amount of code has been lifted from an earlier implementation and transferred into a new implementation, there are just too many opportunities for one or more the undocumented assumptions made by the original developers to be invalid for the new implementation. Indeed, the fact that the implementation of a particular function involves

previously developed code should prompt a more rigorous regime of safety verification for any hazards that may be implicated by this function.

Conflicts Between Availability and Safety

For the general public, the safety of an ATM system is largely perceived to be a matter of assessing the likelihood of a "system crash", i.e., a sudden, unplanned and severe loss of system functionality. Such concerns have, for example, have recently prompted headlines in the U.K. with the deployment of a new ATM system at Swanwick.

Without a doubt, the sudden, unplanned and severe loss of system functionality is a very significant event that could potentially contribute to an accident in a busy airspace. However, there is a risk that too much emphasis on system availability could actually jeopardize safety.

There are many potential causes of a severe failure in a complex software system – for example, the exhaustion of limited resources such as memory, an invalid reference to a memory location or a computational error such as attempting to divide by zero.

In general, fault tolerance strategies involve the detection, containment and, in some cases, correction of a problem. To mitigate system crashes, software mechanisms called "exception handlers" can be used to tolerate faults. For servers, there are also coarser-grain mechanisms for tolerating faults such as automatically switching over to a standby machine in a manner that is mostly transparent to controllers. For workstations connected to the servers, part of the overall fault tolerance strategy may involve providing controllers with a means to disconnect the workstation from a failed server or failed network and use the standalone workstation with locally stored data as an emergency measure.

In some cases, the problem may be benign. But in other cases, it may be symptomatic of a problem that could potentially cause a controller to be misled by the system in an unsafe way. For example, a problem that interferes with the timely update of Mode C altitude data on a situation display could rapidly and directly result in a loss of separation.

Safety concerns arise when faults are not merely tolerated by the system, but occurrences of the fault are effectively hidden from end users. To put it simply, it is not safe to design a system to tolerate faults "at any cost", especially if this involves masking problems and creating an illusion that the system is operating normally.

To avoid the introduction of new safety risks due to overzealous fault tolerance, the use of fault tolerance as a means of achieving availability objectives must be considered in terms of its visibility to end users. Sotirovski [4] describes an approach to "large-grained objects" based on experience with the development of the Canadian Automated Air Traffic System (CAATS). A key insight is to choose an appropriate level of granularity for fault tolerance so that system users such as controllers can be notified of how the service has been degraded. This avoids the dangerous situation of creating an illusion that the system is operating normally.

Choosing the appropriate level of granularity is important because the notification provided to users needs to be understandable in such a way that the users can work around the problem. It is not meaningful to notify a controller that a "divide by zero" exception occurred in a particular function. But it is meaningful to notify a controller that a problem with the flight data for a particular flight has been detected. In this case, the controller can probably work around this problem by re-creating the flight, or resorting to a paper strip approach for this particular flight.

Fault tolerance strategies are intended to allow the system to survive the occurrence of a fault that might otherwise bring the system to a screeching halt. In general, fault tolerance should be viewed as "life raft" rather than a repair to the system. It would usually be prudent for a controller to regard the notification of a failure as a prompt to look for a more durable solution. In some cases, this might mean emergency measures to shed jurisdiction to adjacent sectors or centers depending on the nature of the failure.

The difference between viewing fault tolerance as a "life raft" and viewing it as a form of repair

could have a significant bearing on the design of an ATM system. The "life raft" view of fault tolerance is likely to encourage relatively simple mechanism design to be extremely robust for a limited period of time. But if fault tolerance is viewed as a form of repair, then there is likely to be more emphasis on designing degraded modes of operation to "look and feel" as much as possible like the fully operational system.

The concept of safety espoused in this paper favors the "life raft" view of fault tolerance, while a different concept of safety that places less emphasis on a distinction between safety, reliability and availability would likely favor an approach that views fault tolerance as a form of repair.

Safety-Related Standards and Guidelines

A variety of general sources of safety risk for advanced ATM functionality have been discussed in this paper. Most of these problems would not necessarily be recognized as potential sources of safety risk in an approach to system safety that mostly views safety in terms of other qualities such as reliability and availability. As discussed, there may even be situations in which the pursuit of reliability or availability objectives might actually jeopardize safety.

Unfortunately, some of the standards and guidelines called upon in the development of advanced ATM functionality fail to adequately distinguish safety as a separate quality of a software system. This is typically expressed by an emphasis on the detection of defects in the implementation that could cause the actual behavior of the system to deviate from its required behavior. It is also often expressed by an emphasis on attempting to quantify safety risk in terms of the estimated likelihood of latent defects in various portions of the implementation.

In the author's experience, a significant portion of the unsafe behaviors of a safety-related system are consequences of the requirements, rather than deviations from the required behavior. In other words, the safety problem often originates with the requirements. In such cases, testing (against the requirements) is not likely to be very effective as a means of minimizing safety risk. If some aspect of the requirements entails unsafe behaviour, requirements-based testing will only ensure that this unsafe behaviour has been implemented. For similar reasons, an attempt to quantify safety risk in terms of latent defects is not likely to provide a sound basis for deciding to deploy a safety-related system.

On a positive note, MIL-STD-882C ("System Safety Program Requirements") is an example of a standard that could be used to establish the foundation of an effective approach to the safety analysis of an advanced ATM system. A key aspect of this standard is its emphasis on the identification of hazards. Each hazard postulates the occurrence of an unsafe condition that could directly contribute to the occurrence of a mishap. For example, the display of a corrupted Mode C altitude is likely to be one of the hazards identified for any ATM system that involves the acquisition, processing, distribution or display of Mode C altitudes.

The identification of hazards creates a framework for the assessment of safety risk that allows questions about safety to be separated from questions about reliability, availability and other qualities. Ideally, the overall test methodology for development of an advanced ATM system will dedicate a portion of the testing effort specifically to a safety verification activity that is driven by the hazard analysis rather than the functional requirements.

Summary

Stakeholders must look beyond the safety-net functions to thoroughly understand the safety impact of introducing advanced ATM functionality to an operational environment. This includes consideration of the various categories of problems highlighted in this paper. Stakeholders must also be careful to distinguish safety objectives from other objectives related to system reliability and availability. With a sound approach to safety, it is possible to minimize safety risks to an acceptable level and fulfill promises of greater efficiency and improved safety for the air navigation system.

Acknowledgements

I am indebted to a group of extraordinary Raytheon engineers, subcontractors and technical representatives of the customer who worked on the development of the Canadian Automated Air Traffic System (CAATS) for many insights into the safety of advanced ATM systems.

References

[1] Joyce, Jeffrey J., 2002, *Generic Safety Criteria for Air Traffic Management Systems*, Second Meeting of the U.S. Software System Safety Working Group, Massachusetts Institute of Technology, Boston.

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

[3] Leveson Nancy G. and Clark S. Turner, 1993, An Investigation of the Therac-25 Accidents, IEEE Computer, Vol. 26, No. 7, pp. 18-41

[4] Sotirovski, Drasko, 2001, Towards Fault-tolerant Software Architectures, Working IEEE/IFIP Conference on Software Architecture (WICSA 2001), Amsterdam, pp. 28-31