Generating Procedures and Recovery Sequences: A Formal Approach

Michael Heymann
Technion, Israel Institute of Technology, Haifa, Israel
heymann@cs.technion.ac.il

Asaf Degani & Immanuel Barshi
NASA Ames Research Center, Moffett Field, California, USA
{adegani, Ibarshi}@mail.arc.nasa.gov

This paper presents a formal approach for the analysis and development of effective, safe, and efficient procedures for abnormal and emergency situations. The focus is on methods for describing the behavior of the underlying machine, specification of desirable and unsafe regions of operation, and an algorithmic approach for computation of optimal action sequences. We discuss current gaps in procedure development and conclude with some of the challenges that lie ahead.

Introduction

A procedure is "a particular course of action or way of doing something" (Webster, 1989). It is also defined as "the act of proceeding from a source" and the "action of proceeding or going on to something" (Oxford English Dictionary, 1991). As such, any given procedure begins with some recognition of the system's initial (or current) state, details a specified course of actions, and provides a notion of a desired end state (Degani and Wiener, 1994). The act of executing (proceeding with) a procedure implies dynamics, and, of course, time. Timing issues are a critical and not-so-well understood aspect of procedures in general, and emergency procedures in particular. Timing issues arise in a given procedure at various levels: (1) in the temporal sense (when it should be performed), (2) in the sequential sense of what follows what (Degani, Heymann, & Shafto, 1999), (3) in the interaction between action sequences and the environment (e.g., "wait 60 seconds for the engine to cool down before proceeding to the next step"), and (4) in the overall execution period and the opening and closing of "windows of opportunity" to accomplish action sequences (e.g., "we have 15 seconds to accomplish the procedure before the engine shuts down automatically") (see Raby & Wickens, 1994). Furthermore, when multiple procedures are executed concurrently, delicate timing and synchronization among the procedures (such that an action sequence in one procedure does not block an action in another procedure, for example) are additional critical, yet poorly understood, design issues (see Degani, 2004 Chapter 13). Finally, with respect to emergency procedures, it is important to note here that such procedures always constitute a "race against time." It involves the sudden appearance of a degraded condition with an imminent path to a catastrophe on the one hand, and the existence of measured steps to prevent this catastrophe on the other.

What follows is a general framework for analyzing procedures in the context of a dynamic and complex system. We begin with a theoretical discussion of how to view emergency procedures from a formal perspective and then introduce a working example to illustrate the concepts. We then describe four generic phases in executing a typical emergency procedure, and introduce an algorithmic approach for computing the action sequences of a given procedure.

A Formal Approach for Procedure Analysis

Technological models of systems (e.g., engines, hydraulic systems, life support systems) and their dynamic behaviors are essential to analysis of procedures because they provide detailed descriptions of the system's behavior and functions. Therefore, the first step in our approach is system descriptions and representations that allow the designer to analyze which action sequences are available to mitigate the consequences of failures and drive the system to recovery.

We describe the system's operating domain as a finite state set (see Figure 1). At any instant of time, the system resides in some state or region of the state set. As the system evolves, it undergoes state changes called "transitions." These are normal changes in the system and represent dynamics, mode changes, and various configurations that take place. In this context, we distinguish between two types of such transitions: *controlled transitions* that are manually triggered by the user; and *dynamic transitions* over which the user has no control. These dynamic transitions are

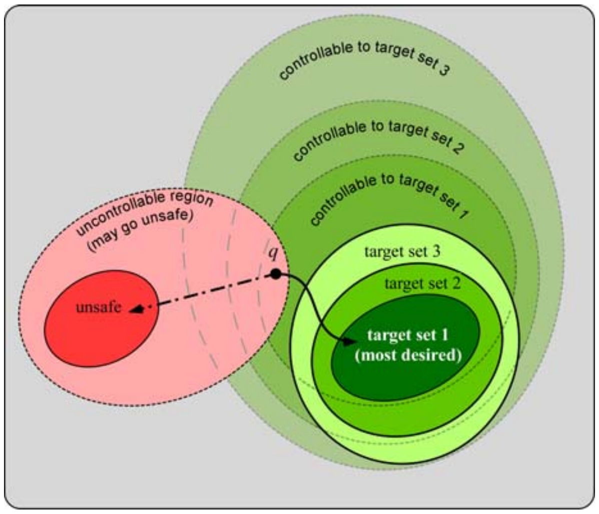
controllable to target set 3
controllable to target set 2
controllable to target set 1
uncontrollable region (may go unsafe)
q
unsafe
target set 3
target set 2
target set 1 (most desired)

Figure 1. *Regions in the system's state space*

triggered either by the system itself (timed or automatic transitions) or by the environment (as disturbances).

When a serious malfunction occurs, the system is thrown out of its normal operating region and an imminent path to a catastrophe appears. Ideally, we would like to drive the system back into a safe and normal operating region, which in Figure 1 is called target set 1. However, a complete recovery is not always possible. Therefore, if we cannot drive the system back to the most desired region (target set 1), we at least want to drive it to a minimally degraded region—which may not be the most desirable solution, but is at least second best, given the situation. We mark this "second best" region as target set 2. If we can't drive the system to target set 2, then we would try to at least achieve target set 3, and so on.

Target set 1 is a complete—and the most desirable—recovery; the rest of the indexed regions represent degraded performance. We therefore collectively call these regions, ranging from a low-indexed target set (target set 2) to a higher-indexed set (target set *n*), the *degraded region* set. In addition to the desirable (target set 1) and degraded (target sets 2 to *n*) regions, there is another region in the state set. This is the *unsafe region*. Entrance into this region is considered to be catastrophic (e.g., engine exploding, loss of control), and must be prevented at all costs.

Subregions and Action Sequences

When and if the system, for whatever reason, gets thrown outside of its normal region of operation (e.g., into state *q* in Figure 1), we want to determine a sequence of actions that will drive the system to the lowest indexed target set possible, without ever entering the unsafe region. In addition, we may place other requirements or constraints on the selection of an action sequence, such as a bounded time of execution (in most cases, as fast as possible), preferred paths, the likelihood of cascading failures, and minimization of impact on related subsystems.

But before we consider the possible action sequences, we need to identify some additional regions in the state set that relate to transitions. First, let us identify and mark the set of states, or region, from which there exists a sequence of transitions that could lead the system to the unsafe region. The pink area in Figure 1 defines this region; from every state within this region, a sequence of dynamic transitions may potentially lead the system to a catastrophe. For example, if, following an engine fire, the users or the automated system do not take action to stop fuel flow into the engine, the engine will inevitably transition towards an unsafe state and may eventually explode. We denote such a transition a with dash-dot line.

Next, we identify the regions from which the system can be driven finitely (and, this time, controllably) to the desirable and degraded target sets. Thus, the controllable region to the desired region (target set 1) consists of all states from which the system can be driven (either by the user or an automated system) in a finite sequence of transitions to target set 1. Along the same lines, we show concentric areas describing the controllable regions to target sets 2 and 3.

As can be seen in Figure 1, some of the regions interact; there are region-inclusions and region-intersections. State *q*, for example, where the system landed following the malfunction, resides in the intersection of the "uncontrolled region to unsafe" and the "controlled region to target set 1." Therefore, from state *q* the system can uncontrollably pass into the unsafe region, while at the same time there exists another set of transitions that can potentially drive the system to the desired target set 1 region. The existence of such an intersection with its two distinct paths (one going to an unsafe region and the other to a target set) is what defines an emergency procedure. (If there is no path to an unsafe region there is no need for an emergency procedure, and if there is no path to a target set the situation is already hopeless). It is important to note here that in most cases these two distinct paths compete temporally, and the requirement of an effective and safe procedure is to minimize the likelihood or risk of unwanted behaviors and to avoid, at all costs, going to an unsafe condition.

Example: Hot Engine During Start

Three elements must be in place to perform an formal analysis of a given system and its procedures: (1) a model of the machine's behavior; (2) a representation of the various regions of operations (unsafe and target states, controllable and uncontrollable); and (3) description of the procedure's specifications (e.g., goals and constraints such as time to execution, preferable paths, minimization of impact on subsystems, and various engineering and cost analysis trade-offs). The resulting model, which incorporates all three elements, can be based on any one of several existing or emerging modeling formalisms for discrete-event systems (e.g., Petri nets, Statecharts, etc.) or hybrid-systems models that combine discrete-event representation with dynamics (Heymann & Meyer, 1997; Ramadge & Wonham 1987).

Figure 2 is a simplified discrete-event model of an engine. The model describes the various states of the engine and the dynamic transitions among them. The initial state of the engine is OFF. The user (or an automatic controller) starts the unit by engaging the starter; now the engine is cranked and RPM increases. Once the RPM value has reached a specified set point, fuel is injected and the engine's speed and temperature begin to increase. The engine can either settle to within the normal operating range, or over-speed and overheat. Whether the system will transition to normal operation or to the high-temperature state is non-deterministic. That is, most of the time the start will be normal, but every so often a start will result in an over-speed and high engine temperature. When and if the engine temperature is extremely high, the engine can explode.

In the event of a high engine temperature (overheat), an effective and safe action is to first shut off the fuel valve. Through this action, it is possible block the potential transition into the unsafe and catastrophic region (explosion). Note that at the onset of "overheat" there exists a path, denoted with a dash-dot line, that can take the system uncontrollably to an unsafe region (marked in red). At the same time, there also exists a path to recovery, as will be discussed next.

Say that we were able to turn the fuel switch "off" in time and avoid explosion. Although no fuel is injected into the engine, the situation is still dire. If we sit on our hands and do nothing, the engine temperature will remain high and eventually the engine's internal components will disintegrate.

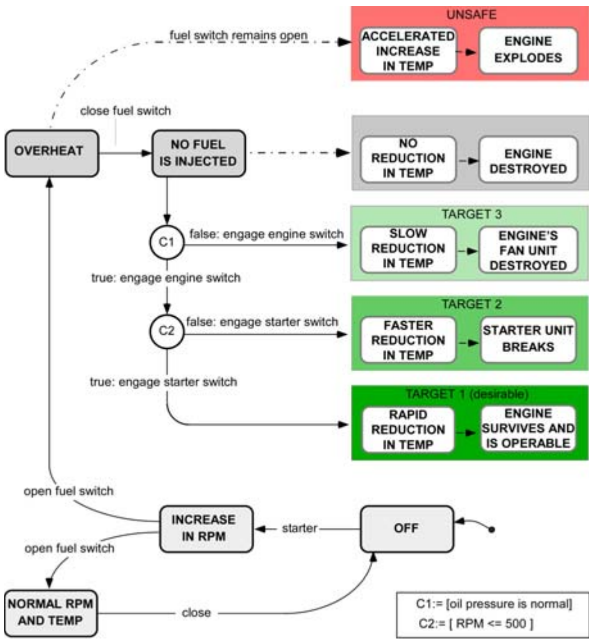

Figure 2. Machine model of a power unit with operational regions

Since such an outcome is undesirable by any account, our next step is to improve the situation. This is done by engaging the "engine control switch" that will allow the engine's fans to freely rotate and cool down the engine. However, the particularities of the situation, namely the extent to which overheating has affected the internal components of the engine, can impede our efforts. For instance, condition C1 represents the requirement for normal oil pressure: If the oil pressure is normal (*true*), it is possible to engage the engine control and rotate the fan, thus cooling down the engine. If, on the other hand, the oil lines that supply lubrication to the fan unit have heated up and ruptured, there will be no oil pressure (*false*).

In the latter case, any rotation of the fans, albeit reducing the temperature, will destroy entire the fan unit. We are thus faced with a dilemma—either to rotate the fans, cool the engine, and destroy the fan unit, or to let the engine burn up slowly. Based on engineering cost analysis, we conclude that it's better to sacrifice the fan unit (which can be replaced) for the sake of avoiding engine destruction. In that case, when we rotate the fans, the system transitions to degraded target set 3 and there we stop. This is the best we can do, given the situation at hand.

Proceeding with the analysis, we consider the case when the oil pressure is normal (condition C1 is *true*). Once the engine temperature has dropped, it is possible to engage the starter and let the engine shaft rotate and further cool down the engine for a longer period of time. This, however, can only be done when the temperature has dropped significantly, and when the engine is at a low RPM (so as not to grind the starter). Specifically, if the RPM is at or below 500 RPM (condition C2 is *true*), and we engage the starter for more than 60 seconds, it is possible to achieve full recovery and save the engine. That's our most desirable, target 1, goal.

However, if the RPM is well above 500 (condition C2 is *false*), continual cooling of the engine cannot be achieved and permanent damage to the engine will occur. It is possible to engage the starter when RPM is somewhat above 500 to continue cooling the engine, but this will destroy the starter unit. Again, based on engineering cost analysis we are willing to sacrifice the starter unit for the sake of cooling the engine temperature to normal. This is our target set 2—not the best outcome, but definitely a recoverable one (after replacing the starter) with respect to future engine operation.

By now we have identified and accounted for all pertinent regions (unsafe, target, etc.) in the state set. Likewise, we identified all the transitions among the states and noted their consequences. With such categorizations of both states and transitions, it became clearer what possible paths are available for recovery; what to avoid and what to seek. The analysis helps in producing and evaluating the set of action sequences that will eventually appear in the procedure.

One of the benefits of using such an approach in the process of designing emergency procedures is that the shared knowledge of all people and disciplines involved can be represented in the model. Currently, in every industry—from medicine, to nuclear power, to process control, to aviation and space—procedures are designed in an ad-hoc fashion. The state of the art in procedure design involves calling upon the expertise of engineers, users, and human factors professionals to review the proposed sequences of actions. Yet no shared model is used to support the design group in the analysis, design, and review process. Based on our observations and field studies of procedure development processes, we believe that the use of even a simplified and/or incomplete formal model can go a long way toward improving the design process and the overall effectiveness of the resulting procedure.

Structure and Outline of an Algorithmic Approach for Procedure Synthesis

In the following section we take our approach one step further, from models and concepts that can support analysis to methods for automatically generating procedures. Our motivation for this stems from the observation that with increased use of automatic control (e.g., in aircraft subsystems) and the high complexity of modern systems, the ability of engineers and procedure designers to visualize, inspect, and evaluate the correctness of procedures is reduced, as there may be thousands of possible permutations and possible action sequences. Our intent is to develop methods and tools that can support the design process by generating a set of candidate sequences. Eventually such methods can be incorporated into online systems that will generate an effective and safe action sequence for any anticipated condition, and perhaps for even unforeseen situations.

In the introduction of this paper we defined a procedure as having an initial state, a specified course of actions, and a desired end state. In this section, we will expand on that definition and discuss four main phases in the execution of abnormal and emergency procedures, then touch on several important design considerations. This will lead us into the last part of our paper, where we present an algorithmic approach for generating effective procedures in the context of a dynamic system.

Phases of Procedure Execution

Few emergency situations present unambiguous cues or indications such that the user (or the automated system) can immediately initiate the necessary sequence of actions. In most other cases, the current state of the system can only be determined by some form of diagnosis (e.g., is the fire in air ducts or in the pneumatic system itself?). Often this diagnosis requires that the user or the automation take certain actions on the system itself to determine its actual state. In the case of smoke in the cabin, it may be possible to send a flight attendant to visually inspect and the cabin and determine the type of smoke and its source. However, it may require several manifold reconfigurations to isolate a leak in an hydraulic system. The point is that in many cases it is necessary to first diagnose and determine the current state of the system. This, in turn, determines which action sequence is to be followed. Therefore, the first phase of an emergency procedure is defined here as *determination of the current state of the system.*

Phase 2 in the execution of an emergency procedure is the *blocking of possible transgressions into catastrophic states*. Here our goal is to immediately and effectively block any dynamic transition (e.g., temperature rise, fire) that can drive the system into an unsafe and potentially catastrophic region (e.g., explosion). In this phase, which is commonly time critical, actions are drastic and have serious consequences. In the engine example described earlier, the imperative blocking action is shutting off the fuel valve to the engine to immediately stop the injection of fuel to the hot engine (and avoid transition to an unsafe state). In many systems, such drastic actions also carry the burden of being irreversible. For example, in many aircraft, once an electrical power unit is disengaged from an engine, it cannot be re-engaged in flight.

Phase 3 focuses on *preliminary stabilization of the failed system*. Here, while the system is blocked from accelerating toward catastrophe, it is still not functional and may be unstable. To deal with this situation, we begin measured steps to first stabilize the system. In the engine example, the preliminary stabilization is the action of engaging the engine control to keep the fans rotating and allow immediate venting and cooling of the engine.

Once the system is stabilized and the level of urgency wanes, it is possible to begin Phase 4—*optimized steps toward recovery*. Here we may have time to consider options and try to find a path that will yield the most desirable (lowest-index) recovery. And while we aim for a full recovery, sometimes we must accept the fact that full recovery cannot be guaranteed. In these situations, we accept a degraded recovery and proceed to it judiciously.

Recognizing the occurrence of unpredictable environmental or dynamic-internal events, especially when only partial information about the underlying system is available, is critical for analysis and design of sequences that take place in phases 3 and 4. Many unexpected events can disrupt the sequence of recovery actions. For example, another system can fail and block our ability to take measured steps towards the intended target state. Even worse, another path to catastrophe can open up and force us to defer or abandon the desired procedure altogether. Thus, all abnormal and emergency procedures fall into the category of being contingency driven, where, in principle, each step must be evaluated given the overall situation (and also the probability of additional failures that may disrupt the sequence).

Finally, procedures are rarely conducted in isolation. They usually interact with other procedures that are going on at the same time or will have to be executed later on. Depending on the nature of that interaction, what may appear in the context of the failed subsystem to be the most desirable target state may not be so in the context of the larger system. Consider, for example, a situation where driving a failed system to target state 1 would result in a certain unit being irreversibly disengaged and shut down, yet that specific unit is critical for some future and unavoidable operation (e.g., for landing). In this situation, it may be prudent to drive the failed system to partial recovery (e.g., target state 2) and keep the critical unit online.

In commercial aircraft operation, some of these considerations are listed within the procedure steps so the pilot will be alerted to the consequences of his or her actions when considering the options available, given the situation. The point here is that an emergency procedure should not only be viewed in a local context, but also in the global. In some situations it will be necessary to forfeit local recovery in order to maintain the overall health of the system.

Synthesis of Procedures

In this section we outline the basic algorithmic steps in a typical application of the proposed methodology for synthesis of an effective, efficient, and safe procedure. The computation of the procedure sequence takes into account an important distinction we have made earlier between *controlled transitions* that are triggered by the user and *dynamic transitions* over which the user has no control. The are five main steps in the way we compute the action sequences for a procedure:

Step 1. Computation of the region of uncontrollable to unsafe. This is the set of states from which there exist sequences of dynamic transitions that may lead the system to unsafe states. To accomplish this computation, we consider the machine model in which all controllable transitions have been (temporarily) deleted and only the dynamic transitions remain. We then reverse the directions of the dynamic transitions (each source state of a transition is interchanged with its destination), and compute the set of reachable states from the set of unsafe states. The resultant set of states is the region of uncontrollable to unsafe.

Step 2. Computation of the controllable region to target j, j=1, 2, 3, n. The algorithm used here is based on previous work by Brave & Heymann (1990) on stabilization. The essence of the algorithm consists of finding the maximal region in the state set that (1) has no dynamic transition sequences that might loop indefinitely without ever reaching the target state, thus rendering the procedure ineffective, and (2) which from each state can reach the target set in a finite and bounded number of transitions.

The algorithm proceeds iteratively, starting from the target set outwards. At the start (0^{th} iteration), the candidate set consists of the target set itself. At iteration i, the algorithm creates the i^{th} candidate set by adding to the $(i-1)^{th}$ set all states which have at least one emanating controlled transition that enters the $(i-1)^{th}$ candidate set and all their emanating dynamic transitions enter the $(i-1)^{th}$ candidate set, as well. The algorithm terminates at the iteration for which no new states with the mentioned properties can be found. The last candidate set is the sought-after controllable region to the target set.

Step 3. Transition and state cost assignment. Once the state set has been classified as described above, we assign costs to the various states so as to express the *undesirability* of reaching these states. Thus, we assign a higher cost to a state in a higher-indexed target set than to a state in a lower-indexed target set. Likewise, we assign high cost to states in the controllable region to the high-indexed target states and low cost to states in the controllable region to low-indexed target states. Next, we assign different costs to the transitions based on operational considerations (e.g., irreversibility of actions will get a high cost, while availability of components such as fire suppression bottles that help in the recovery will get a low cost). Along the same lines, states that require complicated and time-consuming diagnosis receive higher costs than states that do not. Finally, we assign very high cost to states in the region of uncontrollable to unsafe, and the highest cost to states in the unsafe region.

Step 4. Probability assignment to dynamic transitions. Dynamic transitions emanating from a given state may depend on the given state, time of entry into the state, time of residence in the state, and various other case-dependent considerations. Likewise, it is possible to obtain historical data about the probability of a given dynamic transition (e.g., 20 percent of the time after the system fails, it also burns). These probabilities are expressed quantitatively and assigned to all relevant transitions.

Step 5. Optimal procedure synthesis. The optimal procedure is synthesized so as to minimize the cost of blocking, stabilization, and recovery. To understand how this is accomplished, note that each recovery execution may include, along with its designated probabilities, dynamic transitions. Therefore, its sequences may terminate at more than one possible end state. We initially assign to each possible recovery sequence a cost that is equal to that of its end state. A sequence that enters an unsafe state is not permitted to continue beyond that state, and hence is assigned the cost of the unsafe state. Other sequences are permitted to continue to the lowest achievable end state (with correspondingly lower cost assignments).

All executions of minimal cost (in case there are more than one) are then chosen as candidates for selection as the optimal action sequences. The final selection can either be made manually by the user, or it can be computed by minimizing transition costs (e.g., weighted with respect to probabilities of occurrence of dynamic transitions).

Conclusion and Future Directions

In this paper we suggest a formal approach to the problem of developing emergency and abnormal procedures. The approach proposed here aims to enhance the current practice of procedures development by augmenting it with a formal methodology.

Designing effective, efficient, and safe procedures for abnormal and emergency situations is a complex process that has not been systematically addressed. Current practices are well intended, but insufficient to meet the challenges of future systems. The framework presented here is a first step towards meeting such challenges.

In this context, many problems arise that require further research: For example, current procedures often assume a single failure. When there is more than one failure, it is left to the users to prioritize their actions and interleave all the emergency procedures into a single sequence strand. Under extreme time pressure and stress, users performing such prioritization and interleaving may take actions that are potentially unsafe. Currently, there is very understanding of such interleaved procedures, let alone guidelines to support users in this difficult task.

Another topic that requires further research concerns the design of annunciations and indications of systems state to aid users and automated systems in determining the current state of the failed system and initiating the appropriate procedure. This involves criteria for instillation of sensors and various aspects of diagnosis processes, partial observations, and information abstraction (Heymann & Degani, 2007).

Generally speaking, we lack formal definitions of the class of problematic situations (e.g., incorrect sequences, timing problems, deadlocks, blocking) that render a procedure ineffective. This set of properties is critical for analysis, as it is the input for any methods for formal verification of procedures. Likewise, formal and heuristic criteria addressing basic human-machine interaction problems that make a procedure prone to error are still missing. Finally, any analysis, generation, and verification process must be extended beyond technical and basic human-machine interaction issue to include users' cognitive and perceptual limitations, crew coordination (e.g., two pilots coordinating the execution of one or more emergency procedures), and crew-automation coordination (in the case of an online system for dealing with emergencies).

Acknowledgments

The work described in this paper was funded by NASA's basic research and technology effort on human-automation interaction as well as the Integrated Intelligent Flight Deck project, Aeronautics Research Mission Directorate. The first author was supported by Grant NCC 2-798 from the NASAAmes Research Center to the San Jose State University. Michael Shafto and Kevin Jordan provided support and encouragement for this research work.

References

Brave, Y. & Heymann, M. (1990). On Stabilization of Discrate Event Processes. *International Journal on Control*, 51, pp.1101-1117.

Degani, A. (2004). *Taming HAL: Designing interfaces beyond 2001*. New York: Palgrave-MacMillan.

Degani, A., Heymann, M., & Shafto, M. (1999). Formal aspects of procedures: The problem of sequential correctness. *Proceedings of the 43rd Annual Meeting of the Human Factors and Ergonomics Society*. Houston, TX: Human Factors Society.

Degani, A., and Wiener, E. L. (1994). *On the design of flight-deck procedures*. NASA Technical Memorandum #177642. Moffett Field, CA: NASA Ames Research Center.

Heymann, M., & Degani, A. (2007). Formal analysis and automatic generation of user interfaces: Approach, methodology, and an algorithm. *Human Factors.* 49.

Heymann, M. Lin, F. & Meyer, G. (1997). *Synthesis of Minimally Restrictive Legal Controllers for a Class of Hybrid Systems in Hybrid Systems.* In P. Antsaklis, W. Kohn, A. Nerode and S. Sastri, Eds., LNCS 1273, pp. 134-159, Springer Verlag.

Raby, M., & Wickens, C. D. (1994). Strategic workload management and decision bias in aviation. *International Journal of Aviation Psychology,* 4, pp.211-240.

Ramadge, R. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event. processes. *SIAM J. Control and Optimization, 25*(1), pp. 206-230.