A Formal Methodology, Tools, And Algorithm For The Analysis, Verification, And Design Of Emergency Procedures And Recovery Sequences

Asaf Degani
Computational Science Division
NASA Ames Research Center

Michael Heymann
San Jose State University Foundation

Immanuel Barshi
Human Factors Research and Technology Division
NASA Ames Research Center

EXECUTIVE SUMMARY

Central to the success of the Exploration mission is the utility of the Crew Exploration Vehicle (CEV) as a safe and efficient system for transporting humans into space. One of the most critical aspects of ensuring the continual safety of the CEV lies in the crew's (and the onboard automation's) ability to cope with and recover from emergency situations. If one or more emergency conditions arise, astronauts must quickly initiate and successfully execute procedures to mitigate the failures and then recover; these procedures must be performed under tight timing constraints, e.g., during ascent to the Main Engine Cut Off (MECO) point in the current space shuttle system. Similarly, when the CEV is flown in fully autonomous mode, it will be the responsibility of the onboard computers to automatically initiate and execute emergency procedures and recovery sequences. Most likely, however, future exploration systems will be designed such that some (emergency) actions will be initiated by humans and some (recovery) actions will be performed automatically. Whether preformed manually, automatically, or in a manual/automatic combination, it is critical that such emergency procedures be correct, efficient, usable and, most importantly, safe. The analysis, verification, and design of these highly critical and time urgent emergency procedures and recovery sequences are the focus of this project.

Current approaches and best practice for designing procedures are based on employing the knowledge and experience of system designers, simulator trainers, and astronauts, so as to collectively formulate the "best" perceived way to mitigate and recover from an abnormal or emergency situation. After their formulation, procedures undergo extensive testing to validate their correctness and usability. Nevertheless, both the increasing complexity of underlying space exploration systems–engines, fuel, life support, and computers–and the growing reliance on automation as an agent that reconfigures itself and also executes some action sequences, make such traditional design approaches progressively difficult to employ (Degani and Wiener, 1994).

In particular, procedure developers have considerable difficulty fully comprehending all the possible paths in the systems and making sure that the procedures are indeed *correct*. That is, that the action sequence(s) specified in the procedure are guaranteed to be executable, achieve the procedures' goals, and insure the best course of recovery. Another problem common to current procedure design approaches is that there is no assurance that the selected action sequence is the "best" in terms of execution time, mitigation of potential failures, and maximization of recovery; many procedures developed in the traditional trial and error approaches are unnecessarily complicated because not all the possible recovery paths have been considered. As it currently stands, the procedure development and validation process for the International Space Station takes more than 27 months. It also requires extensive coordination, documentation and development traceability, as well as the use of expensive simulation facilities (Systems Operations Data File Procedure Validation/Data Source Information Plan, 2003).

To overcome some of the abovementioned difficulties, we propose to develop a formal approach for the design of emergency procedures and recovery sequences. In this formal approach we employ mathematical methods and tools to systematically analyze, verify, and synthesize action sequences. This methodology can be used both for verifying the correctness and suitability of an existing or proposed procedure, and for (algorithmically) constructing an efficient, reliable, and safe procedure. This new approach has already been utilized in the commercial aviation domain, and its main principles have been incorporated in a new industry standard (FAA Advisory Circular 120-80) for cockpit procedures dealing with in-flight fires. The design of procedures for in-flight fires is an area in which the aviation industry has

acknowledged its vulnerability following the fire onboard Swissair Flight 111 and the aircraft's subsequent crash.

Our approach is general can be applied to all exploration systems, such as CEV systems, habitat systems, powerplants, and rovers. The methodology is also general in the sense that it can be utilized for procedures in which actions sequences are [1] performed manually by the astronauts, [2] executed automatically by the system, and [3] where both the humans and the onboard automation perform and execute action sequences (which is the most common, yet, complicated, type of procedural interaction).

There are four primary components to our approach:

- First, we propose to extend an existing approach and methodology for analyzing the sequential correctness of procedures (Degani, Heymann, and Shafto, 1999). This formal approach, which was successfully employed in analyzing and identifying design errors in emergency procedures for commercial aircraft, will be augmented to cope with more complex systems, such as the CEV abnormal and emergency procedures. A key benefit of such an advanced approach for procedure design is that it will give designers of CEV procedures a rigorous and systematic method for considering and analyzing procedures at every development stage.
- Second, we intend to develop a formal modeling framework and an appropriate representation to describe the behavior of the underlying physical system (e.g., manifolds, valves, regulators, tanks) and the dynamics of these systems. In addition to the system behavior and dynamics, we will also incorporate into the modeling framework the information content (cockpit indications, warning lights, pressure values) that are monitored and acted upon by the astronauts in the process of executing the procedures. Here we draw on existing work in formal modeling of interfaces (Degani and Heymann, 2002) to insure that the information, upon which the procedure execution takes place, is also correct. As a result, the framework will allow us to combine the description of the system's behavior, information about the displayed and communicated state of the system, and the procedure's action sequences. Creating a composite model of these three elements will reveal potential problems such as errors, inefficiencies, and excessive-complexities in the action sequences.

- Our third step will be to capture in formal language the essence of what constitutes a "good" procedure and what constitutes a "bad" procedure. For any analytical and verification process to take place, procedure correctness criteria, or properties, must be developed. Here we will draw on existing work in interface verification in which related criteria (such as error states, restricting states, blocking states, and deadlocks/livelocks) have already been defined and used for verification of cockpit displays for the Boeing 737 Next Generation aircraft (Degani, Heymann, Meyer, Shafto 2000). Of primary interest, we will ensure that the proposed action sequences are achievable under all possible system configurations and that recovery is possible. We will explore other verification criteria relating to deployment of more than one procedure concurrently (i.e., multiple procedures) as well as to astronauts' physical and cognitive abilities in executing a given sequences. By employing an extensive set of procedure design criteria, we will be able to identify procedure design errors prior to actual fielding of the system. Likewise, this process will also allow us to consider issues of procedure efficiency, learnability, and training.
- The fourth part of the project is the development of an algorithm for synthesis of procedures. This algorithm will be used to generate a single sequence of actions, or set of contingency actions, to overcome a potentially catastrophic failure. Naturally, we want this (contingency) sequence to be efficient, reliable and most of all, safe. For example, to synthesize a procedure for a condition in which the Main Propulsion System Helium Pressure (Pre-MECO) is below normal and can lead to an explosion, the algorithm will search for a path that will result in mitigation of the condition and prevention of digression into an unsafe (explosion) situation.

Finally, we emphasize that the proposed work with its development of a modeling framework, procedure verification criteria, and algorithms, will be performed in the context of spaceflight operations. For this, we will be using Ames's ISIS lab—a simulator facility for CEV development. In addition, by using videotapes of actual astronauts performing emergency procedures in the ISIS lab, we will obtain valuable information on how astronauts execute emergency procedures and obtain insight into what kind of astronaut-procedure interaction is more prone to error. This information will be fed back into the procedure correctness criteria so as to make them more comprehensive.

INTRODUCTION AND BACKGROUND

In January 2004, NASA established a long-term program to extend human presence across the solar system, a primary goal of which will be to establish human and robotic presence on the moon and Mars (NASA, 2004). A central concept of this new vision is that future space exploration systems must be sustainable. Achieving sustainability of exploration systems, especially at great distances, will require a capacity to deal with both foreseen and unforeseen failures and, most importantly, enable recovery. In order to meet this technological challenge and maintain sustainability, the state of the art in designing procedures, and in particular emergency procedures and recovery sequences, must be dramatically improved. This is critical not only for evaluation and validation of "canned" procedures, but also for the design of automated procedure systems that will be able to guide recovery given unforeseen and unexpected failures.

Objectives

The objective of this project is to improve the safety of various major systems such as the CEV, habitat systems, rovers, and any critical system in the exploration vision. Our specific aim is to develop procedure design methods that will facilitate and support sustained space exploration, with special emphasis on safety and reliable operation. In particular, we want to create computational methods and tools that will allow designers to develop better procedures to deal with abnormal and emergency situations. We will develop sophisticated approaches, methods, tools and algorithms for verification and synthesis of procedures for humans, automated systems, and human interaction with automated systems.

As it stands today, there is a dearth of research and development in the area of procedure analysis and design. In particular, there are no systematic and rigorous methodologies for designing procedures. This is a serious shortcoming, since all high-risk and complex systems employ procedures and action sequences. Consequently, the work proposed here will improve the process of procedure design, resulting in better procedures and improving the current procedure design process by making it more efficient. We believe that the approach and methodology described in this project has the potential to make a significant impact on the safety of current and future space systems.

As discussed earlier, there is a fundamental problem with existing procedure design processes. In every industry, from medicine, nuclear power, process control, to aviation and space, procedures are designed in an ad hoc fashion and there exist no theory, principles, methods, or guiding tools for procedures design (Degani and Wiener, 1994). This is a serious deficiency that is well recognized by aerospace companies such as Boeing and Airbus -- yet no design methodologies or solutions are being offered. The current state of the art in procedure design involves calling upon the expertise of engineers and users and are, to a large extent, a "design by committee" approach. Furthermore, with increased use of automation and the high complexity of modern systems, the ability of engineers and users to visualize, inspect, and evaluate the correctness of procedures is reduced, since there are thousands of possible permutations and action sequences possible. As a consequence, there are many incorrect procedures still in use today (Degani, 2004 – see Ch. 13). Specifically, our experience in commercial aviation shows that these lacks of rigor in procedures design permeate normal, abnormal, and emergency procedures (Degani, Heymann, Shafto, 1999; Degani and Wiener, 1994).

Moreover, current procedures (e.g., for the space shuttle) assume a single failure. When there is more than one failure, it is left to the astronauts to prioritize their action and interleave all the emergency procedures into a single sequence strand. When under extreme time pressure and stress, astronauts performing this ad hoc prioritization and interleaving process may take actions that are potentially unsafe. Currently, there is very little in a way of understanding such interleaved procedures, let alone a methodology for analyzing them.

We propose to create a general methodology for procedure design. This methodology will enable the development of procedures for fully automated systems as well as for partially automated systems in which the user (astronaut) is asked to perform critical actions such as commencing an abort, or shutting down engines and power units. The proposed work will be designed to enable better system management and operation, supporting a range of systems for exploration. In addition, our work will investigate methodologies and techniques for automating the process of procedure design, enabling the design of onboard computers algorithms that will synthesize and generate, on the fly, tailored procedures to deal with unique (e.g., unanticipated) failures. This capability will provide one of the missing components in the space exploration

vision -- the ability to recover from failures and problems while operating with limited and untimely contact with mission control.

RELEVANCE OF THE WORK FOR NASA MISSION, VISION, AND GOALS

Complex operations depend on proper procedures. Procedures form the backbone of any operation, and the more critical the operation, the more critical the procedures. NASA's exploration vision represents a new frontier in complex operations, a frontier for which current methods of procedure design are no longer adequate.

Current approaches to space missions assume reliance on Mission Control and on continuous real-time communication links. These assumptions are no longer valid as long-duration space exploration requires a much greater degree of vehicle and crew autonomy than do low earth orbit operations. Such exploration missions also involve a much higher level of interaction across systems and across components than ever before. Mission success assumes correct procedures which include recovery from both foreseen and unforeseen malfunctions. Incorrect or incomplete procedures may prove to be the limiting factor in NASA's mission. Thus, a new approach to procedures design and verification is needed.

Reliable, efficient, and safe procedure and recovery sequences are a critical building-block for any space exploration system, with or without human onboard. A systemic and systematic methodology for procedure design allows early identification of risks and enables mitigating such risks, thus increasing safety. Furthermore, clear, consistent, and comprehensive procedures maximize system efficiency on many levels ranging from minimizing astronaut and operator's training to facilitating maintenance operations. The proposed approach is generalizable, and is applicable across all NASA missions.

APPROACH AND METHODOLOGY

The approach proposed here aims to enhance the current practice of procedures development by augmenting it with a formal methodology that provides a rigorous, mathematically sound, and verifiable method for procedure analysis and design. There are four primary components in our approach:

- Our first step is to develop a formal approach for procedure analysis and development. This approach, which is based on previous work in this area (Degani, Heymann, and Shafto, 1999), will provide designers with a rigorous framework for the analysis and design of emergency procedures and recovery sequences.
- Second, we will be using computationally tractable methods to model the system components (e.g., engine) and their behavior (see Degani, 2004 for the use of the finite state machine models for describing such systems). Of primary interest is the use of the models to map out the behavior of the system and identify the most efficient and safe action sequences (see Degani, Heymann, and Shafto, 1999).
- Third, we will be using a formal approach to define and encapsulate what constitutes a "bad" and what is a "good" procedure. Here we shall develop criteria for (or properties of) desirable action sequences in terms of well defined operational goals. These properties will allow us to quantitatively judge the "goodness" of a given procedure and will enable the development of verification and synthesis techniques. The procedure design criteria will also serve as the basis of procedure specification and design guidelines.
- The last element in our proposal is an algorithm for synthesis of procedures with the goals of generating action sequences that are correct, reliable, efficient, and safe. The focus in this element is on generating contingency action sequences in abnormal and emergency situations that will insure the best possible system recovery (under adverse circumstances). Using the theoretical approach described above and previous work done by Brave and Heymann (1990), we can automate the process of generating action sequences, thereby, enabling tools that compute, on the fly, recovery procedures for unforeseen emergencies.

A Formal Approach for Procedure Analysis

A procedure is "a particular course of action or way of doing something" (Webster, 1989). It is 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, a given procedure contains three stages: its beginning (source or initial) state, a particular course of actions, and an end, or terminal, state.

The act of executing (proceeding) with a procedure implies dynamics, and, of course, time. And indeed, timing issues are a critical and not so well understood aspect of many procedures

(see Degani, 2004 Ch. 13). Timing issues arise in a given procedure at various levels: [1] in the temporal sense (what follows what), [2] 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 procedure step"), and [3] 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"). Furthermore, when multiple procedures are executed concurrently, synchronization among the procedures (e.g., such that an action sequence in one procedure does not block an action in another procedure) is another critical, yet hardly understood, design issue. Finally, it is important to note here that every emergency procedure constitutes a "race against time." The sudden appearance of a degraded and imminent path to a catastrophe, and at the same time the existence of measured steps to prevent this catastrophe from happening and to begin recovery -- is what defines an emergency procedure

What follows is a general framework for analyzing procedures in the context of a complex and dynamical system:

The system's operating domain is its (finite) state set. At any instant of time the system resides in some state or region of the state set. As the system evolves it undergoes (discrete) state changes called "transitions." These are normal changes in the system and represent the passing of time, dynamics, mode changes, and various re-configurations that take place.

And then a malfunction occurs! At that instant the system is thrown out of its normal operating region and a degraded and 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, we recognize the fact that perfect recovery is not always possible; and hence, if we *cannot* drive the system to back to the desired region (target set 1), we would 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 denote these regions as target set 2 in the Figure 1. And if we can't drive the system to target set 2, then we would at least try to push the system to target set 3, and so on. Excluding target set 1, which is a perfect recovery, all these indexed regions represent degraded performance. We therefore

collectively call these regions, ranging from a low indexed target set (2) to a higher indexed set (n) -- the *degraded* region set.

In addition to the desirable (target set 1) and degraded (target set 1+n) regions, there is an important region in the state set that we distinguish qualitatively from all others. This is the unsafe region. Entrance into this region is considered to be catastrophic (e.g., engine exploding, loss of life-support systems) and must be prevented at all costs.

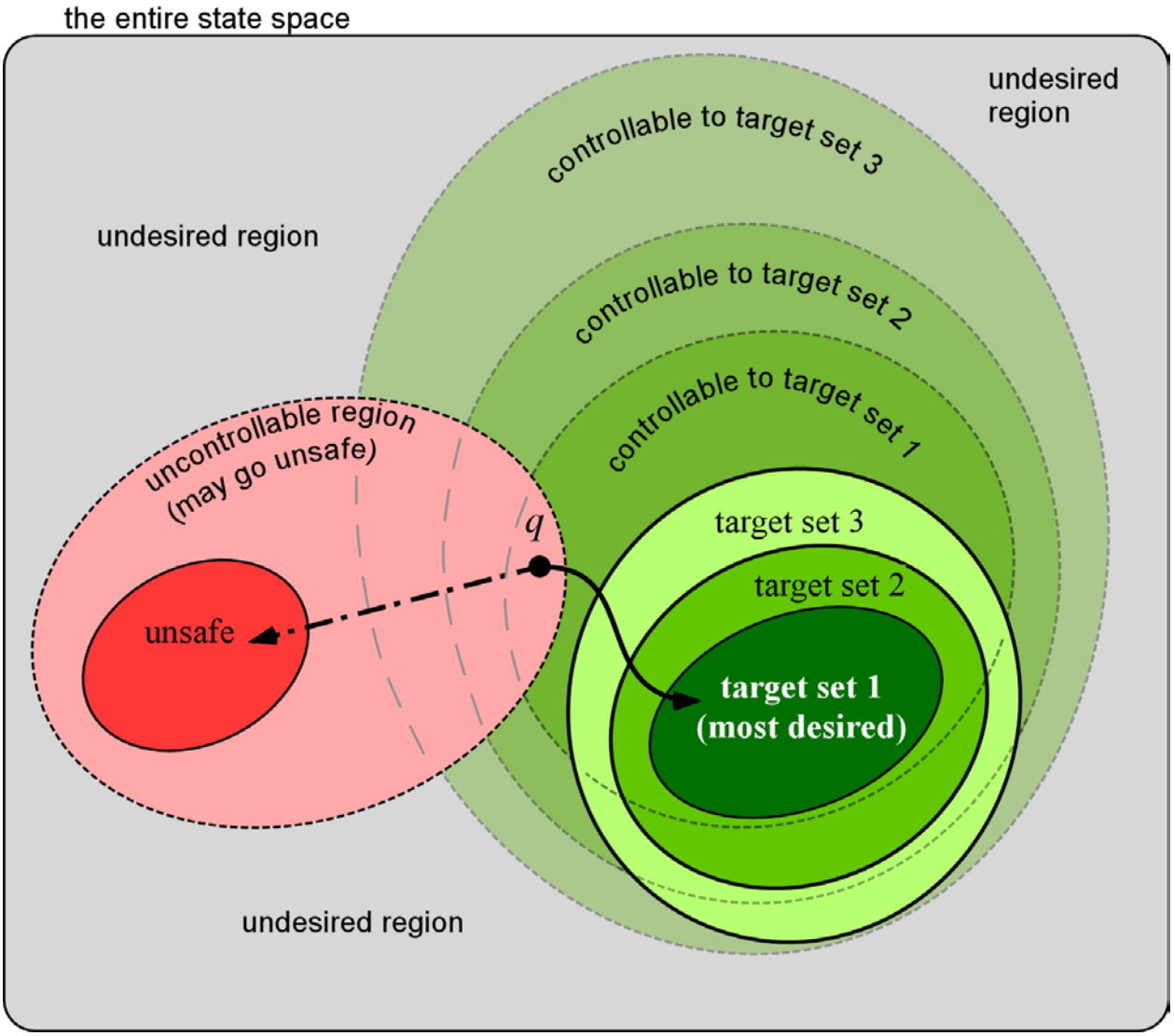

Figure 1. Regions in the system's state space.

Finally, we define the remaining region of the state set -- which is outside of the target sets and outside of the illegal region -- as the undesirable region. Staying in the undesirable region means that while we may have taken some action (e.g., emergency procedure) to mitigate the failure and block the system from becoming unsafe, the system as a whole is dysfunctional. For the system to become even marginally functional and operable we must begin recovery sequences.

Regions and Action Sequences

Now we come to the issue of procedure steps, or action sequences, to deal with the situation. When and if the system, for whatever reason, gets thrown outside of its normal operation region (to state *q* in figure 1) – we wish to determine a sequence of actions that will drive the system to the lowest indexed target set possible, without ever entering the illegal region. In addition, we may place other constraints on this procedure, such as a bounded time of execution (sometimes as fast as possible), preferred paths, the likelihood of cascading failures, and minimization of impact on related sub-systems. To analyze the situation and consider the possible action sequences, we first need to identify additional (sub) regions in the state set:

First, let us identify the set of states from which there potentially exists a sequence of dynamic transitions that may lead the system to the unsafe region. The pink area in Figure 1 defines this region; from every state within that region, a sequence of dynamic transitions (e.g., accelerated rise in engine temperature, rapid loss of oxygen) may potentially transition the system to a catastrophe. Such dynamic transitions, which are not directly controlled by the user or the automated system (and denoted as dash-dot lines in Figure 1), are critical for the analysis of emergency procedure. 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 system will inevitably transition towards an unsafe state and may eventually explode.

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 have concentric areas describing the controllable regions to target sets 2, and 3.

Naturally, and this is clearly seen in Figure 1, some of the regions interact; there are region-inclusions and region-intersections. State q, 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." Thus, from state q the system can, on its own (and uncontrollably), digress to the unsafe region; while at the same time there exists a set of transitions that can drive the system to the desired, target set 1, regions. 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 a precondition for considering emergency procedure and recovery sequences. (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 doomed). It is important to note here that in most cases the two distinct paths compete temporally, and the requirement of a correct procedure may be to minimize the likelihood or risk of unwanted behaviors and to avoid, at all costs, going to an unsafe condition.

Modeling of System Behavior and Analysis of Procedures

Technological models of systems (e.g., engines, hydraulic systems, life support systems) and their dynamic behaviors are essential to procedure analysis and design because these models provide a description of the system's behavior and functions. In other words, they provide the context for the analysis. Therefore, the second component of the project deals with development of system descriptions and representations that allow the designer to "map out the territory" (according to the theory described in Section 1) and see which action sequences are available to mitigate the consequences of failures and, ultimately, drive the system to recovery.

Three elements must be in place to perform such an analysis: [1] a formal model of the machine's behavior, [2] a formal representation of the operational regions (unsafe, undesirable, and all target states), and [3] a formal description of the procedure's specifications (e.g., goals and constraints such as time to execution, preferable paths, minimization of impact on sub-systems). The resulting model, which is a composition of these three elements, can be based on any one of several existing or emerging modeling formalisms for (untimed) discrete-event systems or (timed) hybrid-systems (Ramadge and Wonham, 1987; Heymann, Lin and Meyer, 1997).

What follows is an example of how we superimpose a formal representation of the operational regions (unsafe, undesirable, and all target states) on top of a formal model of the

machine's behavior so as to describe an emergency situation and analyze (or synthesize) a procedure. Figure 2 is a simplified finite state model of a certain power unit, or engine. The initial state of the engine is idle. An automatic system (or an astronaut) starts the unit by engaging the starter; and now the engine is cranked and RPM increases. Once the RPM value has reached a specified set point, fuel is injected and the engine speed and temperature begin to increase. The engine can either settle to within the normal operating range and be ready for operation, or the engine can over-speed and overheat (which is the abnormal case). The point is that whether the system will transition to normal operation or to the high temperature state, is non-deterministic. Most of the time the start will be normal, but from time to time (and we may have historical data about the likelihood) a start will results in an over speed and high temperature engine. And we know that when and if the engine temperature is very high, the situation is dangerous (because the engine can explode), and a procedure for mitigation of failure and recovery must be immediately initiated.

In the event of a high engine temperature (overheat), an efficient and safe action is to first close the fuel switch. By this action we block the potential digression into the unsafe and catastrophic region (explosion). Note that at the onset of 'overheat,' there exists a path that can take the system uncontrollably to an unsafe condition; at the same time there exists a path to safety and recovery. Thus, if we act quickly it is possible not only to prevent the catastrophe, but also to recover. We pursue that option fully recognizing that we are competing "against" the path to an unsafe state.

Let's say that we were able to turn the fuel switch to "off" before anything bad happened. The fuel is no longer injected into the engine. Now, if we do nothing, the engine temperature will stay high and eventually the engine will burn up and be destroyed. This is undesirable by any account, and therefore 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 rotate, thereby cooling the engine.

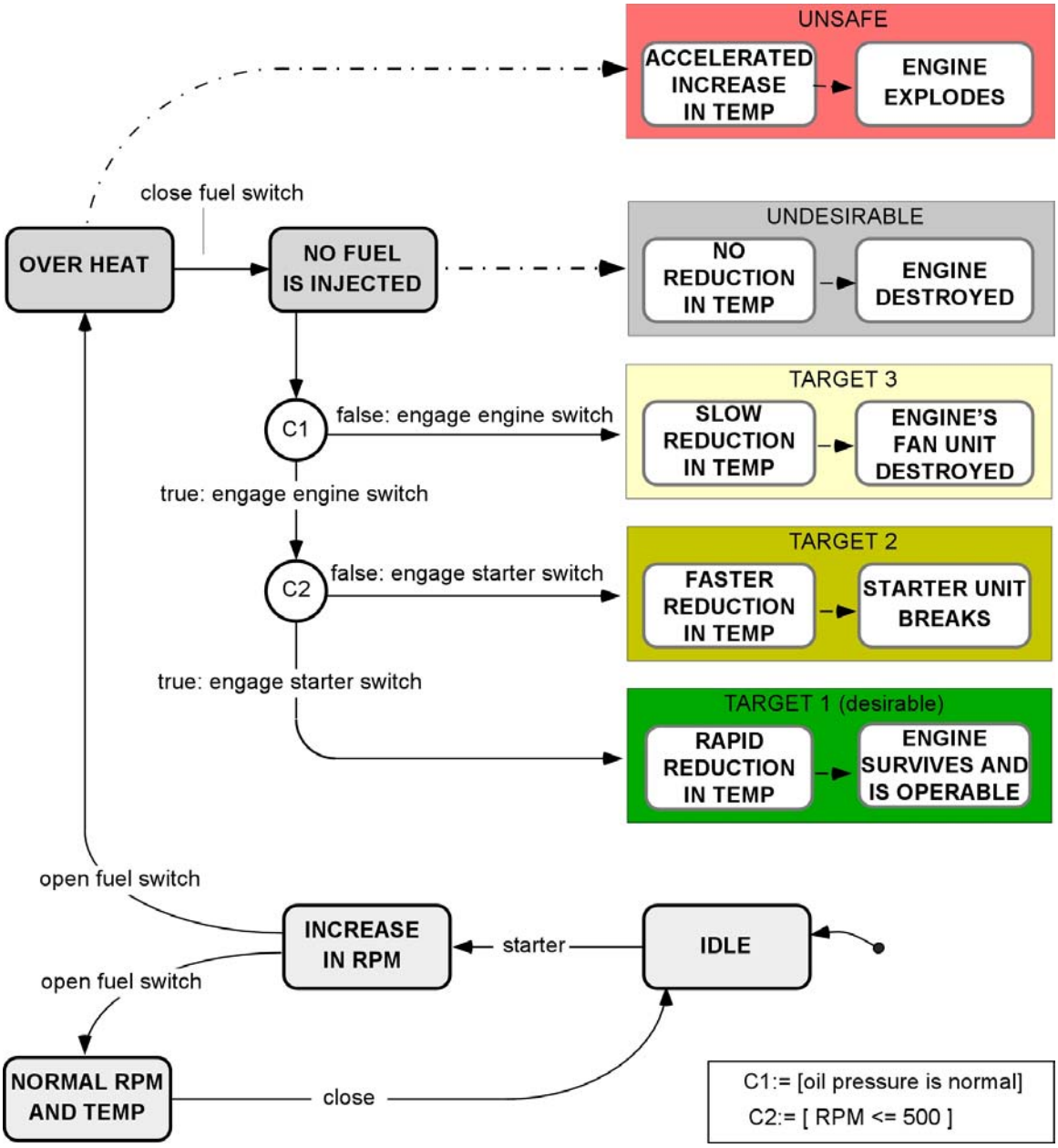

Figure 2. Machine model of a power unit with operational regions (color coded)

However, the particularities of the situation, namely the extent to which the overheat has affected the internal components of the engine, can impede our efforts: If any of the oil lines which supply lubrication to the fan unit have heated up and ruptured – there will be no oil pressure. In that case, rotating the fans, albeit reducing the temperature, will destroy the fan unit. However, if the oil pressure is O.K. (*true:* in Figure 2), we will engage the engine control switch to cool the engine down. If the oil pressure is not O.K. (*false:*), we are faced with a dilemma, to

rotate the fans or not to rotate them. We know that if we don't rotate the fans, the engine will eventually burn up and be destroyed -- definitely an undesirable situation. If we rotate the fans, we will transition to target set 3, a much better condition, but the fan unit will be destroyed in the process. Based on our engineering and cost analysis, we conclude that we are willing to sacrifice the fan unit (which can be replaced) for the sake of avoiding engine destruction.

Let's say the oil pressure is O.K. If that's the case, we want to proceed towards a yet better recovery. Once the engine temperature has dropped lower, 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 is at a considerably lower RPM value (so as not to grind the starter). If the RPM is within limits (500), and we engage the starter for more than 60 seconds, it is possible to achieve full recovery, save the engine, and eventually operate it again as is. That is our desirable target 1 goal.

However, if the RPM is not reduced, continual rotation of the engine cannot be achieved and permanent damage to the engine will occur. Here again, we are willing to engage the starter, even at relatively high RPM (false:), so as to continue cooling the engine -- recognizing the fact that after such abuse the starter will no longer be operable. We are willing to sacrifice the starter unit (which is rather cheap to replace) for the sake of cooling the engine temperature to normal, and saving the engine and fans. 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.

To conclude, we have shown how it becomes possible, by superimposing the operational regions on top of the machine model, to map out a contingency plan to deal with an emergency situation. Our ability to map out the necessary actions that need to be undertaken stems from our theoretical approach for procedure analysis (discussed earlier) and the use of the unsafe, undesirable, and target regions to define alternative courses of action and allow for degraded recovery.

Procedure Design Criteria

In this third element we will define criteria for proper procedure execution and define what constitutes inefficient, error prone, unsuccessful, and, at the bottom of the heap – incorrect and unsafe sequences. The cornerstone of our analytical approach for procedure execution is the observation that there are three major phases in dealing with an emergency situation:

Phase A is concerned with *imperative blocking* of possible transgressions into catastrophic states. Here our goal is to [1] immediately and [2] efficiently block dynamic transitions (e.g., temperature rise, fire) that can drive the system into an unsafe and catastrophic region (e.g., explosion). In this phase A – which is time critical – actions may be irreversible and drastic. In the engine example described in Section 2, the imperative blocking is the action of closing the fuel switch to the engine so as to immediately remove the "fuel from the fire" and avoid transition to catastrophe.

Phase B is about *preliminary stabilization* of the failed system. Here, while the system is indeed blocked from accelerating toward catastrophe, the system is not fully functional and may be unstable. To deal with this undesirable situation, we begin measured steps to stabilize the system. In the engine example described in Section 2, the *preliminary stabilization* is the action of engaging the engine switch so as to free the fans to rotate and thereby allow venting and cooling of the engine. In terms of our theoretical construct, we have blocked the path to catastrophe and taken steps to stabilize the system.

The last phase (C) is *optimized actions* toward recovery. Given the preliminary actions that we have taken in phase A and B, the system is no longer heading toward a catastrophe and is stabilized. Now we can begin to take measured steps towards recovery. And while we aim for a full recovery, we accept the fact that in the face of unpredictable environmental or dynamic-internal events, especially when only partial information about the system is available, full recovery cannot be guaranteed. In this situation we will accept a degraded recovery.

The second objective is it to develop criteria that highlight and define a class of problematic situations (e.g., incorrect sequences, timing problems, as well human-machine interaction problems) that will render a procedure inefficient, and prone to error (both in execution and design). This set of properties is critical for our work as it is the input for any formal verification of procedures. Here we will also focus on the problem of executing several procedures concurrently, defining, in the context of this formal approach for procedure analysis, problems such as race conditions, livelocks, and deadlocks and use them as properties for verification (see Degani, 2004, Ch. 13 for a formal approach for modeling and identifying timing and synchronization problems among three (concurrently running) procedures. Finally, we intend to extend the verification beyond technical problems into the users' cognitive and perceptual

limitations while they are executing a procedure under stress, as well as into some aspects of crew coordination (e.g., two astronauts coordinating the execution of one or more emergency procedures).

An Algorithm for Analysis and Synthesis of Procedures

Next, we shall outline the basic algorithmic steps that must be executed in a typical application of our proposed methodology for synthesis of an efficient, reliable, and correct procedure. Recall that in our earlier discussion the transitions in the system consist of two distinctly classed, *controlled transitions* that are triggered by the user and *dynamic transitions* that the user has no control over and are triggered by the system itself (timed or automatic transitions) or the environment (disturbances).

There are 5 main steps in the way we synthesize and compute the action sequences (referring to Figure 1):

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, one considers the machine model in which all the controllable transitions have been (temporarily) deleted and the only remaining transitions are the dynamic ones. One then reverses the dynamic transitions' directions (each source state of a transition is interchanged with its destination), and computes 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, … The algorithm is based on previous work by Brave and Heymann (1990). The essence of the algorithm consists of finding the maximal region in the state set which (1) has no dynamic transition sequences that form loops (that might be traveled indefinitely without ever reaching the target) 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 then 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 and, respectively, to states in the sets controllable to the various target sets. Next, we assign costs to the various transitions based on operational considerations (e.g., irreversibility of actions, availability of components such as fire suppression bottles, etc.). Finally, we assign a very high cost to states in the unsafe region, and respectively to states in the region of uncontrollable to unsafe.

Step 4. Probability assignment to dynamic transitions. Dynamic transitions emanating from a given state occur with certain probability that may depend on the given state, time of entry into the state, time of residence in the state, and various other case dependent considerations. These probabilities are expressed quantitatively and are assigned to all relevant dynamic 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, we first note that each recovery execution may include, along with its designated probabilities, dynamic transitions and, therefore, sequences that may terminate at more than one possible end state. We first 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 corresponding cost assignments).

All executions of minimal cost (in case there are more than one) are then chosen as candidates for selection as the optimal recovery execution. The final selection is performed by minimizing transition costs (e.g., weighted with respect to probabilities of occurrence of dynamic transitions).

Research Process

In the first phase of this project we will develop a theoretical approach for procedure analysis. We will apply this approach to a variety of procedures and systems under a range of

operational scenarios to test the theory and modify it accordingly. The main objective is to develop a theory that is general enough to deal with a broad range of exploration systems.

The second phase of this project will be to develop a modeling framework for analysis of procedures. This will allow us study the interaction between the system behavior, procedure constraints, and procedure design criteria. The objective here is to model the Shuttle and CEV system, superimpose the procedure constraints and design criteria, and perform analysis. During this phase, the approach and methodology will be extended to deal with several procedures running concurrently.

In the third phase of this project we will primarily focus on the development of algorithms for synthesis and automatic generation of procedures. The objective is to integrate the theory, methodology, and procedure design criteria into an algorithm that will enable automatic construction of emergency procedures and recovery sequences. The algorithm will be used for developing new and more efficient procedures for the CEV and other exploration systems. In this phase we will also investigate the application of this approach, methodology, and algorithm toward an onboard system that can automatically generate a procedure for a given situation. The idea is to develop a tool that can deal with unforeseen emergencies by automatically synthesizing and generating a unique procedure.

REFERENCES

Brave Y. and Heymann, M. (1990). On Stabilization of Discrete Event Processes *International Journal on Control, Vol. 51*(5)*, 1101-1117* .

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

Degani, A., Heymann, M., Meyer, G. and Shafto, M. (2000). *Some formal aspects of human-*automation *interaction*. NASA Technical Memorandum #209600. Moffett Field, CA: NASA Ames Research Center.

Degani, A., Heymann, M., and 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 Heymann, M. (2002). Formal verification of human-automation interaction. *Human Factors, 44*(1), 28-43.

Heymann, M., and Degani, A. (2002). *On abstractions and simplifications in the design of human-automation interfaces*. NASA Technical Memorandum #211397. Moffett Field, CA: NASA Ames Research Center.

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. Lin, F. and 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.

Transportation Safety Board of Canada (2003). *In-flight fire leading to collision with water. Swissair Transport Limited, McDonnell Douglas MD-11 HB-IWF Peggy's Cove, Nova Scotia. 2nd September, 1998*. Report Number: A98H0003.

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.