# Developing a Risk Management "Flight Simulator" for Manned Space Programs: A User Interface to a Systems Dynamic Simulation of System Safety at NASA

by

**Stephen R. Friedenthal**

S.B. Nuclear Engineering, Massachusetts Institute of Technology, 1992

Submitted to the System Design and Management Program in
Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology
February 2004



Signature of Author_____
Stephen Friedenthal
System Design and Management Program
February 2006

Certified By_____
Nancy Leveson
Thesis Supervisor
Professor of Aeronautics and Astronautics

Certified By_____
Patrick Hale
Director
System Design and Management Program

# Abstract

Simulators are a staple of any engineering project and manned space flight in particular. From pilot and crew training to maintenance and emergency repairs, very little is done without it first being thoroughly practiced and refined during advance simulation. Whether the simulation uses a computerized flight simulator that recreates the physics and experience of flight, or a simple mock-up with paper cutouts and hand tools, the end result is the same: people learn to make better and safer decisions through advanced simulation and practice. However, there are no simulation tools in use to help NASA managers to understand the dynamics of systemic risk, or how to evaluate the inherent risk within an organization.

This thesis describes the development of a risk management simulator that will enable NASA managers to explore the dynamics of risk using an advanced system dynamics simulation of the NASA safety culture prior to the Columbia Shuttle accident. This simulation model was developed by MIT Professor Nancy Leveson and her students as part of a NASA USRA research grant and has over 700 variables and several sub models. While the model is eminently useful to those familiar with system dynamics, the Vensim software application and the specific model structure, it is not very useful as a learning tool for those who are not.

The simulator tool developed for this thesis simplifies and consolidates the overall model behavior into 8 decision variables and 35 display variables. Moreover, 18 of those display variables are grouped into one of 5 categories of "leading indicators" of risk. This simulator enables any user to quickly begin to explore the system model and to discover the consequences of different decisions on system risk, without any need for the user to know system dynamics theory or any details of the model design. In a video game the user doesn't know how it is programmed, but is still able to learn the rules of the game, how it behaves and—most importantly—how to win. Similarly, the goal of the risk management flight simulator is to help NASA managers to learn the rules of system risk, how system risk behaves in response to management decisions, and, most importantly, how to make the best informed risk decisions.

# Acknowledgements

A little under three years ago I showed my wife, Michele, a brochure I had just received describing MIT's SDM program. Michele's response was immediate, enthusiastic and supportive, "You have to do this. It's perfect for you!" As I type these last sentences for my thesis and reflect upon the past two years—January "boot camp," two summer sessions, many, many long nights and weekends and two children—I cannot begin to express my gratitude as you have been with me all the way. Michele, thank you. I love you. I couldn't have done this without you.

My thanks also to my son, Eliot, who at the age of two exclaimed during my first week of classes, "Daddy's going to ABC!" and has maintained that wonderful joy throughout the last two years of his life.

I especially want to thank my advisor, Nancy Leveson, whose kindness, wisdom, patience, great humor and support has helped to make this the most wonderful academic experience of my life. I feel truly blessed to have had the privilege to work with you.

Finally, I would like to acknowledge a few of the people in SDM without whom my life would be poorer without: Fellow SDM students Ram Krishnaswami, Fernado Cela and Mal Atherton, and on the SDM staff, SDM administrative coordinator, Ted Hoppe for his support, humor and introducing Eliot to bowling, and SDM Directors' Pat Hale and Denny Mahoney.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Problem outline

This thesis summarizes the development of a risk management tool that was designed to better enable NASA managers and decision makers to understand the complex system dynamics that underscore system risk. Following the Columbia shuttle accident in 2003, Leveson et. al. developed a sophisticated system model of the engineering and safety culture at NASA to better understand and characterize the cultural and socio-political aspects of risk. The model was developed using a commercial off-the-shelf ("COTS") product, *Vensim,* which provides an easy method to describe a system and the interactions between the system components. Vensim, and other tools like it, were born from the ground-breaking work of Jim Forrester and John Sterman and have been used widely to describe systems varying from "predator-prey" population models (i.e., the inter-relationship between rabbit and fox populations in a closed environment) to supplier-consumer distribution models (i.e., the infamous "beer game") to product marketing "flight simulators" (i.e., the complex dynamics of pricing a product and resource allocation) and many others. Leveson, however, took system dynamics into a relatively new area: modeling system safety culture. Specifically, they sought to develop a tool that could be used to better understand the dynamics behind the breakdown in safety *culture* that led to the Columbia disaster. The model was based upon previous work using STAMP (a "Systems Theoretic Accident Modeling and Processes") that "...integrates all aspects of risk including organizational and social aspects."[1] In its simplest form, the model is relatively easy to describe. For

example, with each successful launch, NASA managers[1] may come to believe that safety concerns and objections are increasingly overrated based on the erroneous belief that success = "proof" of system safety. (i.e., insulating foam has come off the shuttle external fuel tank on all previous launches and there wasn't a problem, hence falling foam is "safe".) This increasing (false) confidence in the inherent safety of the system, along with budgetary concerns and external pressure, then leads to an increasing performance pressure to "do more with less" and a concomitant pressure to have more (successful) launches. This relationship can be described schematically below.

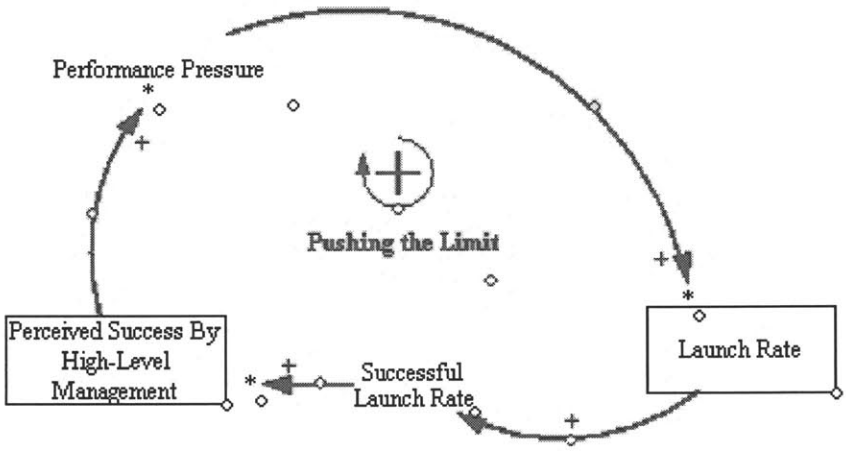



**Figure 1-1 Schematic representation of the performance feedback loop. With each succesful launch, pressure from management leads to further increases in the launch rate**

[1] The term "Manager" leaves a lot of room for ambiguity. In our context when we say manager we are referring to a person that has decision-making authority over the allocation of personnel and resources as well as those persons who have primary authority and approval for when NASA can proceed with a shuttle launch. While it may be true that "everyone is responsible for safety", and anyone with a significant safety concern can impact a shuttle launch decision, in practice, when "everyone" is responsible, no one is. Hence, the decision to base the model on the assumption that there are those who by nature of their unique position have the ultimate decision-making authority in many ways represents the reality of system safety vs. organizational goals of system safety.

Of course, we recognize that such a positive feedback loop cannot go on forever and there are numerous feedbacks and other systems that support, inhibit and feed off of this loop. This was developed and shown as Figure 3 in [7] (below).

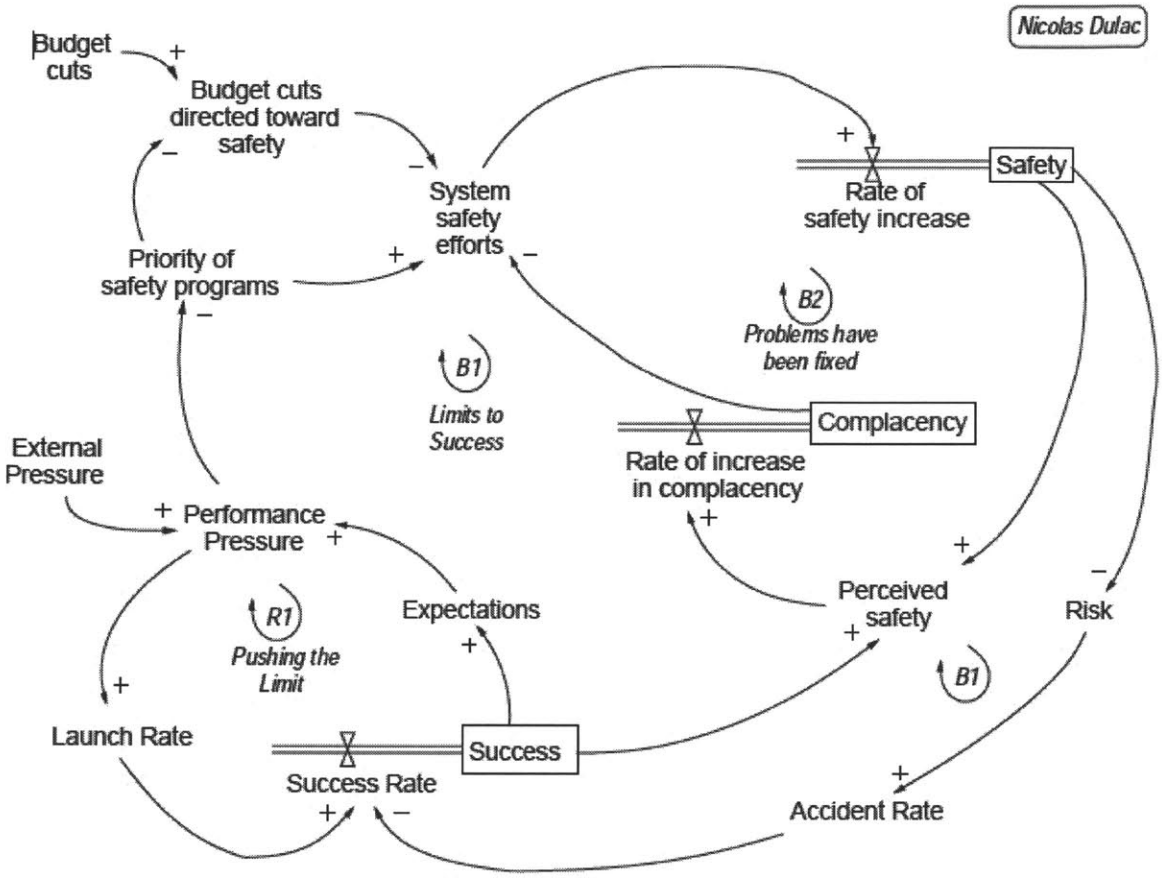



**Figure 1-2 Simplified model of the Dynamics Behind the Shuttle Columbia Loss by Leveson & Dulac**

We see that there are several inter-connecting loops and it is a little more complicated, but intrinsically, it looks pretty straightforward. Were that this was all there was to the model, we would be done. But, of course, real systems are significantly more complex than this and to develop this "simple" model there are, in fact, more than 700 variables in the Vensim model that underlie this diagram and the overall model becomes so complex that all but those who are intimately familiar with it are unable to understand how it works. In other words, we have a

model that is so complex that it is usable only by a few academics. And, while this might provide job security for those so inclined, our goal is disseminate and share this model with NASA managers so that they can use it to better understand system safety and, hopefully, gain a greater appreciation of the complex dynamics of system safety at NASA. There are two fundamental ways to approach this problem. We can either, (1) teach all NASA managers a graduate course in system dynamics, the Vensim application and the specific model design, or (2) develop a simplified "front end" to the model so that someone with very little training can immediately begin to use the model as a learning tool without becoming bogged down in the minutiae of the model design and system dynamics theory.

This, then, is the problem statement and the goal of this thesis: To develop a simplified interface to the system dynamics model so that NASA managers can use the model as a risk management tool.

## 1.2 Thesis goals

In addition to developing a simpler user interface to the system dynamics model, the goal of this thesis is that it can be used, as a learning tool, to help decision makers better understand systemic risk. This understanding is particularly important as it becomes increasingly apparent that the traditional methods of risk management such as probabilistic risk assessment (PRA), fault tree analysis and so forth, do not adequately account for programmatic and human-system-interface risk (See, for example, a discussion of *E. Coli* poisoning of the public water supply in Walkerton, ON that led to several deaths in [5], or even a discussion of the "Therac-25" incident in [6]. In both cases the standard safety methods and tools failed to protect lives due to a lack of awareness of the complexity of system safety.)

Put more simply, in a NASA frame of reference, the Columbia Shuttle accident was not a case of what are referred to as, "unknown unknowns", causing the loss of shuttle and crew. On the contrary, NASA management was fully aware that foam had been a recurring problem, that Atlantis only 3 months prior had foam impact the shuttle a mere 6 inches from a critical electrical junction box and—most importantly—within hours of the launch of the Columbia a post-launch film review indicated that the largest piece of insulating foam to have ever fallen off in the history of the shuttle program had impacted the critical leading edge of the wing at a speed of over 600 mph and appeared to have possibly caused some damage to the shuttle. But, due to an overconfidence in the *perceived safety* that insulating foam had never previously caused an accident, NASA management did not take any steps to further investigate the damage (e.g., having the military take high resolution photos of the damaged area) or plan for the possibility of a damaged thermal tiles on reentry. (Recall the loop in Figure 1-1). We can be sure that this type of accident is less likely to happen again, and certainly without a thorough analysis. But, the weakness of existing safety analyses is that by definition they rely upon the unstated assumption that human-system interfaces are "working" and that a risk is adequately described and appropriately evaluated. PRA and safety analyses are not enough. NASA decision makers need to understand the dynamics that underlie system safety so they can evaluate the "health" of their safety system and its ability to appropriately evaluate risk.

To this end, the work of this thesis is to accomplish several goals. These are:

1. To develop a working version of the model shown in [5], that encapsulates the model complexity so that users can more readily understand the broad macroscopic effects of decisions on system safety.

2. To design and program a user interface to the system dynamics base model that will enabled any user with only a short introduction into its use to be able to explore the model dynamics.

3. To develop the interface program so that it can be used (or further developed) into a risk management tool so that NASA managers can modify a subset of the model inputs (such as launch rate or the percentage of contracting to 3rd parties) and see how these changes affect risk over time.

4. To develop the interface program so that users can track the "leading [risk] indicators" from [2] as the model inputs are modified (over time) and to display them in a likelihood-severity 3 x 4 risk matrix as is commonly used by NASA management to assess risk.

5. To develop the user interface so that it can be upgraded and expanded to incorporate different case scenarios such as external budget and performance requirements.

# 2 Risk Management Tools and Management Simulators

## 2.1 History of simulators

It is no coincidence that interactive simulations are often called, "flight simulators" regardless of their aeronautical or terrestrial goals. Some of the earliest training and simulation tools were first developed to help train pilots. As early as the Wright brothers, it was recognized that system complexity required a different level of training. In WW-I, Edwin Link developed the first mechanical flight simulator which later proved to be used extensively by the US military in WW-II to train pilots how to fly without jeopardizing equipment or personnel.

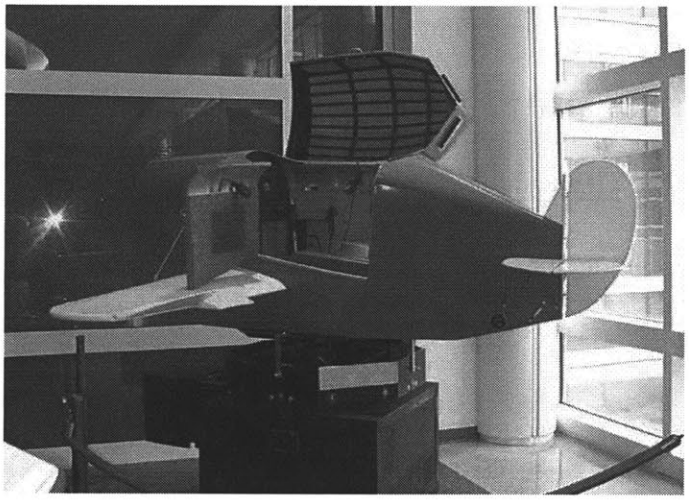


**Figure 2-1 WW-II Link Training Simulator**

These training simulators were very simple and did not mimic the actual aerodynamics and feedback of a real airplane, but they did enable a student pilot to become familiar with the basic control and operation of an airplane. It was not until the late 1940's, with project "Whirlwind" developed at MIT, that the first (analog) computer simulation was used to provide immediate feedback to the user based on mathematical modeling of aerodynamic response. Not only was this to become a seminal event in digital computing and computer simulation, but the lead

computer designer of the project and inventor of "core memory" was Dr. Jay Forrester who would later develop the theory of system dynamics upon which this thesis is based.

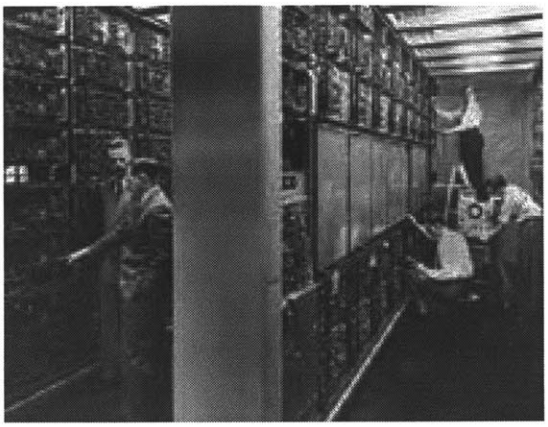


**Figure 2-2 Photograph of the Whirlwind computer. Jay Forrester is on the far left next to Dr. Norman Taylor who is pointing. (1952)** [Picture used with the permission of The MITRE Corporation. Copyright © The Mitre Corporation. All rights reserved.]

Eventually flight simulators became more and more complex to the point where we are today: flight simulators so closely mimic the actual flying experience that they have become a necessary staple of training pilots in flight safety, procedures and aerodynamics. In fact, in most any complex system that involves severe consequences when there is mishap—and where it is impractical or unsafe to train on the real system—we are likely to find a computer simulator to help train operators how to respond in the event of an accident. The nuclear power industry, for example, has full mock-ups of the reactor control room with a sophisticated computer model behind it for operators to practice emergency procedures without the risk of an accident or a loss of plant availability (which is costly).

With the advent of the computer, another type of simulator arose in the 1980's and 1990's—business simulations. These simulations require the development of a complex mathematical model to simulate some aspect of a business cycle or process such as new product development or resource allocation. These tools enable business managers to learn about the dynamics of the

particular challenge and to practice and hone their skills and understanding in a safe environment.

## 2.2 Learning with simulators

There are, of course, many different ways that people learn. We tend to learn incrementally, building upon previous knowledge on step at a time. Figure 4.1 of *Learning Through Simulations* (Fripp [18] ) shows the Kolb learning model reproduced below.

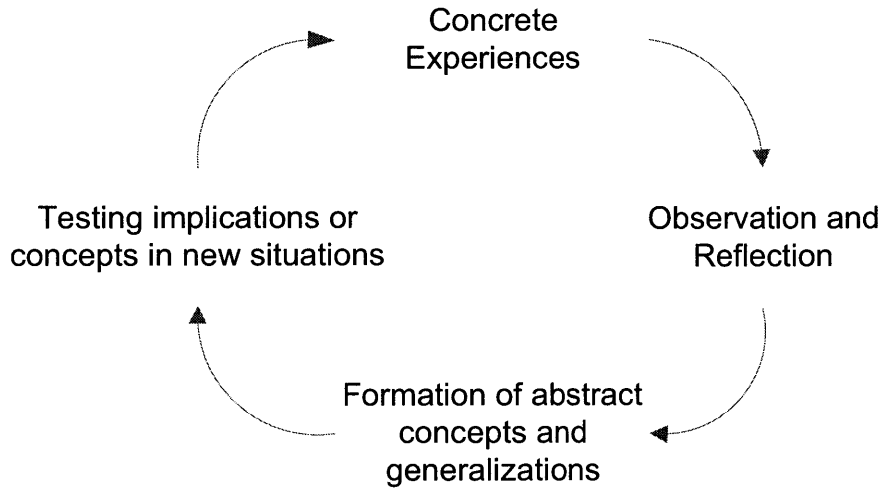



**Figure 2-3 The *experiential learning model* [Kolb 19, reproduced with permission in Fripp p.41]**

The key point of this diagram is that learning is a process that is based upon a combination of concrete and abstract experiences that are then tested and observed/reflected upon. Different people learn in different ways and Fripp spends some time talking about the importance of different learning styles, i.e., What works for some does not work well for others. For our purposes, however, we are less interested in an individual's particular learning style than the observation that, if the Kolb model is correct, regardless of style if one is to truly *learn* a new concept then it must involve the whole cycle. If any one leg is missing then regardless of how well the other three are mastered the ability to apply the new skill is limited. For example, one can read about tennis (abstract concepts), watch others play tennis (observation & reflection),

and even bounce a tennis ball on a racquet or over a net a few times (concrete experiences). If you have never actually *played* tennis (testing concepts in new situations), however, it is highly unlikely that you will be ready for tournament play. Furthermore, it is not enough for the learning structure to be in place for effective learning to take place. The participant has to want to learn so that he can be an active participant. As pointed out by Fripp (p 41), "For people to learn most effectively, the subject matter must therefore either have intrinsic interest to them, or they will need to have their interest stimulated in some way." (Many of us may remember a great professor who made a boring class fun and interesting by stimulating our interest).

So, how do we learn? The options are varied: lectures, classes, on-the-job training, mentoring and coaching, on-line information, case studies and, of course, simulations. All of us have used one or more of these at different times of our lives, but simulations offer several distinct advantages not available with any of the other methods. These include:

- They provide immersion—the participant can immerse oneself within the simulation and act as if it is not a simulation.
- They are a safe environment to make mistakes with no risk of financial loss or equipment damage
- They can be team-based, or individual
- They can simulate extreme situations that would not ordinarily be available
- They work with all four stages of the Kolb model: encouraging the participant to test new concepts, gain concrete knowledge and facts and the opportunity for reflection during and after the simulation

- A well designed simulation can convey system interactions and dependencies
- They can be FUN, which make the learning all the more enjoyable and memorable.

What are the disadvantages of simulations?

Fripp discusses several: They can be complex and time-consuming to design and build: If the simulation is too easy then participants try to "beat the game" and if they are too complicated then participants may give up and lose interest.

But, overall, a well-designed simulation can be an ideal learning environment—especially for endeavors such as space flight, which have such extreme risks and costs associated with failure.

What are some of the features of a simulation that might encourage effective learning?

Fripp discusses three kinds of managerial learning, "cerebral learning," "skills learning" and "transformational learning," and—as with the Kolb model—effective learning involves all three. Aldrich in, *Simulations and the future of learning : an innovative (and perhaps revolutionary) approach to e-learning* (Aldrich [8]) emphasizes the importance of simulations that have the following features (Aldrich, Pg. 9):

- Authentic and relevant scenarios
- Applied pressure situations that tap user's emotions and force them to act
- A sense of unrestricted options and,
- Replayability

In terms of how one learns through a simulation "game" Aldrich discusses three different "contents", what he calls the three "primary colors" of learning in game simulations: Cyclical, Open-Ended and Linear.

### 2.2.1 Cyclical learning

Cyclical learning is familiar to all of us as learning by rote. With each iteration we gain a better understanding of the system and improve our skills. Simulators based upon cyclical learning teach through repetition and practice. Arguably, all learning involves some fraction of repetition as we imprint a neural behavior. Certainly in aeronautical flight simulators we see cyclical learning in areas such as emergency response procedures so that "common" mishaps become familiar to the pilot. However, cyclical learning as a primary simulation method is more prevalent in other industries such as medicine, where the student needs to practice a procedure "until he gets it right." Examples of simulators run from the very simple—such as the "resusci-annie" doll familiar to Red Cross CPR students—to the increasingly more complex and realistic simulation tools for medical students to practice medical procedures.[2]

If we extend the definition of "simulator" to include system mock-ups and dry runs, we can see that cyclical learning is used in many more industries where it is desirable to have personnel practice a complicated procedure in some form of a simulated environment. Nuclear technicians, for example, will practice a nuclear refueling procedure until management is confident that there will be safety through familiarity. Other examples include aircraft and spacecraft maintenance, surgical operating procedures, military "war games", and even school fire drills. How do we differentiate between cyclical learning simulations vs. simply practicing a skill (such as playing tennis or golf) through repetition? I would suggest that the principle difference is that a

[2] See for example the products available from the company "Limbs and Things Ltd." www.limbsandthings.com

simulation requires some degree of imagination on the part of the participant. If she is acting *as if* there is some alternate reality (a fire, emergency, real radioactive fuel present, etc.), then it is a simulation. If there is no suspension of disbelief, then it is not.

### 2.2.2 Linear learning

As the name implies, linear learning is a time-based, one step follows the previous form of learning. Most classes are taught this way, as well as story narratives and it is the most common method for learning a new skill. Linear learning is familiar to us because we experience time as linear. Most simulators, however, are not based upon linear learning. A pilot in a flight simulator can re-play the same event over and over until he is comfortable with his performance regardless of the flow of time. Interestingly, however, because system dynamics uses time as its integrating variable, linear learning does come into play with respect to system dynamics based simulators such as this thesis. As described later in this thesis, the risk management tool has a user step forward through time (the default is 6 months per iteration) so that she can understand and evaluate the time-based nature of safety and risk accumulation. If a user wants to re-examine or improve her performance, then she must go back to the beginning and step forward through time again making the appropriate choices along the way. So, while linear learning does not see much use in class simulators, it is used to a degree in this context.

### 2.2.3 Open-ended learning

Open-ended learning is perhaps the "holy grail" of all good simulations. It involves an understanding of trade-offs and recognition that a system is dynamic with various feedbacks. It is also very much the same concept as Fripp's "transformational" learning. Humans are uniquely gifted with the ability to adapt to a new situation and apply their knowledge of *the system* to take the best course of action when presented with an unknown. Modern aircraft flight simulators are

designed to combine open-ended learning—presenting the student with an unanticipated event —with cyclical learning until he or she "gets it right." Open-ended learning is about teaching the participant about the system dynamics so that he can make an informed decision based upon that knowledge. This is the ultimate goal of this thesis as well—to teach NASA managers about the dynamics of system safety; not in terms of component and mechanical systems (for which NASA has numerous tools to evaluate the risk, cost and severity of a failure) but in terms of the overall socio-technical system itself.

## 2.3 Risk management tools in use by NASA

NASA uses several risk management tools depending on the type and nature of the project. If managing cost and schedule are of primary import, a project is likely to use continuous risk management (CRM) along with a database tool to simplify the integration and reporting of system risks. Supporting CRM, or perhaps as a separate effort, there will be probabilistic risk assessments (PRA), often combined with a fault tree analysis, that seek to quantify and identify the most significant risks of mechanical or system failure. PRA is also used (and misused) to normalize risk so that different risks can be compared in terms of failures per number of events and even to make risk "trade offs"—literally moving risk from one system to another as if it were a physical quantity that could be traded between systems. (See, for example, "Risk as a Resource" [11] which grew from NASA's "faster, better, cheaper" initiative)

### 2.3.1 Continuous risk management (CRM)

An outcome of NASA's faster, better, cheaper (FBC) initiative is extensive use of continuous risk management. CRM aims to track and control budget, schedule and program risk through six steps: identify, analyze, plan, track & control.

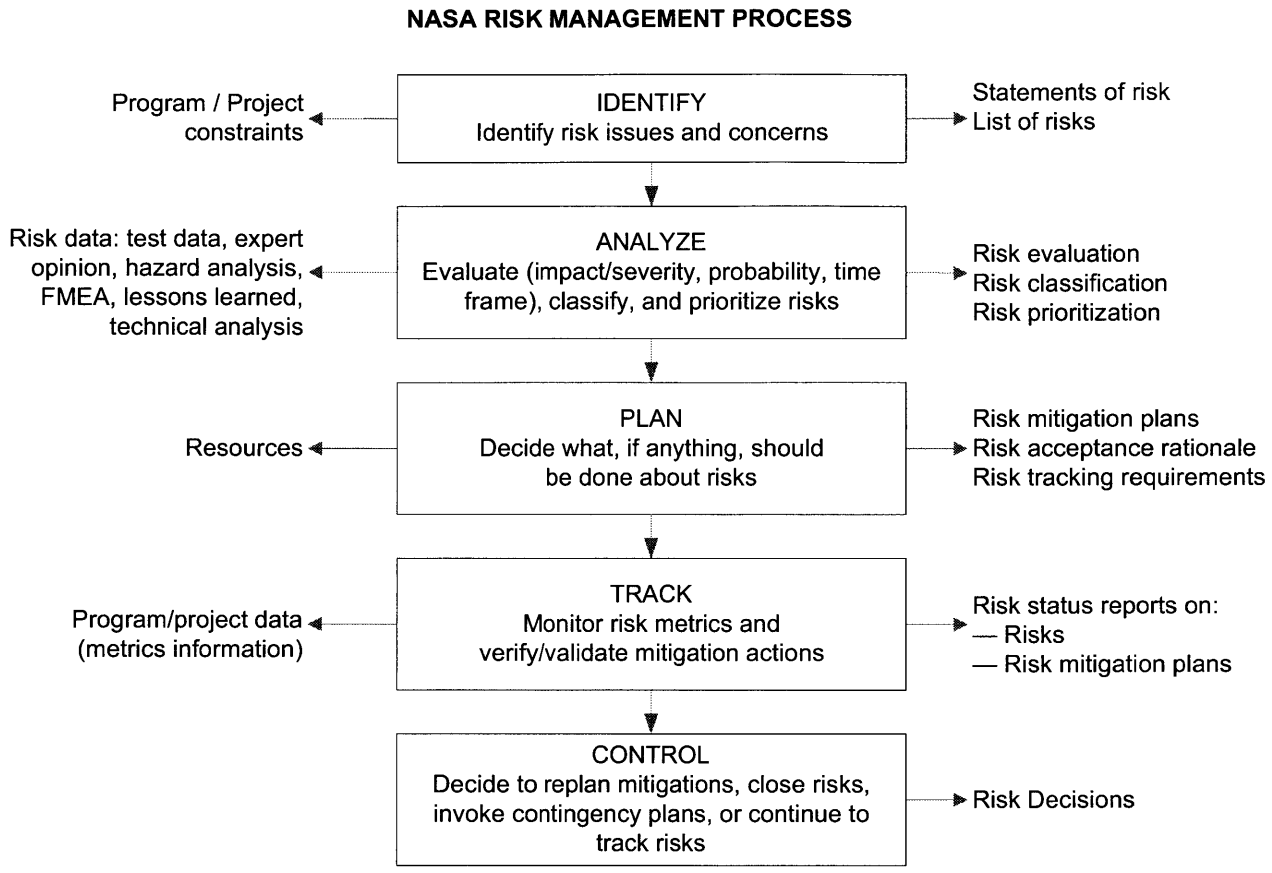



**Figure 2-4 NASA risk management process [11]**

This is often shown as a cyclic process indicating that the process is "continuous".

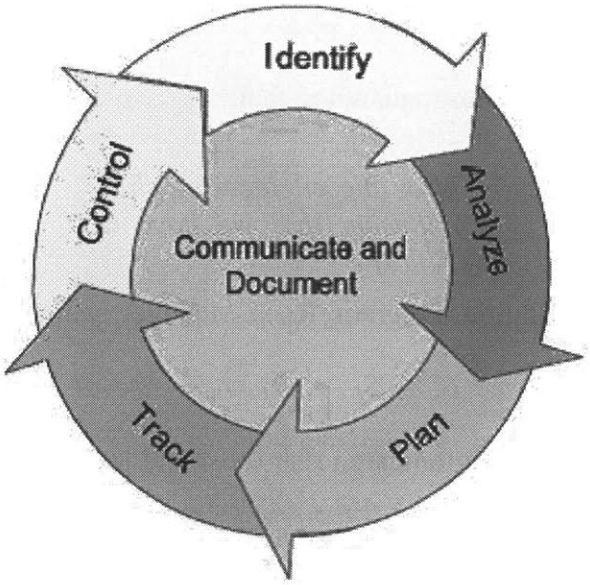



**Figure 2-5 NASA CRM cycle diagram**

The primary output of CRM is the generation of a 5x5 likelihood (probability) vs. severity (impact) risk matrix that attempts to prioritize and focus attention on the most important risks.

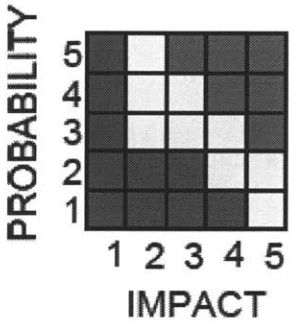



**Figure 2-6 NASA likelihood vs. severity risk matrix**

From a purely safety and risk perspective, the CRM process does not prioritize "risk" over cost and schedule. A high likelihood and severity impact can easily be schedule risk or excessive cost overruns. From a project risk management perspective, this is perfectly fine. But, from a system technical risk perspective, it appears to be problematic. While, "of course" technical risk is treated with respect, there is nothing in the CRM process that explicitly treats it any different.

There are some signs, however, that this may be changing. At the NASA Risk Management Conference VI, Wayne Hale, Manager of the Space Shuttle Program, noted that:

> *"The RM [Risk Management] system is only as good as the engineering that informs it. The RM system can also lead to the illusion that we have a comprehensive portrait of program risk when we don't." [13]*

At the same conference, John Turner, the Space Shuttle Program Risk Manager warned participants [to] "*Avoid Calculus with Crayons Syndrome (CWCS)—risk scores are at best fuzzy*"[14]. So, while CRM is still primarily focused on programmatic risk (cost and schedule) vs. safety, there appears to be a greater appreciation for the limitations of it as well.

### 2.3.2 NASA Risk management applications

NASA has several risk management database and integration tools to assist in the management of risk and the coordination of resources. These include, "Integrated Risk Management" (IRMA) [16], "Risk Management Implementation Tool" (RMIT) [17], "Active Risk Manger" (ARM) [15] as well as the "Glenn Risk Database", developed with Microsoft Access®. A common feature of all of the tools is that they assist the user in following the risk management process whenever a new risk is entered or updated. Typically, users can enter detailed information, and generate reports and a 5x5 risk matrix to highlight the most important risks. The most important take-away from all of these programs is that, as pointed out above, they are project risk management tools, and not safety focused.

## 2.4 The case for a risk management simulator

The case for a risk management simulator is actually quite simple. Can anyone imagine an astronaut piloting the shuttle without extensive practice in a flight simulator? The same goes for emergency repairs and maintenance: workers will practice ("simulate") making the repair in a safe environment to minimize the chance of a mistake on the real system. The answer to both of

these hypotheticals is clearly, "no." In fact, we know that practically every facet of space flight is practiced and simulated until the chance of operator error is remote. Given the great prevalence of simulation in manned space flight it seems almost antithetical that there are no tools available to help managers to practice (safety) risk management decision-making. But, that is exactly the case because, quite simply, they do not appear to exist. More importantly, there are no tools currently available to help managers to understand the risk consequences of project management tradeoffs such as:

- If the budget requires either reducing the launch rate or contracting more work, which decision is safer?
- If you are going to increase the amount of work contracted out, what measures should you take to minimize the safety impact?
- Launching too many missions clearly has a safety impact, but is there a minimum launch goal as well, below which risk increases?
- To what degree does schedule pressure—trying to do more with less—impact safety? Is a little pressure good, or is it always bad?
- What are the leading indicators of a risk problem?
- Is it good to have a string of successful launches? What are the risks?

The goal of this thesis is to provide NASA with a tool that will enable just these types of analyses and eventually, with further work, an on-line real-time risk management simulator as well.

# 3 System Dynamics Model of NASA Safety Culture

This simulation uses the system dynamics model developed by Leveson et.al. for the Phase I final report for a USRA grant in February 2005 [7] with some minor modifications described below.

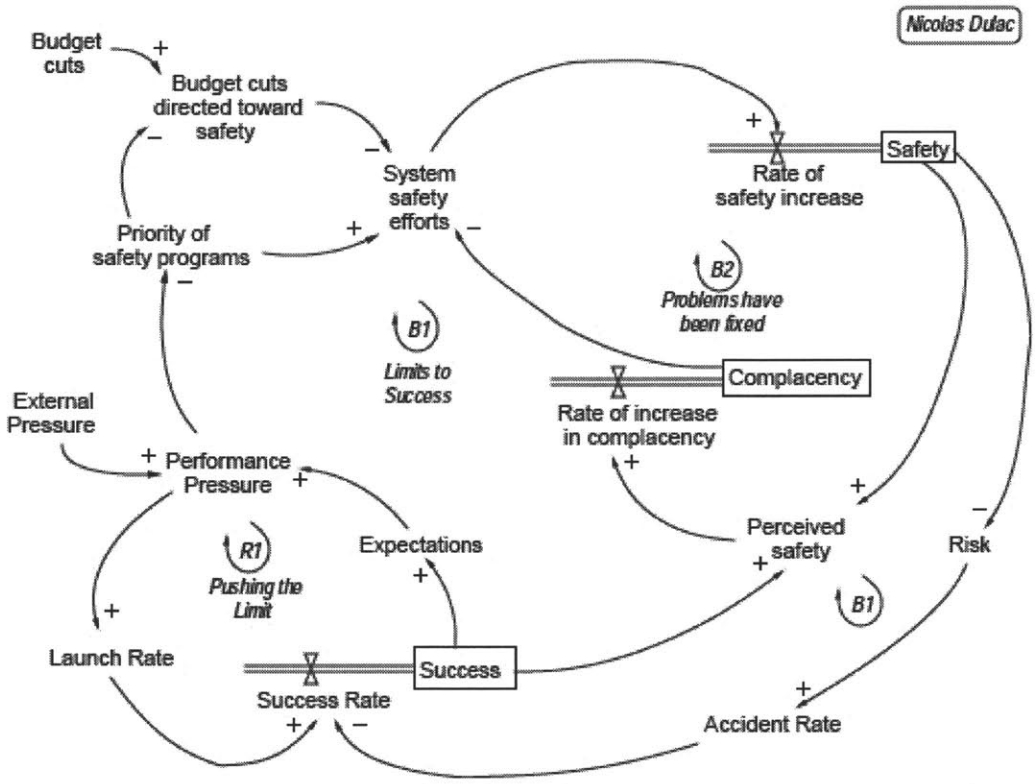



**Figure 3-1 Simplified model of the dynamics behind the Shuttle Columbia loss [7]**

## 3.1 Model description

With the exception of the modifications described in Section 3.2, the model is essentially unchanged from the original USRA, pre-Columbia model. The description of the model is reproduced from that report. The model has several hundred variables grouped into one of 9 separate sub models. These models are:

1. Launch Rate
2. System Safety Resource Allocation
3. System Safety Status
4. Incident Learning and Corrective Action
5. Technical Risk
6. System Safety Efforts and Efficacy
7. Shuttle Aging and Maintenance
8. System Safety Knowledge Skills and Staffing
9. Perceived Success by High-Level Management

It's important to note that the 9 sub models were developed to simplify creating the simulation and do not necessarily correspond to major system dynamic driving forces.

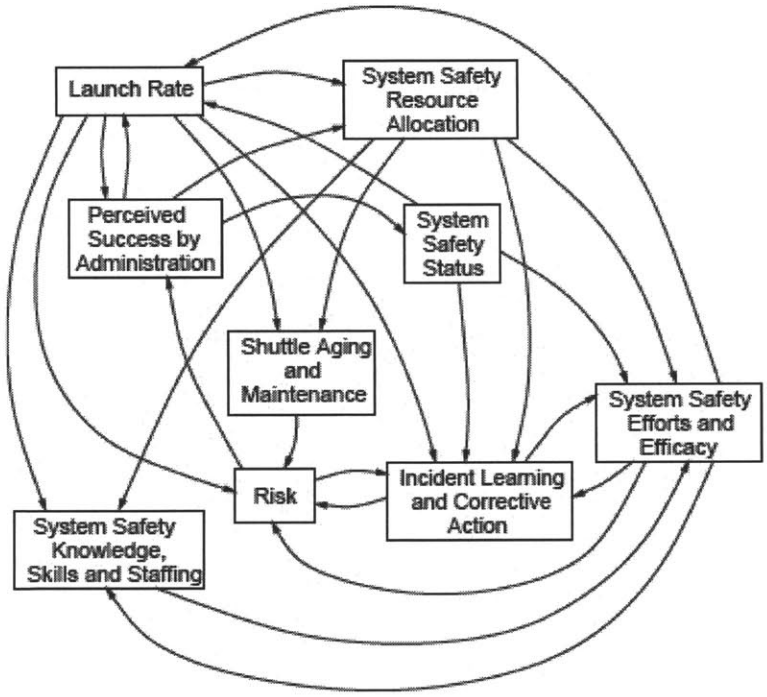



**Figure 3-2 The 9 sub models in the system dynamics model**

The description of each of the 9 sub models from [7] is reproduced below.

#### 3.1.1.1 Technical Risk

The purpose of the technical risk model is to determine the level of occurrence of anomalies and hazardous events, as well as the interval between accidents. The assumption behind the risk formulation is that once the system has reached a state of high risk, it is highly vulnerable to small deviations that can cascade into major accidents. The primary factors affecting the technical risk of the system are the effective age of the Shuttle, the quantity and quality of inspections aimed at uncovering and correcting safety problems, and the proactive hazard analysis and mitigation efforts used to continuously improve the safety of the system. Another

factor affecting risk is the response of the program to anomalies and hazardous events (and, of course, mishaps or accidents).

The response to anomalies, hazardous events, and mishaps can either address the symptoms of the underlying problem or the root causes of the problems. Corrective actions that address the symptoms of a problem have insignificant effect on the technical risk and merely allow the system to continue operating while the underlying problems remain unresolved. On the other hand, corrective actions that address the root cause of a problem have a significant and lasting positive effect on reducing the system technical risk.

#### 3.1.1.2 System safety resource allocation

The purpose of the resource allocation model is to determine the level of resources allocated to system safety. To do this, we model the factors determining the portion of NASA's budget devoted to system safety. The critical factors here are the priority of the safety programs relative to other competing priorities such as launch performance and NASA safety history. The model assumes that if performance expectations are high or schedule pressure is tight, safety funding will decrease, particularly if NASA has had past safe operations.

#### 3.1.1.3 System safety status

The safety organization's status plays an important role throughout the model, particularly in determining effectiveness in attracting high-quality employees and determining the likelihood of other employees becoming involved in the system safety process. Additionally, the status of the safety organization plays an important role in determining their level of influence, which in turn, contributes to the overall effectiveness of the safety activities. Management prioritization of system safety efforts plays an important role in this sub model, which in turn influences such safety culture factors as the power and authority of the safety organization, resource allocation, and rewards and recognition for raising safety concerns and placing emphasis on safety. In the model, the status of the safety organization has an impact on the ability to attract highly capable personnel; on the level of morale, motivation, and influence; and on the amount and effectiveness of cross-boundary communication.

#### 3.1.1.4 Safety knowledge, skills and staffing

The purpose of this sub model is to determine both the overall level of knowledge and skill in the system safety organization and to determine if the number of NASA system safety engineers is sufficient to oversee the contractors. The System Safety Effort and Efficacy sub model use these two values.

In order to determine these key values, the model tracks four quantities: the number of NASA employees working in system safety, the number of contractor system safety employees, the aggregate experience of the NASA employees, and the aggregate experience of the system safety contractors such as those working for United Space Alliance (USA) and other major Shuttle contractors.

The staffing numbers rise and fall based on the hiring, firing, attrition, and transfer rates of the employees and contractors. These rates are determined by several factors, including the amount of safety funding allocated, the portion of work to be contracted out, the age of NASA employees, and the stability of funding.

The amount of experience of the NASA and contractor system safety engineers relates to the new staff hiring rate and the quality of that staff. An organization that highly values safety will be able to attract better employees who bring more experience and can learn faster than lower quality staff. The rate at which the staff gains experience is also determined by training, performance feedback, and the workload they face.

#### 3.1.1.5 Shuttle aging and maintenance

The age of the Shuttle and the amount of maintenance, refurbishments, and safety upgrades affects the technical risk of the system and the number of anomalies and hazardous events. The effective Shuttle age is mainly influenced by the launch rate.

A higher launch rate will accelerate the aging of the Shuttle unless extensive maintenance and refurbishment are performed. The amount of maintenance depends on the resources available for maintenance at any given time. As the system ages, more maintenance may be required; if the resources devoted to maintenance are not adjusted accordingly, accelerated aging will occur.

The original design of the system also affects the maintenance requirements. Many compromises were made during the initial phase of the Shuttle design, trading off lower development costs for higher operations costs. Our model includes the original level of design for maintainability, which allows the investigation of scenarios during the analysis where system maintainability would have been a high priority from the beginning.

While launch rate and maintenance affect the rate of Shuttle aging, refurbishment and upgrades decrease the effective aging by providing complete replacements and upgrade of Shuttle systems such as avionics, fuel systems, and structural components. The amount of upgrades and refurbishment depends on the resources available, as well as on the perception of the remaining life of the system.

Upgrades and refurbishment will most likely be delayed or canceled when there is high uncertainty about the remaining operating life. Uncertainty will be higher as the system approaches or exceeds its original design lifetime, especially if there is no clear vision and plan about the future of the manned space program.

#### 3.1.1.6 Launch rate

The Launch Rate sub model is at the core of the integrated model. Launch rate affects many parts of the model, such as the perception of the level of success achieved by the Shuttle program. A high launch rate without accidents contributes to the perception that the program is safe, eventually eroding the priority of system safety efforts. A high launch rate also accelerates system aging and creates schedule pressure, which hinders the ability of engineers to perform thorough problem investigation and to implement effective corrective actions that address the root cause of the problems rather than just the symptoms. The launch rate in the model is largely determined by three factors:

1. Expectations from high-level management: Launch expectations will most likely be high if the program has been successful in the recent past. The expectations are reinforced through a "Pushing the Limits" phenomenon where administrators expect ever more from a successful program, without necessarily providing the resources required to increase launch rate;

2. Schedule pressure from the backlog of flights scheduled: This backlog is affected by the launch commitments, which depend on factors such as ISS commitments, Hubble servicing requirements, and other scientific mission constraints;

3. Launch delays that may be caused by unanticipated safety problems: The number of launch delays depends on the technical risk, on the ability of system safety to uncover problems requiring launch delays, and on the power and authority of system safety personnel to delay launches.

#### 3.1.1.7 System safety efforts and efficacy

This sub model captures the effectiveness of system safety at identifying, tracking, and mitigating Shuttle system hazards. The success of these activities will affect the number of hazardous events and problems identified, as well as the quality and thoroughness of the resulting investigations and corrective actions. In the model, a combination of reactive problem investigation and proactive hazard mitigation efforts leads to effective safety-related decision making that reduces the technical risk associated with the operation of the Shuttle.

While effective system safety activities will improve safety over the long run, they may also result in a decreased launch rate over the short run by delaying launches when serious safety problems are identified.

The efficacy of the system safety activities depends on various factors. Some of these factors are defined outside this sub model, such as the availability of resources to be allocated to safety and the availability and effectiveness of safety processes and standards. Others depend on characteristics of the system safety personnel themselves, such as their number, knowledge, experience, skills, motivation, and commitment. These personal characteristics also affect the ability of NASA to oversee and integrate the safety efforts of contractors, which is one dimension of system safety effectiveness. The quantity and quality of lessons learned and the ability of the organization to absorb and use these lessons is also a key component of system safety effectiveness.

#### 3.1.1.8 Hazardous event (incident) earning and corrective action

The objective of this sub model is to capture the dynamics associated with the handling and resolution of safety-related anomalies and hazardous events. It is one of the most complex sub models, reflecting the complexity of the cognitive and behavioral processes involved in identifying, reporting, investigating, and resolving safety issues. Once integrated into the combined model, the amount and quality of learning achieved through the investigation and resolution of safety problems impacts the effectiveness of system safety efforts and the quality of resulting corrective actions, which in turn has a significant effect on the technical risk of the system.

The structure of this model revolves around the processing of incidents or hazardous events, from their initial identification to their eventual resolution. The number of safety-related incidents is a function of the technical risk. Some safety-related problems will be reported while others will be left unreported. The fraction of safety problems reported depends on the effectiveness of the reporting process, the employee sensitization to safety problems, the possible fear of reporting if the organization discourages it, perhaps due to the impact on schedule. Problem reporting will increase if employees see that their concerns are considered and acted

upon, that is, if they have previous experience that reporting problems led to positive actions. The reported problem also varies as a function of the perceived safety of the system by engineers and technical workers. A problem reporting positive feedback loop creates more reporting as the perceived risk increases, which is influenced by the number of problems reported and addressed. Numerous studies have shown that the risk perceived by engineers and technical workers is different from high-level management perception of risk. In our model, high-level management and engineers use different cues to evaluate risk and safety, which results in very different assessments.

A fraction of the anomalies reported are investigated in the model. This fraction varies based on the resources available, the overall number of anomalies being investigated at any time, and the thoroughness of the investigation process. The period of time the investigation lasts will also depend on these same variables.

Once a hazard event or anomaly has been investigated, four outcomes are possible: (1) no action is taken to resolve the problem, (2) a corrective action is taken that only addresses the symptoms of the problem, (3) a corrective action is performed that addresses the root causes of the problem, and (4) the proposed corrective action is rejected, which results in further investigation until a more satisfactory solution is proposed. Many factors are used to determine which of these four possible outcomes results, including the resources available, the schedule pressure, the quality of hazardous event or anomaly investigation, the investigation and resolution process and reviews, and the effectiveness of system safety decision-making. As the organization goes through this ongoing process of problem identification, investigation, and resolution, some lessons are learned, which may be of variable quality depending on the investigation process and thoroughness. In our model, if the safety personnel and decision-makers have the capability and resources to extract and internalize high-quality lessons from the investigation process, their overall ability to identify and resolve problems and do effective hazard mitigation will be enhanced.

#### 3.1.1.9 Perceived success by high-level management

The purpose if this sub model is to capture the dynamics behind the success of the Shuttle program as perceived by high-level management and NASA administration. The success perceived by high-level management is a major component of the Pushing the Limit reinforcing loop, where much will be expected from a highly successful program, creating even higher expectations and performance pressure. High perceived success also creates the impression by high-level management that the system is inherently safe and can be considered operational, thus reducing the priority of safety, which affects resource allocation and system safety status. Two main factors contribute to the perception of success: the accumulation of successful launches positively influences the perceived success while the occurrence of accidents and mishaps have a strong negative influence.

## 3.2 Model customization

The original Columbia model was essentially unchanged, but some modifications were made to provide a better user experience or to add new capabilities. These are described below.

#### 3.2.1.1 Discrete resources vs. one "Safety Resource"

The Phase I model had a single variable, "Safety Resources," that represented the combined resources all safety-related functions. To provide the end user with more granular control, this variable was replaced with individual resource variables for investigation, maintenance, SM&A and staffing. A multiplier was added to keep the basic model behavior the same, but it is now possible to investigate the effect of preferentially assigning (or removing) resources from a particular area.

#### 3.2.1.2 Outstanding accumulated waivers added

One of the outcomes from the ITA report [2] was the determination of several "leading variables" that provide an indication of increasing risk in advance of risk (hence the term, leading). The ITA model had a leading indicator, "Outstanding accumulated waivers," which is a stock (i.e., integrated value) given by the Waiver Issuance Rate—Waiver Resolution Rate. Since Waiver Issuance Rate was not in the Phase I report it was added to this simulation to provide a complete list of leading indicators.

#### 3.2.1.3 Substitution for zero values

The nature of closed loop system dynamic models is that the model will result in a null set if many variables are set to 0, even if that is the appropriate real world value. To correct for this, any variables that result in a null set were modified to substitute a very small number (i.e., 0.0001) instead, which provides the same real-world result without breaking the simulation.

#### 3.2.1.4 Simplified model view

Another change was to create a new working view in the model that would mimic the simplified model of Figure 3 in [7], reproduced below in Figure 3-1.

In addition, the new simplified view has several different layers to make it easier to explore the different loops. This is shown below (with all layers displayed)

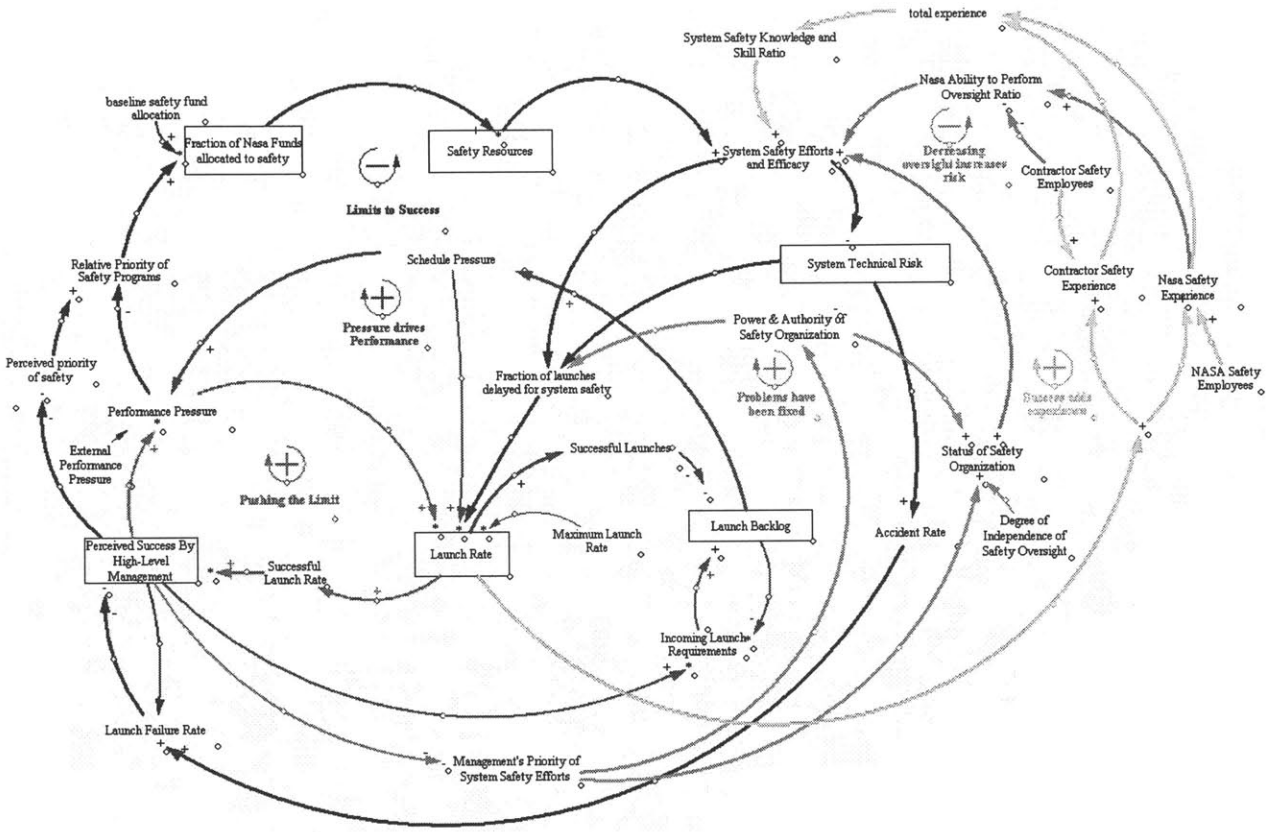



**Figure 3-3 Simplified view from the Vensim system dynamics model**

# 4 Design and Implementation of a Risk Management Flight Simulator

## 4.1 Overview

The application has a single main program, which is designed to provide quick access to all of the key metrics and analysis tools. To start a simulation as user simply types a descriptive name and presses, *Start*. She can then modify any of the input variables and advance the time by 6 months to see the impact on the leading risk indicators and performance indicators. Figure 4-1 shows the main program display for a sample simulation run.

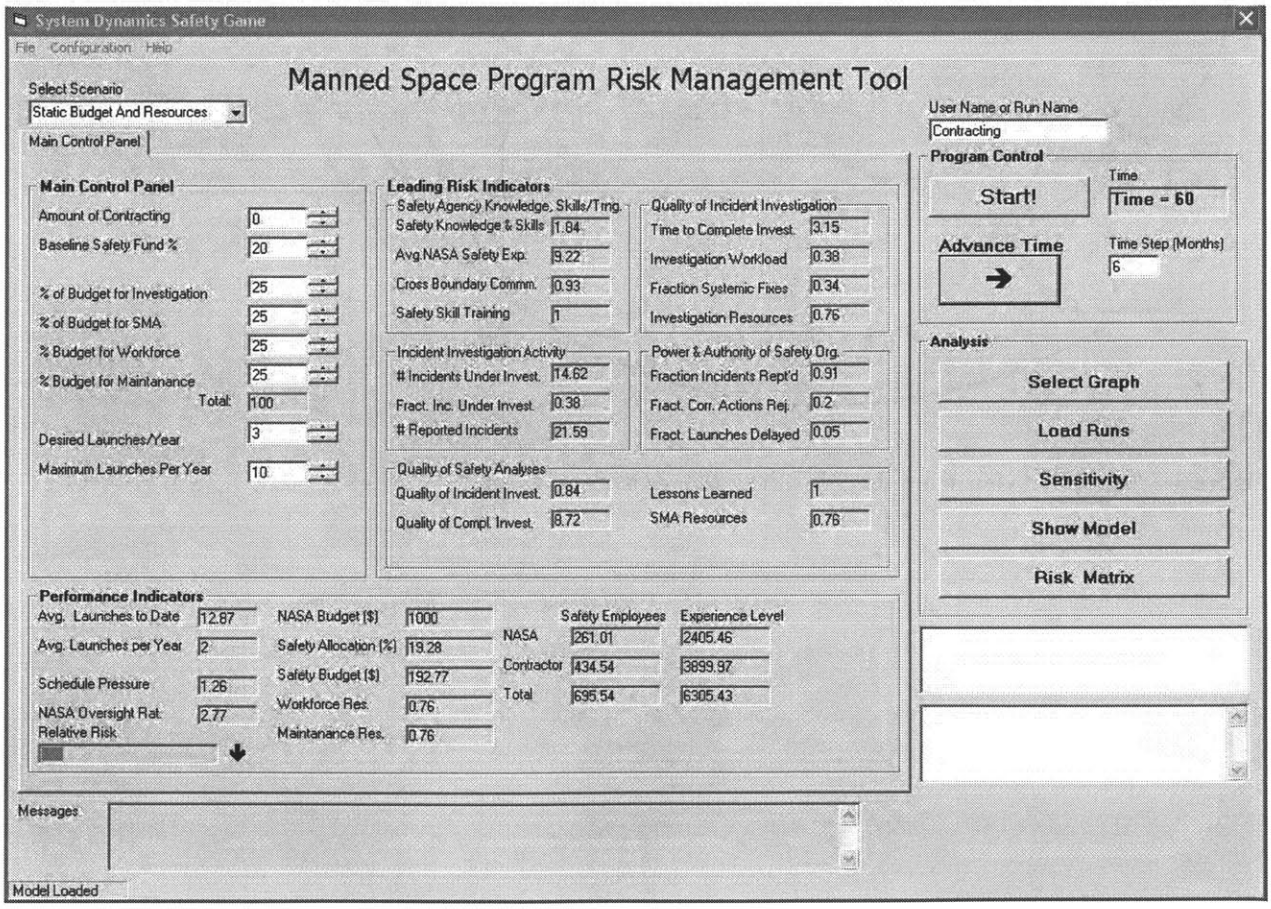



**Figure 4-1 Risk management program main screen**

In this example we can see the current values for several variables at time = 60 months. Clicking the left mouse button on any variable displays an historical trend of the variable. In addition, the user can load (display) the data from any previous run to easily compare different boundary conditions. For example, Figure 4-2, below, is a sample trend of the leading risk indicator, "NASA Safety Experience." The bottom line (HiContracting) and the top line (LoContracting) bound the range of possible values for NASA Safety Experience from modifying the amount of contracting. The middle line is the value of the variable for the current simulation run at time = 60 months.

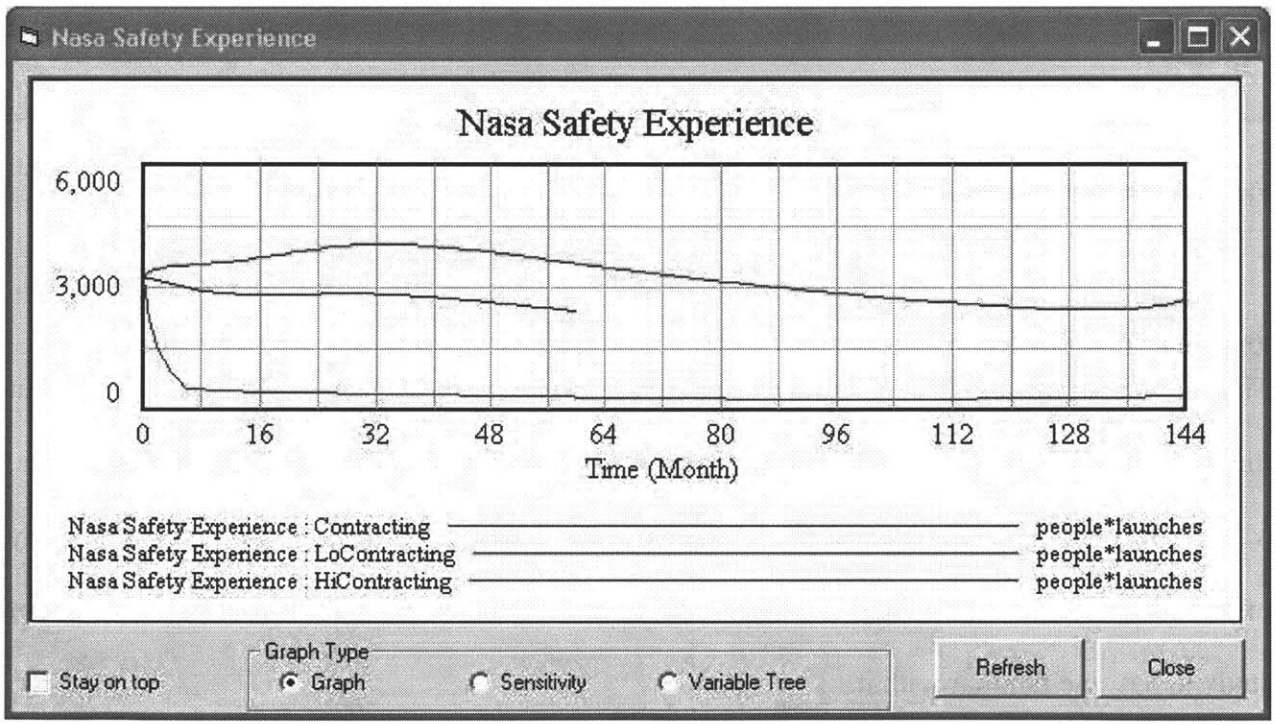



**Figure 4-2 Screen shot of a trend comparing different ratios of contracting**

Another feature of the application is risk matrix display. At any time of the simulation the user can display a severity vs. likelihood risk matrix of the 16 leading risk indicators and their potential impact on system risk.

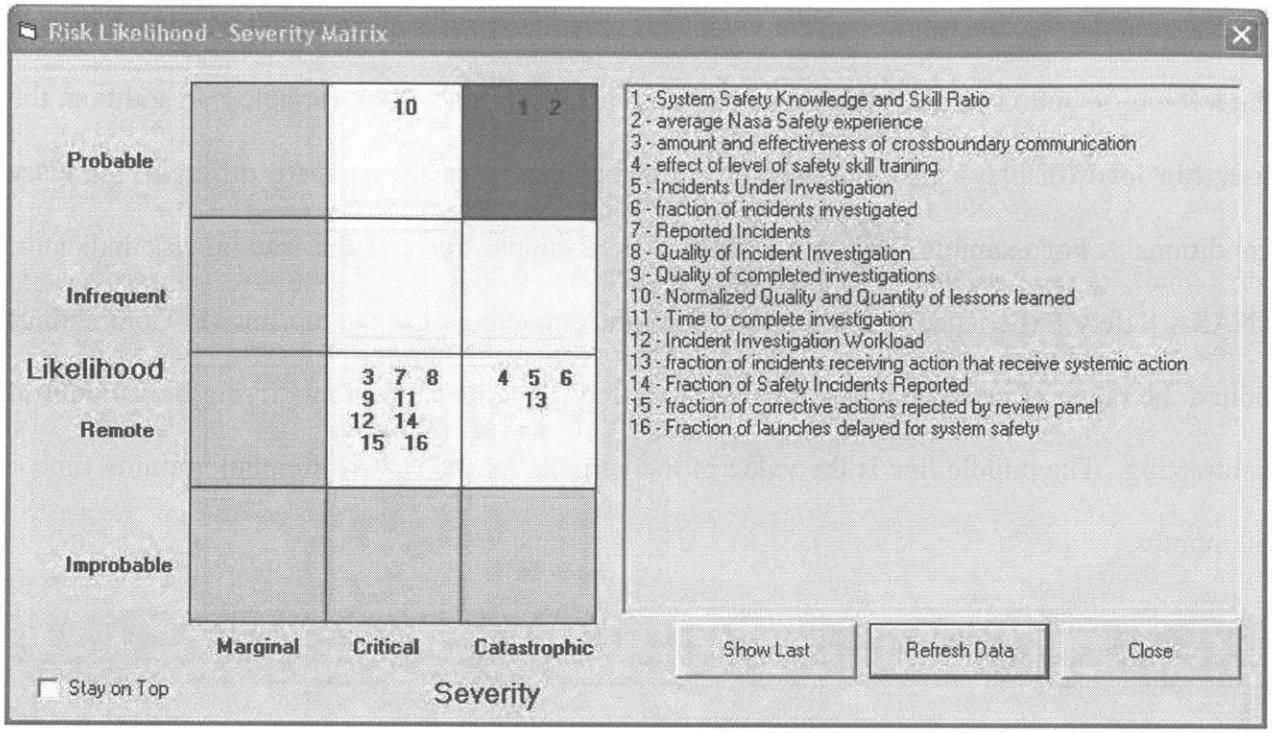



**Figure 4-3 Sample risk matrix display**

## 4.2 System purpose

The software application is being developed so that NASA managers can explore and learn about the dynamics of safety without getting bogged down in either the theory of system dynamics, Vensim ©, or the specific system model developed in Vensim. This latter point is particularly important given that the model has over 700 variables and requires several days to weeks of study to become familiar with it.

### 4.2.1 Program goals

The program has the following goals:

#### 4.2.1.1 Enable a user to explore a simplified representation of the overall system model.

As noted above in 4.1, we want to abstract the user from the details of the Vensim model, but we still want them to be able to explore the basic driving forces of the model. Thus, part of this program will require that the user have a means to easily explore a simplified model that should be readily understandable with only a minimal understanding of system dynamics.

#### 4.2.1.2 Enable a user unfamiliar with either system dynamics or Vensim to be able to explore and learn from the model dynamics.

This is closely related to the previous item, but whereas 4.2.1.1 referred to a model diagram, this program goal only requires that the user be able to interact with and learn from the underlying model dynamics. In other words, just as they might play with a computer game such as *SimCity,* the user should be able to explore and discover model behavior and rules without any need for knowing the actual model structural details.

#### 4.2.1.3 To develop the interface program so that it can be used (or further developed) into a NASA risk management tool.

The requirement of developing a tool that will be *used* by NASA to improve system safety and the understanding of safety concepts is the real purpose of this application. As such, it must present risk metrics and information in terms that are familiar to NASA, as well as being consistent with any other programmatic requirements.

### 4.2.2 High level functional requirements

#### 4.2.2.1 Display relevant safety variables using a risk matrix.

The 5x5 risk matrix (see below) is a standard used within NASA for describing risk and ascribing a relative importance. For this program the leading risk indicators and their severity are already identified in the ITA report [2]. Since the ITA report uses a low-medium-high severity scale, we will use a 4x3 risk matrix as opposed to a 5 x 5.

#### 4.2.2.2 Develop the application so that it can incorporate different scenarios.

As discussed in section 2.2.3, our goal is to enable open-ended learning through dynamic simulation. Just as an aeronautical flight simulator can run different scenarios (e.g., poor visibility, failed equipment, etc.) to enable learning so, too, should this application be able to run different organizational scenarios. For example, reducing the budget while increasing launch requirements. The program should be designed so that the user can select different scenarios and as time "progresses" within the simulation the program will automatically input the appropriate value(s).

#### 4.2.2.3 Allow the user to modify system variables at different simulation time steps

The user should be able to modify selected variables at any time of the simulation and verify that inputted values were accepted

#### 4.2.2.4 Provide trend displays of all variables

The user should be able to easily view a trend of any available variable of its value over time

#### 4.2.2.5 Enable a user to trace dependent variables

The user should be able to trace how a variable is used within the model

#### 4.2.2.6 Enable a user to explore the high level model one "loop" at a time

To meet requirement 4.2.1.1 the user will be able to explore a defined high level view and take advantage of the hidden layers property to successively show additional loops, so that she can more easily see the relationship between the different loops.

#### 4.2.2.7 Perform sensitivity analysis runs and displays

To simplify making test runs at different values (e.g., evaluating the results of setting a variable to a low, medium or high value), the application should provide for an easy means to create a linear or random sensitivity run for one or more variables and display the results along with any other non-sensitivity runs.

#### 4.2.2.8 Enable a user to save and compare runs

To facilitate learning the user should be able to create descriptive names for different simulations and compare them using the available trending tools of 4.2.2.4.

#### 4.2.2.9 Develop the application using commonly available programming tools and applications

To support re-use and further development by others, the application should use commonly available programming tools, databases and other third party applications.

#### 4.2.2.10 Develop the application so that it can be upgraded and expanded with a minimum of rework

To the extent possible, the application should be developed so that it can be adapted to other system models with a minimum of rework. This does not mean no work—,but a designer should not have to start over simply due to a new model being used.

### 4.2.3 System limitations

#### 4.2.3.1 Game and non-game versions of the Vensim model

To enable the user to modify variables at different times (high level requirement 4.2.2.3) we will take advantage of a Vensim feature called "game mode." In game mode the user is able to modify pre-defined game variables and replace the value of the variable at any time of the simulation. Unfortunately, game mode is incompatible with Vensim's "synthsim mode" which enables a user to modify a value and see the entire model response from the beginning to end of the simulation.[3] Since an additional requirement is to perform sensitivity analyses (4.2.2.6), the program needs to support running sensitivity runs with the game variables. However, a limitation of Vensim is that a game variable cannot be used as a sensitivity variable. Therefore,

[3] Fundamentally, game mode lets you step through the model one time period at a time and modify variables along the way to see how the model will subsequently respond. Synthsim mode views the simulation as a whole and does not let the user arbitrarily change values at any number of time steps.

we need to have two similar versions of the model: one with defined game variables (for game mode) and another with the same variables defined as simple constants (non gaming).

#### 4.2.3.2 Inability to easily transition between game mode and "synthsim" mode

Game mode and synthsim are two different operational modes of Vensim and cannot be displayed simultaneously.

#### 4.2.3.3 Single user play

This version of the program does not support multi-user play. However, Vensim supports this and the application could be modified at a future date to support it.

### 4.2.4 System constraints & requirements

#### 4.2.4.1 Vensim DSS

The application was developed with the full version of Vensim ("Vensim DSS") using the free MIT license for academic use. It has not been tested with other versions of Vensim (i.e., Vensim PLE).

#### 4.2.4.2 Minimum 1024x768 display screen

The minimum display resolution is 1024x768, but 1280x1024 is preferred.

#### 4.2.4.3 Microsoft Windows and required Windows components

This application was developed to run on the Microsoft Windows operating system. It was designed and tested on Windows XP Professional Service Pack 2. The user does not need a license for Microsoft Access ® for the application to read and write to the scenario database, but they will need to have the Microsoft Data Access Components (MDAC) v2.7 or above. MDAC is commonly installed with many COTS applications, and is also downloadable from http://msdn.microsoft.com/data/mdac/downloads/default.aspx. The English MDAC install is also included with the risk management installation program.

#### 4.2.4.4 Application support for MS COM-based DLL's

This application extensively uses the Vensim dynamic link library (DLL). Because the Windows .NET programming platform (C#, VB.Net, etc.) has limited support for COM based DLLs in "native mode," the application was developed using Visual Basic v6.

## 4.3 System design principles

### 4.3.1 Description

The high level diagram of the system major components is shown below in Figure 4-4. This specification refers primarily to the user interface portion of the diagram, although various modifications had to be made to the Vensim model to provide the gaming interface and some additional capabilities.

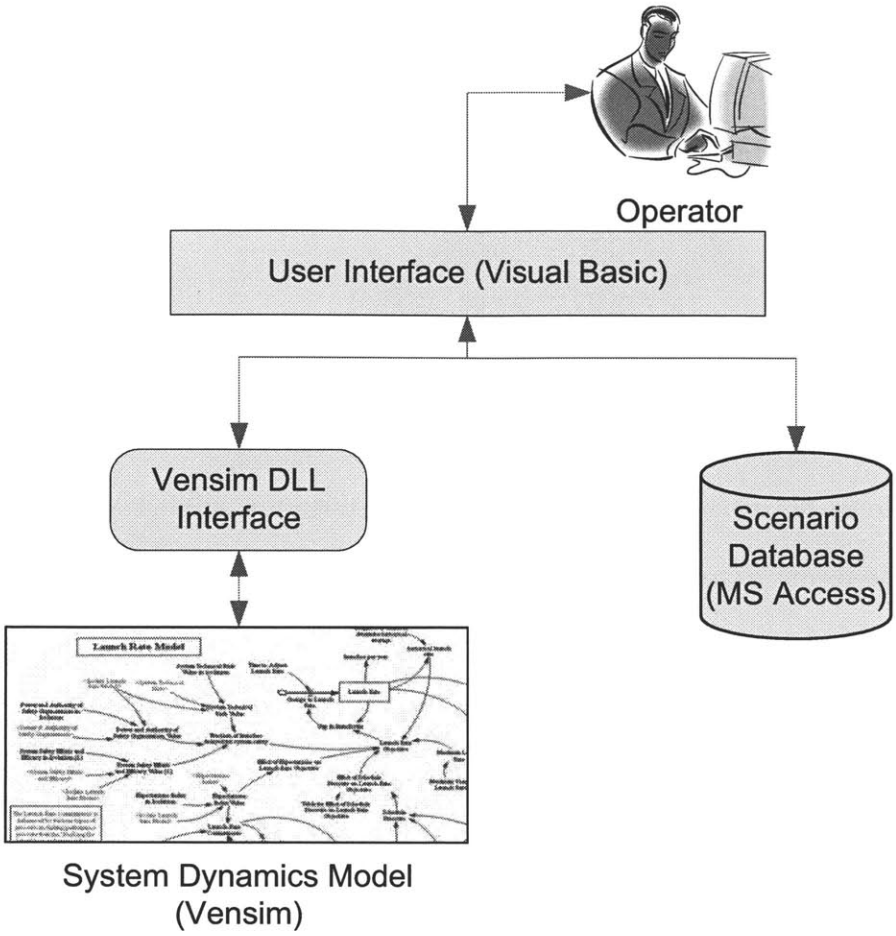



**Figure 4-4 High level component diagram of risk management tool**

### 4.3.2 System components

#### 4.3.2.1 Vensim Models

As discussed in section 4.2.3.1 we need to have two versions of the Vensim model: one with game variables and one without.

##### *4.3.2.1.1 Non-game model*

The non-game model is nearly identical to the original model developed by Dulac and Leveson in [7]. The only modifications were to create greater granularity in the variable, "safety resources" so that an operator could split budget resources between investigation, Safety and Mission Assurance (SMA), workforce staffing and maintenance. In addition, a new view was

created that essentially consolidates the model behavior to match the simplified diagram of the dynamics behind the Columbia loss in Figure 1-2.

#### *4.3.2.1.2 Game model*

The game model is identical to the non-game model except that any variables that we want the ability to modify were re-designated as gaming variables. For other than constant or lookup variables, this is of no real consequence. However, level and dynamic (i.e., formula) variables can also be designated as game variables, which has the effect of "turning off" their normal dynamic behavior. The ability to override calculated results can be advantageous in instances where one is playing a game and wants to try and "beat the system." However, since our goal was to have a minimal impact on the normal operation of the model, we restricted gaming variables to constants that might normally be under the (manual) control of a decision maker e.g., resource allocation, desired launches per year and so forth.

### 4.3.3 User interface

#### 4.3.3.1 Simple display screen with all variables readily visible

The user interface should be relatively clean and uncluttered so that all meaningful information is readily visible without requiring the user to select different buttons, tabs, pages, etc.

#### 4.3.3.2 Extensive use of Vensim charting and display tools

To simplify development and to provide consistency between the Vensim reports and the risk management tool, the application should utilize Vensim's graphing and charting capabilities whenever possible. This will also eliminate the need for a 3rd party graphing and charting tool.

### 4.3.4 Software reuse through modularity

A primary functional requirement is for the program to be adaptable to other system dynamic models (4.2.1.3). While the application does not fully meet this goal, it does have some of the

basic underpinnings through the use of an Access database to store different scenarios and eventually the ability to redefine system variables.

#### 4.3.4.1 Microsoft Access ® database for scenario configuration

An Access database will be used to store scenario names, variables and when their values are updated. In addition the database should provide a short description to accompany each change so that the operator can be informed about the updated value.

#### 4.3.4.2 Microsoft Access database for Vensim integration (Not implemented)

The access database should provide tables that define the model variable names, types, descriptions, limits, where they should be displayed on the user interface and other simulation parameters as later determined. (The functions and methods necessary to implement this feature were not implemented for this thesis, but several test cases proved that the idea could be further developed and implemented.)

### 4.3.5 Object-oriented design

#### 4.3.5.1 Extensive use of class-based design to minimize programming

The application should use classes where they can reduce the programming and debugging in this current application.

### 4.3.6 Rapid development environment

#### 4.3.6.1 Microsoft Visual Basic v6

As noted in the section 4.2.4.4, Microsoft .NET programming environment does not always "play nice" with COM-based DLLs and past experience has shown (painfully) that legacy application interoperability is best developed with a legacy development environment. Since Visual Basic 6 is significantly easier and faster to develop a graphical user interface (GUI) than Visual C/C++, and all of the Vensim program examples are provided in either VB or VC notation, this application was developed in Visual Basic v6 with the latest service packs.

#### 4.3.6.2 Microsoft Access

Microsoft Access was chosen as the database layer due to wide support and availability.

### 4.3.7 Integration with Vensim ©

The application will integrate with Vensim using the vendor's proprietary dynamic link library (DLL) interface.

### 4.3.8 A risk matrix for leading risk indicators

The application will provide a means for displaying all leading indicators using a 4 x 3 (likelihood vs. severity) matrix. Note that the severity is fixed for each variable from reference [2] and consequently it can only move up or down in likelihood.

## 4.4. Black box behavior

### 4.4.1. Scenario control

The user will be presented with a combo box that will be populated with a short description from the scenario database. With each time step the application will query the database for any variables that need to be updated and write the appropriate values to the model. Additionally, the program will display any associated message with the updated variable as well as a history of changes.

### 4.4.2. Save & compare user runs

The application will provide a space for the user to enter a descriptive name for each run of the program. When a new game/run is started the user will be prompted for any additional runs that he would like to have displayed alongside the current run's data. For example, if investigating the effect of contracting, he might call one run "High Contracting" and another "Low Contracting" so that the effect of each can be readily compared against each other or other runs.

### 4.4.3. Modify input variable values

The user will be able to modify the value of the defined input variables throughout the game. To validate that the model accepted a value, the program will query Vensim for the current value immediately after it is written and display the returned result.

### 4.4.4. Display graphs of variables over time

The user will be able to display a graph of any variable by clicking with the left mouse button one time on the textbox or label that displays the value of the variable. There will be no limit to the number of variables displayed.

### 4.4.5. Rescale, refresh and keep graphs on top

The user will be able to resize any trend display by simply dragging the trend display window to make it larger. Because more than one graph may be displayed and to allow the user to follow the specific behavior of a variable as the simulation progresses, the user will be able to keep a graph on top of other windows and easily refresh the displayed data.

### 4.4.6. Display graphs from the Vensim model

The user will be able to list and display any predefined graphs in the Vensim model. These graphs will have the same capabilities as described in 4.4.5.

### 4.4.7. Explore variable dependencies

The user will be able to "walk" the dependency tree of any of the model variables to see how they are used within the model. This display will not use the Vensim "uses" or "depends" trees as their similarity to fault-event trees may lead to confusion. The dependencies should be displayed using a Windows "tree" (i.e., Microsoft Windows Explorer interfaced) to preclude this confusion and to present the information in a familiar context. The dependency screen will have the same capabilities as described in 4.4.5.

### 4.4.8. Perform sensitivity analyses

Vensim DSS has the ability to modify one or more variables as functions and display the accumulated results as a sensitivity analysise.g., Ramp a variable from 1 to 100 and execute the model at each step to show model behavior as the value increases. The application will provide for a similar capability, but will limit the user to only a linear ramp or Gaussian function. The user will be able to select one or more variables to modify, the range of modification and, if more than one will be modified, their order. Once the sensitivity run is completed the user will then be able to display the sensitivity run along with any other loaded runs simultaneously. These sensitivity graphs will have the same capabilities as described in 4.4.5.

### 4.4.9. Load & save configuration data to disk

The user will be able to load and save the program parameters such as the scenario database name, Vensim model name and path, and other parameters to disk using .INI files. When the program is closed any modified parameters will be written to disk.

The sensitivity analyses variable configuration will be saved and retrieved using the specific .VSC or .LST Vensim file formats.

### 4.4.10.Register missing Vensim entries

It was discovered before starting development of the program that a bug in the Vensim DLL install does not create the proper Windows Registry entries for DLL-based applications to run. Therefore, to facilitate installing and running the program on other machines, the application will check to see if the DLL entry is appropriately entered and create the entry if missing.

### 4.4.11.Display and explore simplified model

The user will be able to show the high-level schematic view of 4.2.2.6 with a slider-type control to show each of the hidden layers successively. This display will also let the user zoom in and out, fit the graph to the display, and resize the window.

### 4.4.12.Display updated risk consequence-severity matrix over time

#### 4.4.12.1 Determine likelihood level

As noted in section 4.2.2.1, the severity level is fixed from the ITA report [2] as low, medium or high. However, there is not a clear definition of "likelihood" and this is later discussed as an area for further research. For this software, likelihood was determined by looking at each leading indicator and applying a logical test to come up with a likelihood function. For some variables, this is straightforward—e.g., "fraction of launches delayed for safety" is a leading indicator that maps from 0 to1 and an increasing value corresponds with greater risk.[4] But, others, such as the "average quality of completed investigations," are more complicated because they do not map from 0 to1 and increasing quality corresponds with lesser risk, not greater. In this case, the likelihood effect function was given by the function, "1 – average quality of completed investigations". The logic diagram for determining each variable's likelihood function is shown below.

[4]As with many dynamic models, the assumption that the import of a variable scales from one extreme to the other does not hold up well at the boundaries. For example, if 0 launches are delayed due to safety then it may be an indicator of a severely broken safety system and if 100% of the launches are delayed then the safety system may be working (to prevent launches), but clearly there are significant safety issues resulting in the cancellation of all flights.

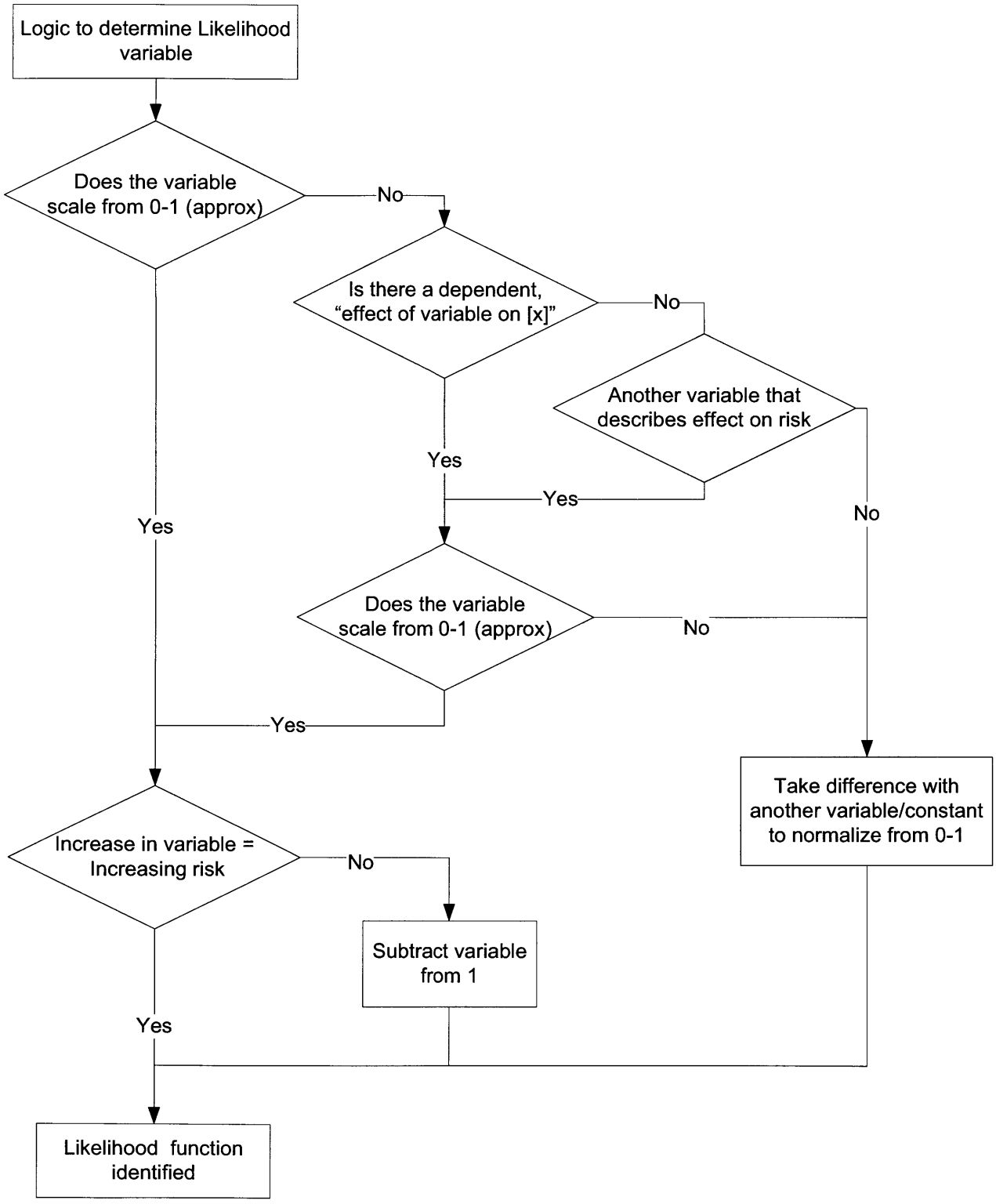



**Figure 4-5 Logic diagram to determine the severity function for the severity-likelihood matrix**

**Table 4-1 Leading indicators and associated likelihood functions (From the ITA report [2].)**

| Leading Indicator Model Variable | Likelihood Effect Function |
|---|---|
| **Safety, Knowledge & Skills** | |
| System Safety Knowledge and Skill Ratio | Effect of System Safety Knowledge and Skill Ratio on System Safety E&E |
| Average Nasa Safety experience | Effect of Nasa ability to perform oversight ratio on Safety E&E |
| Amount and effectiveness of crossboundary communication | Amount and effectiveness of crossboundary communication |
| Effect of level of safety skill training | Effect of level of safety skill training |
| **Investigation** | |
| Incidents Under Investigation | Incident Investigation Workload |
| Fraction of incidents investigated | 1 - fraction of incidents investigated |
| Reported Incidents | 1 - Fraction of Safety Incidents Reported |
| **Quality of Safety Analyses** | |
| Quality of Incident Investigation | 1 - Quality of Incident Investigation |
| Quality of completed investigations | 1 - average quality of completed investigations |
| Normalized Quality and Quantity of lessons learned | Effect of Root Cause Incident Learning Value on System Safety E&E |
| **Quality of Incident Investigation** | |
| Time to complete investigation | Time to complete investigation - normal time to complete investigation |
| Incident Investigation Workload | Incident Investigation Workload |
| Fraction of incidents receiving action that receive systemic action | Effect of Corrective Actions adressing Systemic Factors on Risk |
| **Power and Authority of Safety Authority** | |
| Fraction of Safety Incidents Reported | 1 - Fraction of Safety Incidents Reported |
| Fraction of corrective actions rejected by review panel | Fraction of corrective actions rejected by review panel |
| Fraction of launches delayed for system safety | Fraction of launches delayed for system safety |

Note: These are the actual variable names and any apparent typographical errors merely represent the naming convention used in the model

In addition to determining the likelihood function the value must then be mapped onto one of four levels: improbable, remote, infrequent, and probable. [5] In a PRA-centric approach, the attribution of a risk to likelihood level is a simple matter of choosing even breakpoints. But, as noted earlier, an area of further research is developing a consistent scale for all of the leading variables. Until then, trial and error indicated that the following break points result in a good distribution of likelihoods.

[5] This terminology follows the NASA convention to make the results more familiar to NASA managers, but their adaptation from a hardware/system failure domain to a leading risk indicator domain is a bit problematic. Given the qualitative nature of the system dynamics model it may be more appropriate to use terms that reflect the degree of focus that should be devoted to a particular area to address safety concerns. E.g., "Immediate attention required", "long-term problem that should be addressed soon", "long-term problem that should be addressed", and "no long term or immediate safety concerns"

**Table 4-2 Likelihood classification break points**

| Likelihood Classification | Value range |
|---|---|
| Improbable | $\leq 0.33$ |
| Remote | $\geq 0.33$ and $\leq 0.66$ |
| Infrequent | $\geq 0.66$ and $\leq 1$ |
| Probable | $\geq 1.1$ |

#### 4.4.12.2 Risk matrix display functions

Each of the defined leading indicators is mapped to a 4x3 risk matrix to show the changing likelihood of each variable at different time sequences. Each time the user refreshes the display the system keeps the previous value for each variable in memory and lets the user toggle between the previous and current value to more readily see any transitions.

To display each variable, the risk matrix has a 12-square grid with the outer levels in green, the inner levels in yellow and the top right (most severe, highest likelihood) in red. On top of each square will be displayed a number corresponding to a leading indicator listed to the right of the matrix.

To facilitate self-directed investigation, double-clicking on a leading indicator (in the list box) displays that variable's trend display screen.

### 4.4.13.Step model forward in time (user configurable)

The application has an easy method for the user to step the model forward in time by a configurable time period (default = 6 months) until the end of the simulation.

### 4.4.14.Context-sensitive help

Context-sensitive such as "tool tips" are available throughout the program to assist the user. An on-line help document was not developed for this version of the program.

### 4.4.15.Display leading risk indicator values

The main display screen groups the leading risk indicators (see Table 4-1) by functional area on the main display screen. The displayed values are automatically updated with each step forward in model time.

### 4.4.16.Notify user of an "accident"

The Vensim model has an accumulating risk function that simulates an accident when the number of launches equals the risk function. This event is clearly indicated to the user to inform him that the accumulated risk has reached an unsafe level.

### 4.4.17.Log error messages to disk

The application has an error logging function to write errors to disk with a timestamp and short description of the error.

## 4.5. Logical functions

The high-level main program operation is described in this section.

### 4.5.1 Class definitions

#### 4.5.1.1 Vensim class structure

The Vensim class is the core object for this application. An instance of the class is created for each model variable that is displayed and added to a class collection. In addition to reading and writing values to the Vensim model, the class is responsible for trending, dependency lists, calculating the likelihood function for a variable, and updating all user controls. One particularly nice aspect of the class that simplifies the program code is that any label or text box that displays the variable value is assigned to that class's instance so that the class can subscribe to the object for any events such as a mouse click event and respond appropriately. Thus, the main display form has very little code and is easier to troubleshoot and modify.

| Vensim Class (cVensim.cls) |
| --- |
| **Public Properties** |
| GameVariable (Bool) |
| PicFormCollection (Collection) |
| EffectOnSystemSafetyVariable (String) |
| Severity (AccidentSeverityLevel) |
| LeadingVariable (Bool) |
| IsPercent (Bool) |
| VariableName (String) |
| **Public Functions** |
| AddControl (Form Control) |
| PlotVariable() |
| GetVenGraph (Graphname) |
| SetValue (Value as Single) |
| GetValue(Val as Single) as Bool |
| GetEffectOnSS() as Single |
| **Private Functions** |
| AddFormToCollection (WinForm) |
| GetFormValue() as Single |
| UpdateFormControls( Val as Single) |
| HandleError() |
| **Object Events** |
| Label_MouseDown() |
| ProgressBar_MouseDown() |
| TextBox_DoubleClick() |
| Class_Initialize() |
| Class_Terminate() |

**Figure 4-6 Vensim class function overview**

#### 4.5.1.2 Windows INI class

The Windows INI class handles all registry and INI read/write operations. In addition to loading and saving user configuration data the class also updates the registry if the Vensim DLL was not properly installed (4.4.10).

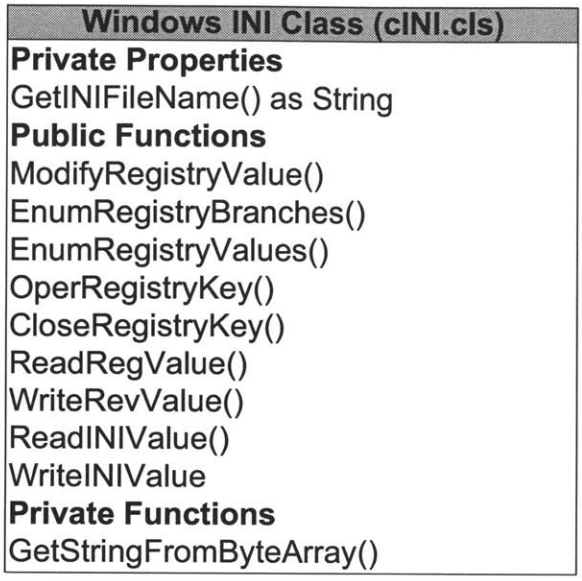



**Figure 4-7 Windows INI class overview**

#### 4.5.1.3 Error handling class

The error handling class writes error and informational messages to disk.

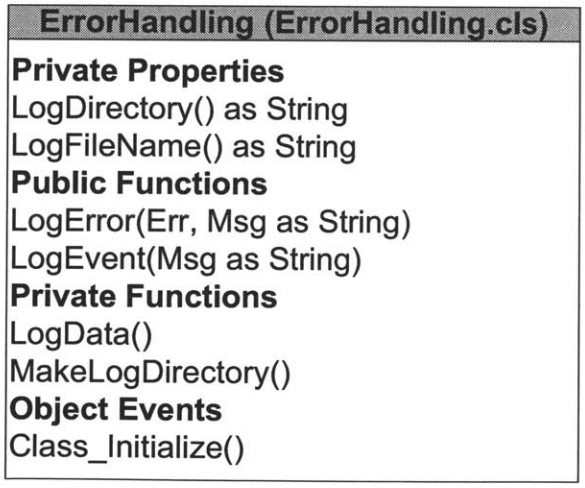



**Figure 4-8 Error handling class overview**

### 4.5.2 Program initialization

Program initialization includes reading user settings from the INI file, checking and fixing the registry (this would only happen once as once fixed the registry is not written to again),

initializing the Vensim classes and connecting to the scenario database. A high level overview is shown below.

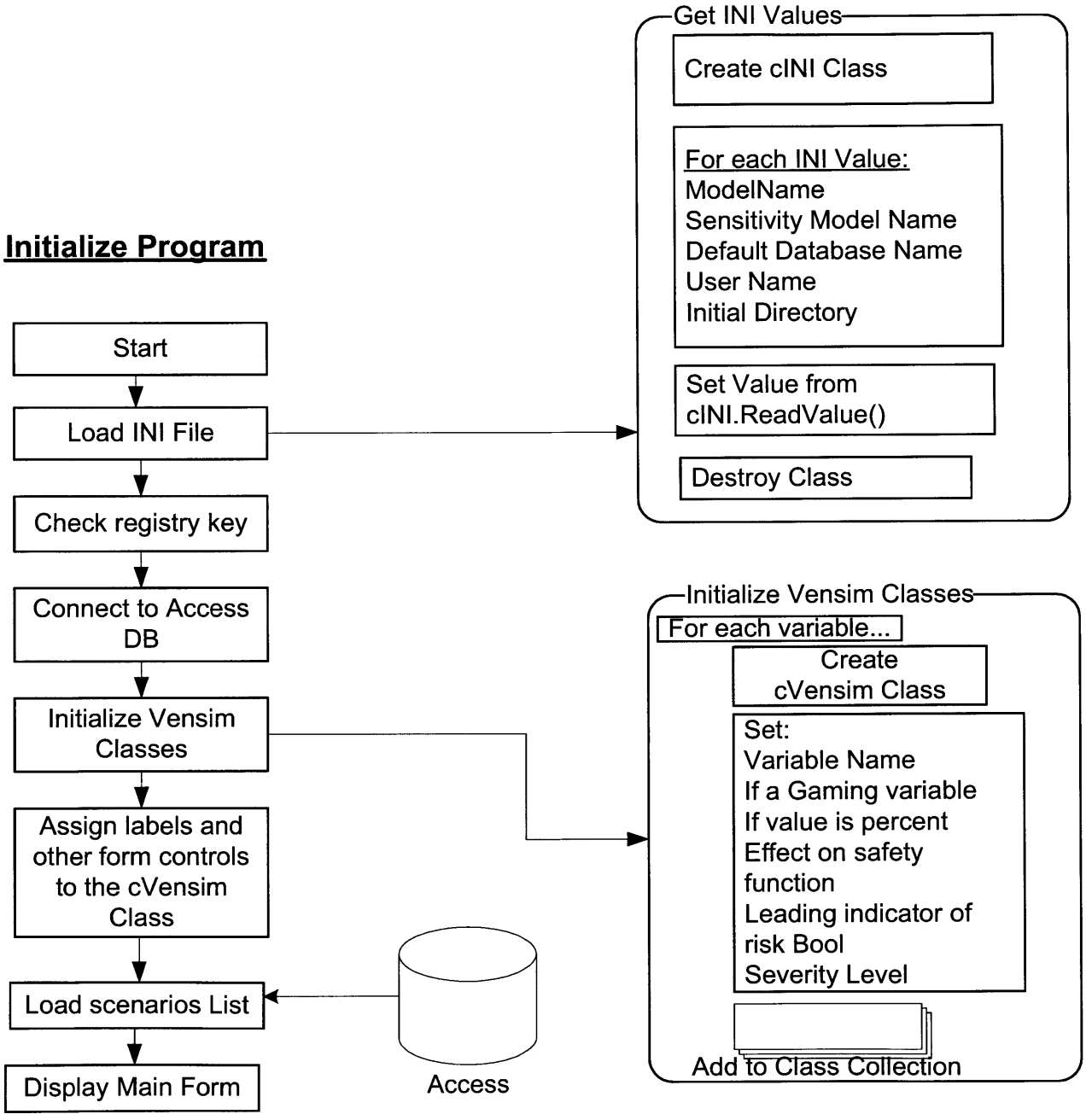



**Figure 4-9 High level overview – Program initialization**

### 4.5.3 Start simulation

A new simulation starts by the user entering a name for the simulation run and pressing the 'Start' button. The program resets the scenario database and then prompts the user to load (i.e. display) any additional runs at the same time. Finally, the main window form is updated with the initial values for all display and user input values. A high level overview is shown below.

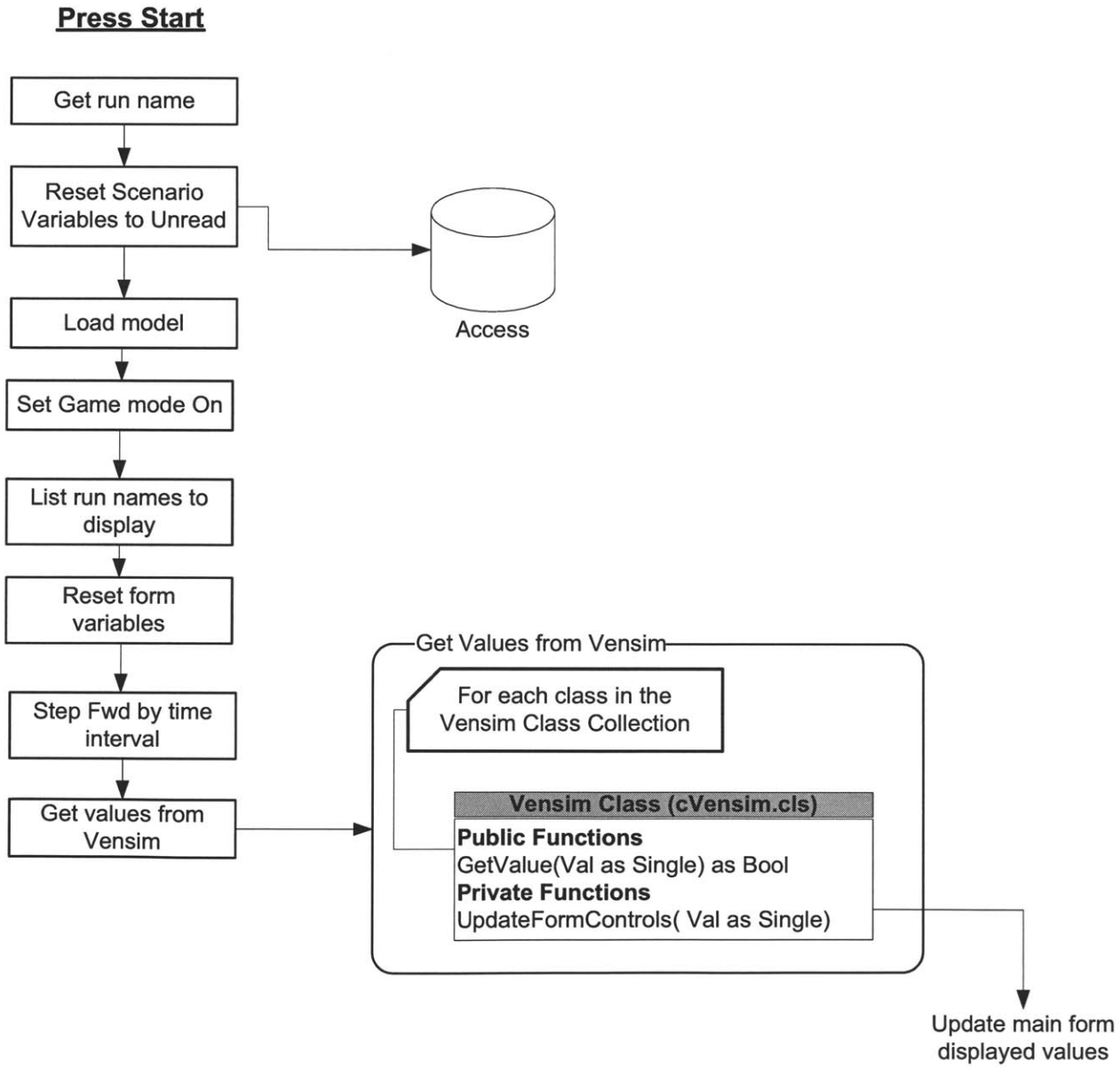



**Figure 4-10 High level overview – starting a new simulation run**

### 4.5.4 Advance simulation

To advance the simulation the user enters a new time step if desired and presses the 'Advance Time' button. The program writes user input values to Vensim, retrieves updated values for all variables and writes values from the scenario database. In addition, an "accident" message is displayed if there was an accident as defined in the model. A high level overview is shown below.

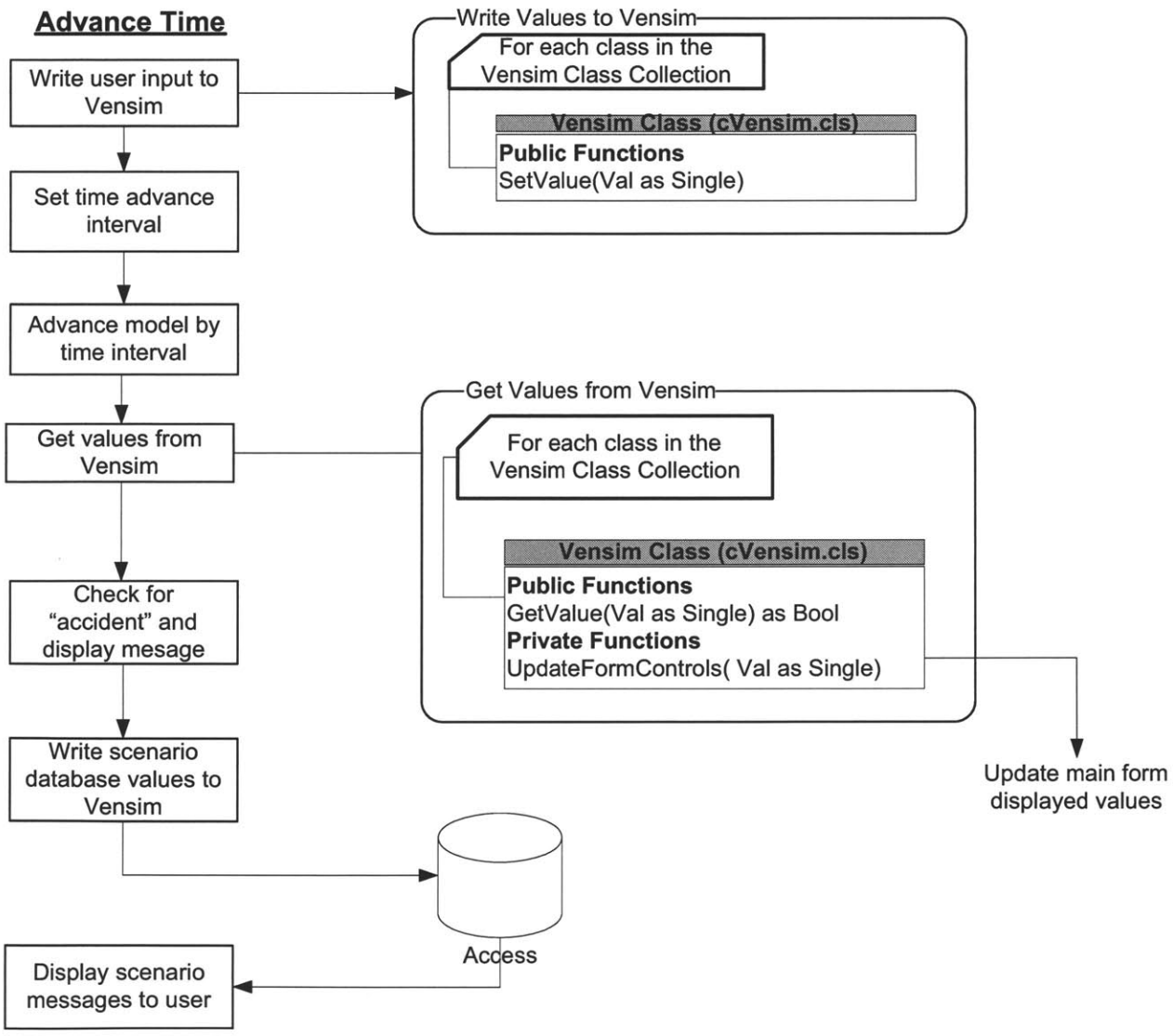



**Figure 4-11 High level overview – advance simulation**

### 4.5.5 Trend/Sensitivity/Dependent variable display

The variable trend display, sensitivity plot and dependency tree share the same form object because they share many of the same methods. By selecting the appropriate option button, the trend updates to show the selected display. Because the application uses the Vensim plotting methods, it was discovered that if a trend was resized that it was necessary to have Vensim recalculate the trend with the updated picture dimensions. Hence, whenever the form is resized it instructs Vensim to refresh the data.

To manage memory and to ensure an easy program termination, each instance of a form is managed in a global forms collection.

In addition, the form takes advantage of some Windows API function calls to provide the option of making it the top-most form on the Windows desktop. This feature makes it possible to keep a trend displayed continuously as the simulation is updated.

A schematic diagram of the program flow for the trend display window is shown below in Figure 4-12.

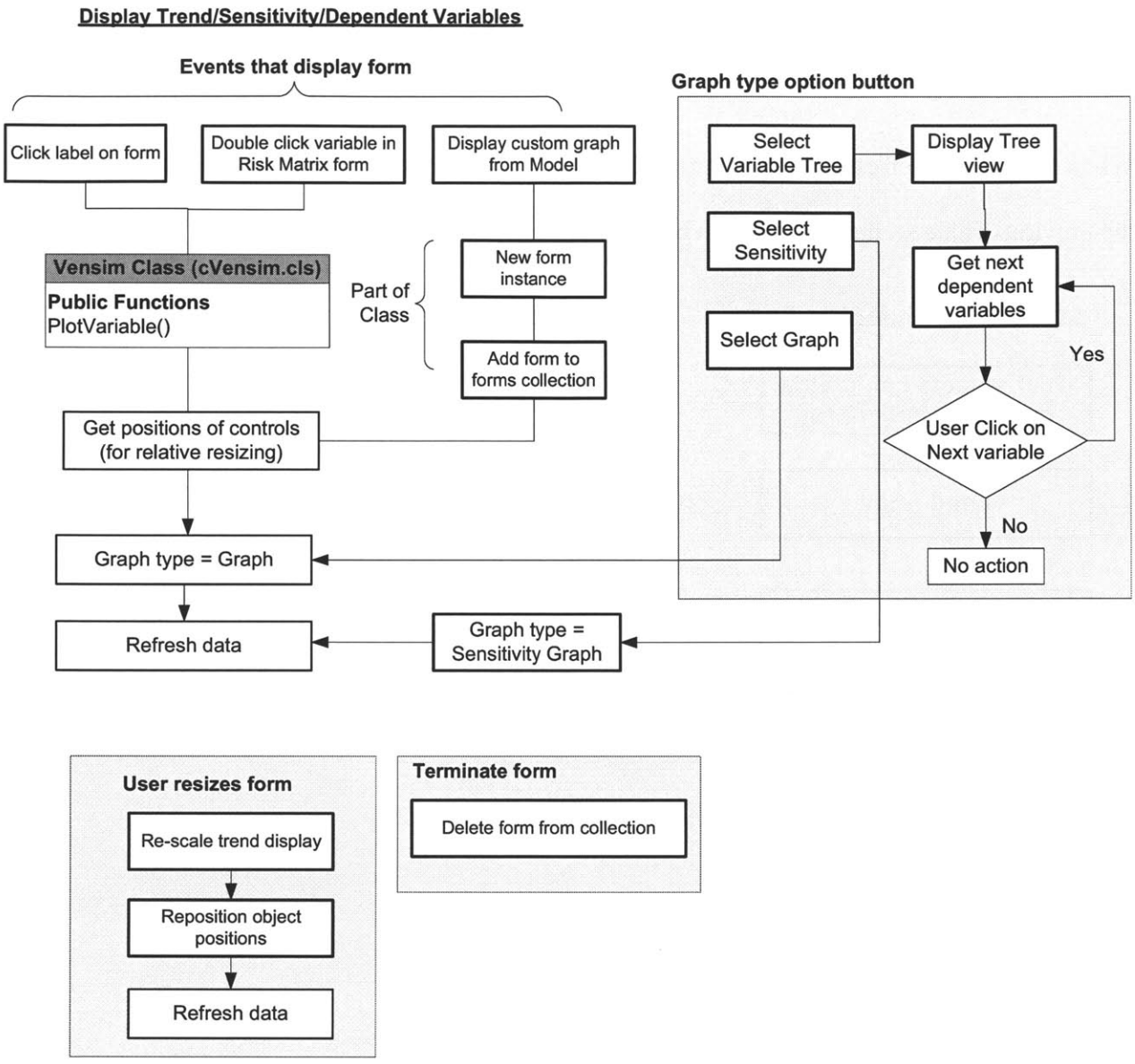


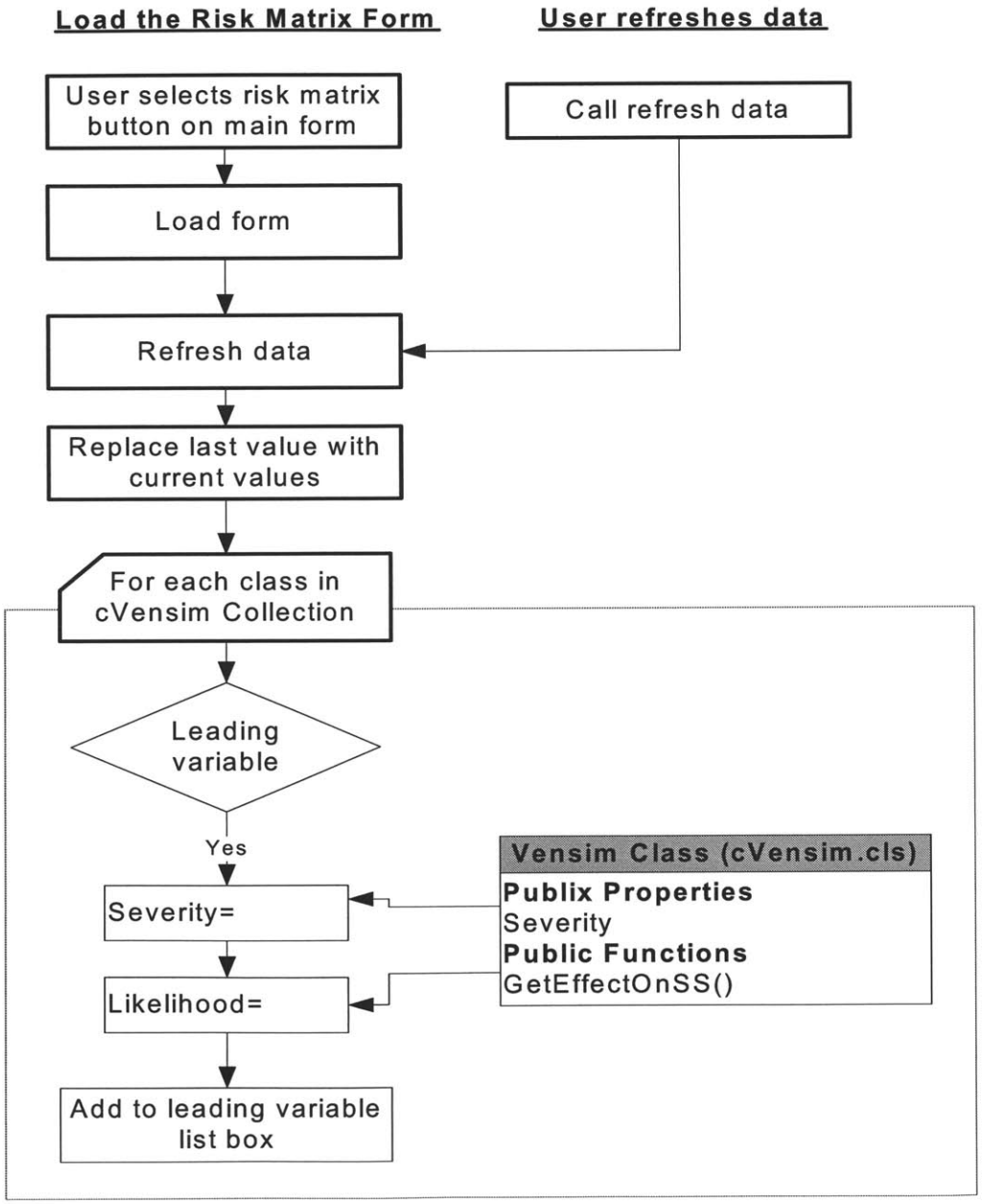

**Figure 4-12 High level diagram of the trend, sensitivity and model dependency tree**

### 4.5.6 Risk matrix form

The risk matrix form maps the leading indicators listed in Table 4-1 to a 4x3 likelihood vs. severity matrix (discussed in Section 4.4.12.2). Upon being loaded, the form iterates through the cVensim class collection to find the classes with the Leading Variable property = TRUE. For each leading variable it adds it to the list box and calls the Severity and GetEffectonSS functions

to get the severity and likelihood, respectively. It then displays the associated variable number in the corresponding likelihood-severity grid coordinate.

When the form is refreshed, it re-queries the class for the updated likelihood value and histories the previous value so that the two can be easily compared.



### 4.5.7 Show model

Selecting the show model button displays a high level view of the system dynamics model as described in 4.2.2.6. The form displays the Vensim view, "Simplified View" and displays the first layer of the view with a single loop. A slider bar is available for the user to show and hide successive loops adding and hiding complexity as needed to help her explore the basic model relationships.

There is also a "show behavior" option that implements the "synthsim" mode of Vensim to display small behavior trends for each variable in the model.

### 4.5.8 User configuration

The user configuration is accessible by selecting Configuration→Select model or Configuration→Game Database from the program title bar. The select model option opens a dialog wherein the user can browse and select the game and non-game versions of the Vensim model. (Refer to section 4.2.3.1).

The select database option lets the user browse and select the access database for the scenario database.

Upon exiting the program, the selected user values are written to the .INI file.

### 4.5.9 Instructions

The user will be able to display basic instructions (read from the scenario database) by selecting Help→Instructions from the program title bar.

## 4.6. Scenario database

### 4.6.1 Table structure

#### 4.6.1.1 Venvariables

The Venvariables table stores the information to be written to Vensim for all of the scenarios.

**Table 4-3 Scenario database Venvariables table structure**

| VenVariables Table | | |
|---|---|---|
| **Field Name** | **Description** | **Data type** |
| ID | Unique index for each field | Long Integer |
| Variablename | The name of the Vensim model variable | String |
| ShortDesc | A short description of the variable | String |
| WhatHappened | Memo to be displayed to operator. Has <Var1> and <Var2> place holders to substitute values | Memo (long string) |
| ElapsedTime | The number of time units (i.e. months) that this value is to be written | Integer |
| TimeEnd | <not used> | |
| Var1 | The value to be written to Vensim. May also be displayed to operator in memo | Single |
| Var2 | A Value to be displayed to the operator, but not written to Vensim. (can be used to express cleaner units such as 4 launches/yr vs. 0.333 launches/month) | Single |
| Inactive | Flag set by program when variable has been processed | Boolean |

#### 4.6.1.2 Scenarios

The scenarios table holds all of the scenario descriptions.

**Table 4-4 Scenario database Scenarios table structure**

| Scenarios Table | | |
|---|---|---|
| **Field Name** | **Description** | **Data type** |
| ScenarioID | Unique index for each field | Long Integer |
| Scenario | A short description of the scenario to be displayed by program | String |
| Description | A longer descriptive field | String |

#### 4.6.1.3 Scenario2Variable

The Scenario2Variable table is a one-to-many map of the Scenarios table to the VenVariables table. In addition to minimizing the chance of error when associating an entry in the VenVariables table with a scenario, it also enables us the re-use entries in more than one scenario.

**Table 4-5 Scenario database Scenario2Variable table structure**

| Scenario2Variable Table | | |
|---|---|---|
| **Field Name** | **Description** | **Data type** |
| ID | Unique index for each VenVariables field | Long Integer |
| ScenarioID | Unique index for each Scenario field | Long Integer |

#### 4.6.1.4 Game play

The Game play table stores on-line help and informational messages. They are accessible when selecting Help→Instructions from the program title bar menu.

**Table 4-6 Scenario database Game Play table structure**

| Game Play Table | | |
|---|---|---|
| **Field Name** | **Description** | **Data type** |
| ID | Unique index for each field | Long Integer |
| ShortDesc | A short description indicating when the message is to be displayed. Presently only 'Intro' | String |
| Message | Memo to be displayed to operator. | Memo (long string) |
| TimeStart | <not used> | <not used> |
| TimeEnd | <not used> | <not used> |
| Var1 | <not used> | <not used> |
| Var2 | <not used> | <not used> |

### 4.6.2 Table relationships diagram

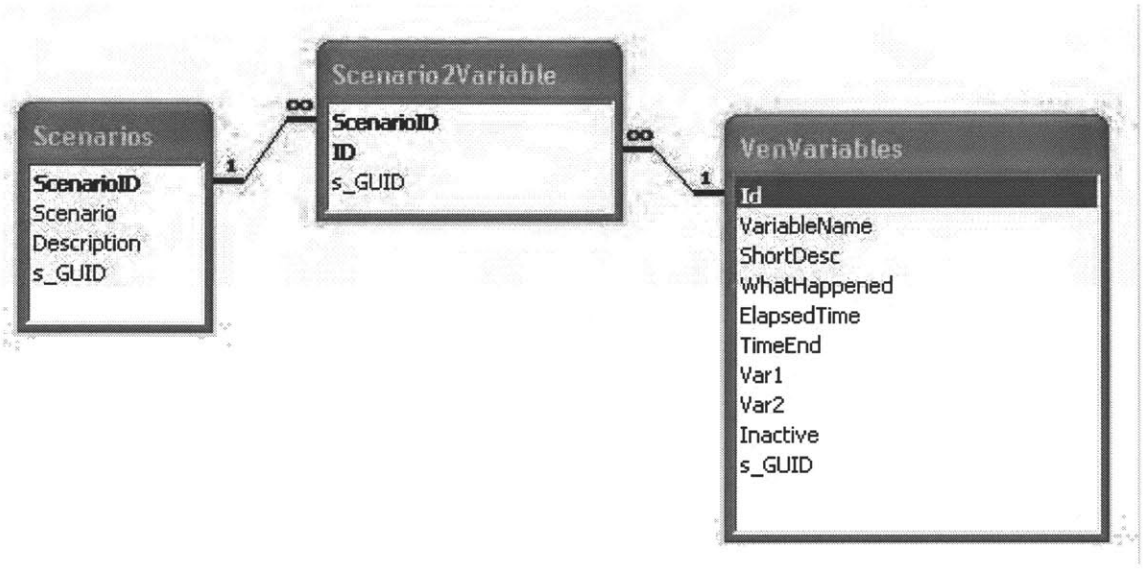



**Figure 4-13 Scenario database table relationships diagram**

## 4.7. Implementation

### 4.7.1 Screen snap shots

#### 4.7.1.1 Main program

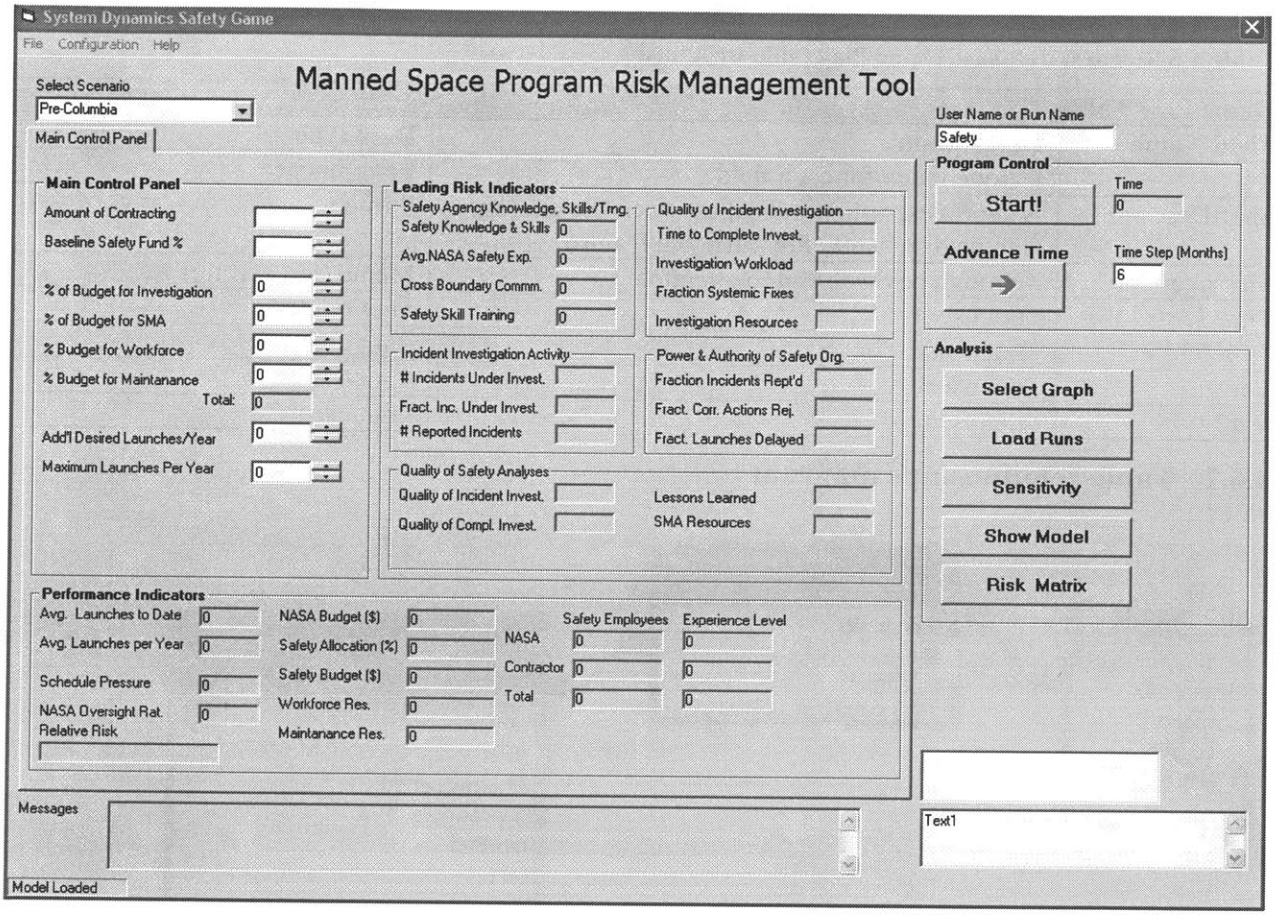



**Figure 4-14 Screen image of main program**

#### 4.7.1.2 Sample trend

Clicking on any variable's value on the main form accesses the sample trend. The example below shows the current simulation (short line), and two previously completed simulations: lo contracting (top line) and hi contracting (bottom line)

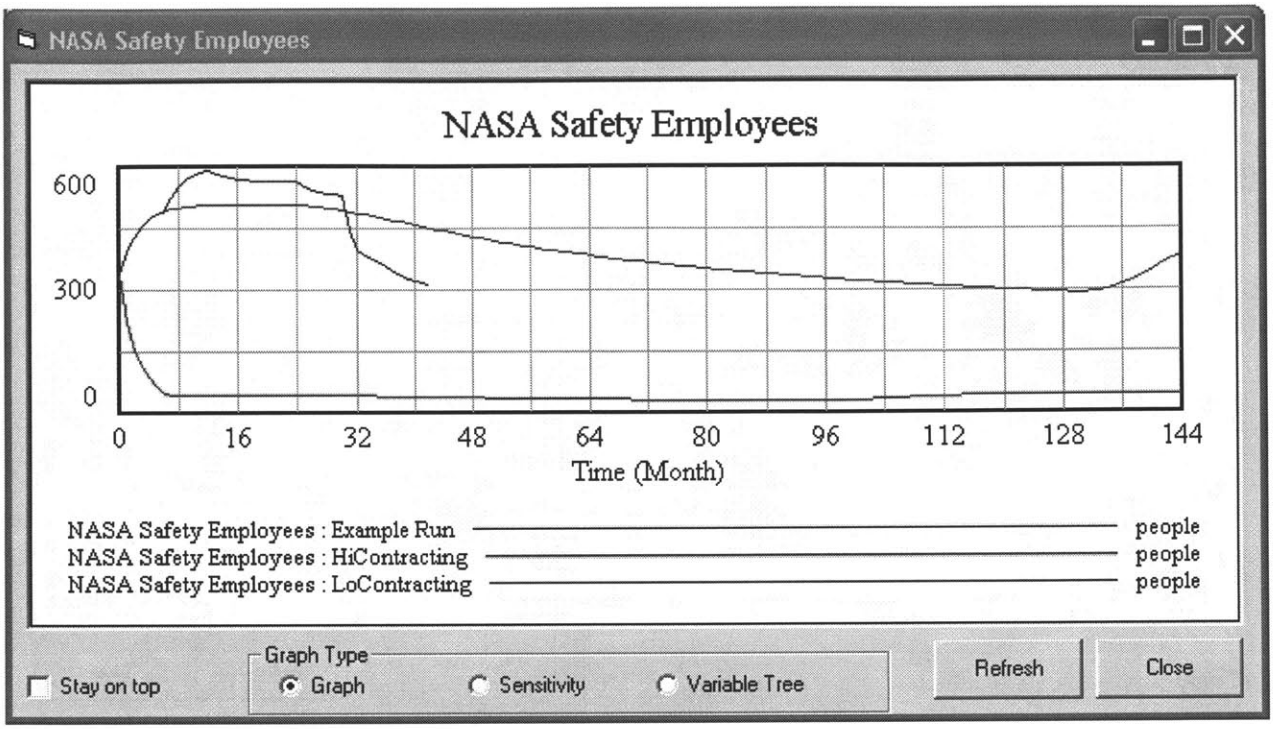



**Figure 4-15 Screen image of variable trend display**

#### 4.7.1.3 Variable tree

The variable tree option button lets the user explore how the variable is used in the model.

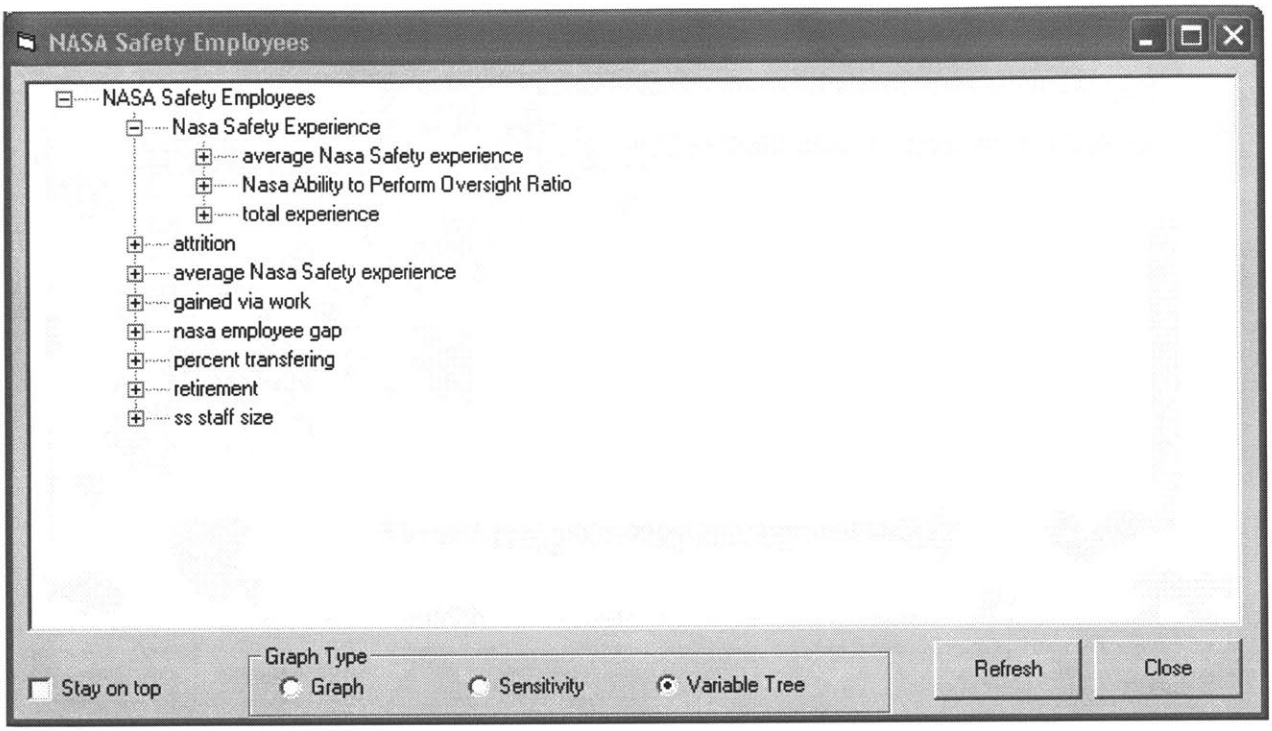



**Figure 4-16 Screen image of variable dependency tree**

#### 4.7.1.4 Sensitivity control

The sensitivity control lets the user select which variables should be modified and the order of simulation. The user then saves and runs the sensitivity to view the results.

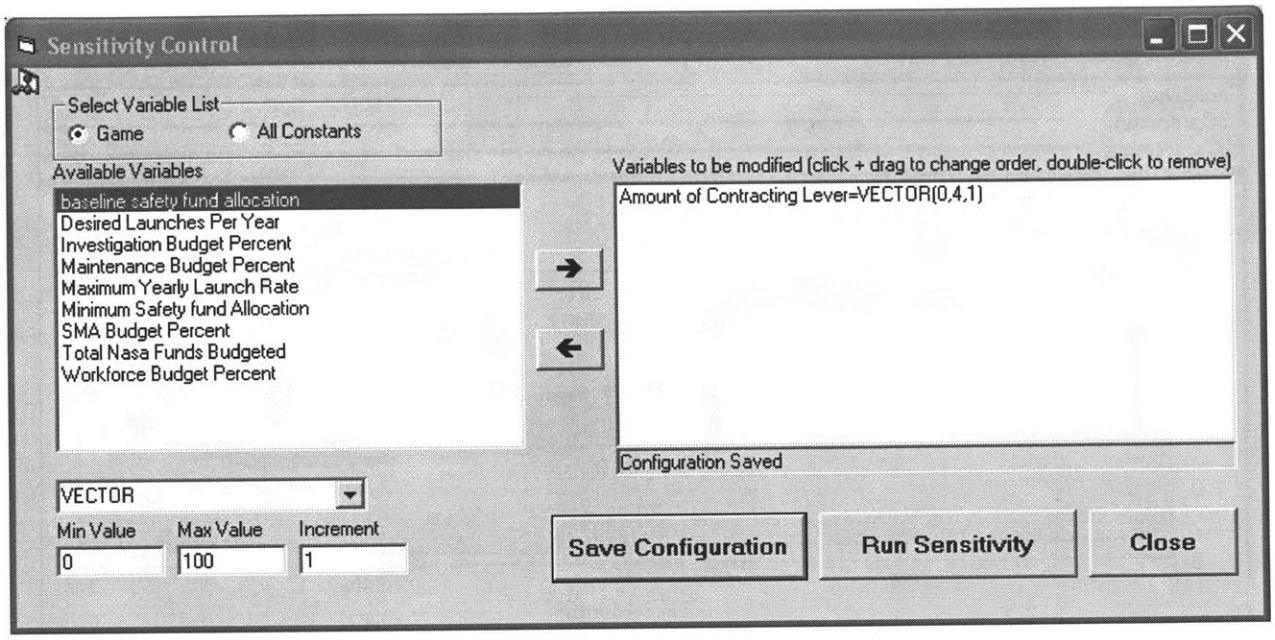



**Figure 4-17 Screen image of sensitivity configuration**

#### 4.7.1.5 Sensitivity display

The sensitivity display shows the range of possible values for a variable from the simulation. Shaded sections indicate which portion of the runs fell within a confidence bound (i.e., the bottom shaded portion indicates a 75% confidence that values won't exceed that band.). Additionally, three other loaded runs are visible as well. The top dark line is the boundary for the low contracting run, the bottom dark line is the boundary for the high contracting run, and finally the short gray line that in the first quartile indicates that the current example run has a particularly high number of NASA employees.

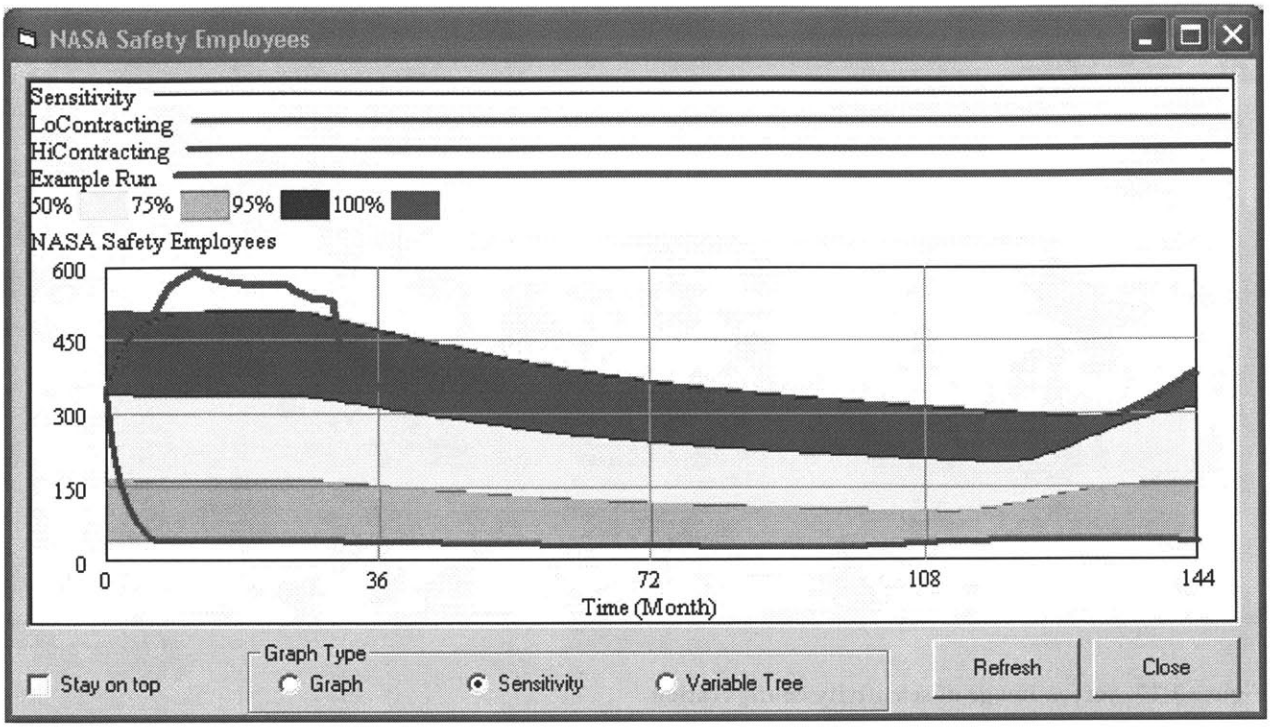



**Figure 4-18 Screen image of sensitivity plot**

#### 4.7.1.6 Risk matrix

The risk matrix displays the leading variables and where they fall on the likelihood-severity matrix.

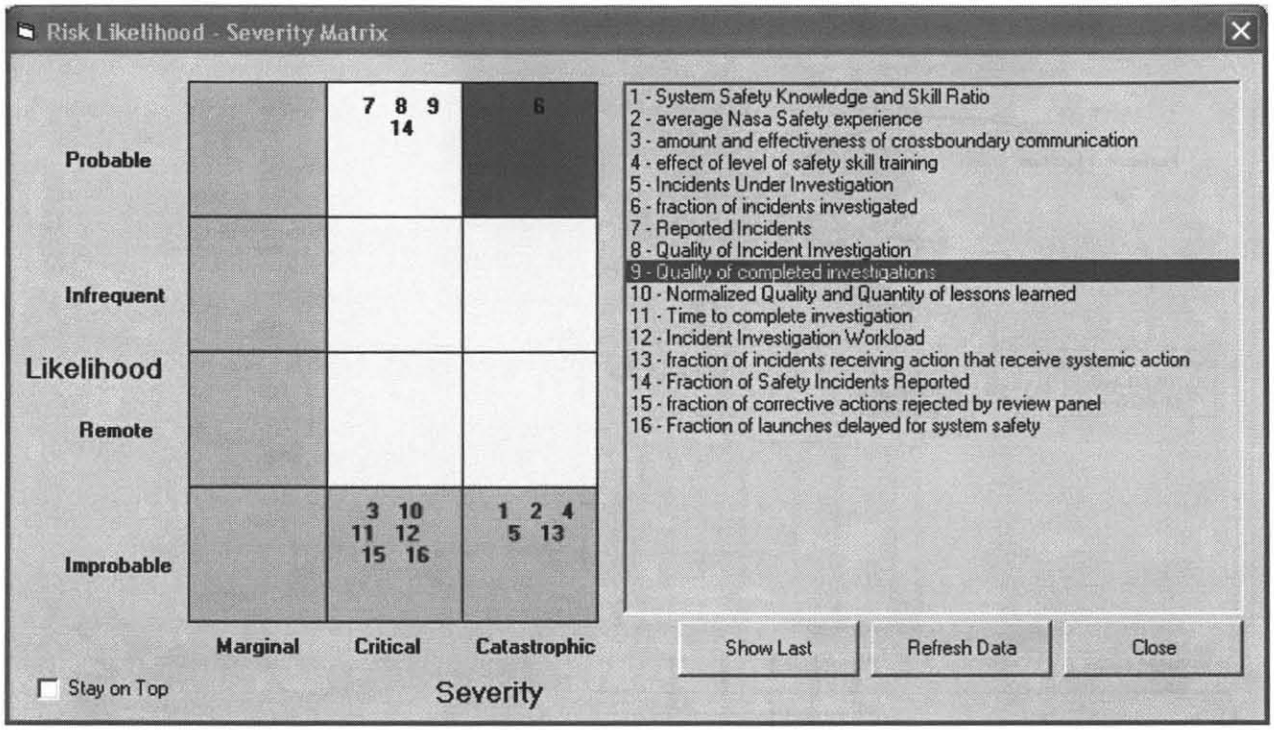



**Figure 4-19 Screen image of risk matrix display**

#### 4.7.1.7 Show model

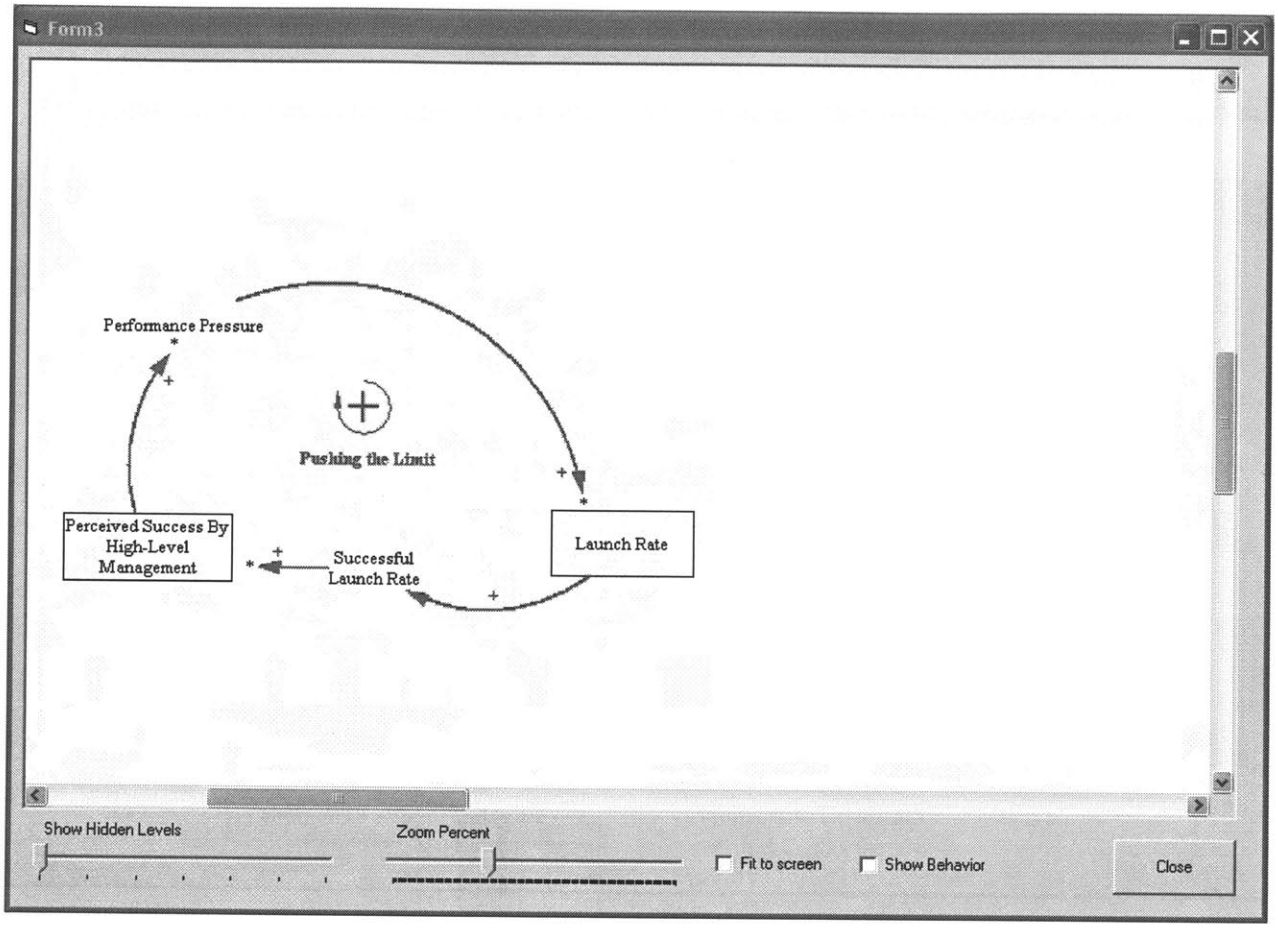



**Figure 4-20 Screen image of high level model with one loop (layer) displayed**

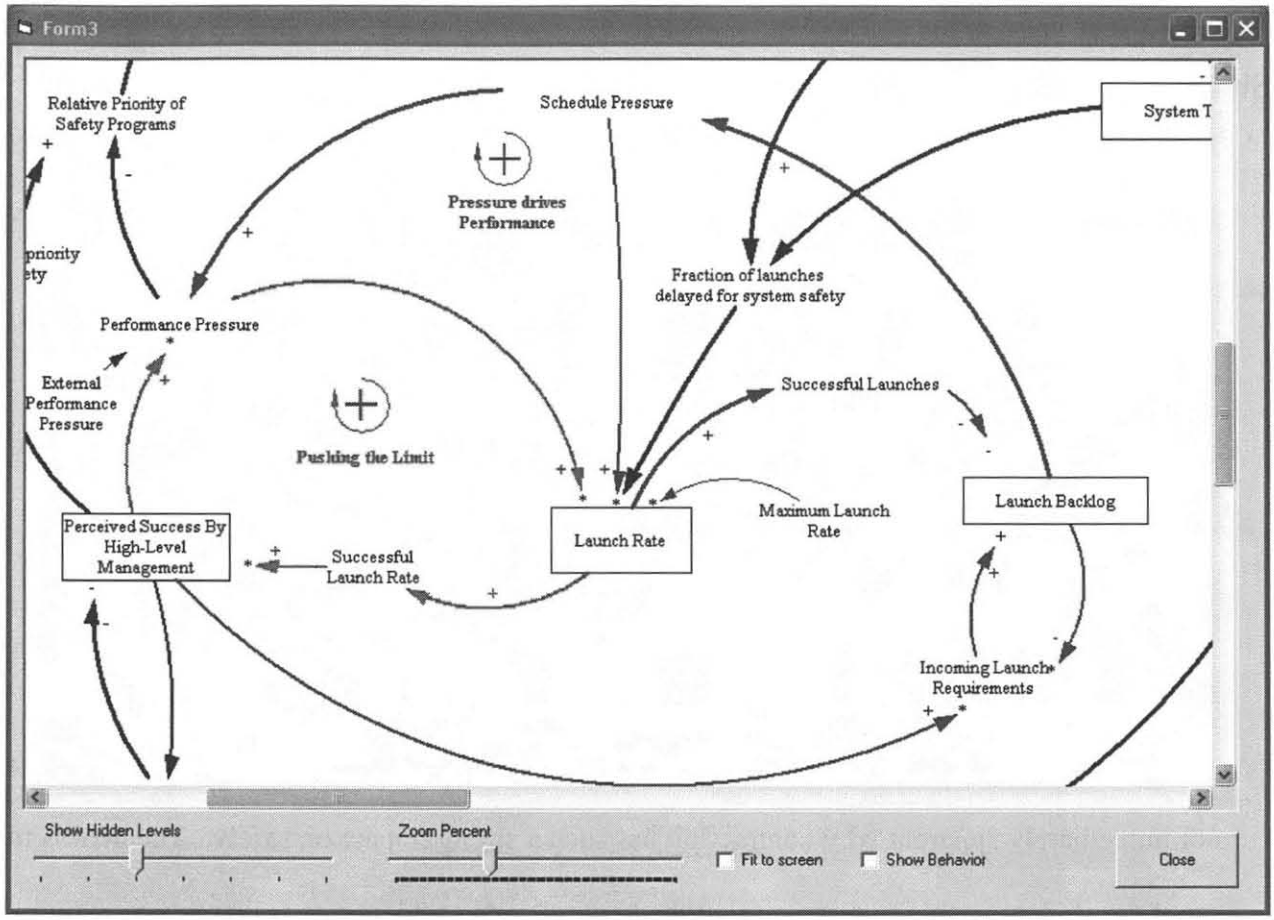



**Figure 4-21 Screen image of high level model with three loops (layers) displayed**

# 5 System Evaluation & Testing

To help determine the effectiveness of this software as well as to get user feedback, three MIT graduate student volunteers were solicited to use the risk management simulator. They were given a short introduction (about 30 minutes) into the basic operation of the tool and a very high overview of the NASA model and system dynamics.

## Sample test scenarios

### Simulating high and low contacting ratios

In this scenario the users are asked to explore the impact of contracting on system safety. They should discover that a high contracting ratio leads to higher risk and an accident sooner than low contracting. The participant is then asked the following:

- Is there a difference between high contracting and low contracting on system safety?

  The student should notice that high contracting results in a higher risk and an accident much sooner than with low contracting

- What is the root cause?

  In both cases (high and low contracting) the total safety staff size (NASA and contractors) and knowledge are about the same (even though individually there are very different), so it is not immediately apparent why contracting has such a strong impact on safety. The goal is to see if the student is able to discover that the high risk and accident rate is driven primarily by NASA's much smaller ability to oversee contracting work ("Nasa ability to perform oversight ratio" in the Vensim model).

### Simulating performance pressure through launch requirements

There are two ways to control the launch rate in the model. The user can change either the yearly launch goal or the administrative safety limit. Think of the former as the goal that is set at the beginning of the year and the latter as a stop check on system safety that can be applied at any time. By setting these values appropriately, one can create nearly identical launch rates (i.e. 2 per year), but with very different safety results. The participant is then asked the following:

- Is there a difference between the two launch scenarios on system safety?

The student should notice that when the desired launch rate is set much lower than the upper limit so that risk is very low and that there are no accidents, however, when the two values are reversed and the desired launch rate is much higher than the max allowed that risk is higher and an accident occurs.

- What is the root cause?

This is a problem that, at first, seems very confusing. The average launch rate is almost the same and all other values are fixed; yet there is a very clear difference in safety. The goal is to see if the student is able to discover that the much greater performance pressure drives the higher risk and accident rate.

## Test results

- The test was administered to three graduate MIT students. From observations and post-test discussion it became clear that the simplified interface of the simulator program still presents the novice student with too much information. The potential for confusion is lessened significantly if the student works through a sample case with the test administrator to gain familiarity with the tool and to learn some investigation techniques. This both shortened the total test time and exposed the student to an additional test scenario on safety dynamics. It also seemed that the student that worked through a sample case first enjoyed the experience more, had greater confidence using the tool and seemed to have greater depth of understanding of the system dynamics.

## Test observations

- Students found it useful and informative to have an overview of the Vensim model and the purpose of the simulation tool, but it did not make it any easier to solve the cases

- It was immensely useful to first work through a problem with a student so that they can see how to use the tool and possible strategies
- For initial training it may be useful to have a simplified display screen with only a few variables and trends so that students can immediately see how the model operates
- Maximum learning occurs through collaboration with a test administrator. The student is able to ask questions about the model, and can get hints if he appears to get stuck.
- The case question approach (i.e., why safety changes with contracting) appears to be a very good method for teaching basic system behaviors and how to use the leading indicators.
- The risk matrix was of no value to the test participants
- The sensitivity analysis was useful at the conclusion of a test during discussion to explain and show the model behavior, but was confusing if used as a diagnostic tool (but this could be due to the students' unfamiliarity with the tool and greater use could change this.)
- One student suggested that he would have found it very beneficial if the tool could indicate somehow (i.e., via color) which variables/leading indicators had the most divergent values for different test runs to help him focus his attention on the most important variables.

## Case-based learning model

Based upon student test results a "case method" can be a very instructive method for teaching system safety concepts with the risk management tool. A suggested flow chart for case-based learning is based upon Figure 7.1 in [18] and is shown below.

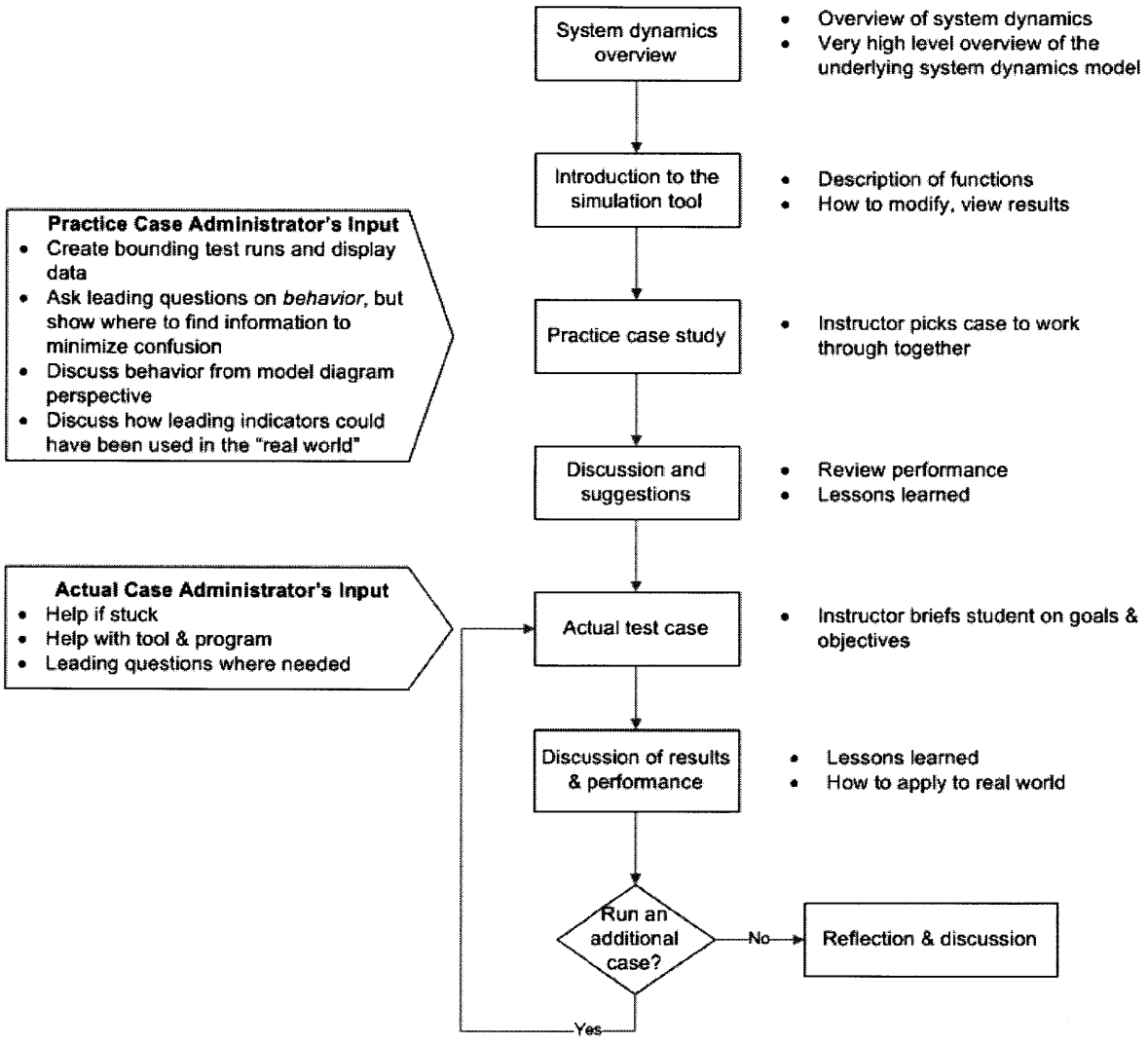



**Figure 5-1 Case-based learning flow chart for teaching users system safety concepts using the risk management tool. (Based upon Figure 7.1 of [18]**

# 6 Conclusions and Recommendations

## 6.1. Discussion of thesis goals

Test takers confirmed that they found the simplified model diagram useful for visualizing the basic system loops and interactions. Color-coding the loops and a adding loops one step at a time also helped with their comprehension.

### Enable a user unfamiliar with either system dynamics or Vensim to be able to explore and learn from the model dynamics.

The user interface was designed to display model variables and user controls in a familiar Microsoft Windows context. While not a design goal – the fact that they found the simulation "fun" kept them engaged and interested as well. It was originally thought that the tool would be given to a user with a very general goal such as "minimize risk without reducing launch rate", but preliminary testing suggests that this goal is too advanced for a novice. The simplified interface still has too much information and it is too easy to become overwhelmed. On the other hand, the case method appears to be very instructive for creating a well-defined problem that leads to an understanding of basic safety concepts as well as a springboard for further discussion.

## 6.2. Conclusions

It appears that practically every high-risk endeavor that works with complex systems has some kind of training or modeling simulator available. From flight simulators to product marketing simulations, if a bad decision can lead to the loss of life or severe financial loss, there is likely to be a simulator available to help train people how to make the right decision. This is especially true at NASA, where it seems that practically every evolution, from maintenance to launch procedures is practiced repeatedly to minimize any chance of a mistake. With one glaring exception: there is presently no simulator available to help managers to make informed

decisions to minimize system risk. The goal of this thesis is to bridge this gap and bring risk simulation programmatic decision-making and resource allocation. Based upon the student testing and feedback the software application has made significant inroads towards developing a dynamic risk management flight simulator. Students reported that after using the software for only a short period they had a better intuitive understanding of the system dynamics underlying the safety culture at NASA. For example, when working through the contractor case, none of the students had previously considered the importance of "contractor oversight" on safety when increasing the number of contractors involved with a project. This was in spite of the fact that they were experienced engineers with an average work experience of about 10 years each. With further development (discussed below) as well as additional user testing with NASA managers the risk management simulator can significantly help to train people about systemic risk and provide NASA with a powerful tool for risk analysis.

## 6.3. Suggested future work

### 6.3.1. Calibrate the model

At present, the model is entirely theoretical based upon our understanding of NASA, its policies and procedures and staff interviews. However, as with all computer models, there are numerous assumptions with respect to the size and scale of input variables as well as the model response to some inputs. While this may not change the overall behavior, the lack of "real numbers" will cause NASA managers to be distrustful of any model results. Using real historical values and comparing the model results to the known data for the same time and values can rectify this. This is called, "calibrating a model" and results in a model that can accurately predict past

events. As with all models, the ability to predict future events off of past data is dubious[6], but the greater correlation with real values would still be an asset.

### 6.3.2. Dynamic risk matrix "what if" analysis

It is our hope that the likelihood-severity matrix will be a popular tool by NASA. Late in its development, it was realized how eminently useful it would be if a user could dynamically explore the actions available to reduce the likelihood of a particular indicator (recall that the severity is fixed in this model) i.e., if a particular indicator had a high likelihood the user should be able to use a slider bar or other tool to modify one or more model variables to see which one(s) can reduce the likelihood. Right now a user can either make a change and step forward in time (which is a different risk calculation since time is modified), or he can stop and restart the simulation and modify one or more values to see how the matrix has changed. Neither one of these is very dynamic and make it hard for user to explore in real-time e.g.,, "...OK, I'm in a pickle with this variable... let's see what I can move around to reduce this likelihood"

### 6.3.3. Live on-line risk management simulator

Once the model is calibrated the next logical step would be to provide a direct import of real data in near real-time. As opposed to a training simulator, NASA managers could potentially have an on-line risk management tool using real data. Not only would NASA personnel have greater trust and faith in a system using their data, but also the promise of a risk management flight simulator with real data would fundamentally change the safety decision process for NASA management and hopefully help to prevent future space flight disasters.

[6] As the late sports commentator Yogi Berra would say, " It's tough to make predictions, especially about the future"

# References


1 Leveson, N. (2004) "A New Accident Model for Engineering Safer Systems," Safety Science, 42 :4, pp. 237–270.

2 Leveson, N.G., Dulac, N. with contributions by Cutcher-Gershenfeld, J., Carroll, J., Barrett, B and Friedenthal, S., (2005) http://sunnyday.mit.edu/ITA-Risk-Analysis.doc

3 Perrow, C., *Normal Accidents: Living with High-Risk Technology*, Basic Books, New York. (1984)

4 Sterman, J.D. (2000) *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Mc-Graw Hill/Irwin.

5 Leveson, N.G., Daouk, M., Dulac, N., and Marais, K. (2003) Applying STAMP in Accident Analysis, *Workshop on the Investigation and Reporting of Accidents*

6 Leveson, N.G. (1995) *Safeware: System Safety and Computers*. Addison Wesley.

7 Leveson, N.G., Dulac,N. Zipkin,D., Cutcher-Gershenfeld, J., Barrett, B. and Carroll, J., *Modeling, Analyzing, and Engineering NASA's Safety Culture* by Final Report of a Phase 1 NASA/USRA research grant (http://sunnyday.mit.edu/Phase1-Final-Report.pdf )

8 Aldrich, C., *Simulations and the future of learning : an innovative (and perhaps revolutionary) approach to e-learning* San Francisco, Pfeiffer,San Francisco, CA, 2004

9 Roberts, B., Bennet, R., *Risk Management for the NASA/JPL Genesis Mission: A Case Study*, 10th Annual International Symposium of the International Council on Systems Engineering, Minneapolis, Minnesota, June 16-20, 2000 (http://www.futron.com/pdf/INCOSE2000.pdf)

10 Roberts, B., *Integrated Risk Assessment: Results and Lessons Learned*, Proceedings of the Risk Management Symposium sponsored by the USAF SMC and the Aerospace Corporation, February 8-11, 1999. (http://www.futron.com/pdf/aerospace.pdf)

11 Greenfield, M., *Risk Management "Risk as a Resource"*, Presented to the NASA Langley Research Center, May 1998 by Dr. Michael Greenfield, Former Deputy Associate Administrator NASA Office of Safety and Mission Assurance (http://www.hq.nasa.gov/office/codeq/risk/risk.pdf )

12 Cockrell, *Managing Risk Continuous Risk Management at NASA*, Presented at the NASA Risk Management Conference V, NASA Assurance Quality Center, October 27, 2004 (http://rmc.nasa.gov/archive/rmc_v/presentations/cockrell%20managing%20risk.pdf)

13 Hale, W., *Managing Space Shuttle Program Risk for Return to Flight: Why does risk management fail?* Presented at the NASA Risk Management Conference VI, Lake Buena Vista, FL, December 6-8, 2005 (http://rmc.nasa.gov/presentations/Hale_Managing_Shuttle_Program_Risk_For_Return_To_Flight.pdf)

14 Turner, J., *A Risk Manager's Perspective: Lessons Learned for Future Exploration Systems* Presented at the NASA Risk Management Conference VI, Lake Buena Vista, FL, December 6-8, 2005 (http://rmc.nasa.gov/presentations/Turner_Risk_Mgrs_Perspective_LL_Future_Exploration_Systems.pdf)

15 Curiel, P., *Applied Lessons Learned on ARM Implementation for CEV,* Presented at the NASA Risk Management Conference VI, Lake Buena Vista, FL, December 6-8, 2005 http://rmc.nasa.gov/presentations/Curiel_Lessons_Learned_ARM_Implementation_for_CEV.pdf

16 Perera, Sebastian J, *Integrated Risk Management Application (IRMA) Overview/Update*, Presented at the NASA Risk Management Conference V, NASA Assurance Quality Center, October 27, 2004 http://rmc.nasa.gov/archive/rmc_v/presentations/perera%20irma%20overview.pdf

17 Calhoun, C., *RMIT Risk Management Implementation Tool*, Presented at the NASA Risk Management Conference V, NASA Assurance Quality Center, October 27, 2004 http://rmc.nasa.gov/archive/rmc_v/presentations/calhoun%20rmit.pdf

18 Fripp, J, *Learning Through Simulations A Guide to the Design and Use of Simulations in Business and Education*, McGraw Hill, Berkshire, England 1993

19 Kolb, D, Rubin, I., Osland, J., *Organizational Behavior: An experiential Approach*, 5th edition,1991, p 59, Prentice Hall, Englewood Cliffs, New Jersey,.

# Appendices

# A.1. System Dynamics Model Sub Model Views

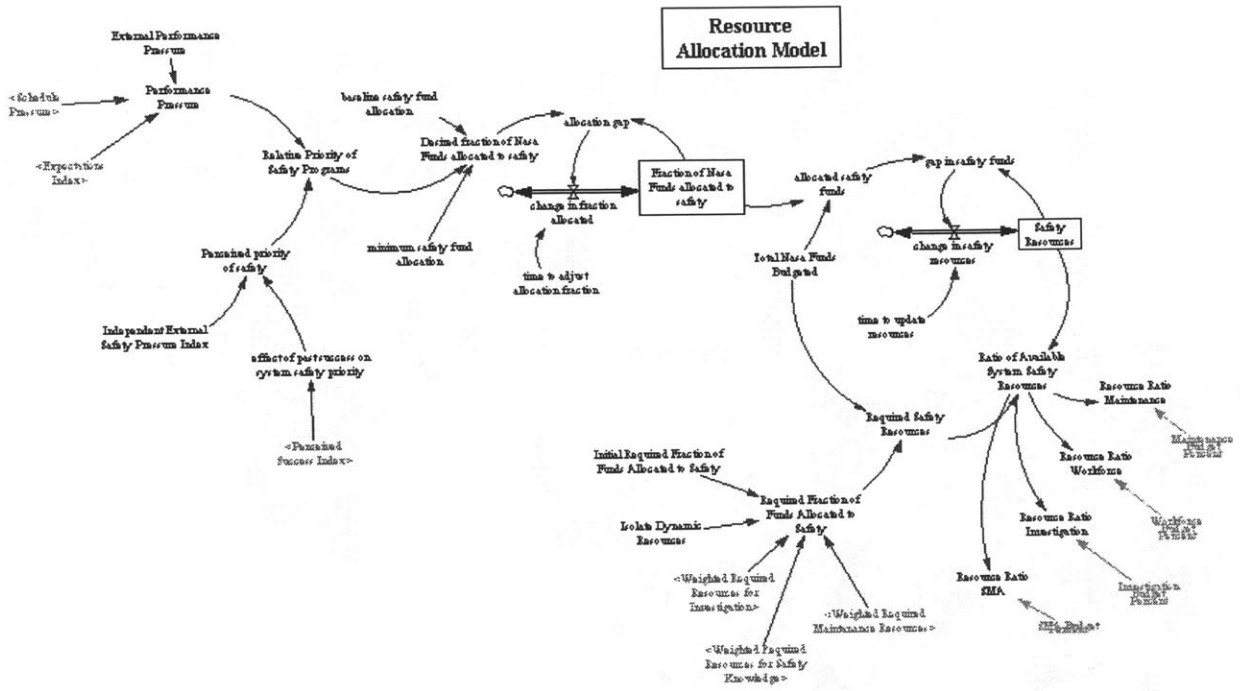



**Figure A.1.1 Resource allocation sub model**

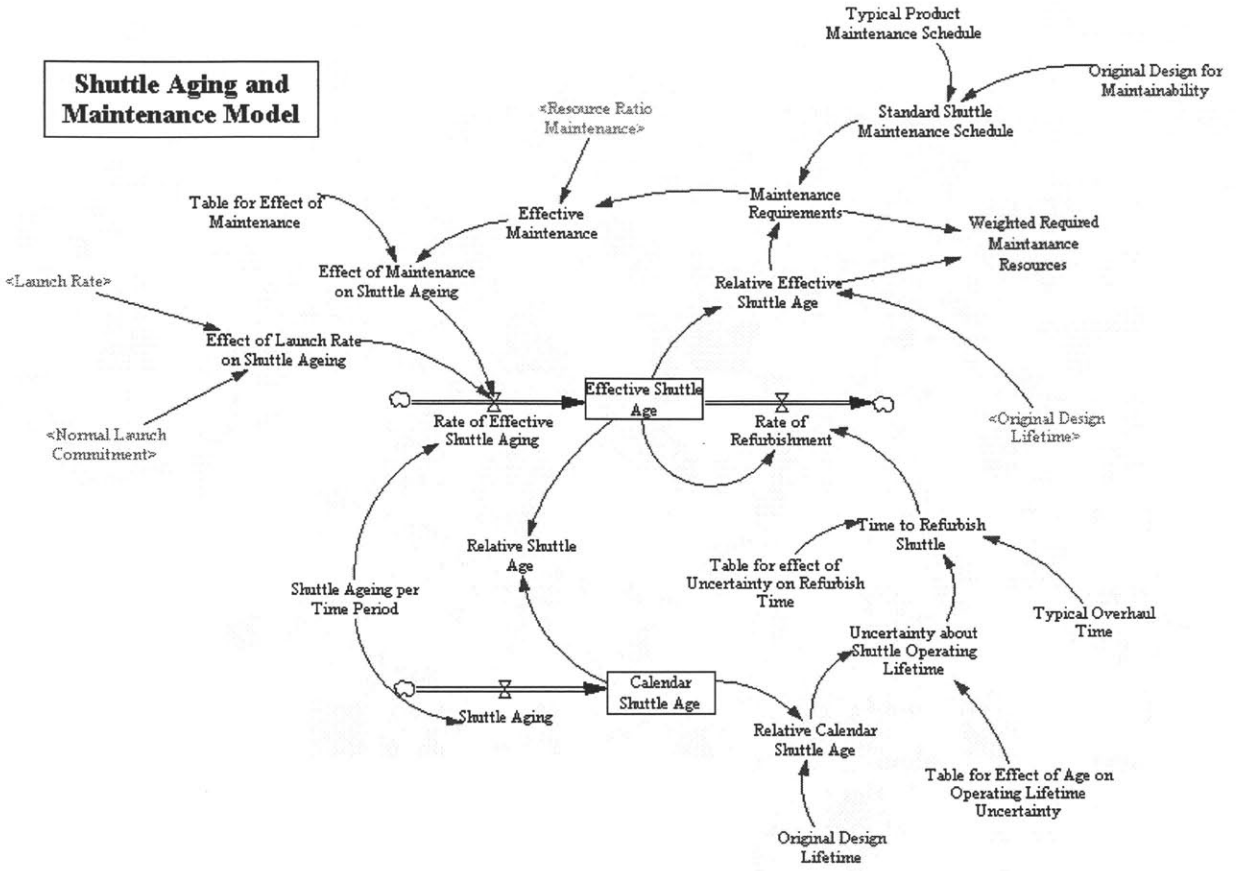



**Figure A.1.2 Shuttle aging and maintenance sub model**

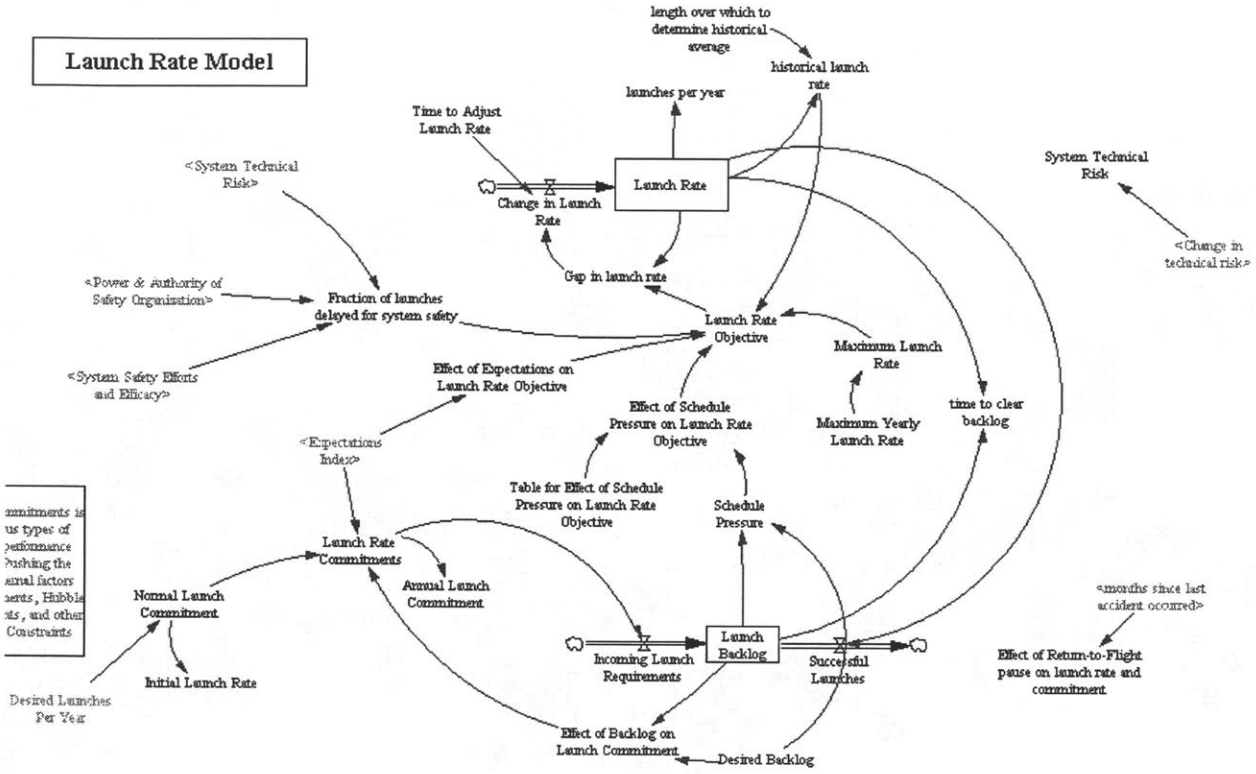



**Figure A.1.3 Launch rate sub model**

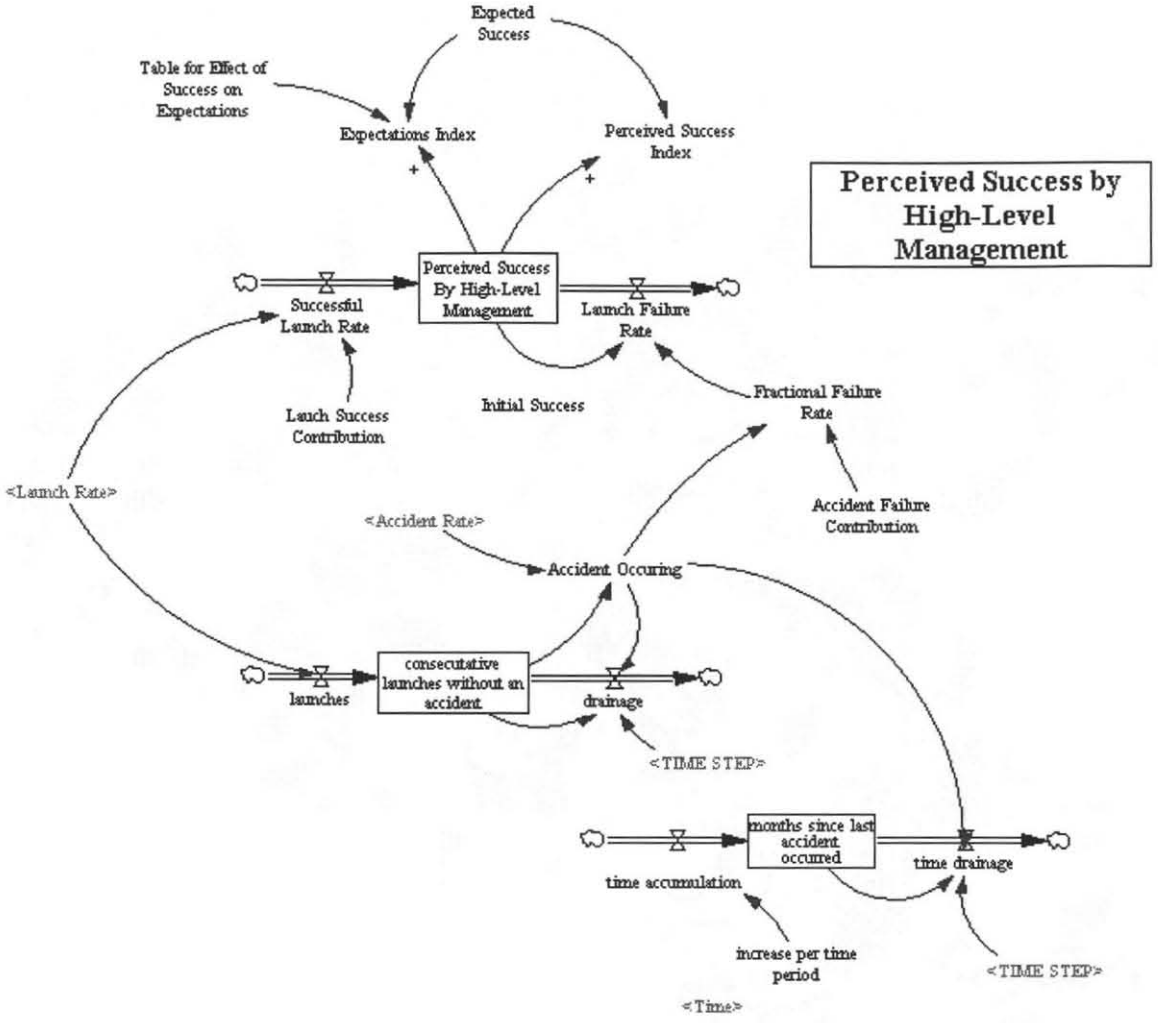



**Figure A.1.4 Perceived success by high-level management sub model**

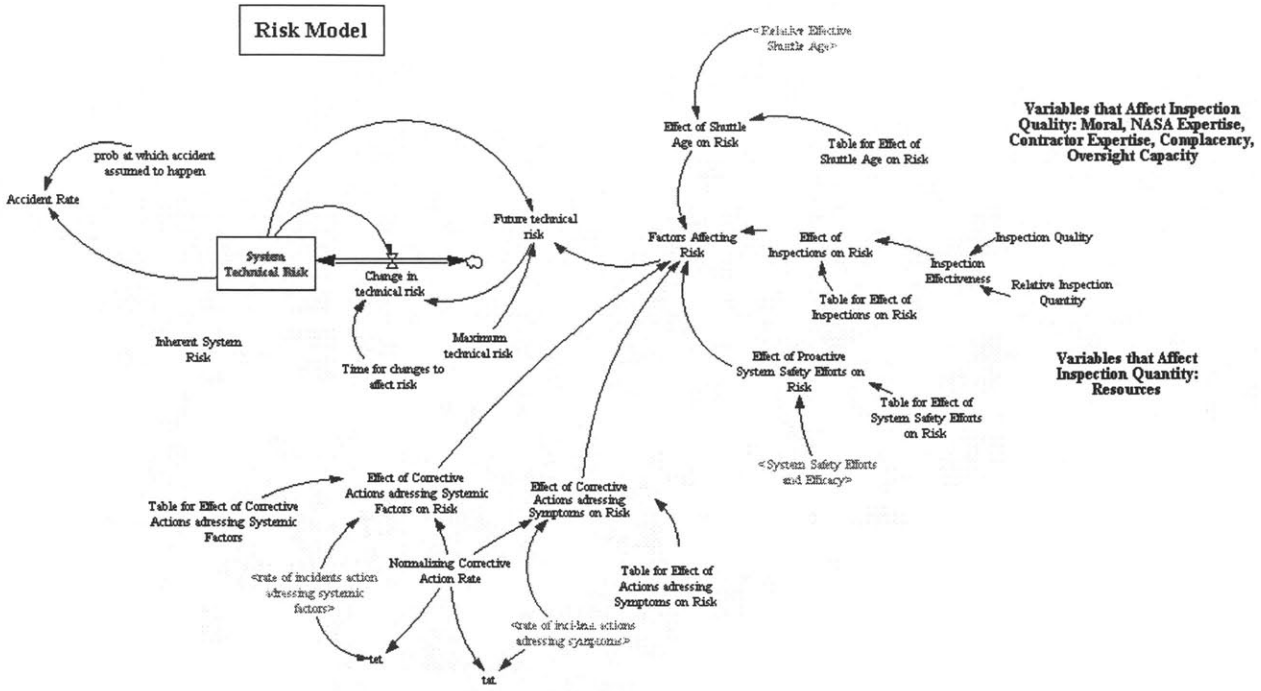



**Figure A.1.5 System technical risk sub model**

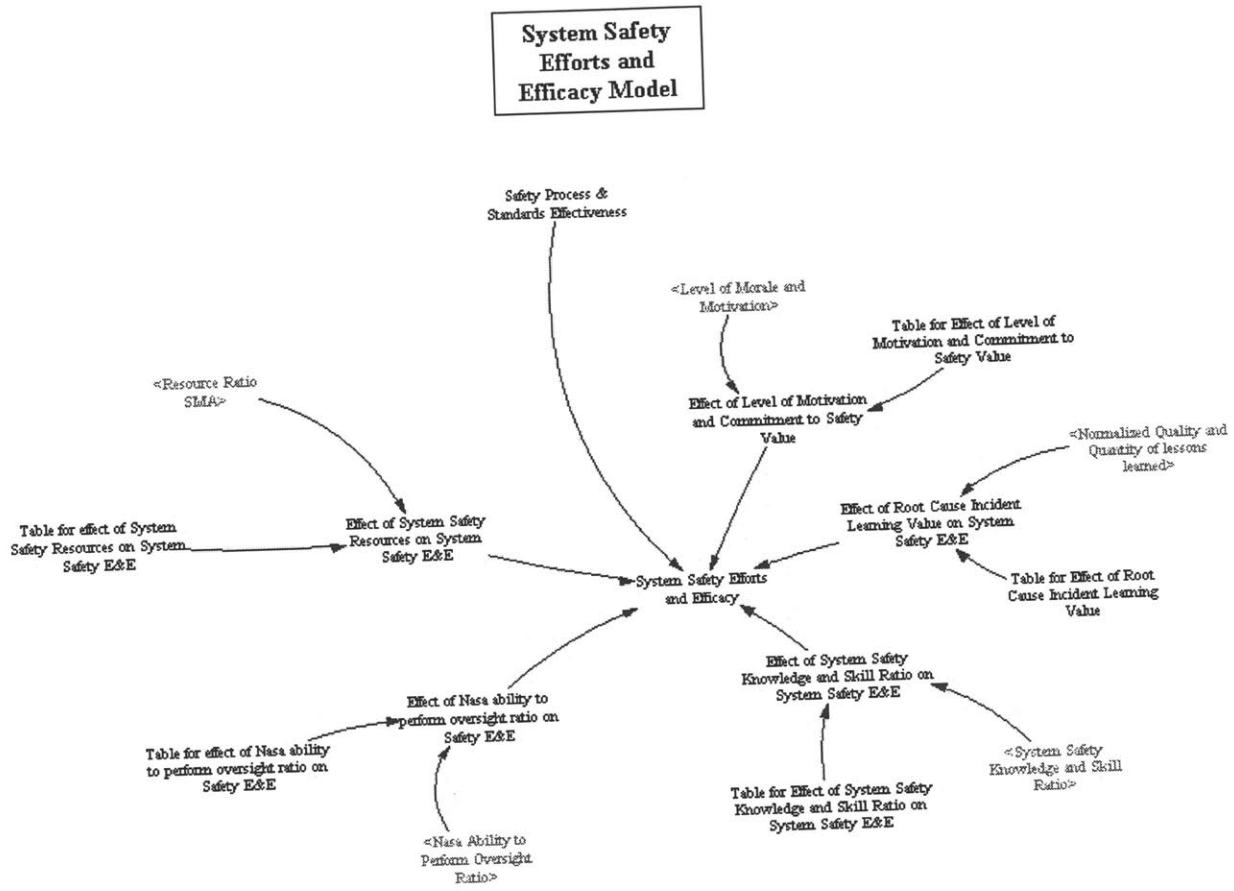



**Figure A.1.6 System safety efforts and efficacy sub model**

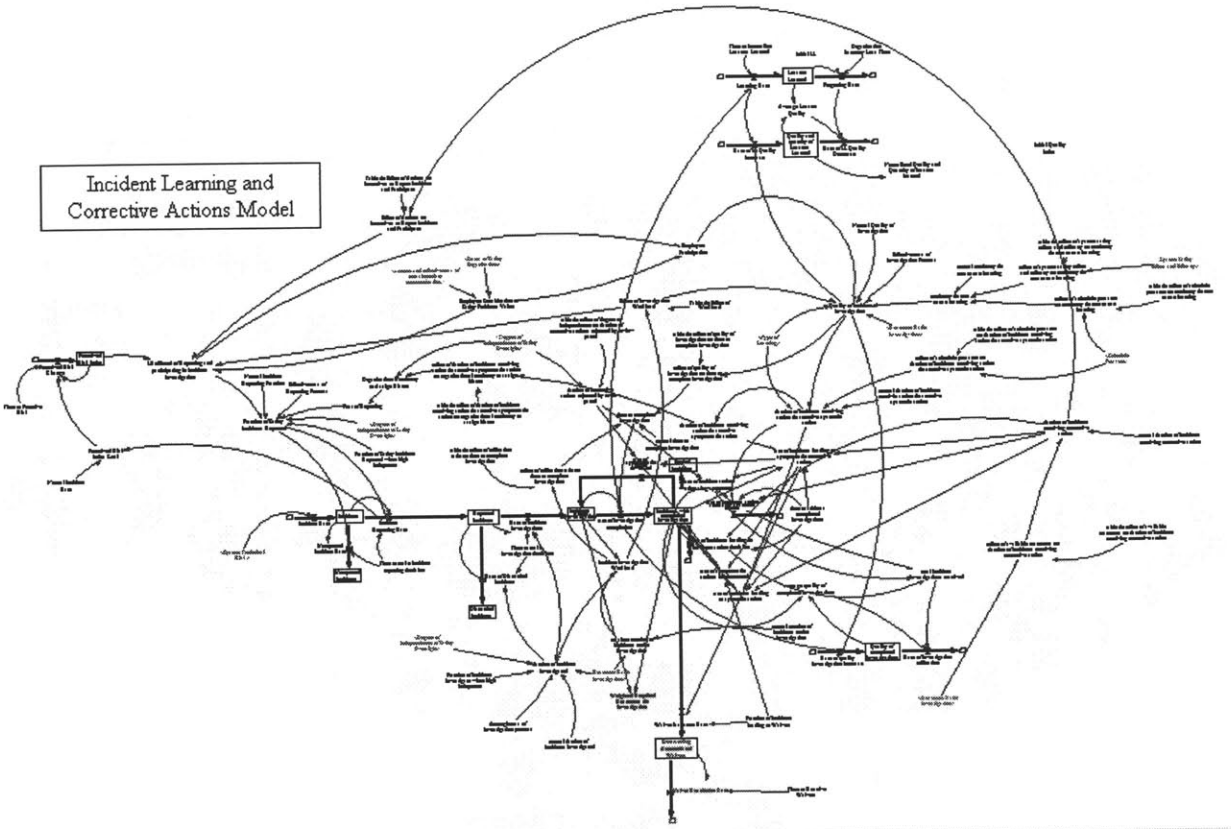



**Figure A.1.7 Incident learning sub model**

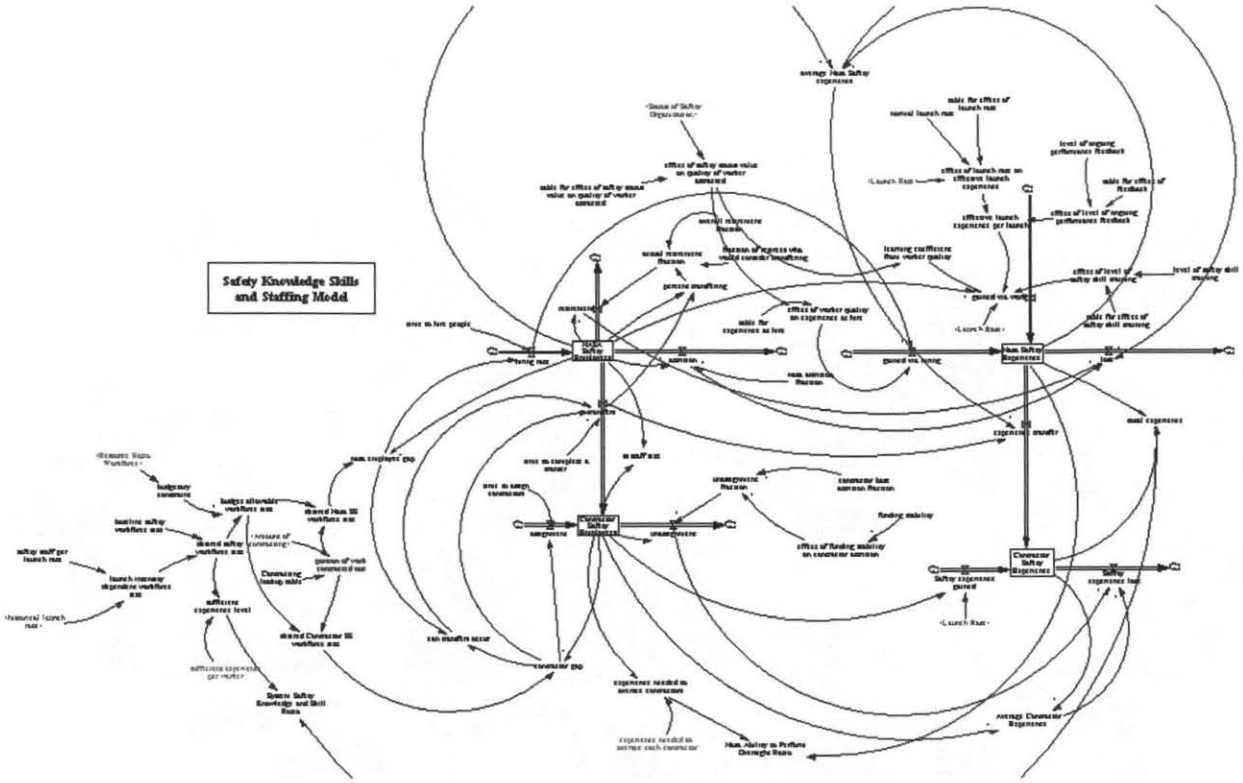



**Figure A.1.8 Safety knowledge, skills and staffing sub model**

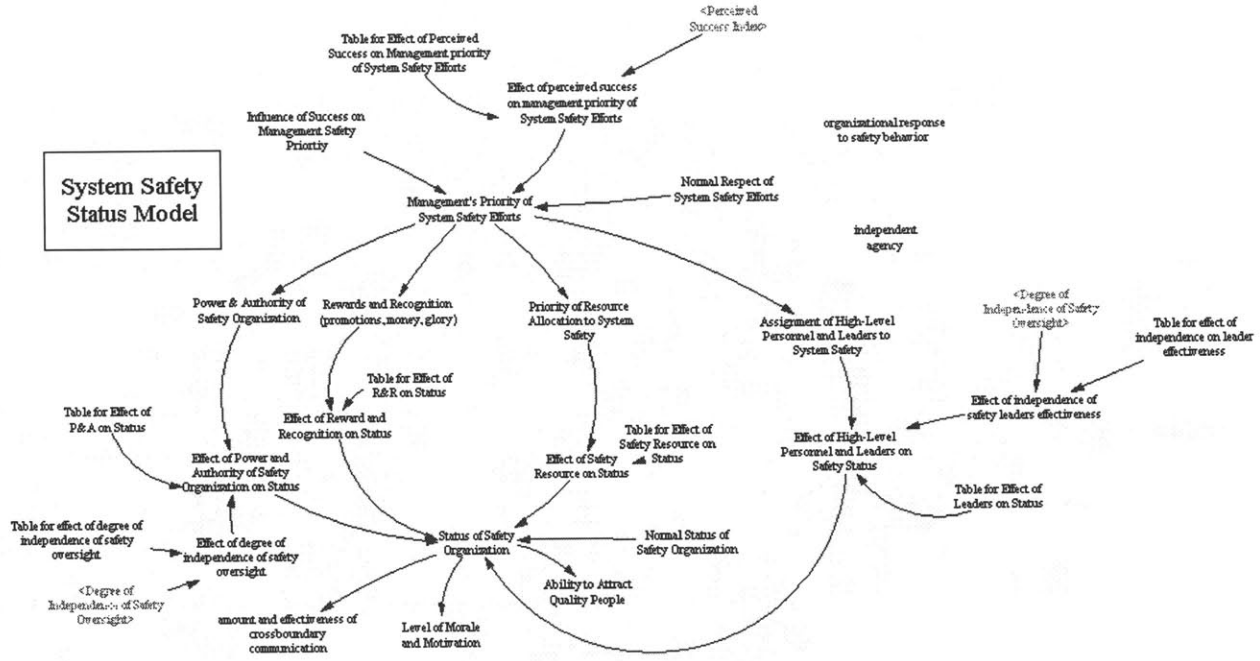



**Figure A.1.9 System safety status sub model**

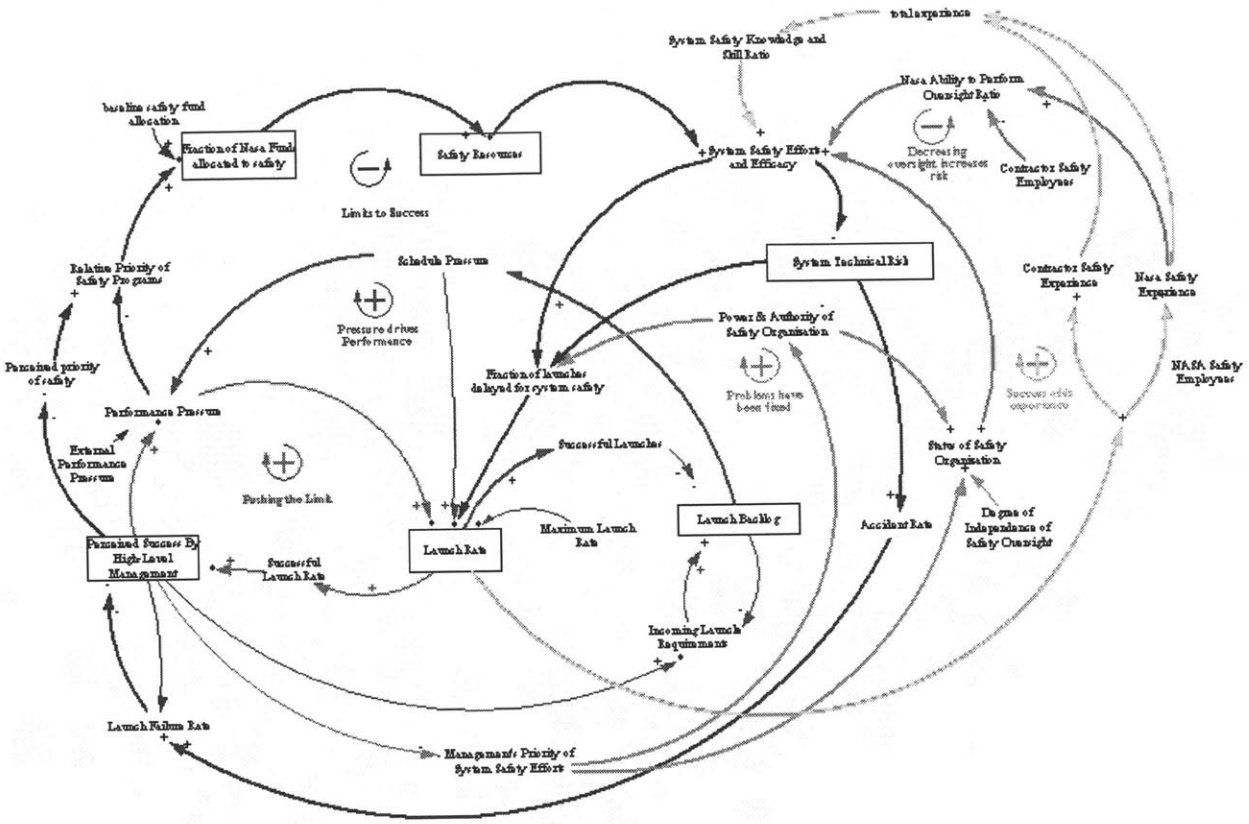



**Figure A.1.10 Simplified overview**

# A.2. Manned Space Program Risk Management Tool User Manual

## A.2.1 Introduction

Thank you for using the Manned Space Program Risk Management tool.

This application provides an easy-to-use interface to a system model developed in Vensim ®, which simulates the NASA safety culture prior to the Columbia Shuttle accident in 2003. With this software you are able to make resource allocation and mission decisions throughout a twelve-year time span and evaluate their impact on key performance and leading risk indicators. You can use this application in a variety of ways. For example, you may just want to explore the basic system dynamics, or you may have a specific goal, such as how to maximize the shuttle launch rate while minimizing risk and the possibility of an accident. Regardless, we hope that you find the application educational, fun and useful.

This user manual is broken down into the following sections:

**Getting started** – Program prerequisites, system configuration and starting a simulation

**Running a simulation** – How to load and run a simulation and evaluate the results

**Modifying the Vensim model** – How to modify the underlying Vensim model

**Advanced topics** – How to use the scenario database to create new simulation scenarios as well as customized user instructions.

**Trouble-Shooting** – Some common problems and their solution

## A.2.2 Getting Started

### Prerequisites

#### *Vensim DSS*

The application was developed with the full version of Vensim ("Vensim DSS") using the free MIT license for academic use. It has not been tested with other versions of Vensim (i.e., Vensim PLE).

#### *Minimum 1024x768 display screen*

The minimum display resolution is 1024x768, but 1280x1024 is preferred.

#### *Microsoft Windows and required Windows components*

This application was developed to run on the Microsoft Windows operating system. It was designed and tested on Windows XP Professional Service Pack 2. You do not need a license for Microsoft Access ® to read and write to the scenario database, but you will need to have the Microsoft Data Access Components (MDAC) v2.7 or above. MDAC is installed a part of many other software applications, such as Microsoft Access. The MDAC install is included with the simulation setup CD and is available at:

http://msdn.microsoft.com/data/mdac/downloads/default.aspx

### Starting the program

The installation program will create a program directory and an entry on your Windows Start menu. Navigate to the Risk Management Tool folder and select the "Safety Game" icon to start the program.

### Overview of the risk management tool

The risk management tool provides a simplified user interface to a system dynamics model developed in Vensim. This main program screen is shown below.

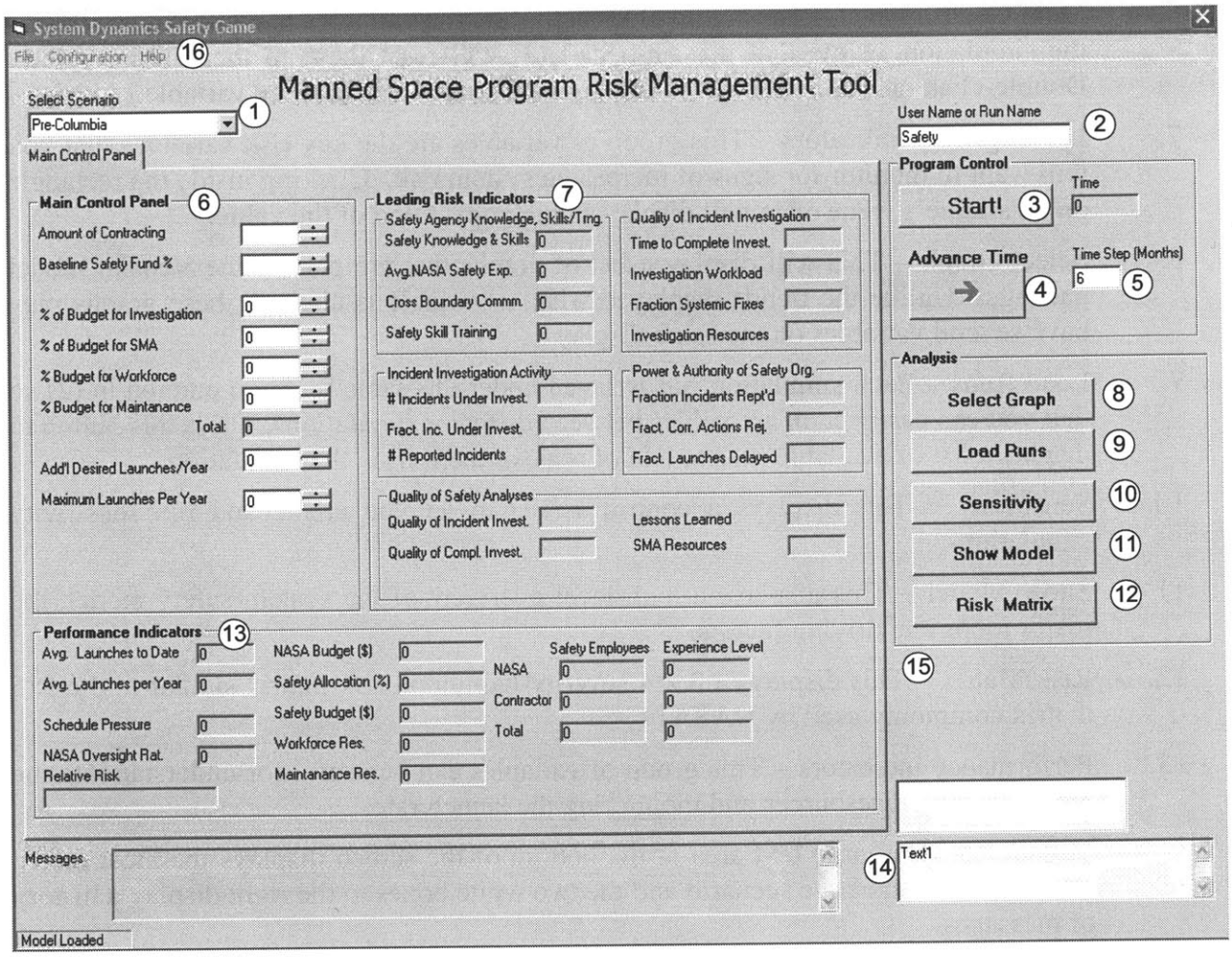



**Figure A.2-1 Main program screen of the risk management tool**

Program Functions

1. Select Scenario – Select the scenario for the simulation. A scenario will automatically modify simulation values (budget, launch requirements, and so forth) to mimic changing external factors. Choose, "Static Resources" to for no changes
2. User name or Run Name – the descriptive name ("run name") for a simulation. It must be entered before pressing Start. As you progress through the simulation all calculated results are stored in a ".vdf" file with the specific run name, e.g., "SampleRun.vdf"
3. Start – Press start to start a new simulation. You will be prompted to select additional runs to load at this time.
4. Advance Time – Advances the simulation time by the number of months in (5).
5. Time Step – The number of months to advance a simulation. Default is (6).

6. Main Control Panel – The eight values below are the variables that you control during the simulation. Note that pressing Start (3) will reset these to their initial values. Double-click on any text block to display an historical trend of the variable.
7. Leading Risk Indicators – This group of variables are the key risk variables that you will want to monitor for signs of increasing system risk. Clicking inside the rectangle on a variable's value once will display an historical trend of the value.
8. Select Graph – This will display a list of graphs that are part of the Vensim model package. Unlike the trends displayed when a variable is clicked, these graphs may have several variables on the same display.
9. Load Runs – Each simulation run is saved under the name that you entered in (2) so that you can easily compare values between different simulations. Press this button to display a list of available runs to add or remove them from the simulation.
10. Sensitivity – This displays a control screen to let you create and run sensitivity simulations.
11. Show Model – This displays a high-level overview of the system safety model, the major loops and driving factors.
12. Risk Matrix – This displays a 3 x 4 severity-likelihood risk matrix similar to the 5x5 matrix commonly used by NASA
13. Performance Indicators – This group of variables can be useful for understanding the model behavior, resources and monitoring the launch rate.
14. Messages – The large text area at the bottom of the screen displays the most recent message from an active scenario and the two white boxes to the right display a history of messages.
15. If the model predicts a shuttle accident the event is displayed here.
16. File Menu –
    a. File, then Exit – Exits the application
    b. Configuration – select the Vensim model and the scenario database locations
    c. Help – Displays instructions from the database

## System configuration

### Select Vensim models

From the File menu (16) select *Configuration*, then Select *Model*. There are two Vensim models that need to be loaded. The main model, "SafetyGame.vpm" is the model used throughout the simulation. If you get an error message when starting the program that it can't find the model

then you may need to update the simulation with the correct file location. The new location is saved in the .INI file when the program closes so that you will not need to do this again.

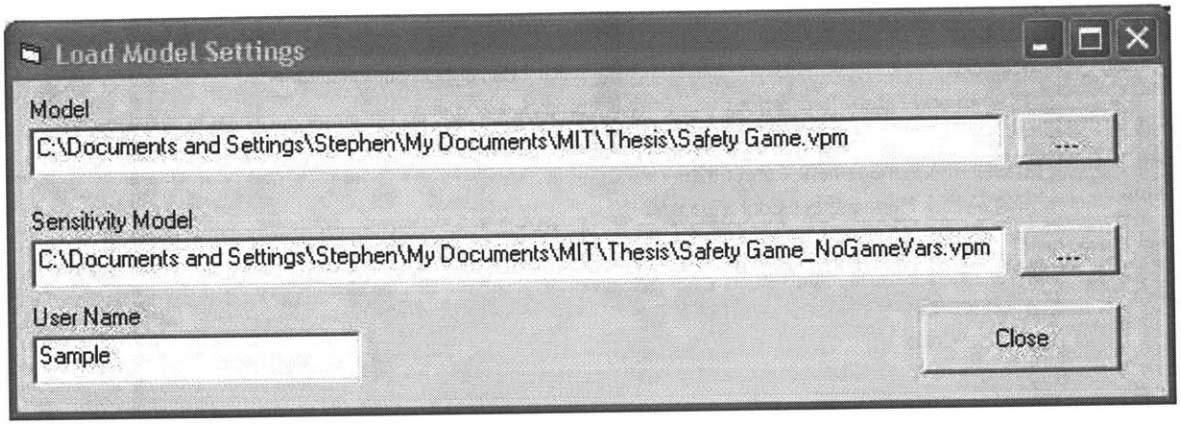



**Figure A.2-2 Load model dialog**

To browse and select a different location press the ellipses button to the right and browse and select the appropriate model as shown below.

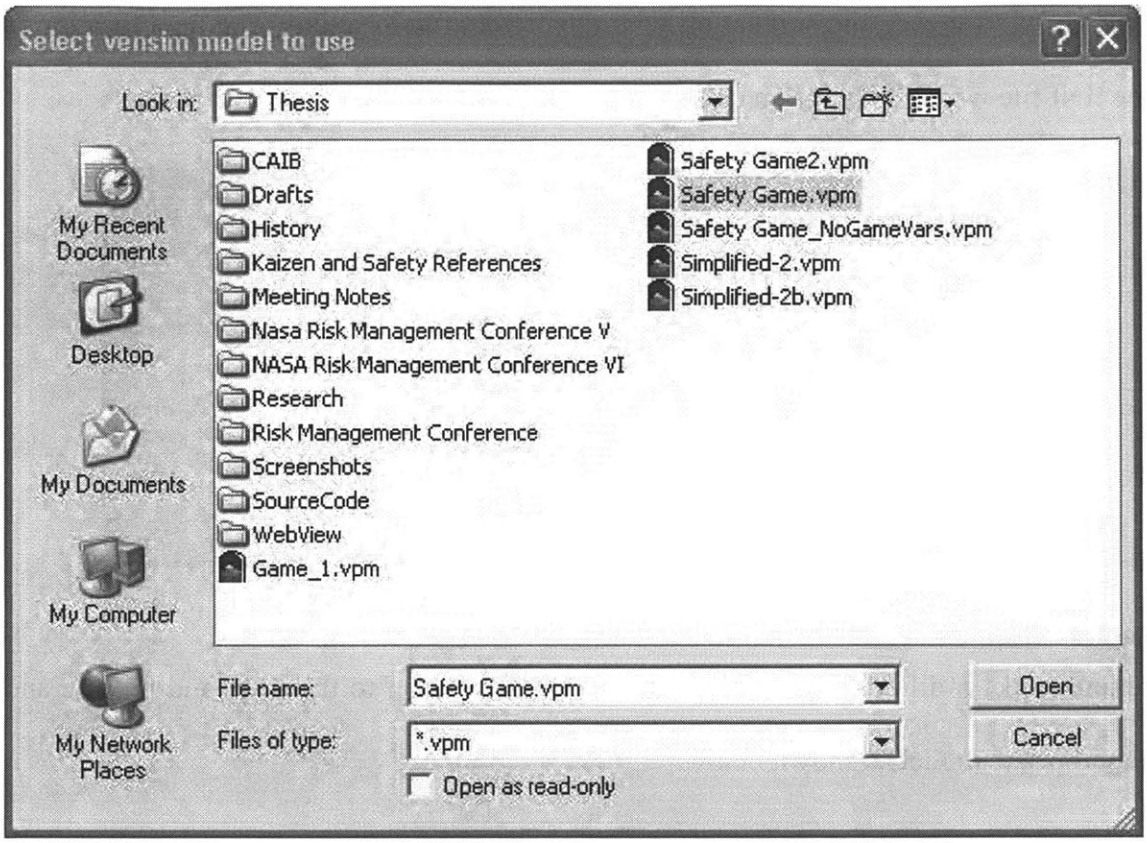



**Figure A.2-3 Browse and select model dialog**

The second option in the configuration screen is the sensitivity model location. This is only needed if you will be performing sensitivity analyses. This model structure is the same as the simulation model , but all of the "game variables" have been converted to constants. This is a limitation of Vensim: it is unable to run a sensitivity simulation with game variables.

### Select scenario database

If you get an error that the simulation tool is unable to locate the scenario database then you need to browse and select the scenario database "Safety Game.mdb" file.

### INI File

The "SafetyGame.ini" file, is in the same directory as the simulation program and stores system and user configuration settings. The SafetyGame.ini file values are read when the program is

started and any changes to the parameters are written when the program closes. If this file is missing the simulation program will automatically create a new one with default values.

### Show model

Select this button to display a simplified view of the system dynamics model. This view has several different "layers" with each level adding another color-coded loop to show the basic model behavior and driving loops.

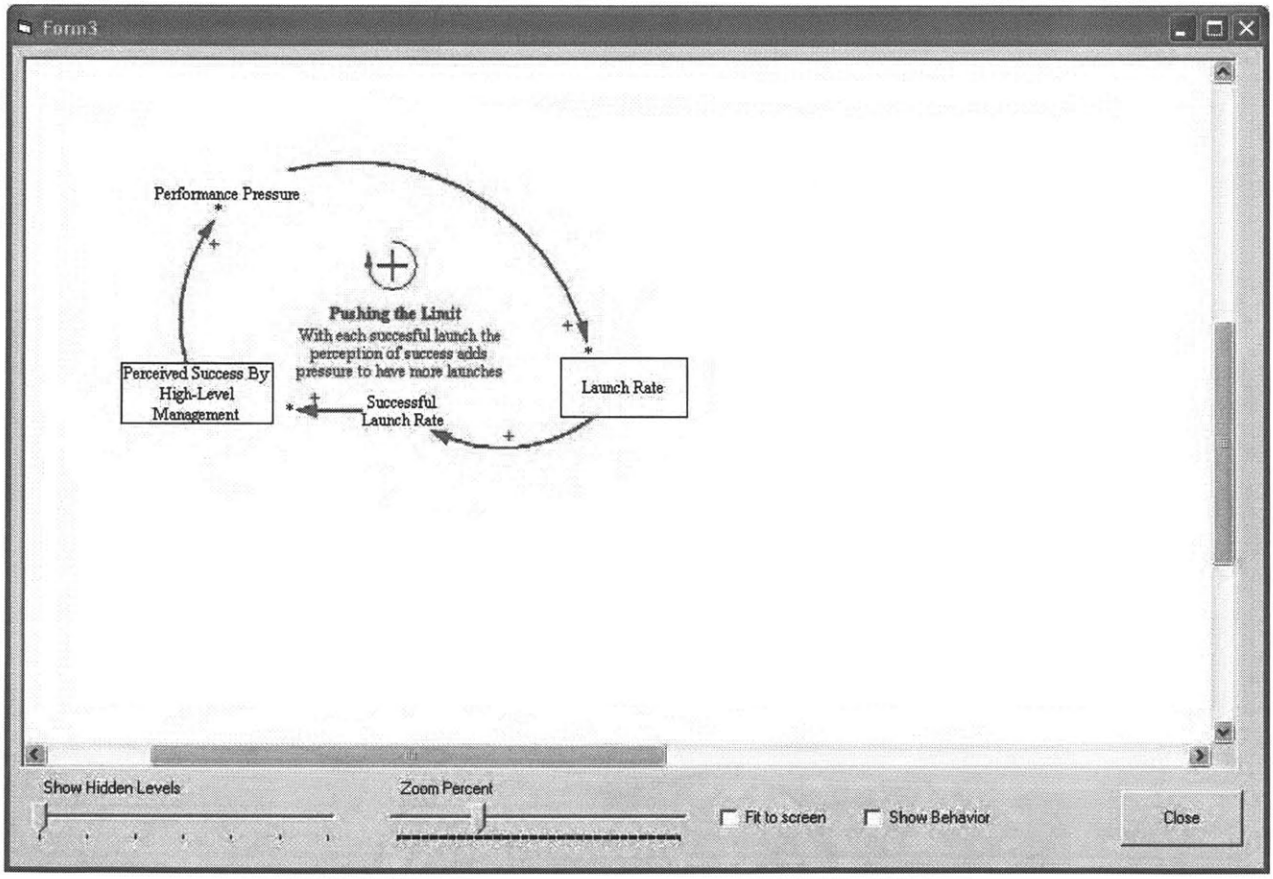



**Figure A.2-4 High level overview of the system dynamics model**

When first opened, the model will only show a single positive feedback loop titled, "Pushing the Limit". This shows how the perception of success by high-level management increases with each successful launch. Moving the "Show Hidden Levels" slider to the right adds new loops until the complete model is displayed as shown below. The "Zoom Percent" slider control lets

you zoom in and out of the display as needed, or you can use the "Fit to Screen" checkbox to display the whole model within the window. (Note: When "Fit to Screen" is checked the zoom slider is disabled.) Checking "Show Behavior" will display a trend of all loaded runs for each variable within each variable box in the window. For example, the screen shot below shows the behavior with two scenarios loaded: high yearly launch rate (blue line) and low yearly launch rate (red line).

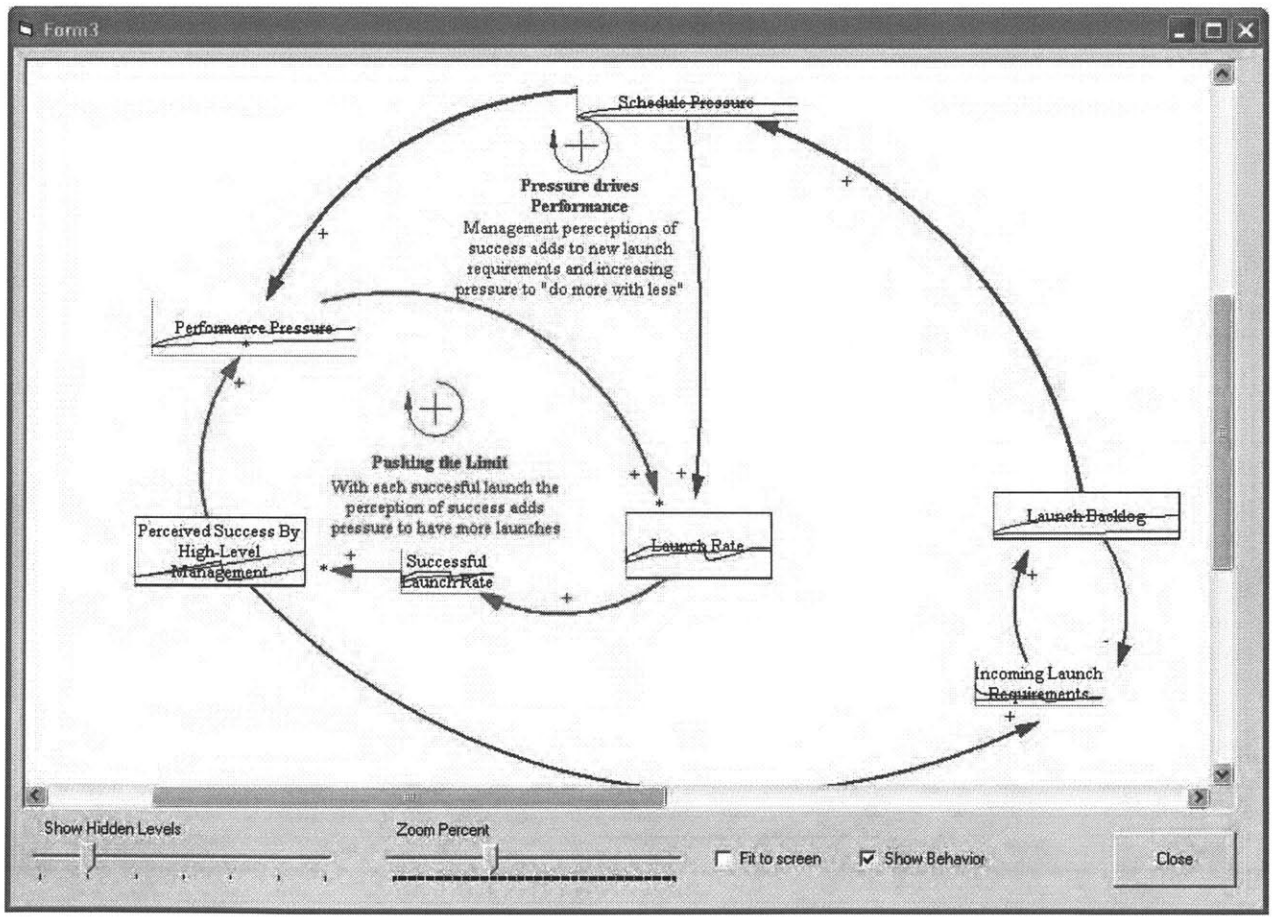



**Figure A.2-5 High level overview of the system dynamics model with two layers displayed and *show behavior* enabled**

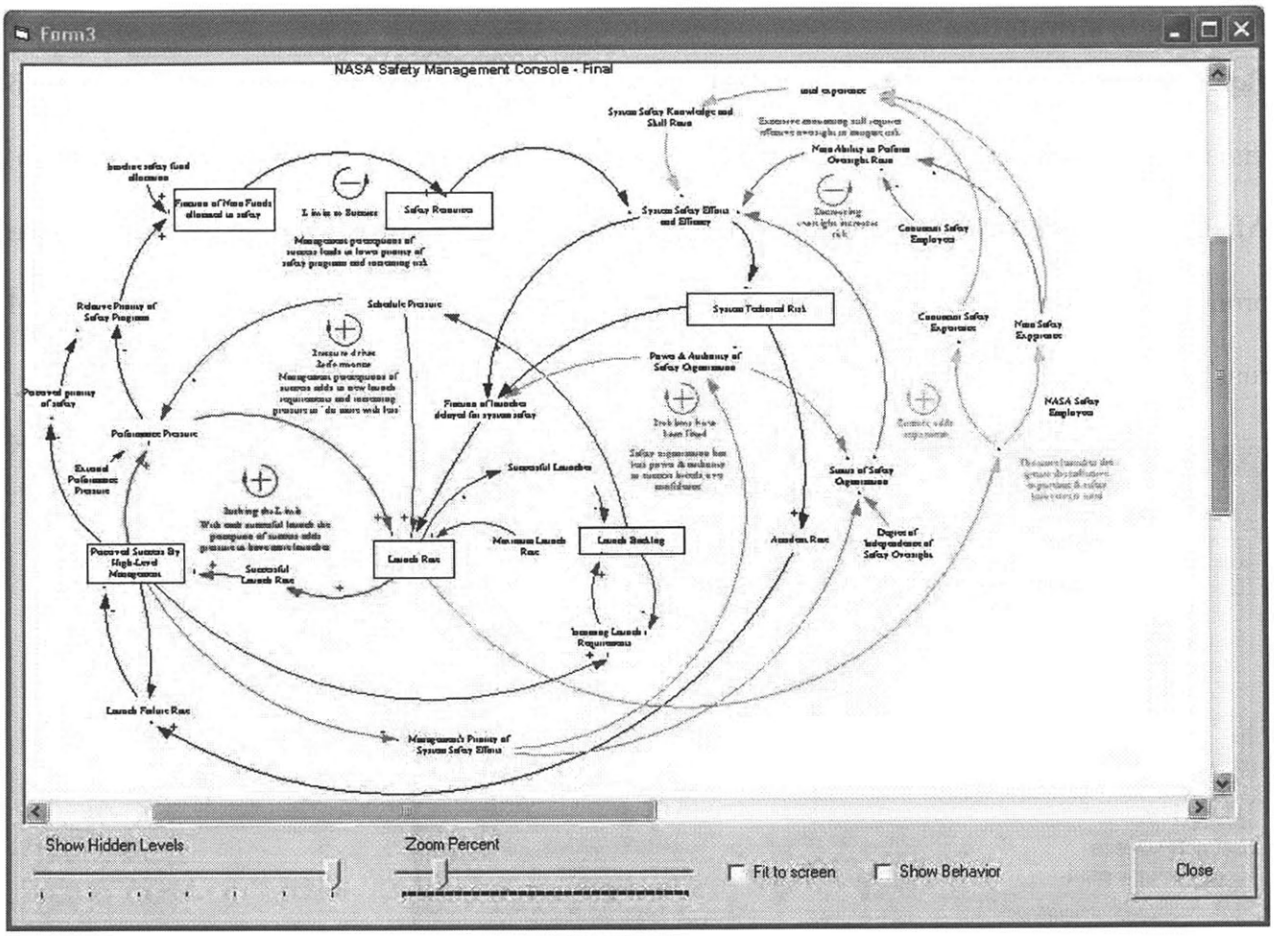



**Figure A.2-6 High level over of the system dynamics model with all layers displayed**

## A.2.3 Running a simulation

### Scenario selection

The simulation tool uses a scenario database that can simulate external events such as changing budget or mission requirements. You control which scenario to run by selecting it from the list box at (1). As you advance in time through the simulation it will check the database for any values that need to be written to the model and display the results in the "Messages" section of the main form.

If you do not want to any data written from the scenario database select "Static budget and resources".

## Starting a simulation

Before starting a new simulation you should type a descriptive name in the run name box (2) so that the data from the simulation run can be viewed at a later date or contrasted with another run. After typing a run name press *Start* (3). If a run with that name already exists you will be prompted if you want to overwrite it. You are then prompted to load or remove any additional runs as shown below.

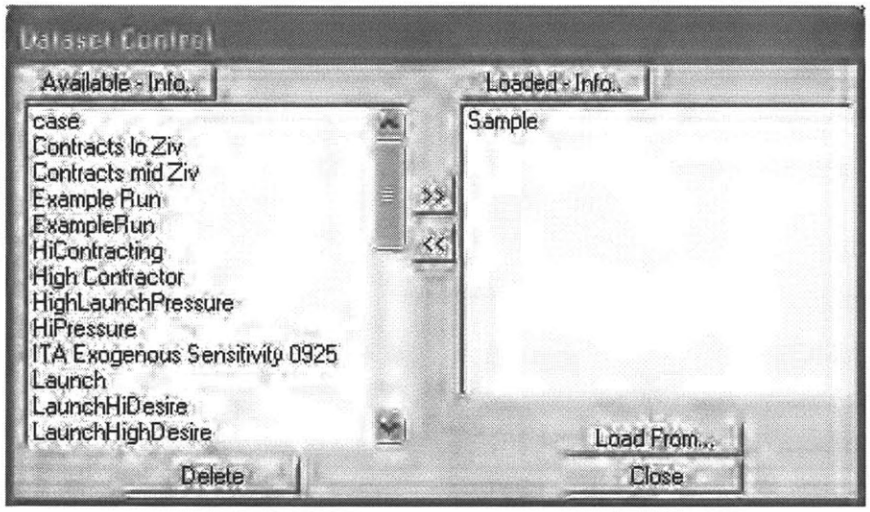



**Figure A.2-7 Load runs configuration dialog**

To load or unload a run either press the appropriate double arrow, or double-click on the run to move it between boxes. You can also press the appropriate buttons to delete a previous model, see the date that it was run and load a run from an alternate location. When you are satisfied press *Close* to close the window.

The main form then updates with the starting default values for all of the user control and display variables. Please note that this will reset any changes you have made to an input variable. The simulation is at time zero.

## Advancing simulation time

The simulation runs from time = 0 months to 144 months (although this can be changed by modifying the Vensim model and republishing it). To advance the simulation press the *Advance Time* button (4). The program writes user inputs from the *Main Control Panel* section (6), advances the simulation time by the number of months in the *Time Step* field (5) and updates the display with the current value of the system variables.

The program also checks to see if the model predicted a shuttle "accident" and displays a message similar to the one below if there was an accident.

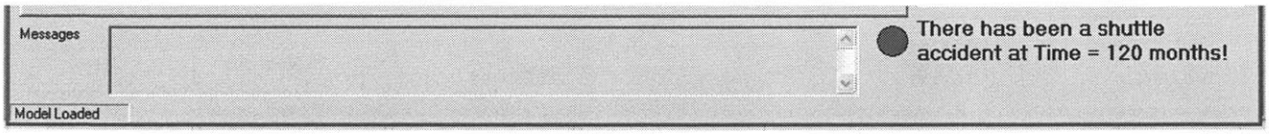



**Figure A.2-8 Message indicating that there has been a shuttle accident**

When the simulation time reaches 144 months a dialog is displayed announcing that the "game is over."

You can end or restart a simulation at any time by pressing the *Start* button.

## Trending data

To display an historical trend of any model variable, left-click one time in the rectangle of any leading or performance indicator. For user input values, double-click in the white text box of the variable. A trend similar to the one below is displayed

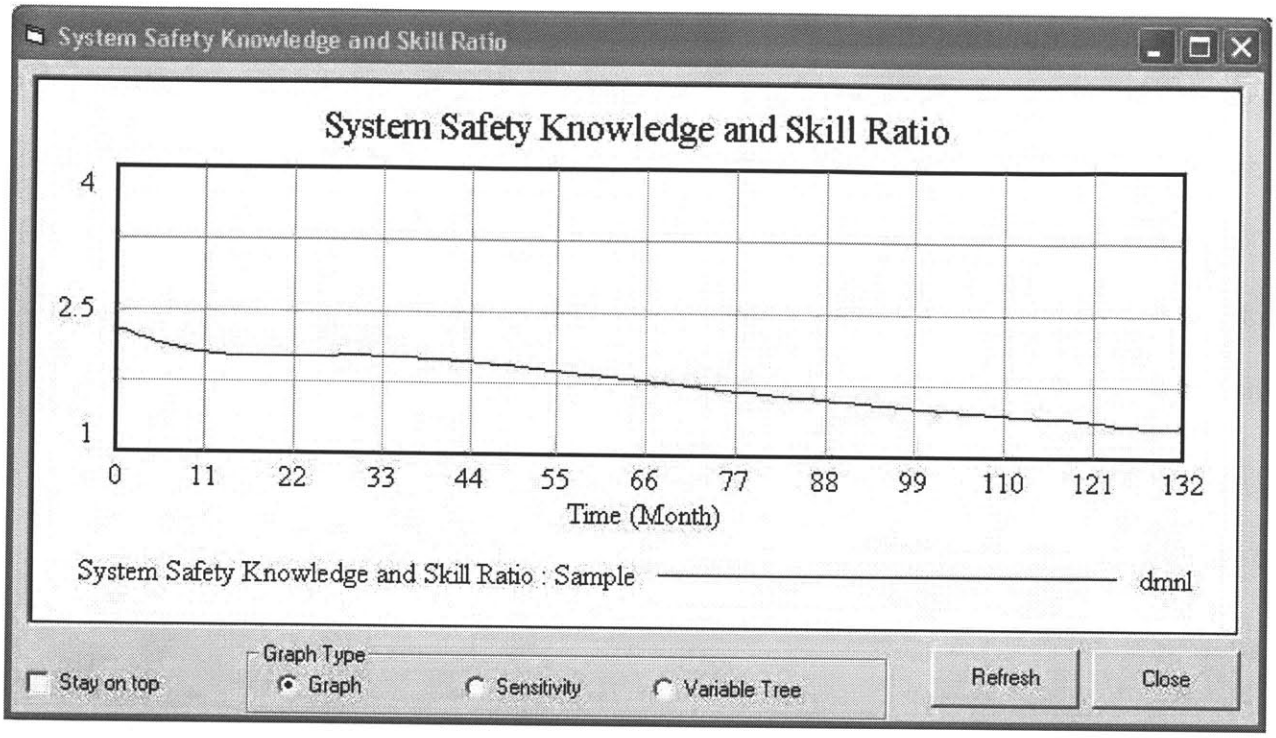



**Figure A.2-9 Historical trend of a model variable**

## Trend display size

To make a trend larger or smaller use your left mouse button to click and drag any of the borders to make the window larger or smaller.

## Trend display controls

### *Graph type control options*

**Graph** The graph radio button is the default when the form opens. Select this option to display the a trend graph of the selected variable

**Sensitivity** Select the sensitivity option to display a sensitivity plot. You will get an error if a sensitivity run is not loaded and the first run listed in the *Load Runs* window. Refer to the section on sensitivity analyses for more information

**Variable Tree** Select the variable tree option to see how a variable is used throughout the model. When first selected you will see the variable name with a "+" sign to the left. Clicking on the + sign displays the variables that depend on the variable as a calculation input. You can then click on any of the newly displayed variables to see what variables depend on them for information and so forth. In this manner you can easily follow the flow of information in the model.

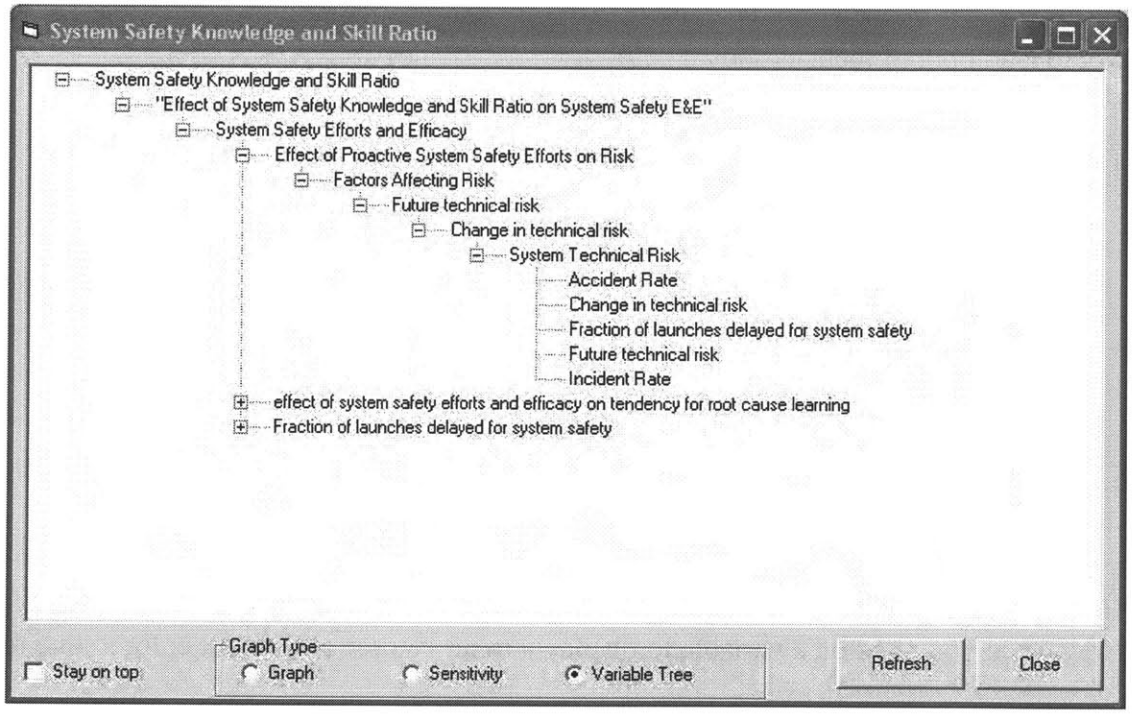



**Figure A.2-10 Variable tree display**

### *Stay on top*

Check the *Stay on top* check box to force the trend to be on top of all other Windows on the desktop. This is useful if you want to monitor a particular variable over time and want the display to be readily visible.

### *Refresh*

Press the *Refresh* button to have the trend display redrawn with the current data.

## Loading other simulation runs

A key feature of the simulation tool is the ability to compare variables between different runs to highlight differences. Whenever you start a new run (with a new name) the data from the simulation is automatically saved to a file with the descriptive name that you gave the simulation. To display the data from other runs press the *Load Runs* button and select the dataset(s) to load. For example, if you wanted to investigate the effect of high and low contracting percentages on the model you would run a simulation for each scenario, and then you would load both of them as shown below.

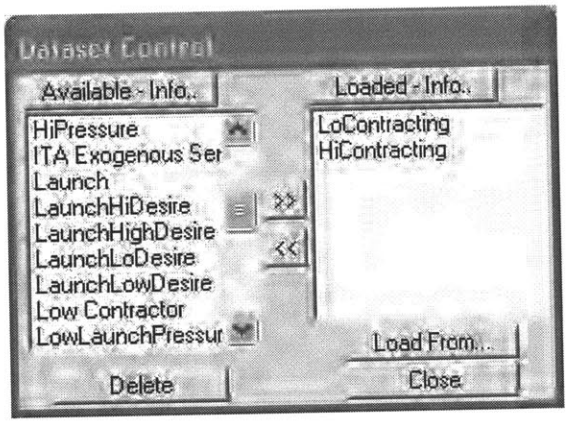



**Figure A.2-11 Screen shot of the load runs dialog with two data runs loaded for analysis**

Now when you click on a variable to display a trend you see a different colored line for each data run. Looking at the charts below, we see that, as expected, the number of contractor safety employees is higher under the Hi Contracting model, but NASA's ability to oversee the contractors is significantly reduced.

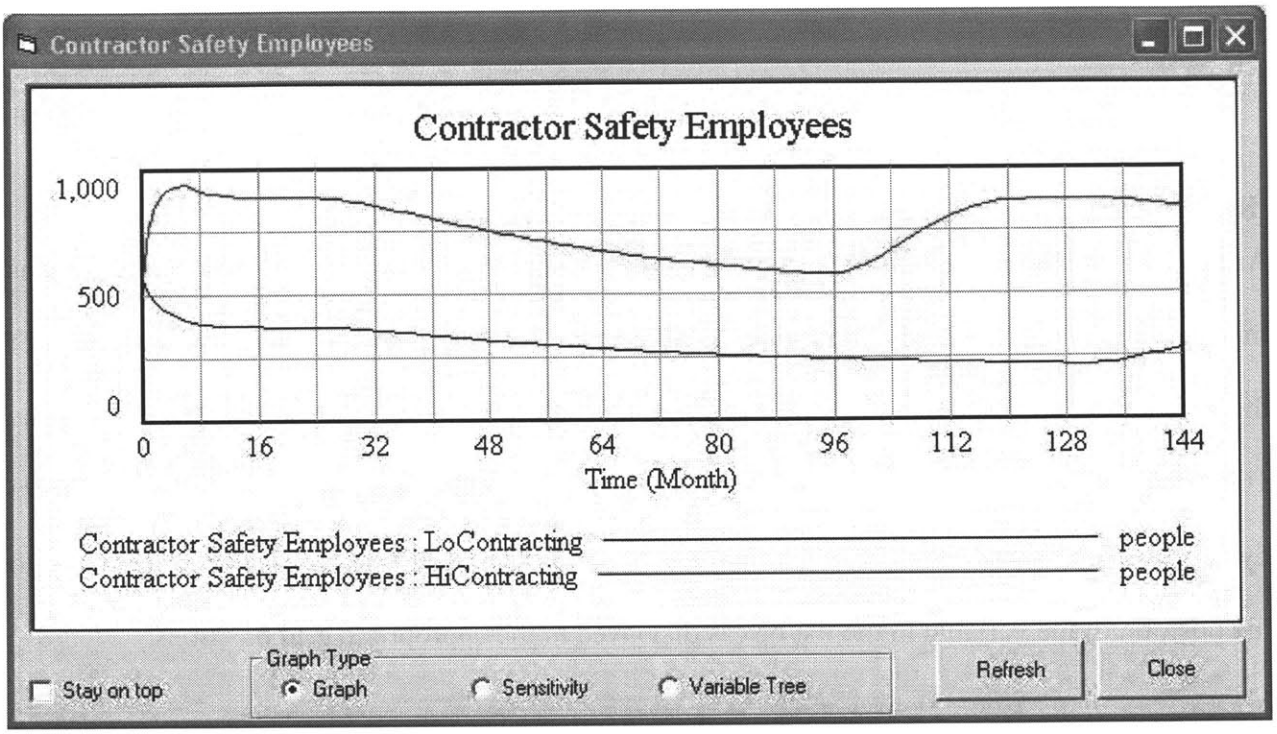



**Figure A.2-12 Trend comparing the number of contractor employees under high and low contracting scenarios**

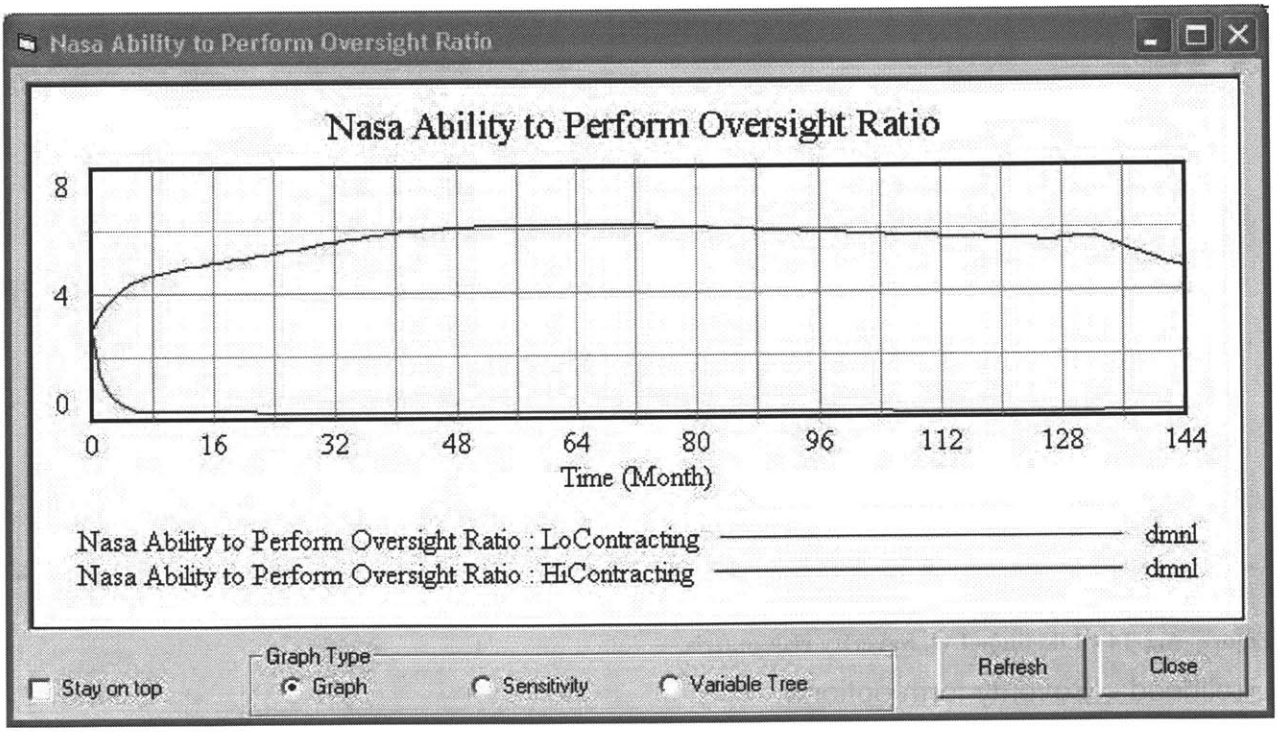



**Figure A.2-13 Trend comparing NASA's ability to oversee contractors under high and low contracting scenarios**

## Risk matrix

Another key feature of the simulation tool is the likelihood vs. severity 4x3 risk matrix that is modeled after the 5x5 matrix commonly used by NASA.. The matrix displays an indication of the potential impact of a leading variable on system safety and risk. The severity of each of the 16 leading variables is fixed from the ITA report, but the likelihood is calculated from the system dynamics model and varies with time. As the likelihood changes, the number corresponding to the position of the variable in the list box is displayed in the appropriate grid positions.

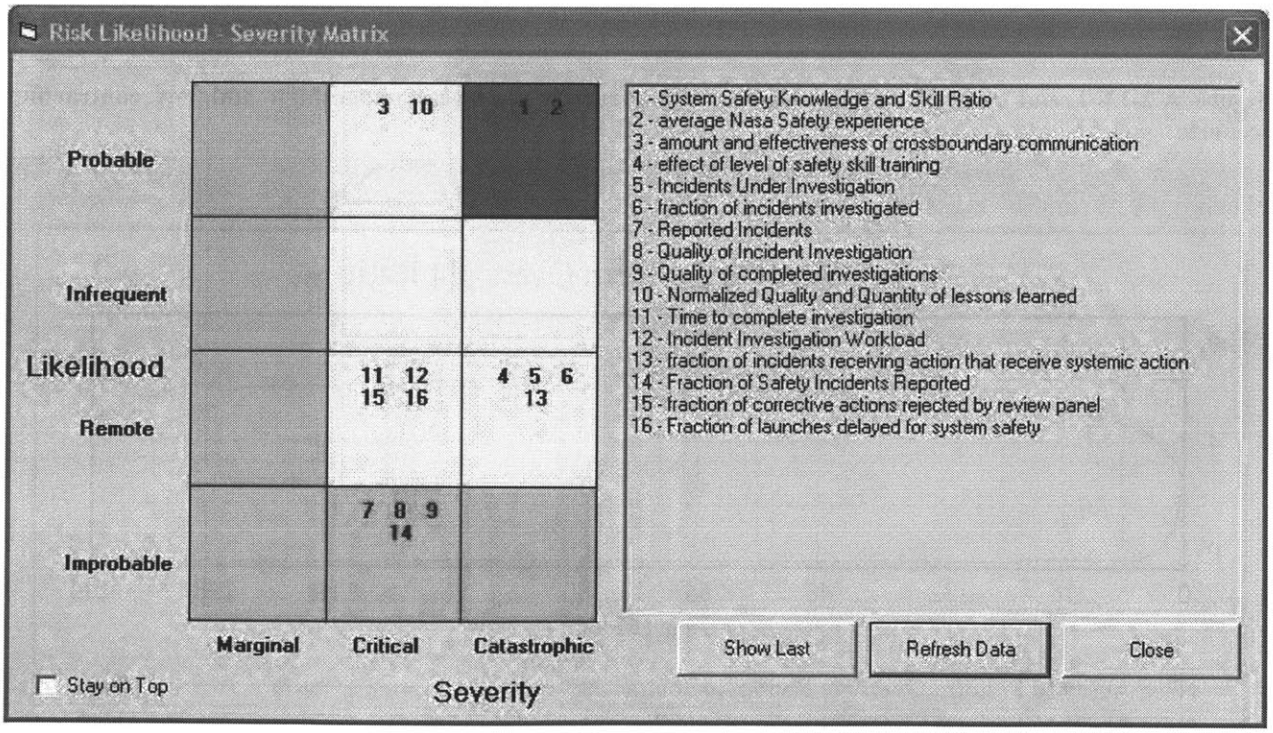



**Figure A.2-14 - Likelihood vs. Severity risk matrix**

Likelihood – Severity form options

***Show Last***

Every time you press the *Refresh Data* button the currently displayed data is buffered in memory and the data is then refreshed. Press and hold the *Show Last* button to display the buffered values in blue, when you release the button the current values are redisplayed. By alternating a couple of times you can easily see any changes in the matrix.

### *Refresh Data*

Press *Refresh Data* to update the matrix with the most current data. Before the matrix is updated the old data is copied to the buffer for the *Show Last* button. Each time you press *Refresh Data* the buffer is updated with the previous data set even if nothing has changed.

### *Stay on Top*

Click on the *Stay on Top* checkbox to force the matrix display to be on top of all other window applications.

## Select graph

*Select Graph* displays a list box of any pre-configured graphs that are stored with the Vensim model. Unlike the trend displays when you click on a variable, the pre-configured graphs can show multiple variables and different graph types such as x-y graphs.

## Sensitivity Analysis

### Creating a sensitivity run

*Sensitivity Analysis* is a powerful tool that makes it easy for you to see the effect of changing a variable incrementally or randomly without having to many simulation runs. For example, suppose you wanted to see the effect of changing the launch rate from 0 to 12 launches per year. You could create 13 different test runs, increasing the launch rate by 1 each time. Or, you could run a sensitivity analysis and have the software calculate the all possible values for you and

display the results on a sensitivity plot. A screen shot of the sensitivity control form is shown below

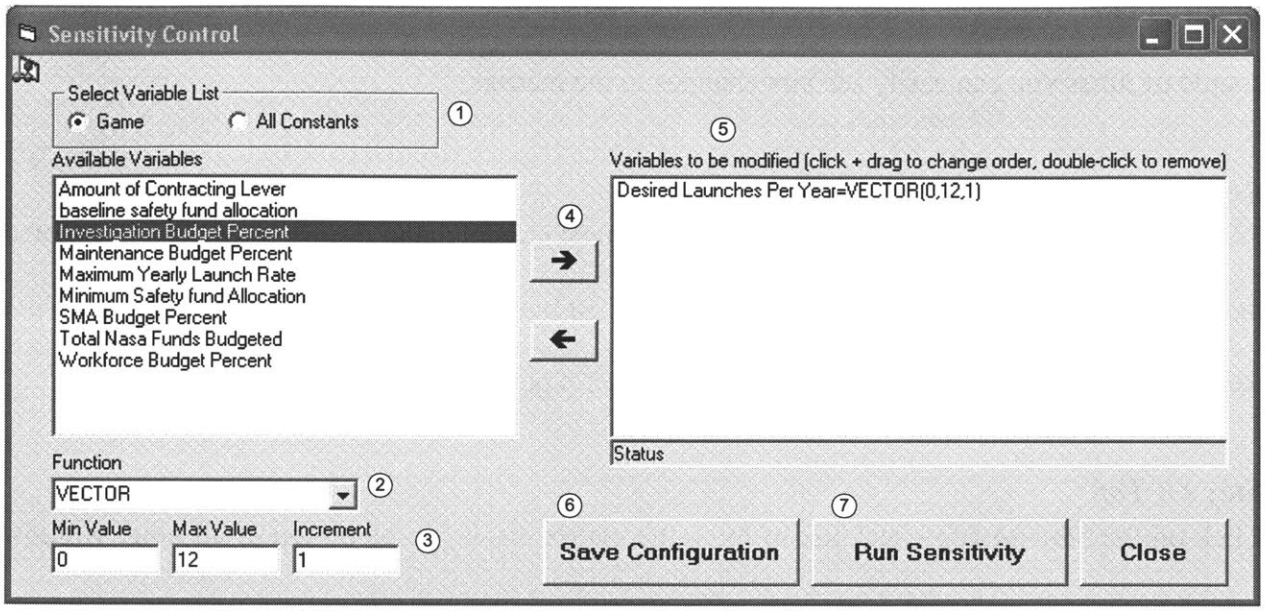



**Figure A.2-15 Sensitivity control configuration dialog**

To procedure for running a sensitivity analysis is as follows:

1. Select a variable and it's sensitivity function and limits
2. Copy the variable and function to the window on the right.
3. Save the configuration
4. Run the sensitivity

Sensitivity Functions

1. Select Variable List – Select *Game* to limit the display to the user control variables in the risk management software. Select *All Constants* to display all user control variables in the Vensim model even if they are not displayed in the risk management software.
2. Function – There are two functions available for modifying a variable, "Vector" and "Random_Uniform".
   a. Select Vector to change a variable linearly. Enter a minimum, maximum and how much to increment the variable in the appropriate input boxes (3)

b. Select Random_Uniform to have Vensim generate a normal distribution between the specified minimum and maximum values (the increment text box is disabled).

3. Min, Max & Increment boxes are where you enter the desired limits for the sensitivity analysis. For example, in the shuttle launch rate example described earlier, you would enter 0, 12 and 1, respectively.

4. Use the left and right arrows to copy the highlighted variable and function parameters to the sensitivity configuration window. You can also double-click a highlighted item in either window to the item between the windows as well.

5. This window shows the selected variable(s) and sensitivity function that will be executed.

6. After modifying or creating a sensitivity configuration you must save it before attempting to run it.

7. Press the *Run Sensitivity* button to execute the saved sensitivity configuration. The results are stored in a data run called, "Sensitivity"

A sensitivity analysis is not limited to modifying only one variable. By adding more than one variable you can evaluate compound effects. For example, suppose you wanted to investigate the effect of changes in the workforce budget with different launch rates. This sort of problem can be extremely laborious to do manually. For example, just three different budget percentages and the aforementioned 13 launch rates (0 to 12) would require 39 different runs. With the sensitivity control you could instead calculate a random distribution of budget percentages for each launch rate, yielding a much more detailed analysis with a fraction of the work. This example is shown below.

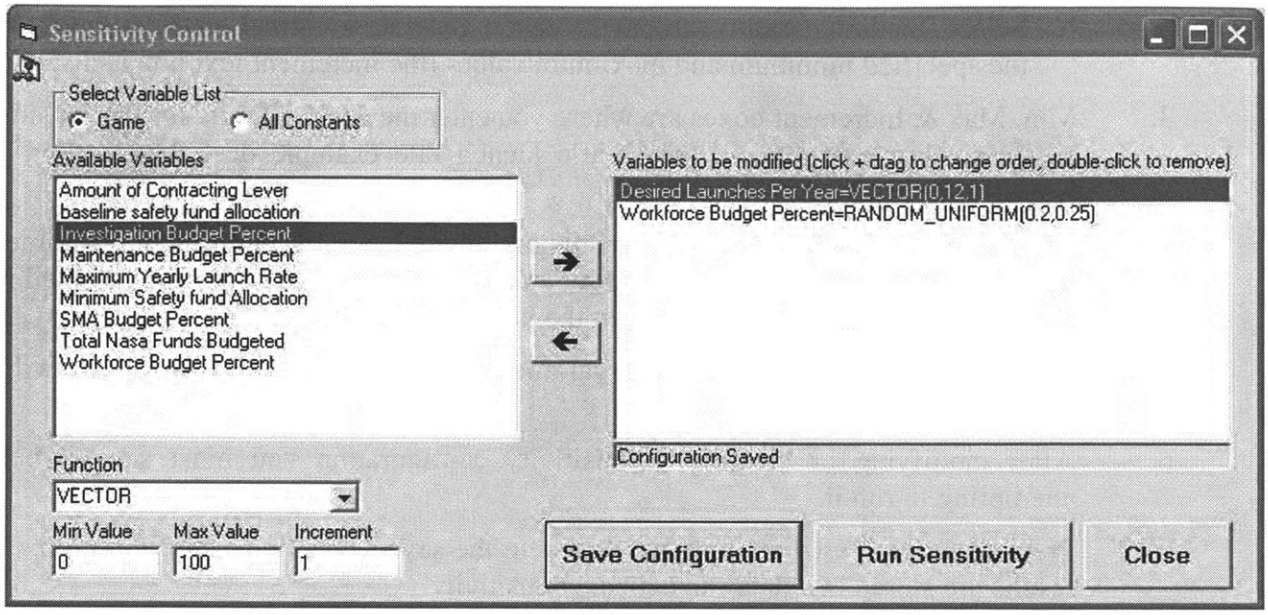



**Figure A.2-16 Sensitivity control configuration for two variables**

## Displaying sensitivity results

After you run a sensitivity analysis the results are automatically loaded and saved in a data set called, "Sensitivity" and *any previously loaded runs are removed.* To see the results, click on any variable box as would normally to display a trend and select the *Sensitivity* option button on the form.

To reload the data from a previous simulation run press the *Load Runs* button and re-select the run.

Note that to display a sensitivity plot the sensitivity run *must be the first (top) data run listed in the Load Runs dialog bo,x or you will get an error when you try to display a sensitivity plot.*

The sensitivity plot is displayed as a probability distribution function. For example, in the example below there is a 50% probability that the value is bounded by the yellow band, a 75% probability of being bounded by the green band.

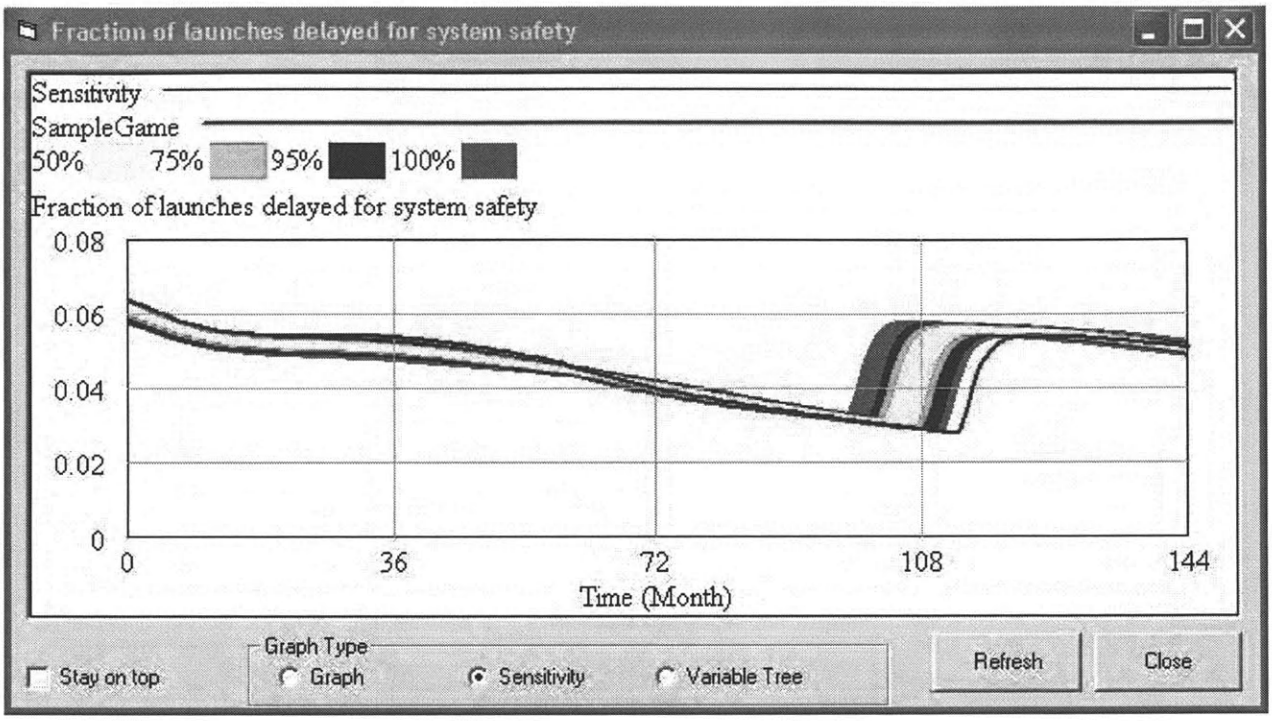



**Figure A.2-17 Sample sensitivity plot**

## A.2.4 Modifying the Vensim Model

You may find it necessary to make changes to the underlying Vensim model. When you are done with your changes you will need to "Publish" the model to make it readable by the simulation software. To publish a model select *Publish* from the Vensim file menu. A dialog box similar to the one shown below will appear. Select the same options as shown below and save the file with the same, or a new name as desired. (If you select a new name then you will need to update the simulation tool configuration to load the new model.)

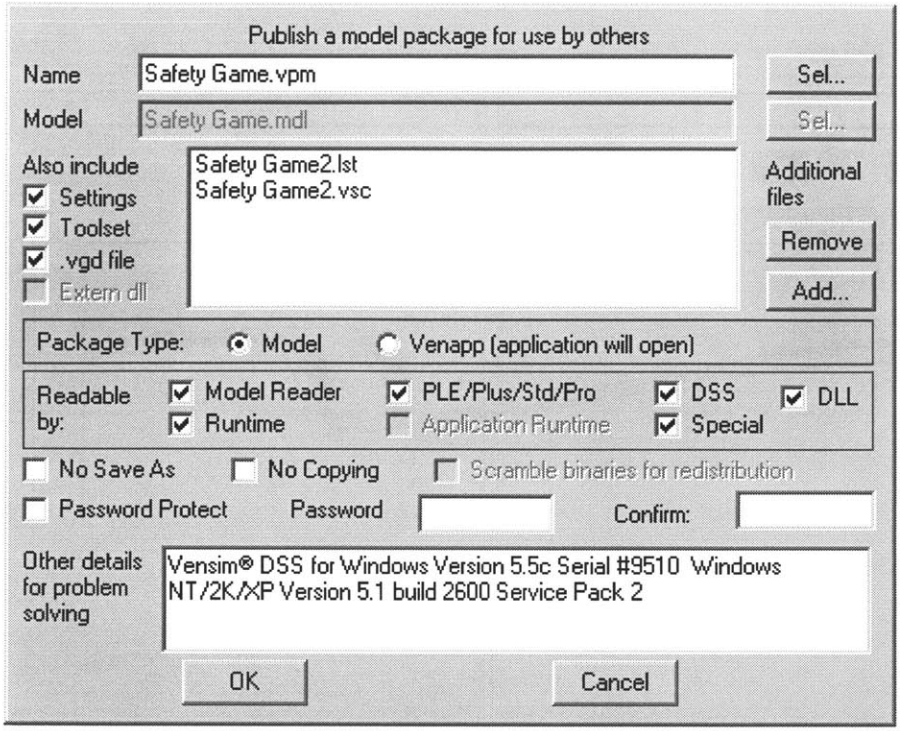



**Figure A.2-18 Vensim publish model dialog when modifying the Vensim model**

## A.2.5 Advanced Topics

### Creating and modifying scenarios

The scenarios are saved in a Microsoft Access database called, "Safety Game.MDB". You will need Microsoft Access to open and modify the database.

#### Modifying existing scenarios

Refer to the diagram below. The scenario database is comprised of three tables

**Scenarios** – Holds a short description of each scenario that is displayed to the operator

**VenVariables** – Has an entry for each unique event

**Scenario2Variable** – Links the Scenarios table to the VenVariables table

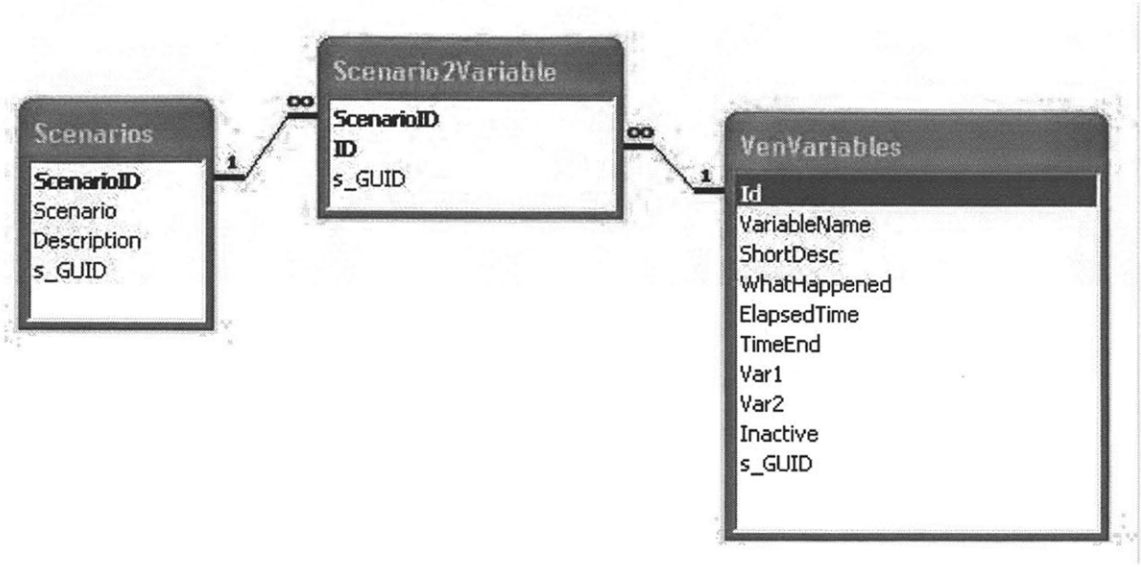



**Figure A.2-19 Scenario database table relationships diagram**

To modify the text or value of an existing event simply open he VenVariables table and make the change in the appropriate field. However, to add new events to a scenario you need to edit both the VenVariables table (to create the event) and the Scenario2Variables table to link the newly created event to the appropriate scenario. The procedure is three steps:

1. Create the new entry in the VenVariables table. Note the unique ID number for the event (.e.g, 22)
2. Open the Scenarios table and note the Unique ID of the scenario that you want to associate the event with.
3. Open the Scenarios2Variable table and on a new row type the scenario ID and the event ID in the ScenarioID, and ID fields, respectively.

Refer to Figures below, which show the relationship between the three tables.

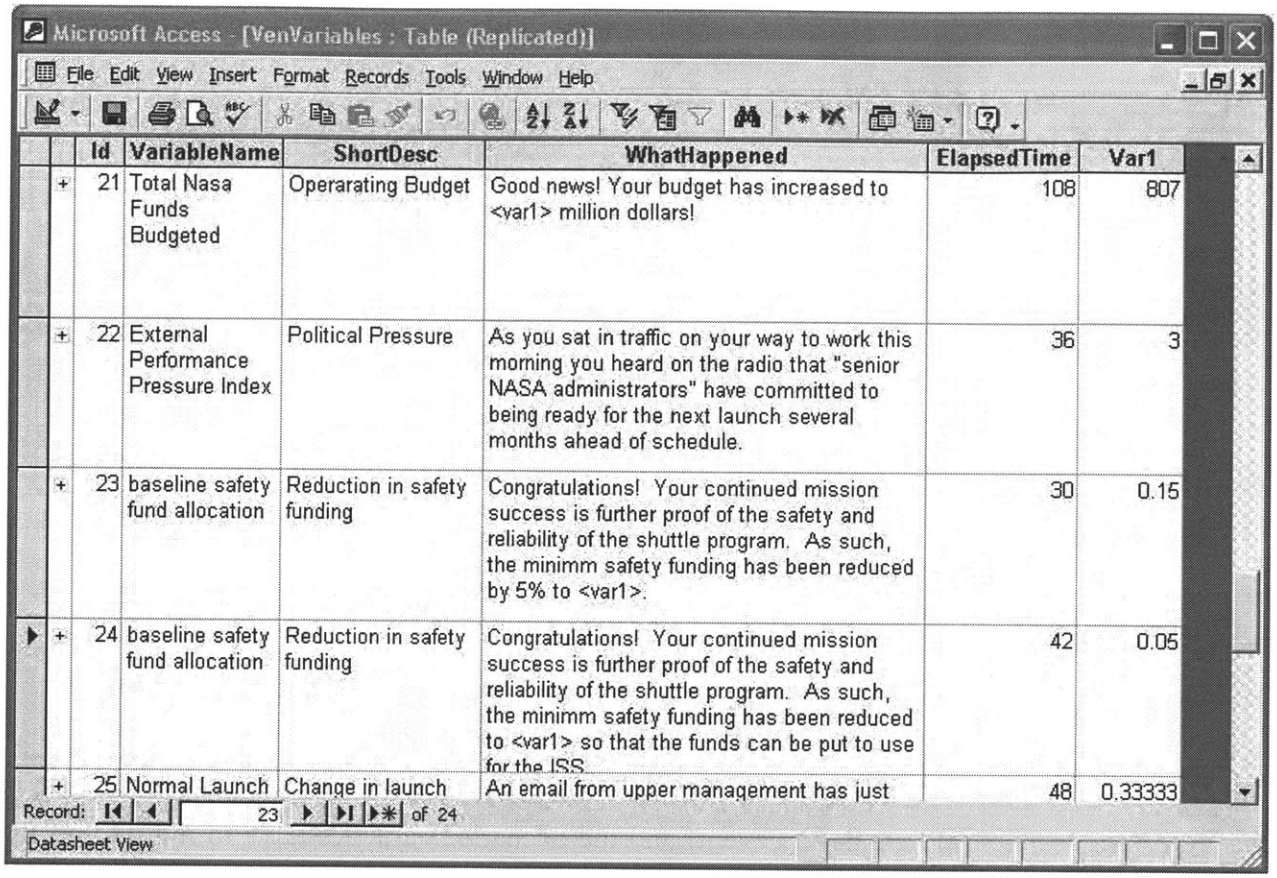



**Figure A.2-20 Screen shot of the VenVariables table**

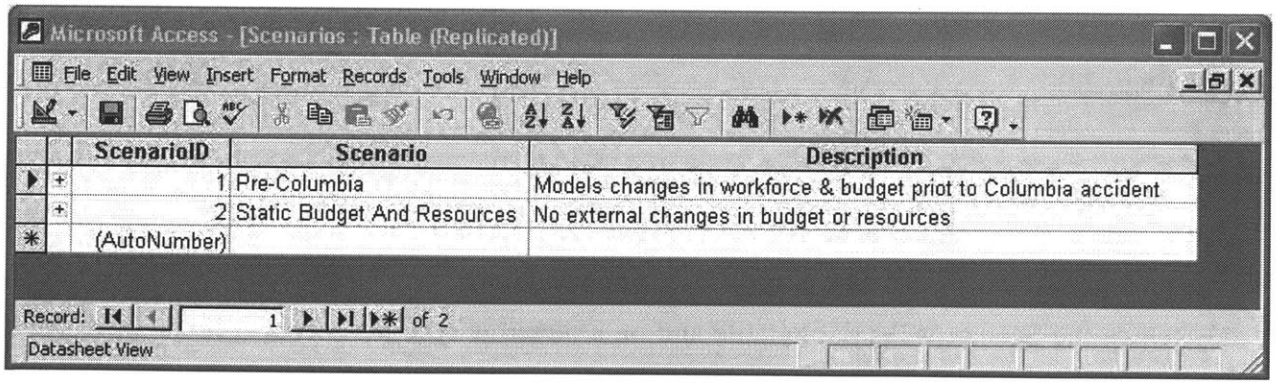



**Figure 2-21 Screen shot of the Scenarios table**

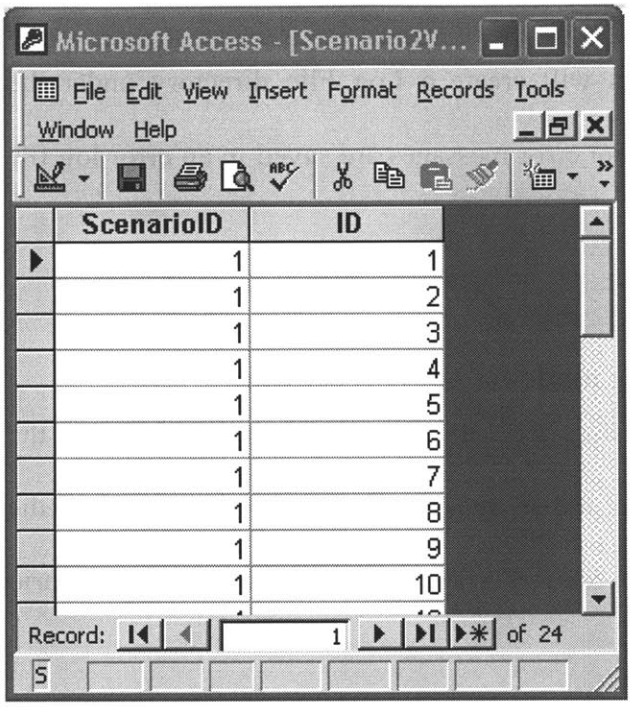



**Figure A.2-22 Screen shot of the Scenario2Variable table which links each event to the right scenario**

## Creating and modifying system instructions

The system instructions are saved in the table, GamePlay. To create a new instruction simply enter a new row and ensure the word, "Intro" is entered in the ShortDesc field

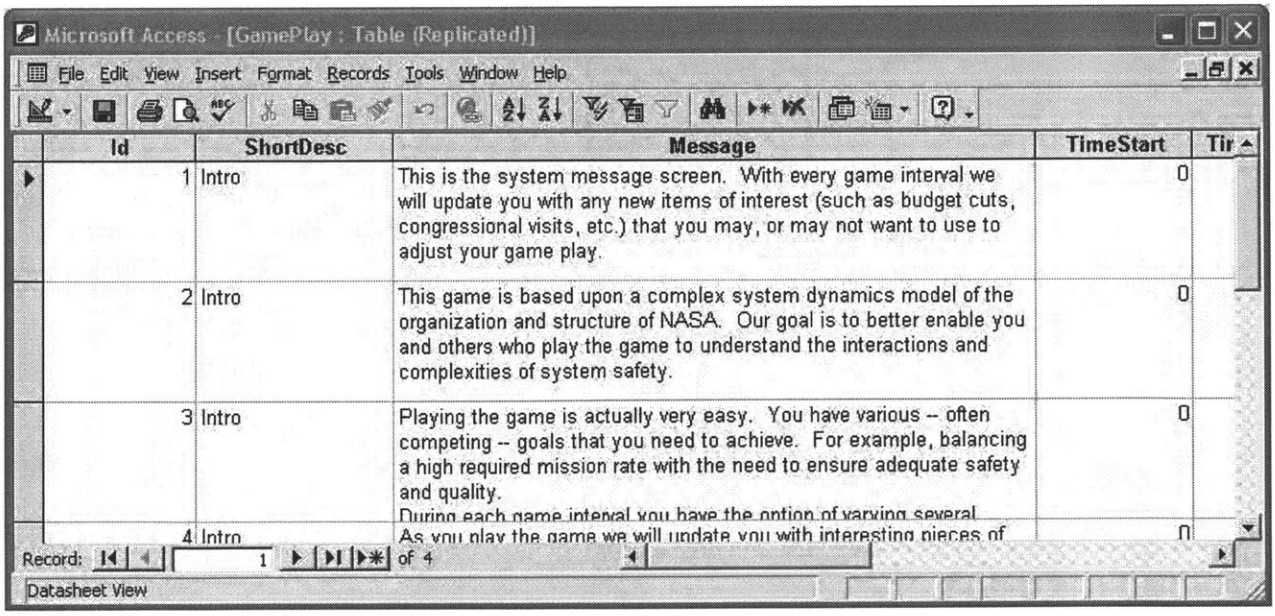



**Figure A,2-23 Screen shot of the GamePlay table with user instructions**

### Error logging

When you first start the software application it will create a Log File directory under the directory where software is installed. Any software error messages are saved in an error log file in the log file directory.

## A.2.6 Trouble-Shooting

### Scenario values are not being written to the model at the correct time

Make sure that you are not advancing the simulation too fast. The simulation tool checks the database after each advance in time to see if there are any values that need to be written to the model. If there are, then they are written at the "current" time – *not* the time in the scenario database.

### Program states that "model has errors" when starting

Sometimes the Vensim model gets corrupted in some fashion. This is usually fixed by republishing the Vensim model. If that doesn't work, open the model and select, "Reform and Clean" from the Vensim file menu (under "Model"), save and republish the model