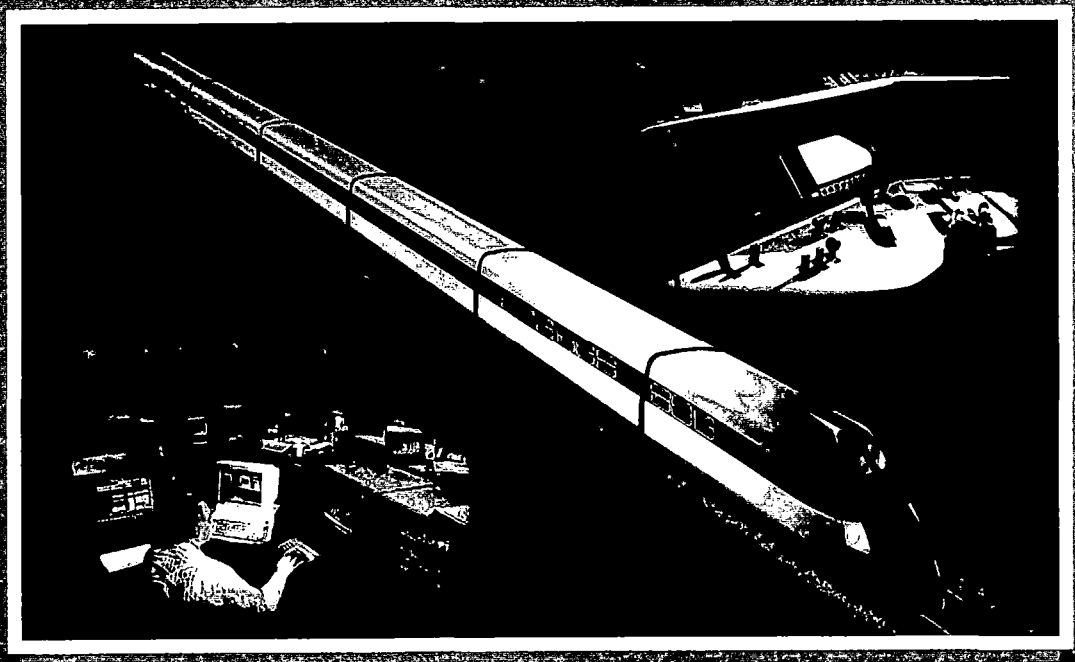
DESK COPY

U. S. Department
of Transportation
**Federal Railroad
Administration**

Human Factors Phase IV: Dynamic Risk Estimation Using The Safety State Model

Office of Research
and Development
Washington, D.C. 20590

Safety of High-Speed
Ground Transportation Systems



)OT/FRA/ORD-
)OT-VNTSC-FRA-

Final Report
March 1999

mc

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*LEAVE BLANK*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Final Report August 1993-July 1997

4. TITLE AND SUBTITLE

Human Factors Phase IV: Dynamic Risk Estimation Using the Safety State Model

5. FUNDING NUMBERS

RR993/R90.32

6. AUTHOR(S)

Edward J. Lanzilotta and Thomas B. Sheridan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

*Human-Machine Systems Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

8. PERFORMING ORGANIZATION

DOT-VNTSC-FRA-

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Department of Transportation
Federal Railroad Administration
Office of Research and Development
Washington, DC 20590

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

DOT/FRA/ORD-

11. SUPPLEMENTARY NOTES

Safety of High Speed Ground Transportation Systems Series

* Under Contract to: U.S. Department of Transportation
Research and Special Programs Administration
John A. Volpe National Transportation Systems Center
Cambridge, MA 02142

12a. DISTRIBUTION/AVAILABILITY STATEMENT

This document is available to the public through the National Technical Information Service, Springfield, VA 22161

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report covers the theoretical development of the safety state model for railroad operations. Using data from a control automation experiment, experimental applications of the model is demonstrated.

A stochastic model of system behavior is developed which is used to estimate the dynamic risk probability in a human-machine system. This model is based on a discrete Markov process model. Based on observer behavior of an existing system, the model is used to determine a instantaneous risk probability function, which is dependent on the system state.

14. SUBJECT TERMS

automation, high-speed trains, human factors, human-in-the-loop simulation, Markov modeling, risk estimation., safety, supervisory controls, transportation

15. NUMBER OF PAGES

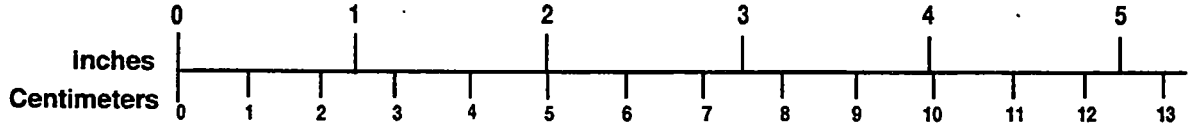
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified

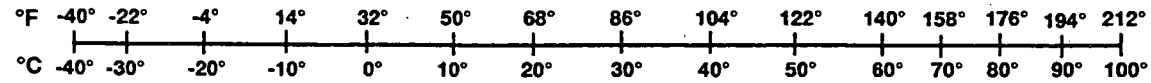
METRIC/ENGLISH CONVERSION FACTORS

ENGLISH TO METRIC	METRIC TO ENGLISH
LENGTH (APPROXIMATE)	**LENGTH (APPROXIMATE)**
1 inch (in) = 2.5 centimeters (cm)	1 millimeter (mm) = 0.04 inch (in)
1 foot (ft) = 30 centimeters (cm)	1 centimeter (cm) = 0.4 inch (in)
1 yard (yd) = 0.9 meter (m)	1 meter (m) = 3.3 feet (ft)
1 mile (mi) = 1.6 kilometers (km)	1 meter (m) = 1.1 yards (yd)
	1 kilometer (km) = 0.6 mile (mi)
AREA (APPROXIMATE)	**AREA (APPROXIMATE)**
1 square inch (sq in, in^2) = 6.5 square centimeters (cm^2)	1 square centimeter (cm^2) = 0.16 square inch (sq in, in^2)
1 square foot (sq ft, ft^2) = 0.09 square meter (m^2)	1 square meter (m^2) = 1.2 square yards (sq yd, yd^2)
1 square yard (sq yd, yd^2) = 0.8 square meter (m^2)	1 square kilometer (km^2) = 0.4 square mile (sq mi, mi^2)
1 square mile (sq mi, mi^2) = 2.6 square kilometers (km^2)	10,000 square meters (m^2) = 1 hectare (ha) = 2.5 acres
1 acre = 0.4 hectare (he) = 4,000 square meters (m^2)	
MASS - WEIGHT (APPROXIMATE)	**MASS - WEIGHT (APPROXIMATE)**
1 ounce (oz) = 28 grams (gm)	1 gram (gm) = 0.036 ounce (oz)
1 pound (lb) = 0.45 kilogram (kg)	1 kilogram (kg) = 2.2 pounds (lb)
1 short ton = 2,000 pounds (lb) = 0.9 tonne (t)	1 tonne (t) = 1,000 kilograms (kg)
	= 1.1 short tons
VOLUME (APPROXIMATE)	**VOLUME (APPROXIMATE)**
1 teaspoon (tsp) = 5 milliliters (ml)	1 milliliter (ml) = 0.03 fluid ounce (fl oz)
1 tablespoon (tbsp) = 15 milliliters (ml)	1 liter (l) = 2.1 pints (pt)
1 fluid ounce (fl oz) = 30 milliliters (ml)	1 liter (l) = 1.06 quarts (qt)
1 cup (c) = 0.24 liter (l)	1 liter (l) = 0.26 gallon (gal)
1 pint (pt) = 0.47 liter (l)	
1 quart (qt) = 0.96 liter (l)	
1 gallon (gal) = 3.8 liters (l)	
1 cubic foot (cu ft, ft^3) = 0.03 cubic meter (m^3)	1 cubic meter (m^3) = 36 cubic feet (cu ft, ft^3)
1 cubic yard (cu yd, yd^3) = 0.76 cubic meter (m^3)	1 cubic meter (m^3) = 1.3 cubic yards (cu yd, yd^3)
TEMPERATURE (EXACT)	**TEMPERATURE (EXACT)**
[(x-32)(5/9)] °F = y °C	[(9/5) y + 32] °C = x °F

QUICK INCH - CENTIMETER LENGTH CONVERSION



QUICK FAHRENHEIT - CELSIUS TEMPERATURE CONVERSION



For more exact and or other conversion factors, see NIST Miscellaneous Publication 286, Units of Weights and Measures. Price $2.50 SD Catalog No. C13 10286

Updated 6/17/98

PREFACE

This document summarizes work done under an ongoing research program at the Volpe National Transportation Systems Center, RSPA/USDOT (Cambridge, MA) in collaboration with the Human-Machine Systems Laboratory at the Massachusetts Institute of Technology (Cambridge, MA). Supported by the Office of Research and Development, Federal Railroad Administration (FRA/USDOT), the program is part of a comprehensive effort to identify and develop the technical information required for safety regulation of high-speed guided ground transportation. This is a final report on one component of the overall research program.

The evaluation of risk of any complex human-machine system must necessarily include study of the interaction between the human operator and the system. The pursuit of safety can be considered as the minimization of risk, under the constraint of available resource expenditure. Thus, risk assessment is the objective component of safety. The risk of an undesirable event involves the probability of the occurrence of that event combined with the severity of the event outcome.

In the area of transportation, the undesirable event is often referred to as an accident. In ground transportation, this class of events includes collisions between two or more vehicles, as well as situations where vehicles become separated from the normal guideway or roadway. An accident has potential for causing personal injury or fatality to passengers or operators, as well as property damage.

It is intuitively clear that the risk probability of a system is dynamic—that is, the risk probability changes with time. This research assumes that the dynamic risk probability is a function of system state. A stochastic model of system behavior is developed which is used to estimate the dynamic risk probability in a human-machine system. This model is based on a discrete Markov process model. Based on observed behavior of an existing system, the model is used to determine an instantaneous risk probability function, which is dependent on the system state. The risk probability function is used for system analysis and identification of high risk system states. The risk probability function is also used to transform observed system behavior into dynamic risk trajectories, which are used for performance evaluation of individual operators with respect to risk.

This report covers the theoretical development of the safety state model. Using data from the control automation experiment (Lanzilotta and Sheridan, 1998), experimental application of the safety state model is demonstrated. An appendix includes a detailed description of the high-speed rail simulation system, used for a series of human factors experiments in high-speed rail.

ACKNOWLEDGMENTS

The authors would like to acknowledge and thank the many people that have been involved with and contributed to the completion of this work. In particular, we would like to thank Dr. Thomas Raslear of the Federal Rail Administration for his guidance and financial support of the project. Dr. Donald Sussman and Mr. Robert Dorer, at the Volpe National Transportation Systems Center, lent significant contributions to the shape and direction of the work, and were able to provide key contacts and references. Dr. Jordan Multer provided valuable insight and guidance during the experimental phase of the research. In addition, we would like to thank Dr. Peter Mengert and Dr. Robert DiSario for graciously reviewing the theory and application of the theoretical model.

At the Massachusetts Institute of Technology, we would like to thank our colleagues in the Human-Machine Systems Laboratory for their contributions and support. Through collaborative work in development of the high-speed rail simulation system, as well as through complementary experimental work, my association with Shumei Yin Askey was fruitful and enlightening. Nicholas Patrick was especially helpful with statistical analysis and interpretation. Jacob Einhorn, Bernardo Aumond, Steven Villareal, and Helias Marinakos have all been of assistance while in the process of taking on the next phases of the project. Our undergraduate assistants, Sapna Augustine, Wendy Chang, Grace Park, Joseph Wenish, and Jill Chen, were helpful in preparing for and conducting the experimental portion of the research. Dr. Jie Ren built the analog-to-digital converter used with the simulator throttle.

Finally, there are special thanks to Dr. Judith Bürki-Cohen, of the Volpe National Transportation Systems Center, for her guidance and enthusiasm throughout this work. Her experimental experience and editorial input were critical to the successful completion of the experiments and subsequent reports.

TABLE OF CONTENTS

Section	Page
PREFACE	iii
ACKNOWLEDGMENTS	v
CONTENTS	vii
EXECUTIVE SUMMARY	xi
1. INTRODUCTION	1
2. BACKGROUND: SAFETY AND RISK ASSESSMENT	3
2.1 THE RELATIONSHIP BETWEEN SAFETY AND RISK	3
2.2 RISK: A SYSTEMS PERSPECTIVE	5
2.3 THE RELATIONSHIP BETWEEN SAFETY AND SYSTEM RELIABILITY	6
2.4 THE HUMAN ELEMENT IN RISK	9
2.5 SYSTEM ANALYSIS	11
2.6 CONTROLLING OPERATOR BEHAVIOR	12
3. THEORY: SAFETY STATE MODEL	13
3.1 BRIEF REVIEW OF MARKOV PROCESS THEORY	13
3.2 THE SAFETY STATE MODEL—AN APPLICATION OF FINITE MARKOV PROCESSES	17
3.2.1 Structure of the Safety State Model	17
3.2.2 Finding the State Transition Probabilities	20
3.2.3 Finding the Mean Time to Failure From Each State	22
3.2.4 Converting MTTF to Risk Probability	26
3.2.5 Characteristics of the State Transition Matrix	27
3.3 IMPLEMENTATION—FOUR PHASES OF ANALYSIS	27
3.4 PRACTICAL LIMITATIONS	29

TABLE OF CONTENTS (cont.)

Section **Page**

4. EXPERIMENTAL DEMONSTRATION 31

4.1 MOTIVATION AND GOALS 31

4.2 PHASE 1 RESULTS—RISK IDENTIFICATION 31

4.3 PHASE 2 RESULTS—SYSTEM OBSERVATION 38

4.4 PHASE 3 RESULTS—MODEL CALIBRATION 39

4.5 PHASE 4 RESULTS—RISK ESTIMATION 51

4.6 DISCUSSION 59

5. SUMMARY AND CONCLUSIONS 63

REFERENCES 67

APPENDIX A — DESIGN AND IMPLEMENTATION: HIGH-SPEED RAIL SIMULATION SYSTEM 71

A.1 GOALS AND OBJECTIVES 71

A.2 SYSTEM ARCHITECTURE ISSUES 72

A.2.1 Road Database Representation 72

A.2.2 Network Interconnection 79

A.2.3 OTW View 80

A.3 ACTIVE SIMULATION ELEMENTS 81

A.3.1 CTC Simulation 82

A.3.2 Vehicle Simulation—Display-Aided Version 83

A.3.3 Vehicle Simulation—Control Automation Version 85

A.3.4 Vehicle Simulation—OTW Server 86

A.3.5 Vehicle Simulation—Dashboard Server 87

A.4 SUPPORT SOFTWARE 87

A.4.1 `Pathnet` 87

TABLE OF CONTENTS (cont.)

Section **Page**

A.4.2 Additional Data Analysis Tools 87

A.5 SOFTWARE ENGINEERING ISSUES 88

A.5.1 Shared Libraries 89

A.5.3 Software Build Engineering 91

A.5.4 Revision Control 91

A.6 SYSTEM CONFIGURATIONS TO SUPPORT EXPERIMENTS 91

EXECUTIVE SUMMARY

Safety is a concern in all aspects of life. It is of particular concern in transportation systems, where there is an implicit understanding that the service providers will take reasonable efforts to ensure the safety of the customers of the system. Transportation systems are currently experiencing an accelerated application of high technology, particularly with regard to systems intended to assist the human operators and reduce their task load. Whenever operator aid or automation systems are implemented, the nature of the interaction between the human operator and the machine system is changed. In order to assure that the safety of the system has not been compromised in the process, a method for evaluating the system-level safety performance is required. However, objective measurement of safety is quite difficult.

Safety is often considered to be the freedom from risk. Since risk cannot be completely eliminated, it is perhaps more accurate to define safety as the minimization of risk. The risk of a bad event is generally considered to be some combination of the probability of occurrence of that bad event with the eventual outcome of the bad event, relative to the health and well-being of people that interact with the system. Efforts may be taken to reduce the level of risk, both by reducing the probability of an accident (*active safety*) and reducing the consequences in the event of an accident (*passive safety*). Both classes of effort require expenditure of resources to achieve their goal.

It is relatively easy to evaluate the outcome of a bad event—through deterministic creation of the event and measurement of the resultant effects, the relationship between the event and the outcome can be reasonably determined. For example, the risk to human passengers in head-on automobile crashes can be evaluated by deliberately colliding two vehicles under controlled conditions with instrumented dummies in the passenger seats.

It is, however, much more difficult to objectively evaluate the probability that the event will occur. First of all, accidents are extremely rare events. As a result, it is extremely difficult to identify conditions immediately leading to an accident. To further complicate matters, the relative safety of contemporary systems is higher than before. This implies that most, if not all, of the first-order safety flaws have been identified and corrected. Thus, accidents tend to be the result of complex compounding of causal factors.

Intuitively, we know that the risk probability is not constant, but rather a dynamic state. This is especially true in highly dynamic systems, such as transportation systems. The concept of a *near collision* underscores our understanding that some situations are inherently riskier than others. Furthermore, it is believed that near collisions occur far more frequently than actual collisions.

The goal of this research is the development of a model which allows estimation of the dynamic risk probability as a function of system state. This goal is motivated by the fundamental notion that risk probability is a dynamic system state and is a function of system state which results from human operator input. Accomplishment of this goal will allow objective identification of near collision situations, and will provide a mechanism for identifying causal chains leading to collisions or near collisions.

Some techniques have been developed for estimation of risk probability. Such techniques include fault tree and event tree modeling, among others. However, these techniques estimate risk probability as a constant number, valid throughout all operational modes of the system. While useful for certain types of system safety evaluation, these techniques do not provide a mechanism to estimate dynamic risk probability.

The *safety state model* is a stochastic model, based on a finite Markov process. A set of binary conditions, which are either true or false, are combined into a binary number to form the Markov state number. As a result, the Markov state number is a description of the system state. An additional state is assigned to the accident event. The accident event is classified as a trapping state, corresponding to the irreversible nature of an accident. When the state transition matrix of the Markov process is known, the mean time to failure (MTTF) can be estimated for each state in the system. The MTTF is then transformed into an estimate of risk probability. In order to determine the state transition probabilities contained in the state transition matrix, the system behavior of an operational system is observed and recorded. The collected statistics are combined to determine the state transition matrix.

Demonstration of the safety state model was accomplished in conjunction with an experimental evaluation of the effect of control automation on locomotive engineer performance in high-speed rail operation (Lanzilotta and Sheridan, 1998). A human subject experiment was conducted, using a human-in-the-loop simulation system (described in Appendix A). The accident event selected was a grade-crossing accident, in which a high-speed passenger train collides with a

highway vehicle stuck in a grade crossing. Seven binary conditions were identified, each of which was judged to be potentially contributory in the grade-crossing collision. The seven conditions combine to form 128 system states, with an additional state for the collision event. Over the course of the experiment, system behavior was recorded by monitoring the seven conditions selected. The recorded data is summarized as a safety state trajectory, as a function of time. The collected statistical data is combined to determine the state transition matrix, which was then used to determine the risk probability as a function of system state. The safety state trajectories were subsequently converted into risk probability trajectories, as a function of time, through application of the risk probability transformation function.

Application of the safety state model to determine dynamic risk probability of a system is shown to be a successful technique for estimating dynamic risk probability. It is a generalized model, in the sense that it considers all combinations of the contributory conditions, without prejudice or bias. The observational nature of the method corresponds to conventional human behavior in evaluating safety—we observe the behavior of a system, and make conclusions regarding the relative risk. The safety state model is an omniscient observer, collecting data over a wide scope of conditions. It is a method that can be easily automated.

The primary weakness of the safety state model is the potential magnitude of required computational resources. The number of states in the Markov process grows exponentially with the number of conditions that are considered. Thus, the breadth of effectiveness of the method is constrained by limits in computer technology. An additional shortcoming is related to the observational nature of the method. Calibration of the state transition is only as accurate as the observed data. Thus, the accuracy of the resultant transformation function is limited by the nature of the observed data. As a result, the method is not well-suited to prediction of events in the absence of calibration data.

Overall, the safety state model is demonstrated to be a useful tool for estimating the dynamic risk probability in a complex system. Application of parallel processing computer technology can be exploited to address the limitations of the technique due to computational limitations. The technique does not require special laboratory conditions, and it can be applied to virtually any operational system. It is believed that the method is generally applicable across a wide scope of complex human-machine systems, including most forms of transportation systems.

1. INTRODUCTION

As the U.S. progresses toward high-speed passenger rail service, questions have emerged regarding the usefulness and implications of automation in the vehicle cab. In the study of existing foreign high-speed passenger rail operations (Train à Grande Vitesse (TGV) in France, Inter-City Express (ICE) in Germany, and Shinkansen in Japan), it is clear that automation is implemented in these systems in differing degree and with differing philosophy.

The primary motivation for incorporating automation in rail vehicles is to simplify the task of vehicle operation. Some believe that the use of automation for mundane tasks will free a portion of the attention bandwidth of the locomotive engineer, thereby allowing the operator to focus more intently on higher-level tasks, such as monitoring for vehicle failures and other emergency situations. Thus, the overall performance of the train operator is expected to improve through the use of automation.

The application of automation often carries a concern for safety, especially in the case of transportation systems. It is generally agreed that human error is a significant factor in a majority of transportation accidents. As the level of automation in any system increases, the interaction between the human operator and the system necessarily changes. It is not yet clear how these changes will impact safety. On one hand, some believe that increasing the level of automation will free the attention resources of an operator from mundane, repetitive tasks, thus potentially increasing the level of safety by allowing the operator to focus more intently on safety-related concerns. On the other hand, it is possible that increasing the level of automation will sufficiently reduce the workload of the operator that s/he might lose some degree of situation awareness. This condition, often referred to as an *operator out-of-the-loop* situation, has serious negative safety implications when an operator must make a timely response to an unexpected emergency situation.

Accidents are relatively rare occurrences, especially in transportation systems. Estimation of risk, as a means of evaluating relative safety, is a difficult task. To complicate matters, monitoring a system for accident analysis is quite difficult because the time period of highest interest occurs immediately prior to the accident. This situation leads to great difficulty in obtaining data that could help identify accident causality. An additional complication is that

accidents are generally caused by a compound set of events that interact in an unexpected fashion. Under these conditions, it is very difficult to estimate risk probabilities or to determine causality among the chain of events that lead to an accident.

The goal of this research is the development of a method for estimating the dynamic risk probability of a human-machine system. A driving force in this work is the notion that risk is dynamic, changing over time. Considering an example in highway transportation, it is clear that an automobile driver is at higher risk of having an accident when s/he is drunk and tired, compared to when s/he is well-rested, alert, and sober. The level of risk varies with changes in operator fitness, vehicle positions and speeds, vehicle condition, weather, obstacles, and so on. Identification of risk as a dynamic state, as a function of a set of potentially causal factors, opens the door to methods that will allow identification of causal factors and events.

The resulting method, termed the *safety state model*, utilizes a stochastic system model, based on a discrete finite Markov process, to estimate the mean time to failure (MTTF) from each of the defined system states. The system states are defined as combinations of binary (i.e., true or false) conditions, each of which are potentially causal factors to an accident. The accident is modeled as a trapping state, which corresponds to the notion of irreversibility of an accident.

As a related component of the research project, a human subject experiment was conducted to evaluate the effects of control automation on operator performance (Lanzilotta and Sheridan, 1998). The experiments were conducted in a laboratory setting, using a high-speed rail simulation system (described in Appendix B). During the course of this experiment, data was collected and used for demonstration of the safety state model.

2. BACKGROUND: SAFETY AND RISK ASSESSMENT

Whether we consciously realize it or not, risk assessment is an integral component of our lives. Throughout our daily activities, we continuously make decisions which are impacted by our understanding of safety. In a sense, each decision is a form of risk assessment—we weigh the costs against the benefits, and make a decision. In general, our understanding of safety is based on intuition.

The analysis of safety is an active field of research. Application of the techniques generated by safety research occurs in major decision processes, often in business. As a result of this research, many of the concepts and terms have been formalized. This chapter serves to provide background from the safety research literature as it applies to the development of the safety state model and its applications.

2.1 THE RELATIONSHIP BETWEEN SAFETY AND RISK

Because of the often intuitive understanding of safety, the words used to describe safety-related concepts can sometimes have multiple or ambiguous meanings. To prevent ambiguity, the terms are defined and discussed here. The dictionary definition of *safety* refers to freedom from risk of human injury or death (Webster's Dictionary, 1994). However, this represents an ideal (and realistically unachievable) goal, as it is not possible to completely eliminate all risk. While complete elimination of risk is not achievable, reduction of risk is possible, although at some cost. Therefore, it is more appropriate to discuss the *pursuit of safety*, in which the level of risk is traded against the costs of reducing risk. In effect, the pursuit of safety is the minimization of risk, subject to constraints on available resources.

Lowrance (1976) defines safety as the "judgment of acceptability of risk." This definition provides a working framework in which safety is separated into a subjective component and an objective component. The subjective component, which is the *judgment of acceptability*, evaluates whether a given level of risk is acceptable to the society which is affected. Based on that judgment, policies are set, which determine the trade between a desired level of risk and resources expended to achieve that level of risk.

The objective component is *risk assessment.* There are a variety of definitions of risk in the literature. Rowe (1977) defines risk as "the potential for unwanted negative consequences of an event or activity," alluding to the notion of chance and reflecting the undesirable nature of the outcome. Lowrance leans toward a more explicit definition, defining risk as the "measure of probability and severity of adverse effects." Rescher (1983) echoes the notions of probability and severity: "Risk is the chancing of negative outcome. To measure risk we must accordingly measure both of its defining components, the chance and the negativity." Gratt (1987) specifies the mathematical relationship between probability and severity in risk assessment as a product of two value, stating that the "estimation of risk is usually based on the expected result of the conditional probability of the event times the consequences of the event given that it has occurred." Wharton (1992) offers that "a risk is any unintended or unexpected outcome of a decision or course of action," including both positive and negative outcomes.

There is a tacit presumption among the various definitions of risk that zero risk is an unattainable goal. This reflects the fact that, whenever there is human interaction with a system, there is some finite potential, no matter how slight, that there could be injury from that system. Taking a fatalistic viewpoint, a truly probabilistic event (i.e., an event that is well-modeled by probability theory) will eventually occur, given enough time. The only way to avoid the occurrence of such an event is to "get out of the game" before that event occurs. In fact, this is what happens to most of us with regard to rare catastrophic events. This notion forms the basis of risk exposure—given a constant risk probability, the expected number of failures over a prescribed period rises with the size of the period. The colloquial notion of risk exposure is that the "laws of probability catch up with you." While this presents a convenient rationalization, in fact the relationship is reversed—the occurrence of the accident provides statistical data to calibrate or validate the probabilistic model.

Whenever we, as individuals, are part of a system which has potential for personal injury, safety is of concern to us. The system in question might be a transportation system that is used for commuting to the workplace, or might be a production line in the workplace. Whatever the system in question, the goal of the pursuit of safety is to reduce the potential risk of that personal injury.

Through expenditure of resources, risks can be reduced. This is the task of risk management—strategically choosing the amount and application of resources used to reduce risk. Risk can be reduced either a) by reducing the probability of occurrence of the undesirable event, or b) by reducing the severity of the outcome when the event does occur.

Thus, we have the basis for distinction between active safety and passive safety devices, respectively. *Active safety* is a term typically applied to those devices or systems which assist in preventing accidents, while *passive safety* is applied to those devices or systems which reduce the severity of an accident when it does occur. In practice, the pursuit of safety involves a combination of the two.

The challenge of risk management is adequate determination of the relationship between safety expenditures and the resultant reduction of risk. This relationship is difficult to determine, in part because it is difficult to quantify the potential for risk. In addition, the risk manager must consider that there is always some point of diminishing returns on investment—as risk is reduced, the relative expenditure required to further reduce risk is increased. In order to improve the quality of safety decisions, it is essential to improve the quantitative understanding of risk potential.

2.2 RISK: A SYSTEMS PERSPECTIVE

From a systems engineering perspective, a system can be considered to be an operator that transforms resources into a product or service. A manufacturing system is one which combines raw materials, machinery, and power into a product. A transportation system uses vehicle and power resources to provide the service of moving people and goods from one place to another.

A generalized system can be represented as a black box operator (Figure 2-1). Contained within the box is the equipment that constitutes the system. Inputs to the system are the consumable resources, such as energy, labor, and money. Outputs of the system are measurable units of production or service. These outputs are used to evaluate the overall performance of the system.

Risk can be considered as one of the outputs of a generalized system. Risk is not a physical output. Rather, it is an information output, which is but one component of an overall system performance evaluation. As mentioned earlier, whenever humans interact with the system, either

as consumers of that system or labor in the system, there is potential for risk of injury; minimization of risk is the pursuit of safety.

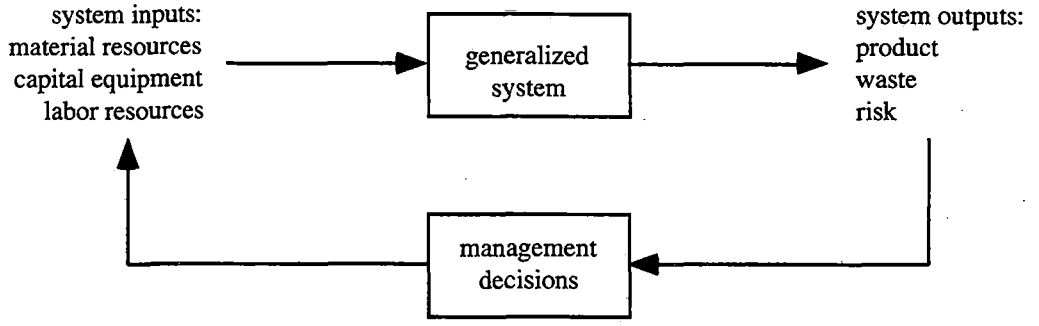


Figure 2-1. Closed-Loop System Diagram

The risk performance can be controlled through the system inputs. By changing the equipment used in a system, or the resources that are input into a system, it is possible to cause change in the safety performance of that system. Ultimately, these changes require resources—reduction in risk requires expenditure of time, materials, and money.

2.3 THE RELATIONSHIP BETWEEN SAFETY AND SYSTEM RELIABILITY

Risk assessment is, in effect, a subset of reliability engineering, which is focused on estimating the probability and effects of system failures. Quantitative reliability techniques are used to predict the likelihood of failure or estimate the availability of a machine system. When a system failure can result in injury or death to a human, it becomes a safety issue. Assessment of system reliability with respect to a failure of this type is risk assessment.

In general, the occurrence of human injury is preceded by an event known as an accident. As defined by (Senders and Moray, 1991), an accident is an "unwonted and unwanted exchange of energy." This definition captures two distinct characteristics of an accident: the event is unusual or unexpected, and it is undesirable. Or, expressed another way, an accident is system behavior which is outside the design expectations. In most cases, an accident represents a demarcation point in time—the events occurring before the accident contribute to the probability of accident occurrence, while the events occurring after the accident are in the domain of outcome analysis. In transportation systems, the accident event is alternatively known as a *crash* or *collision.*

Safety measures aimed at reducing the risk probability fall under the banner of crash avoidance, while measures aimed at reducing the risk outcome are known as crashworthiness.

Using the accident as the primary time mark of interest allows classification of two key components of risk. Risk probability is the relative likelihood of an accident event, while risk outcome is the cost of that accident in terms of human injury. It is interesting to note a non-reciprocal relationship between accident occurrence and injury outcome—while it is clear that injury is the result of an accident, accidents do not necessarily result in injury.

Because of high public visibility and reliance, transportation systems have evolved into extremely reliable and safe systems. As a result, accidents have become relatively rare. While this, in itself, is a positive occurrence, it makes more difficult the task of further improving the reliability and safety of the system. The only true measure of the safety of a system is the occurrence of failures, and by improving the safety, we reduce the opportunity to measure the performance.

Furthermore, because much of the more obvious safety flaws have been discovered and rectified, the remaining safety problems involve the occurrence of complex combinations of precipitating events. In other words, the causes of accidents have evolved from single-point failures into complex sequences of causal events, and the absence of any would effectively avert the accident. In effect, we have moved from the realm of first order effects into higher order effects. This, in combination with the rarity of accidents, makes the task of identifying causal factors much more difficult.

Sequences of events almost leading to the point of accident are commonly referred to as *near collisions* (and sometimes mistakenly called *near misses*). A guiding motivation in this work is the notion that these near collisions are far more common than actual accidents. If we have the capability of identifying near collisions and the conditions that lead to them, responses (either in design, operating procedure, or policy) can be formulated to reduce the occurrence of near collisions, and in the process reduce the number of accidents.

Risk probability, especially in transportation systems, is not a static quantity. Instead, risk probability varies as a function of the state of the system, which includes the state of the vehicle and operator as well as the state of the environment. The system state in transportation systems

is quite dynamic with respect to time. Operators are responsible for a constant stream of control decisions and actions which determine the state of the vehicle in relation to the state of the environment. Thus, through these control decisions, the operator has a profound impact on the risk probability of the system. Many accident scenarios are the result of compounding several hazard conditions, each of which may be relatively innocuous when occurring in isolation. Some of these hazards may be due to operator errors, while others may be due to machinery failures in vehicle or wayside equipment. The collected set of potential hazard conditions leading to a particular accident scenario can be considered to be a system state. Because this state varies with time, and the risk probability is a function of this state, risk probability can also be considered to be a function of time.

Time is an integral component of dynamic risk probability. Using probability theory, we can model the risk probability as the relative likelihood of the occurrence of an accident. However, the risk probability of an accident only makes sense if we compare its occurrence to the alternative event, which is the non-occurrence of an accident. Since nothing "happens" during the non-occurrence, the event can only be considered with respect to some fixed metric. The safety state model considers the probability of an accident with respect to a fixed time frame, known as a *time slice*. Thus, the risk probability represents the relative likelihood of an accident in a single time slice. On the average, it also represents the percentage of time slices that result in an accident. An alternate expression is in terms of the mean time (in terms of the number of time slices) between occurrence of accidents. This form is commonly known as the mean time to failures (MTTF), and is used extensively in the field of reliability engineering. The relationship between risk probability and MTTF plays an integral role in the development of the safety state model.

Many techniques have been developed for quantitative system reliability and safety analysis. In particular, the methods of fault tree analysis and event tree analysis provide useful background (Swain and Guttmann, 1983) (Lewis, 1987) (McCormick, 1981) (Gertman and Blackman, 1994). Both methods utilize a tree structure to organize the events which can lead to failures in complex systems. The event tree method is considered *forward-looking* in that the analysis commences with a precipitating event and explores the possible consequences in light of subsequent decisions and actions. By comparison, the fault tree method is *backward-looking*—the analysis starts with a failure event and works backward to evaluate the possible combination of events

that might have led to that failure. Both techniques can be used quantitatively or qualitatively. While these are powerful techniques with wide application, an acknowledged weakness in both techniques is difficulty in accounting for the time relation of events, especially relative to application in transportation system analysis.

Markov process models have been employed in the related area of system reliability. Babcock (1986) demonstrates the use of Markov process models to simplify the quantitative reliability analysis in fault-tolerant systems. In this work, he contrasts the Markov process modeling technique to both fault tree methods and mean-time-to-failure methods. In addition, several methods for reducing the size of the Markov network are demonstrated, to ease the loads of data storage and computation. Lewis (1987) discusses the use of Markov process modeling to evaluate system reliability. He takes the significant step of using Markov states to represent the system states (in contrast, Babcock uses Markov states to represent events). This concept is used in the safety state model.

The safety state model is an extension and combination of event tree and fault tree models, using a discrete finite Markov process. By considering all combinations of the possible precipitating events, the safety state model is the most general form of these. In addition, the time-based nature of dynamic risk probability is captured through use of the Markov process model. Thus, the safety state model represents a step forward in risk probability estimation techniques. A detailed development of the safety state model is contained in section 3.

2.4 THE HUMAN ELEMENT IN RISK

When comparing the relative safety of transportation modes, the traditional approach is to compare statistics relating the number of fatalities or injuries to a performance metric. For example, it would be reasonable to express risk as the fatalities per passenger-mile, or the injuries per vehicle-mile. While this technique is useful for comparing the safety performance of two or more competing modes of transportation, it does not provide for deeper analysis within one specific mode or system.

Consider, for a moment, highway safety. It is generally believed that human operator error is at least partly accountable for over 80 percent of all highway accidents (Shinar, 1978). If we consider the human operator to be part of a closed-loop control system (Figure 2-2), it is clear

that there could be three general causes: failure to properly sense the system state, failure to make a correct decision, and failure to execute a decision (control action). Yet, it would be very difficult to use accident statistics to identify the faults, relative to these three broad classes, that lead to the accidents.

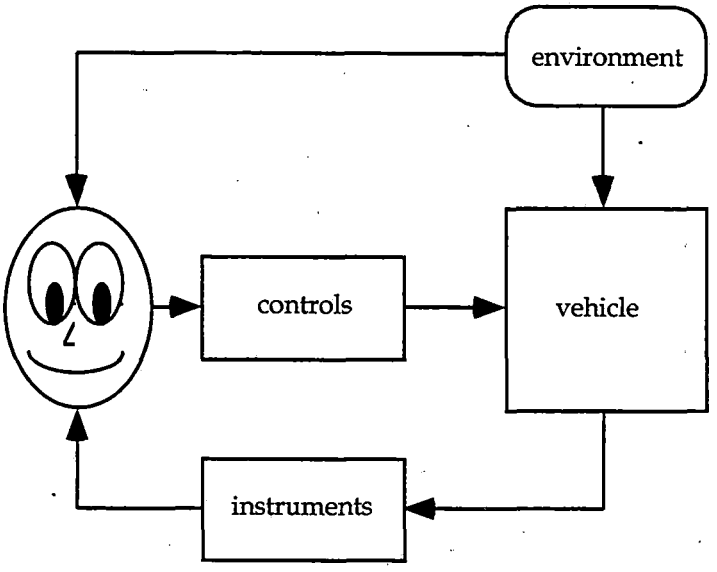


Figure 2-2. Closed-Loop Vehicle Control

Human-machine interaction, by nature, raises the potential for safety issues. First of all, safety has been defined relative to the risk of human injury—if there is no human-machine interaction, then by definition there is no safety problem. The interaction between human and machine in a transportation system can occur from the perspective of either an operator or a consumer (i.e., passenger). Second, human behavior is generally not deterministic, and the inclusion of a human control element in a system represents a significant opportunity for error, which can lead to a higher level of risk.

Fitts (1951) published a comparison of the performance of humans and automation for certain classes of tasks. His list identifies classes of tasks which are well suited for human operators, and those which are not. In many cases, the use of automatic control would eliminate the potential for human error leading to safety problems. Yet, for a variety of reasons, human controllers are often employed in systems where automation might be more sensible. One factor is the difficulty of qualifying sophisticated automatic control elements with regard to overall system safety—it becomes more difficult to prove the safety of an automated system under all

possible scenarios as the system grows in complexity. Safety qualification of human-operated systems should be at least as difficult, especially when the human operator must interface with increasingly complex systems, because it is impossible to prove that a human operator will have sufficient resources to handle any possible scenario. However, from a policy point of view, it is believed that properly trained human operators will have adequate facility to handle any foreseeable emergency scenario.

2.5 SYSTEM ANALYSIS

When analyzing a system with respect to risk, there are several points of interest. The first is identification of the conditions or combinations of conditions that pose a high risk. Simply said, we want to find out which system states (i.e., combination of potential causal conditions) present the highest levels of risk. This is an essential step in system safety analysis, and it must be taken before any form of corrective action can be made to the system.

A second area of interest is identification of the causal chain of events that lead to the high risk states. This is a more subtle effort, and requires observation and monitoring of the system in the time period prior to the event of interest. In the event that it is impossible to completely eliminate the risk via system design, the next possible step toward ameliorating the risk is to reasonably assure that the high risk state is avoided. Identification of the causal chain can help in this regard.

As a simple example, consider the problem of grade crossing safety in rail systems. Grade crossings are an acknowledged safety problem in rail operations, and, as a result of human nature and impatience, they will likely remain a problem in the future. The obvious design solution is separation of the rail guideway from the highway paths, so as to eliminate the crossing altogether. However, this solution is not feasible in many places, due to the high cost. An understanding of the causal chain (i.e., the train and car are approaching the crossing; the car driver does not wish to wait for the train to pass, so an attempt is made to cross before the train; because of the long stopping distances, the train is not able to slow in time to miss the car; the car is demolished) leads to the intermediate solution of grade crossing barriers—by blocking the opportunity to cross before the train, the risk level is reduced.

2.6 CONTROLLING OPERATOR BEHAVIOR

Let us now extend the closed-loop control paradigm to a different level. Consider a control loop, where the "plant" is the operator. The output of the "plant" is the safety-related behavior, and the controller is intended to regulate the safety-related behavior through education.

In many forms of operator training, there are two possible impediments to this system. First of all, there is no effective objective method for measuring the safety-related behavior of the operator. Without this measure, it is virtually impossible to regulate the behavior. Secondly, the input to the "plant" (a.k.a. operator training) occurs only for a relatively short period of time, typically at the beginning period of operator experience. In effect, these factors lead to a control loop which is not closed.

One potential application of the safety state model is to provide an effective measure of the safety-related operator performance in a dynamic sense. This, combined with continuing education, will allow effective closed-loop control of operator safety performance.

3. THEORY: SAFETY STATE MODEL

Having determined that our goal is to express dynamic risk probability as a function of system state, we now proceed with the task of developing the theory that will accomplish that goal. This theory is centered on a model of system behavior, which can be used to predict the future based on some experience.

The systems that we are concerned with include both machines and human operators. (In this sense, any human that interacts with the system and can change the outcome is considered an operator, whether they are trained for the task or not.) Although deterministic models are often useful for machines, humans are much more variable in their perception, decision, and actuation processes. As a result, a stochastic model is more appropriate for modeling a human-machine system. This approach employs a finite-state Markov process model.

3.1 BRIEF REVIEW OF MARKOV PROCESS THEORY

Before considering Markov process theory and analysis, we must define the term *system state*. In the field of system dynamics, the state of the system can be represented as a vector of numbers, with each element of the vector corresponding to a physical entity. In general, a system state vector is continuous with respect to time. Each state element might be measured by an instrument, or perhaps estimated by a mathematical model.

As an example, consider the dynamic state of a rail vehicle. If we know the path of the tracks and assume that the tracks do not move, it might be sufficient to express the state of the vehicle in terms of its position along the tracks and its speed, each of which is a scalar number. Each of these two state variables are used to represent a continuous physical parameter. Combining these into a two-element vector is an expression of the system state.

If a state vector has n elements, it can be considered a vector in n-space. If each unique collection of state variable values is considered to be a separate "state," then there are an infinite number of states possible in this n-space.

The notion of system state, as described above, correlates the individual state elements to physical quantities. This is distinct from the general concept of state as used in discrete Markov process analysis. In finite discrete Markov processes, the number of allowable states is limited to

a finite positive integer. For the moment, let us ignore the precise definition of those states, except that they are mutually exclusive (i.e., the system cannot be in more than one Markov state at any given time). Each individual state has an identification number associated with it, which serves as a label for the state. In most cases, the state numbers constitute a sequential set of non-negative integers—an *n*-state Markov model would have state numbers from 1 through *n* (or perhaps 0 through *n-1*). The notation *S(i)* indicates that the system is currently in the state labeled *i*.

A useful metaphor, used extensively by Howard (1971), is a set of lily pads on a pond. Each lily pad represents a Markov state. A frog, representing the system, jumps from lily pad to lily pad. The jumps are instantaneous, so the frog is on one of the lily pads at any point in time. Whenever the frog moves from one lily pad to another, a state transition is said to have occurred.

A simple example to illustrate this concept is a coin tossing process. Suppose we are interested in the combinations of the three most recent coin tosses. There are only 8 possible combinations: HHH, HHT, HTH, HTT, THH, THT, TTH, and TTT (where the last letter represents the most recent toss in the sequence). We are only concerned with the three most recent tosses—anything prior to that is irrelevant. Let us assign each of these combinations to a Markov state, as shown in Figure 3-1. Each box represents one of the states, and the directed lines represent a possible transition path. We can see, for example, that it is possible to go from state HHH to HHT with the toss of a tails, but it is impossible to go from HHH to HTH, because we cannot change the state of the previous toss. Each of the states is assigned a state number. (In this example, the selection of state numbers has no meaning beyond identification of the individual states.)

In the discrete Markov process, the opportunity for a state transition (i.e., the chance for the frog to jump from one lily pad to another) occurs at specified points in time. In the coin toss example, these state transition opportunities occur each time the coin is tossed. In dynamic systems, an alternative approach is to define transition opportunities which occur at fixed intervals in time. Note that these are couched as "opportunities"—at a transition point, the system may change state, or it may remain in the same state. In the coin toss example, the transition path that leads from HHH back to itself represents a case where the system remains in the same state, when three consecutive head tosses are followed by a fourth.

If we consider that each transition path has a probability associated with it (Figure 3-1), we can organize these probabilities in the form of a matrix, known as the state transition matrix and denoted as P. Each element, $p_{i,j}$, is the probability of transition from state *S(i)* to state *S(j)*. For a Markov process with n states, this state transition matrix will be of dimension $n \times n$. By convention, the row number corresponds to the state *S(i)* before the transition, and the column number corresponds to the state *S(j)* after transition. Because the probabilities of all paths leading from any one state must sum to 1, it follows that each row of the state transition matrix must also sum to 1.

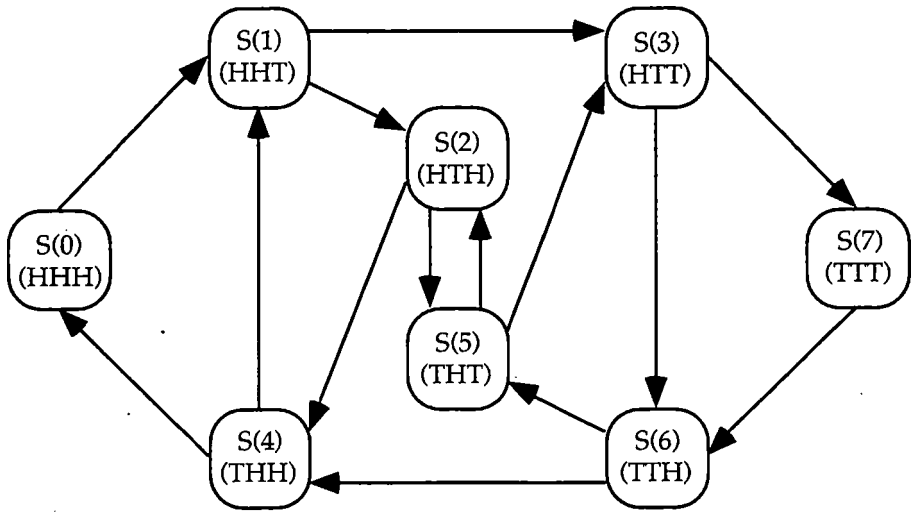


Figure 3-1. Markov Process for Coin-Toss Example

If a Markov state has one or more paths leading to it but no paths leading away, it is known as a trapping state. Once the system enters a trapping state, it remains there permanently. For trapping state *S(k)*, the corresponding row in the state transition matrix has a value of 1 for $p_{k,k}$, and 0 for all other entries.

The state transition matrix serves to summarize the probabilistic nature of the Markov process at each transition. In addition, it can be used to forecast the behavior of the system in the future. Consider the row vector $\pi(0)$, of dimension $1 \times n$, which contains a value of 1 in the position of the current state and 0 everywhere else. Since each row of the state transition matrix contains the probabilities of transition from the current state to any of the possible next states, we can utilize equation 3.1 to find $\pi(1)$, which summarizes the probabilities of being in any of the other states after one transition.

$$\pi(1) = \pi(0)P \tag{3.1}$$

Let's now look ahead one additional step. We can calculate $\pi(2)$ in the same manner ($\pi(2) = \pi(1)P$), based on our knowledge of $\pi(1)$. Substituting for $\pi(1)$, we can evaluate the probabilities of being in any of the other states at the second transition, based only on knowledge of the current state and the state transition matrix (equation 3.2).

$$\pi(2) = \pi(1)P = \pi(0)P^2 \tag{3.2}$$

If we continue the iteration, we can see that this relationship can be summarized in terms of the power of the state transition matrix (equation 3.3).

$$\pi(\tau) = \pi(0)P^{\tau} \tag{3.3}$$

Traditionally, the power of the state transition matrix is notated as $\Phi(\tau)$, following the relationship:

$$\Phi(\tau) = P^{\tau} \tag{3.4}$$

with

$$\Phi = P^{\infty} \tag{3.5}$$

being the limit as τ goes to infinity.

In the case of Markov processes having one trapping state, Φ will always have a single column of ones, in the column corresponding to the trapping state, and the remainder of the matrix is zero. This makes sense intuitively—given an infinite number of transitions, and the system will ultimately reach the trapping state, regardless of the transient states occupied.

This brief summary of Markov process analysis illustrates the basic tools used to form the safety state model, which provides a mechanism for expressing the risk probability of a system as a function of system state. The safety state model is an application of discrete finite Markov processes with a single trapping state.

3.2 THE SAFETY STATE MODEL—AN APPLICATION OF FINITE MARKOV PROCESSES

Our goal is to find a model which allows estimation of dynamic risk probabilities. As mentioned earlier, the motivation is straightforward—existing methods of safety analysis only provide mechanisms for determining the overall average safety of a system, without regard to individual operators or system design features. If we are to evaluate the safety-related behavior of individual operators within a system, we need a dynamic estimation of risk probability.

Ideally, such a model would provide an instantaneous risk probability value, and would be an analytical function which takes all of the pertinent state variables as input. However, there are practical problems with such an approach. Such a model would be deterministic by nature. But humans are far from deterministic, especially in response to real or perceived emergency situations. Another problem is the calibration of such a model—how do you obtain the values of the pertinent parameters? Still another problem is resolution of the model—how sensitive is the model to the accuracy of the parameters?

An intermediate approach is to divide the state space into a set of *domains*. By calculating the probability of reaching the failure event from any of these domains, we can achieve a more coarse estimate of dynamic probability than might have been achieved with an analytical model form. We define these domains to be the states in the discrete Markov process.

The development of the model requires several steps. First, the structure of the model is developed. In this step, the method for assigning Markov states is defined. Next, the method for determining the state transition probabilities is developed. Then, the algorithm for calculating the mean time to failure is derived. Finally, the technique for estimating the risk probability is described.

3.2.1 Structure of the Safety State Model

The safety state model is an application of a discrete Markov process model with a single trapping state. A set of domains in the continuous system state space are mapped into a set of Markov states, which form the model. To define these domains, let us consider a set of n binary conditions (i.e., states that are either true or false). All of these conditions are assumed independent. Each condition can be represented by a single bit of information (i.e., a 1 or a 0). If

we concatenate the set of *n* bits into a single number, that number can have a value from 0 to 2^{n-1}, and that set of 2^n numbers represents all of the possible combinations of those *n* conditions.

Each of the binary conditions used for definition of the safety state model is an expression of a safety related system state which may be a potential causal condition for the event representing the *bad outcome* (accident). A *causal condition* is defined as some system state value or range of values which may lead to or contribute to an accident.

In the safety state model, the definition of the Markov states in the safety state model forms the structure of the model. By using the set of conditions to define a number which is then interpreted as the Markov state number, we have created a link between the system state and the Markov state number used to represent that state. In the general case of a Markov process, this is not necessarily true—the state number is merely a label. In the safety state model application, the state number carries with it a definition of system state. The resultant set of Markov states represents the exhaustive list of all possible combinations of the specified safety-related causal conditions. As a result, the safety state model allows comprehensive investigation of many interacting conditions which, if possible at all, would be tedious and time-consuming using other methods.

Therefore, defining the potential causal conditions defines the structure of the model. It is important to note that the causal factors are described as conditions, and not as events. A condition is defined as a state of being, while an event can be defined as demarcation point in time, which might signal the change to or from a condition. This is an important distinction—an event is a point in time, while a condition occurs over some period of time.

The safety state model is based on the notion that there are a variety of condition combinations that might lead to a particular failure. Each condition is binary—it is either true or false. The binary condition may be based on a continuous state variable. For example, if vehicle speed is considered to be a factor, a binary condition based on speed could be whether the speed is above or below a specific threshold.

The set of binary conditions are then concatenated into a binary number, which is interpreted as the Markov state. This implies that there is an individual Markov state for each possible combination of the specified conditions. There is also the implication that the number of states

grows as a power of 2. That is, if there are *n* conditions specified, there are 2^n possible combinations of those states, leading to 2^n Markov states. In this context, these Markov states are known as operational states, as they represent the non-failure operation of the system.

One additional state must be specified. This failure state is defined as a trapping state, and it is predicated by the defined failure event. Or, alternately stated, the Markov transition to the failure state is used to represent the occurrence of an accident. The definition of the system failure state as a trapping state corresponds to the irreversible nature of a failure event.

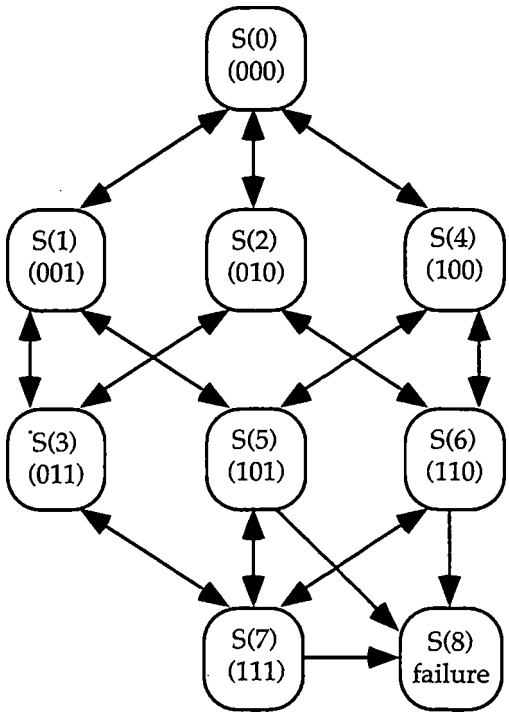


Figure 3-2. Simple Safety State Model

The structure of the safety state model which results from the definition of the safety state conditions provides a framework for observing an existing system. Once the failure event and the causal conditions of interest have been defined, one may observe an operational system and record the state occupancies and transitions. Such an observation results in a *safety state trajectory*, which is simply the statement of occupied state as a function of time.

A simple example of a safety state model is shown in Figure 3-2. Consider that there are three conditions in this example, leading to eight states plus the failure state. This diagram shows the

possible transition paths between the defined states, assuming that only one of the conditions can change in a state transition period.

3.2.2 Finding the State Transition Probabilities

We must now define the method used to obtain the probabilities for state transition matrix. In order to do this, we must first consider the micro-actions of the model.

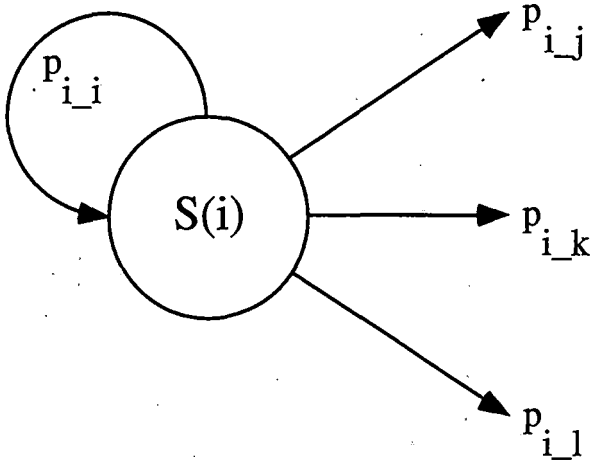


Figure 3-3. Transition Paths for a Single State

Assume that the model is currently in state *S(i)*, and from *S(i)* can transition only to state *S(j)* (Figure 3-3). At the next transition point, the system might remain in state *S(i)* (with a probability of $p_{i,i}$), or it might transition to some other state *S(j)* (with a probability of $p_{i,j}$). Therefore, if the system enters state *S(i)*, remains in that state for *k* transition periods, and then transitions to state *S(j)*, we can estimate that the probability $p_{i,i}$ is equal to $(k\text{-}1)/k$, and the probability $p_{i,j}$ is $1/k$. In other words, of the *k* transition opportunities in which the system was in state *S(i)*, *k-1* were spent holding the state *S(i)*, while 1 involved transition to state *S(j)*. Therefore, the relative likelihood (probability) of holding in state *S(i)* is $(k\text{-}1)/k$, and the relative likelihood of going to state *S(j)* is $1/k$.

To generalize this technique, the state transition probabilities are generated by collecting statistics about the state occupancies. By counting the occurrences of state holding times and transitions, the relative likelihood of each transition is computed. This is accomplished through the use of a construct known as the statistics matrix. This matrix is analogous to the state

transition matrix, and is used for accumulation of the state occupancy statistics. The matrix is denoted by S, and the individual elements are $s_{i,j}$.

Data from the safety state trajectories are accumulated into the statistics matrix. For each state occupancy, we need to know the current state (*S(i)*), the next state (*S(j)*), and the time spent in the state (t_i, expressed as a count of the transition intervals). We then increment $s_{i,i}$ by t_i-1 and increment $s_{i,j}$ by 1 (corresponding to the number of times each transition was experienced). This process is iterated for all the available safety state trajectories.

When all the safety state trajectory data have been incorporated, the state transition matrix is computed from the statistics matrix. This is done by computing the sum of each row in the statistics matrix,

$$s_i = \sum_j s_{i,j} \tag{3.6}$$

and dividing that sum into each element of the row.

$$p_{i,j} = {}^{s_{i,j}}\!/_{s_i} \tag{3.7}$$

As an example, let us consider a three-condition safety state model (Figure 3-2). There are nine states, numbered 0 through 8. Assume that we have observed that the system has entered state *S(1)* on 12 different occasions. The total holding time in state *S(1)* was 88 transition periods, and the system had transitions to state *S(0)* on 3 occasions, to state *S(3)* on 6 occasions, state *S(5)* on 2 occasions, and state *S(8)* once. The total time in *S(1)* was 100 transition periods. The resultant state transition probabilities from state *S(1)*, corresponding to row 1 in the state transition matrix, would be estimated as follows:

$p_{1,0}$	3/100 = 0.03
$p_{1,1}$	88/100 = 0.88
$p_{1,2}$	0
$p_{1,3}$	6/100 = 0.06
$p_{1,4}$	0
$p_{1,5}$	2/100 = 0.02
$p_{1,6}$	0
$p_{1,7}$	0
$p_{1,8}$	1/100 = 0.01

It is a distinct possibility that, in our system observations, certain states are not occupied. This does not imply that state will never get occupied, only that it was not occupied in our experience. To account for the possibility that states might conceivably be occupied, there is an initial default state of the state transition matrix. This initial state says that, until we have gained experience from the real world, we assume the most general form. That is, if the state were to be occupied, it would be occupied for a default amount of time, and the distribution of the transitions out of this state is equal among the possible states.

3.2.3 Finding the Mean Time to Failure From Each State

Let us assume we have a discrete Markov process with a single trapping state. Let us also assume that a state transition occurs at fixed regular intervals in time. Thus, a count of state transitions can be equated with a period of time, as the product of the number of transitions and the length of the time interval. (The system may remain in a given state for longer than one interval—this is captured with the holding probability, which is the probability for each state that the system will return to that state at the transition point.)

One of the fundamental questions to be answered is as follows: Given a particular starting state, what is the mean time for the process to reach the trapping state? This question is of particular importance if we consider the trapping state to indicate an undesirable failure event.

Let us consider a large discrete Markov process with n transient states and one trapping state. The states are numbered from 0 to n, with $S(n)$ as the trapping state. The P matrix is of dimension $(n+1)\times(n+1)$, and has the form

$$P=\begin{bmatrix} p_{0,0} & \cdots & p_{0,n} \\ \vdots & \ddots & \vdots \\ p_{n,0} & \cdots & p_{n,n} \end{bmatrix}=\begin{bmatrix} P_a & P_f \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

where P_a represents an *nxn* sub-matrix of the transition probabilities between operational (or *transient*) states, P_f represents an *nx1* column vector of the transition probabilities into the failure (trapping) state directly from the operational states, a *1xn* row vector of zeros (in the last row), and a single element of 1 for $p_{n,n}$.

The probability of state occupancy, as a function of the initial state, after τ transitions can be expressed by:

$$\Phi(\tau)=\begin{bmatrix} \Phi_a(\tau) & \Phi_f(\tau) \\ 0 & 1 \end{bmatrix}=\begin{bmatrix} P_a^{\tau} & \left(\sum_{i=0}^{\tau-1} P_a^i\right)P_f \\ 0 & 1 \end{bmatrix} \tag{3.9}$$

Note that a similar partitioning occurs, with the last column representing the probability of getting to the trapping state after τ transitions. The matrix Φ is the limit of $\Phi(\tau)$ as τ goes to infinity ($\Phi = \lim_{\tau\to\infty}\Phi(\tau)$). In the case of the P matrix for the safety state model, Φ will always have a column of ones in the rightmost column, and zeros elsewhere. (In other words, $\Phi_a = 0$ and $\Phi_f = 1$.)

However, in order to obtain the mean time to failure (MTTF), we are more interested in the probability distribution of getting to the failure state on the τ^{th} transition, which can be expressed as:

$$\Psi(\tau)=\Phi_f(\tau)-\Phi_f(\tau-1)=\left[\sum_{i=0}^{\tau-1}P_a^i-\sum_{i=0}^{\tau-2}P_a^i\right]P_f=P_a^{\tau-1}P_f \tag{3.10}$$

This is the first-order interarrival probability distribution.

We know that $P_a^\tau = P_a P_a^{\tau-1}$, so $P_a^{\tau-1} = P_a^{-1} P_a^\tau$. Therefore, we can say that:

$$\Psi(\tau) = P_a^{\tau-1} P_f = P_a^{-1} P_a^\tau P_f \tag{3.11}$$

The iterative method for calculating MTTF for each state is to sum the product of the number of transitions and the probability of the failure event occurring on the τ^{th} transition. This can be stated algorithmically as

$$M = \sum_{\tau=1}^{\infty} \tau \Psi(\tau) \tag{3.12}$$

where M is a *(n-1)x1* row vector containing the MTTF for each of the *n-1* operational states.

The disadvantage of this approach is that it is difficult, if at all possible, to predict the number of iterations required to achieve an acceptable answer. In addition, the mean time to failure is expressed as an infinite sum—it is also difficult to determine the bounds that will qualify an acceptable answer, even if we could predict the number of iterations required.

An alternative approach is to use transform analysis. By evaluating, at *z=0*, the differential of the z-transform of the first order interarrival time, the MTTF can be calculated directly.

$$M = \frac{d}{dz} \Psi(z) \Big|_{z=0} \tag{3.13}$$

Using the definition of $\Psi(\tau)$ from above (equation 3.11), we can compute the z-transform as:

$$\Psi(z) = P_a^{-1} \left[I - P_a z \right]^{-1} P_f \tag{3.14}$$

Substituting equation 3.14 into equation 3.13, the MTTF is

$$M = \frac{d}{dz} \Psi(z) \Big|_{z=0} = P_a^{-1} \left[I - P_a \right]^{-1} P_a \left[I - P_a \right]^{-1} P_f \tag{3.15}$$

This expression can be further simplified. We know that

$$\left[I - P_a \right]^{-1} = I + P + P^2 + \ldots \tag{3.16}$$

which means that

$$P_a[I-P_a]^{-1} = [I-P_a]^{-1}P_a \tag{3.17}$$

allowing us to simplify to

$$M = P_a^{-1}P_a[I-P_a]^{-1}[I-P_a]^{-1}P_f = [I-P_a]^{-1}[I-P_a]^{-1}P_f \tag{3.18}$$

Furthermore, we know that

$$P_f = \begin{bmatrix} 1-p_{0,0}-p_{0,1}-\ldots-p_{0,n-1} \\ 1-p_{1,0}-p_{1,1}-\ldots-p_{1,n-1} \\ \vdots \\ 1-p_{n-1,0}-p_{n-1,1}-\ldots-p_{n-1,n-1} \end{bmatrix} = [I-P_a]\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = [I-P_a]1 \tag{3.19}$$

where *1* is an *(n-1)x1* column vector of ones. Using this, the expression can be simplified further to

$$M = [I-P_a]^{-1}1 \tag{3.20}$$

An alternative derivation (Ross, 1985) states that

$$M_i = p_{i,0}M_0 + p_{i,1}M_1 + \ldots + p_{i,n-1}M_{n-1} + 1 \tag{3.21}$$

for each state *S(i)*. (This is based on the method of computing expectation by conditioning, where $E[X] = \sum_y E[X|Y=y]p\{Y=y\}$.) In vector form, equation 3.21 can be stated as

$$M = P_a M + 1 \tag{3.22}$$

which provides to the identical solution

$$M = [I-P_a]^{-1}1 \tag{3.20a}$$

Note that the result of equations 3.20 and 3.20a is a set of MTTF values, expressed as a vector. Each element of the vector represents the MTTF from a single Markov state. Thus, the vector

represents the MTTF as a function of Markov state, which is defined by a set of discrete system states.

This solution has been verified experimentally, using a 129-state Markov model. Using the iterative approach, the calculations required approximately 300,000 iterations to get to a reasonable (but inexact) solution, taking about 20 hours on a mid-range Silicon Graphics workstation. The closed-form solution required less than 10 seconds (on the same computer) to get an exact solution.

3.2.4 Converting MTTF to Risk Probability

In the previous section, we developed a method for calculating the mean time to failure for each non-failure state in the system. However, this is not enough—we are really interested in estimating the risk probability at each operational state in the system. The probability of transition directly from each operational state to the failure state is expressed as P_f. However, for most of the states, this probability is zero, meaning that it is not possible to go directly from a specific state to the failure state.

On the other hand, we know that the model will always eventually reach the failure state, given enough opportunity—this is inherent in a Markov process with a single trapping state (as evidenced in equation 3.5). What we really need to know is the relative likelihood of reaching the failure states, based on the current system state.

To accomplish this, the MTTF vector is transformed to an equivalent risk probability. Let us assume that we have a Bernoulli process (i.e., the "coin toss"), which has probability p_f of failure and probability $1 - p_f$ of success, occurring at points in time corresponding to each transition point. Because the failure event is rare, we expect p_f to be small. The MTTF for such a system is equal to $1/p_f$. (The units are consistent—since the probability values represent the likelihood per interval time, the inverse of the probability values are in units of time, appropriate for the MTTF.)

In the safety state model, we have calculated the MTTF for each non-failure state of the Markov process. By assuming the form of a Bernoulli process, we can thus estimate the equivalent risk

probability by the relationship $R_i \equiv 1/M_i$. Thus, the MTTF vector is transformed into a risk probability vector by an element-wise transformation.

This estimate underscores a very important concept: Although it may be impossible to reach the failure (trapping) state directly from a specified state, by nature the trapping state will always be reached eventually, regardless of the initial or transient state. Restated, although the probability of going directly from an operational state to the failure state may be zero, the probability of eventually reaching that state is one. Computation of the equivalent risk allows us to express the differences in risk between the various defined states in the system.

3.2.5 Characteristics of the State Transition Matrix

As shown earlier, the state transition matrix is used to compute the MTTF, and the MTTF is used to compute the equivalent risk probability. Thus, there is a direct link between the state transition matrix and the resultant risk probability estimation vector. What is the nature of the form or shape of the state transition on that risk probability estimation vector? And what do the risk probability numbers really mean?

We expect that, given a suitable time scale for the transition point intervals, the number of periods spent in each state (holding time) is significantly greater than the number of times the state is visited. Or, stated differently, the average hold time for each state is substantially greater than the transition intervals.

Thus, if we view the state transition matrix as a mesh diagram, we expect the topography to appear as a row of mountains down the main diagonal, with shorter "hills" scattered sparsely about the remainder of the area. In a mesh diagram, the horizontal axes represent the rows and columns of the two-dimensional state transition matrix, and the height of the surface represents the values of the individual state transition probabilities. (An example of the state transition matrix displayed as a mesh diagram is shown in Figure 4-1.)

3.3 IMPLEMENTATION—FOUR PHASES OF ANALYSIS

In the process of applying the safety state model to a practical risk analysis problem, there are four distinct phases of analysis. These are, in approximate chronological order, the risk

identification phase, system observation phase, model calibration phase, and risk estimation phase. Each phase represents a distinctly different type of activity, and can be represented with a set of inputs and outputs.

The risk identification phase is the starting point for the analysis. The analyst enters this phase with a general problem, and uses that problem to define the application of the safety state model. The system observation phase follows with the measurement of an actual system and the state trajectories of that system. The model calibration phase uses the data collected from the operational system to feed the safety state model, which results in a function that relates risk probability to system state. Finally, the risk estimation phase applies that risk probability function onto the measured state trajectories, resulting in a set of risk probability trajectories as a function of time. A flowchart of these phases is shown in Figure 3-4.

These phases of analysis can be considered to proceed in a chronological fashion, as the output of one feeds the input of the next. There is also room for iteration, especially in the latter two phases. However, this places an implication on the earlier two phases—in order to have latitude in the later phases, it is important to leave a number of doors open in the earlier phases. This will become apparent as we discuss the phases in greater detail.

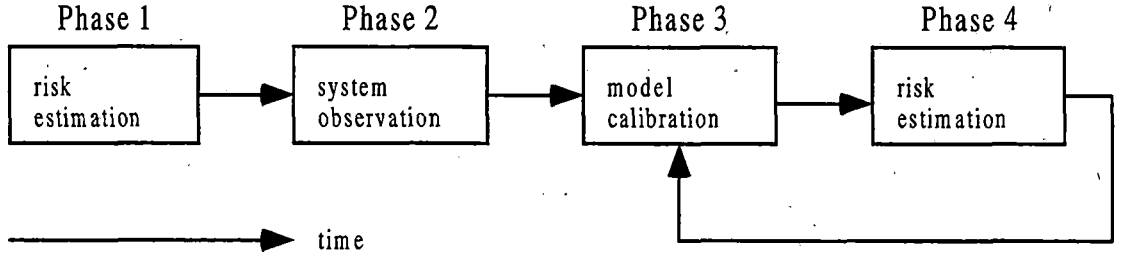


Figure 3-4. Flowchart of Safety State Model Operational Phases

The risk identification phase is the starting point for the analysis. It is in this phase that the analyst clearly identifies the risk event of interest and the conditions that might lead to it. These define the size of the required safety state model. The input of this phase is a problem, while the output is a failure event and a set of conditions that are believed to lead to that failure.

The system observation phase is the period of time during which an operational system is observed and its behavior is recorded. The input to this phase is a failure event and set of

conditions that might contribute to its occurrence, and the output is a set of state trajectories that result from measured system behavior.

At the commencement of this phase, the analyst must transform the failure event and set of contributing conditions into a set of measurements to be made on the system. Each condition of interest must be defined in terms of events that determine the beginning and end of that condition. The required measurements may be of two possible forms: event recording, and continuous variable recording. The former is appropriate for events that take place, such as operators pressing buttons or components failing. The latter form is appropriate for measuring variables which truly are continuous in nature, such as vehicle speed or position, or perhaps the position of a control lever.

Once the specific system measures have been determined, the system is instrumented and observed. In practice, this phase represents the most labor intensive, and consumes the largest proportion of the project time. Because of the amount of effort required in this phase, it is appropriate, in the risk identification phase, to overspecify the conditions to be used. By identifying a large number of conditions, we assure ourselves that enough data will be collected during the system observation phase to allow some flexibility during the subsequent phases.

The model calibration phase consists of a series of calculations, which transform the measured system behavior from the system observation phase into the dynamic risk probability function used by the risk estimation phase. This is the phase which utilizes the Markov process model.

The risk estimation phase is the time when everything comes together—the risk probability function (output from the model calibration phase) is applied to the safety state trajectories (output from the system observation phase), transforming them into risk probability trajectories as a function of time. These represent the grand output of the entire process.

3.4 PRACTICAL LIMITATIONS

One of the disadvantages of using a Markov process model is that the number of states grows very quickly with increasing number of variables, and the number of elements in the state transition matrix grows as the square of the number of states. In the case of the safety state model, this growth occurs exponentially. In a system with n conditions, the number of states is

approximately 2^n, and the number of elements in the state transition matrix is approximately 2^{2n}. In any computer-based system implementation, there are two fundamental constraints: processor speed and memory size. Both of these constraints will have implications on the maximum practical size of the safety state model.

During the model calibration phase, the state transition matrix is converted to an equivalent risk probability function (as a function of safety state) via matrix inversion. This matrix inversion is the single most costly computation required in the method. The number of multiplication operations required to perform a general matrix inversion is roughly $O(N^3)$, and $N=2^n+1$. Therefore, a 15-condition system would require roughly $3x10^{13}$ multiplication operations, which would require about 300,000 seconds (about 3.5 days) using a dedicated computer with 100 Mflop performance. Each additional condition in the safety state model raises the required processing time by a factor of 8.

In terms of data storage, the largest load is due to the model calibration phase for storage of the state transition matrix (which contains approximately 2^{2n} elements). Assuming that high-precision floating point numbers are used (each using 8 bytes), a 15-condition model will require about 8 Gigabytes of memory. Each additional condition in the safety state model raises the memory requirement by a factor of 4.

In summary, under the constraints imposed by current conventional computer technologies, safety state model analysis is limited to roughly 15 contributing conditions. Larger models could be accommodated using more advanced technologies, such as super computers or parallel-processing machines. Investigation of such techniques may constitute a subject for future research.

4. EXPERIMENTAL DEMONSTRATION

The previous chapter outlines the theoretical derivation of the safety state model. In this chapter, the safety state model is applied to an active safety issue, in order to demonstrate the procedure of using the method. The venue of high-speed rail is an excellent application for understanding the safety state model. The experimental demonstration of the safety state model was coordinated with a human subject experiment investigating the relationship between vehicle control automation and operator situation awareness in high-speed rail, the details of which are described in (Lanzilotta & Sheridan, 1998). The research was conducted as a laboratory experiment in human behavioral response, using the high-speed rail simulation system.

In a sense, demonstration of the safety state model was piggy-backed onto the control automation experiment. That is, the control automation experiment was conducted as a formal controlled behavioral study, addressing experimental design issues as necessary. The data used for demonstration of the safety state model was then extracted from the immense reserves of operational data, recorded during the control automation experiment.

4.1 MOTIVATION AND GOALS

The motivation of this component of the work was to demonstrate that the safety state model is a viable technique for estimating the dynamic risk probability in an operational high-speed rail system. Typically, a formal validation process includes comparison of test results from a new technology to those from an existing method. However, in the case of the safety state model, there was no known method for estimating dynamic risk probability in an operational system. Therefore, the validation is limited to demonstration that the method can discern differences in risk probability as a function of system state and that the data generated from the model is useful for system analysis.

4.2 PHASE 1 RESULTS—RISK IDENTIFICATION

In this phase, the task was to identify the risk event of interest, and to identify the conditions believed to contribute to that risk. The risk event used was a grade crossing collision between rail and highway vehicles. Due to the significant momentum involved, rail vehicles have braking distances which are very long (on the order of several miles, in some cases). When a train

approaches a grade crossing, highway vehicles often attempt to "beat the train" in an attempt to avoid waiting while the train crosses. Despite the presence of barrier systems, grade crossing collisions are a serious problem in operational rail systems, and the situation only promises to worsen in high-speed rail operation.

A seven-bit safety state model was defined, using the following five conditions:

1. distance to crossing — four mutually exclusive cases:
    a) near (vehicle cannot be stopped before crossing, even with use of the emergency brake)
    b) medium (vehicle can be stopped before crossing with use of emergency brake, but cannot be stopped with full service braking)
    c) far (vehicle can be stopped before crossing with the service brakes)
    d) vehicle is stopped (distance to crossing is irrelevant)
2. service brakes applied —true if the service brakes are applied
3. emergency brakes applied —true if the emergency brakes are applied
4. automation mode — four mutually exclusive cases:
    a) manual control (operator manually operates throttle and brake under all conditions)
    b) cruise control (operator sets a cruising speed, and automation system modulates the throttle to maintain that speed)
    c) programmed stop (operator selects a stopping point, and automation system modulates the brakes to stop at that point)
    d) autopilot (operator enables system, and automation follows a pre-programmed speed trajectory, as a function of vehicle position along track)
5. obstruction present — true if a highway vehicle is stuck at the grade crossing

Note that conditions 1 and 4 are compound conditions and have four mutually exclusive states each. Using the technique for combining multiple mutually exclusive conditions, as described in chapter 3, each of these conditions was combined into two bits of the safety state number. For example, the four cases in the automation mode are manual mode, cruise control, programmed stop, and autopilot. In the general case, these four conditions would require four independent bits in the safety state number. However, because we know that these four conditions are, by system design, mutually exclusive (i.e., if one mode is active, the others cannot be), these are represented with a two bit sub-word—00 for manual, 01 for cruise control, 10 for programmed stop, and 11 for autopilot.

The seven-bit safety state number ranges from 0 to 127. Thus, there were 128 operational states. The risk event, collision in the grade crossing, is defined as state 128. State 128 is a trapping state—once the system experiences a collision, it cannot be "undone." The resulting state transition matrix is of dimension 129 x 129. The resultant set of system states is listed in Table 4-1.

Another parameter specified is the interval time scale for the Markov process. In this case, we used an interval of one second. In general, it is not required that this parameter be specified during the risk identification phase, as tuning of this interval is useful during the model calibration phase. However, by virtue of the fact that this interval should not be much smaller than the resolution of the time stamp in the data recording, a preliminary specification was set at 1 millisecond, allowing a proper bound in the time resolution of the system observation phase.

Table 4-1a. — Summary of Safety States in Seven-Bit Model

state	hex	description
0	0x00	nobstruct, manual, nestop, nbrake, stopped
1	0x01	nobstruct, manual, nestop, nbrake, far
2	0x02	nobstruct, manual, nestop, nbrake, close
3	0x03	nobstruct, manual, nestop, nbrake, medium
4	0x04	nobstruct, manual, nestop, brake, stopped
5	0x05	nobstruct, manual, nestop, brake, far
6	0x06	nobstruct, manual, nestop, brake, close
7	0x07	nobstruct, manual, nestop, brake, medium
8	0x08	nobstruct, manual, estop, nbrake, stopped
9	0x09	nobstruct, manual, estop, nbrake, far
10	0x0a	nobstruct, manual, estop, nbrake, close
11	0x0b	nobstruct, manual, estop, nbrake, medium
12	0x0c	nobstruct, manual, estop, brake, stopped
13	0x0d	nobstruct, manual, estop, brake, far
14	0x0e	nobstruct, manual, estop, brake, close
15	0x0f	nobstruct, manual, estop, brake, medium
16	0x10	nobstruct, cruise, nestop, nbrake, stopped
17	0x11	nobstruct, cruise, nestop, nbrake, far
18	0x12	nobstruct, cruise, nestop, nbrake, close
19	0x13	nobstruct, cruise, nestop, nbrake, medium
20	0x14	nobstruct, cruise, nestop, brake, stopped
21	0x15	nobstruct, cruise, nestop, brake, far
22	0x16	nobstruct, cruise, nestop, brake, close
23	0x17	nobstruct, cruise, nestop, brake, medium
24	0x18	nobstruct, cruise, estop, nbrake, stopped
25	0x19	nobstruct, cruise, estop, nbrake, far
26	0x1a	nobstruct, cruise, estop, nbrake, close
27	0x1b	nobstruct, cruise, estop, nbrake, medium
28	0x1c	nobstruct, cruise, estop, brake, stopped
29	0x1d	nobstruct, cruise, estop, brake, far
30	0x1e	nobstruct, cruise, estop, brake, close
31	0x1f	nobstruct, cruise, estop, brake, medium

Table 4-1b. — Summary of Safety States in Seven-Bit Model (continued)

state	hex	description
32	0x20	nobstruct, pstop, nestop, nbrake, stopped
33	0x21	nobstruct, pstop, nestop, nbrake, far
34	0x22	nobstruct, pstop, nestop, nbrake, close
35	0x23	nobstruct, pstop, nestop, nbrake, medium
36	0x24	nobstruct, pstop, nestop, brake, stopped
37	0x25	nobstruct, pstop, nestop, brake, far
38	0x26	nobstruct, pstop, nestop, brake, close
39	0x27	nobstruct, pstop, nestop, brake, medium
40	0x28	nobstruct, pstop, estop, nbrake, stopped
41	0x29	nobstruct, pstop, estop, nbrake, far
42	0x2a	nobstruct, pstop, estop, nbrake, close
43	0x2b	nobstruct, pstop, estop, nbrake, medium
44	0x2c	nobstruct, pstop, estop, brake, stopped
45	0x2d	nobstruct, pstop, estop, brake, far
46	0x2e	nobstruct, pstop, estop, brake, close
47	0x2f	nobstruct, pstop, estop, brake, medium
48	0x30	nobstruct, autop, nestop, nbrake, stopped
49	0x31	nobstruct, autop, nestop, nbrake, far
50	0x32	nobstruct, autop, nestop, nbrake, close
51	0x33	nobstruct, autop, nestop, nbrake, medium
52	0x34	nobstruct, autop, nestop, brake, stopped
53	0x35	nobstruct, autop, nestop, brake, far
54	0x36	nobstruct, autop, nestop, brake, close
55	0x37	nobstruct, autop, nestop, brake, medium
56	0x38	nobstruct, autop, estop, nbrake, stopped
57	0x39	nobstruct, autop, estop, nbrake, far
58	0x3a	nobstruct, autop, estop, nbrake, close
59	0x3b	nobstruct, autop, estop, nbrake, medium
60	0x3c	nobstruct, autop, estop, brake, stopped
61	0x3d	nobstruct, autop, estop, brake, far
62	0x3e	nobstruct, autop, estop, brake, close
63	0x3f	nobstruct, autop, estop, brake, medium

Table 4-1c. — Summary of Safety States in Seven-Bit Model (continued)

state	hex	description
64	0x40	obstruct, manual, nestop, nbrake, stopped
65	0x41	obstruct, manual, nestop, nbrake, far
66	0x42	obstruct, manual, nestop, nbrake, close
67	0x43	obstruct, manual, nestop, nbrake, medium
68	0x44	obstruct, manual, nestop, brake, stopped
69	0x45	obstruct, manual, nestop, brake, far
70	0x46	obstruct, manual, nestop, brake, close
71	0x47	obstruct, manual, nestop, brake, medium
72	0x48	obstruct, manual, estop, nbrake, stopped
73	0x49	obstruct, manual, estop, nbrake, far
74	0x4a	obstruct, manual, estop, nbrake, close
75	0x4b	obstruct, manual, estop, nbrake, medium
76	0x4c	obstruct, manual, estop, brake, stopped
77	0x4d	obstruct, manual, estop, brake, far
78	0x4e	obstruct, manual, estop, brake, close
79	0x4f	obstruct, manual, estop, brake, medium
80	0x50	obstruct, cruise, nestop, nbrake, stopped
81	0x51	obstruct, cruise, nestop, nbrake, far
82	0x52	obstruct, cruise, nestop, nbrake, close
83	0x53	obstruct, cruise, nestop, nbrake, medium
84	0x54	obstruct, cruise, nestop, brake, stopped
85	0x55	obstruct, cruise, nestop, brake, far
86	0x56	obstruct, cruise, nestop, brake, close
87	0x57	obstruct, cruise, nestop, brake, medium
88	0x58	obstruct, cruise, estop, nbrake, stopped
89	0x59	obstruct, cruise, estop, nbrake, far
90	0x5a	obstruct, cruise, estop, nbrake, close
91	0x5b	obstruct, cruise, estop, nbrake, medium
92	0x5c	obstruct, cruise, estop, brake, stopped
93	0x5d	obstruct, cruise, estop, brake, far
94	0x5e	obstruct, cruise, estop, brake, close
95	0x5f	obstruct, cruise, estop, brake, medium

Table 4-1d — Summary of Safety States in Seven-Bit Model (continued)

state	hex	description
96	0x60	obstruct, pstop, nestop, nbrake, stopped
97	0x61	obstruct, pstop, nestop, nbrake, far
98	0x62	obstruct, pstop, nestop, nbrake, close
99	0x63	obstruct, pstop, nestop, nbrake, medium
100	0x64	obstruct, pstop, nestop, brake, stopped
101	0x65	obstruct, pstop, nestop, brake, far
102	0x66	obstruct, pstop, nestop, brake, close
103	0x67	obstruct, pstop, nestop, brake, medium
104	0x68	obstruct, pstop, estop, nbrake, stopped
105	0x69	obstruct, pstop, estop, nbrake, far
106	0x6a	obstruct, pstop, estop, nbrake, close
107	0x6b	obstruct, pstop, estop, nbrake, medium
108	0x6c	obstruct, pstop, estop, brake, stopped
109	0x6d	obstruct, pstop, estop, brake, far
110	0x6e	obstruct, pstop, estop, brake, close
111	0x6f	obstruct, pstop, estop, brake, medium
112	0x70	obstruct, autop, nestop, nbrake, stopped
113	0x71	obstruct, autop, nestop, nbrake, far
114	0x72	obstruct, autop, nestop, nbrake, close
115	0x73	obstruct, autop, nestop, nbrake, medium
116	0x74	obstruct, autop, nestop, brake, stopped
117	0x75	obstruct, autop, nestop, brake, far
118	0x76	obstruct, autop, nestop, brake, close
119	0x77	obstruct, autop, nestop, brake, medium
120	0x78	obstruct, autop, estop, nbrake, stopped
121	0x79	obstruct, autop, estop, nbrake, far
122	0x7a	obstruct, autop, estop, nbrake, close
123	0x7b	obstruct, autop, estop, nbrake, medium
124	0x7c	obstruct, autop, estop, brake, stopped
125	0x7d	obstruct, autop, estop, brake, far
126	0x7e	obstruct, autop, estop, brake, close
127	0x7f	obstruct, autop, estop, brake, medium
128	0x80	collision

4.3 PHASE 2 RESULTS—SYSTEM OBSERVATION

In the system observation phase, an operational system was observed and data from that system were recorded. The system used for system observation was the high-speed rail simulation system, while in operation for the control automation experiment.

During the experiment, data records were recorded into a disk file by the train simulation program. The recorded data included periodic summaries of the vehicle state, with the vehicle position, speed, and operator input (throttle position) recorded. The data were recorded nominally at 600 millisecond intervals. A higher sampling rate was used when the operator was moving the control lever, enabling more accurate capture of operator actions. Also included in the data file were records of the operator input at the control buttons, as well as state changes within the vehicle (such as vehicle system failures).

Because such a comprehensive collection of data was recorded, the resultant data files are large. Typically, the raw data file for a three-hour session is on the order of 1.5 to 2 megabytes in size. The raw data file was post-processed by a program named `ss_process`, which reduces the raw data to a safety state trajectory as a function of time. The resultant collection of safety state trajectories was used in phase 3 (model calibration).

Even before entering the model calibration phase, the safety state trajectories provided useful information. It was feasible to quickly identify the occurrences of the risk event, and the sequence of states that led up to each risk event. An intuition may be developed about the relative occurrence of risk events and the causality leading to those occurrences. Such an intuition, in itself, is useful in general risk assessment and safety engineering.

For example, inspection of 96 safety state trajectories recorded during the control automation experiment[1] allowed identification of 10 grade crossing collisions. Table 4-2 lists the sequence of states that occurred in the 300 second (i.e., 5 minute) interval prior to each collision[2]. The

---

[1] The safety state trajectories used for this comparison were gathered simultaneously with data gathered for the control automation experiment. The trajectories included those generated by the formal experimental subjects, as well as those generated by preliminary subjects. In addition, safety state trajectories from training session road tests were included.

[2] The "test type" field in table 4-2 refers to the level of control automation used during the test run, where "manual" refers to the case where the operator is in full manual control of the vehicle, "partial" refers to the case where the

state numbers correspond with the list provided in Table 4-1. We can see that states 70, 78, and 114 can be considered high-risk states, as the state immediately preceding a collision event is always one of these three. State 70 is the condition where there is an obstruction present in the grade crossing, the grade crossing is at close range (i.e., the vehicle cannot be stopped before the crossing using the emergency brake), and the operator is braking with the service brakes only. State 78 is a similar condition, except that the operator has engaged the emergency brakes. State 114 is the condition where there is an obstruction in the grade crossing, the crossing is at close range, but the operator is using the autopilot—neither the service brakes or the emergency brakes have been activated.

Table 4-2. — Summary of Collision Occurrences

subject number	test type	state sequence prior to collision
2	manual	5-1-5-1-3-2-1-3-2-1-65-69-71-70-128
2	full	49-1-5-1-49-51-50-49-113-65-69-71-69-71-70-128
3	partial	17-1-5-1-17-1-5-21-5-1-17-19-83-67-71-70-78-128
3	full	45-44-36-32-33-32-0-4-0-1-49-51-50-49-51-115-114-128
6	full	49-113-115-114-122-74-78-128
10	practice	1-5-7-71-67-71-70-78-128
11	full	49-1-5-1-49-113-115-67-71-70-128
13	practice	1-5-1-5-1-5-1-3-7-6-2-1-5-7-3-67-71-70-128
18	practice	1-5-1-5-1-5-1-65-67-71-70-128
18	full	49-113-115-114-66-70-78-128

4.4 PHASE 3 RESULTS—MODEL CALIBRATION

The purpose of the model calibration phase is to transform the safety state trajectories, generated during the system observation phase, into a risk transformation function. The risk transformation function summarizes the relationship between the safety state and the risk probability of failure. The algorithms for these calculations are presented in chapter 3.

---

operator utilizes both automatic speed control and automated stopping systems, "full" refers to the case where the operator utilizes a full autopilot system, and "practice" refers to operator training sessions.

The collected raw data identifies changes in the binary conditions (used to construct the safety state values) and the time marks at which those changes occurred. Post-processing of the data resulted in a time-based trajectory of the safety state values over the course of each test session. From the safety state trajectory, we have identified the number of transitions into each state and the length of time, in terms of transition periods, that each state remained occupied; this data was accumulated in the statistics matrix (section 3.2.2). The selected time period was one second. From the statistics matrix, the elements of the Markov state transition matrix were calculated (equation 3.7).

A graphical representation of the resultant state transition matrix is shown in Figure 4-1. In this view, the height of the surface represents the state transition probability, and the x-axis and y-axis correspond to the row and column indices of the state transition matrix, respectively. The point furthest from the viewpoint is the trapping state, with a probability of 1. In effect, we are viewing the "back side" of the state transition matrix, as the x-axis corresponds to the columns of the matrix and the y-axis corresponds to the rows of the matrix. Note that the higher transition probability values lie along the main diagonal of the matrix—this is an indication that the average holding time for each of the states is relatively long compared to the time interval used by the model, and can be used as a criterion for judging the quality of the state transition matrix.

An alternate view of the state transition matrix is shown in Figure 4-2. This view is a plan view of the matrix, with contour lines showing ridges of equal probability. This view gives a sense of the sparseness of the state transition matrix, and reinforces the sense that the largest probability values lie along the main diagonal of the matrix.

Once the state transition matrix was calculated, it was used directly in computation of the mean time to failure estimates (using equation 3.20). The resultant MTTF output is shown in Figure 4-3. The MTTF, for a given safety state, is an estimate of the expected time interval (expressed in terms of safety state intervals) that the system will take to reach failure. The MTTF was then converted, element by element, into an equivalent risk probability (using

equation 3.23). Figure 4-4 shows the risk probability function obtained from the MTTF function shown in figure 4-3. Table 4-3 lists the values of the dynamic risk probability function.[3]

Let us pause for a moment to consider the meaning of the numbers calculated using safety state model analysis. The state transition matrix contains numbers that are (literally) the modeled probability values for transition from one state to another. These probability values are specified relative to the sampling time period. Thus, the mathematical statement $p_{i,j} = x$ is interpreted as follows: Given that the system is in state *i*, the system will transition to state *j* with probability x within the next sampling period. The value of x lies in the interval $[0,1]$ inclusive, where $x = 0$ means that the system will never transition from state *i* to state *j*, and $x = 1$ means that the system will always transition from state *i* to state *j*.

The MTTF is an expression of the amount of time, on average, that the system will take to reach the failure state from a given operating state. Thus, a state with large MTTF has less risk than a state with small MTTF. Note that there is no specification of the path leading from the given operational state to the failure state—the MTTF is path-independent.[4]

The equivalent risk probability is used as an alternative means of interpreting the calculated MTTF values. These values are not true probabilities, in the same sense as the state transition matrix elements. Instead, they provide a mechanism for using MTTF values to compare dynamic risk as a function of system state.

---

[3] Note that the probability values listed in table 4-3 are represented in scientific notation, with the number following the *e* representing the power of ten (e.g., 3.665911e-05 is $3.665911x10^{-05}$, which may also be represented as 0.00003665911). Since many of the probability values are extremely small numbers, one method for quickly identifying the riskiest states is a cursory inspection of the exponent values.

[4] It is important to keep in mind the test conditions when inspecting the numbers that result from safety state analysis. In the example presented, where the occurrence of accidents was much higher by design than an acceptable operational rail system, the MTTF numbers are relatively low. In fact, the absolute level of the MTTF values depends on the overall risk level of the system—a relatively risky system will have lower MTTF numbers than a less risky system, by virtue of the fact that accidents are more common. The true value of the MMTF values lies in the relative differences between MTTF values for a set of states for a given system, allowing identification of higher risk conditions and the occurrence of near collisions. If a system under analysis has a set of MTTF values which are all relatively close in value, then the analysis, as designed, is not able to identify high risk conditions or causal paths.

The risk probability function was subsequently applied to the safety state trajectories. The outputs of the model calibration phase (i.e., the dynamic risk probability trajectories) are also useful. Through knowledge of the risk probability function, we can identify the states that have a high relative risk probability. For example, by inspecting Figure 4-4 and Table 4-3, we can identify state 70[5] as the highest risk state. Other high-risk states include 78, 66, 98, and 114[6]. This analysis provides quantitative evidence of the intuition developed in the system observation phase.

[5] State 70 corresponds to the case where the operator is using manual control, the track is obstructed, the emergency stop system has not been engaged, the operator is braking, and the vehicle is near enough to the obstruction that it could not be stopped even with the emergency brake applied (Table 4-1c).

[6] State 78 is the same as state 70, except that the emergency brake has been engaged. State 66 is the same as state 70, except that the operator has not applied the brakes. In state 98, the operator is using the programmed stop system, the track is obstructed, neither the service nor emergency brakes are being used, and the vehicle is too near the obstacle to stop. State 114 is the same as state 98, except the autopilot system is in use.

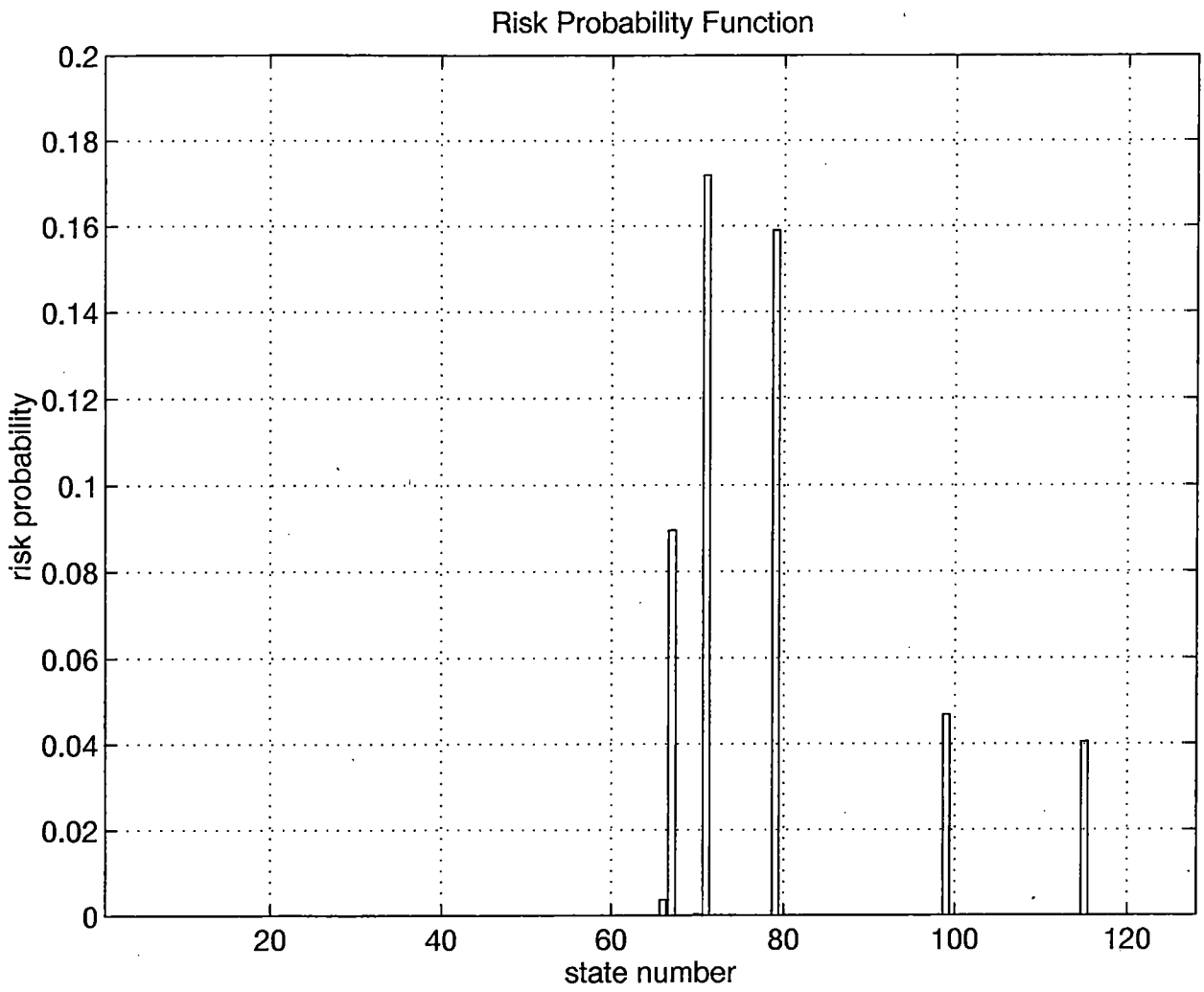


Figure 4-1 — State Transition Matrix (Mesh View)

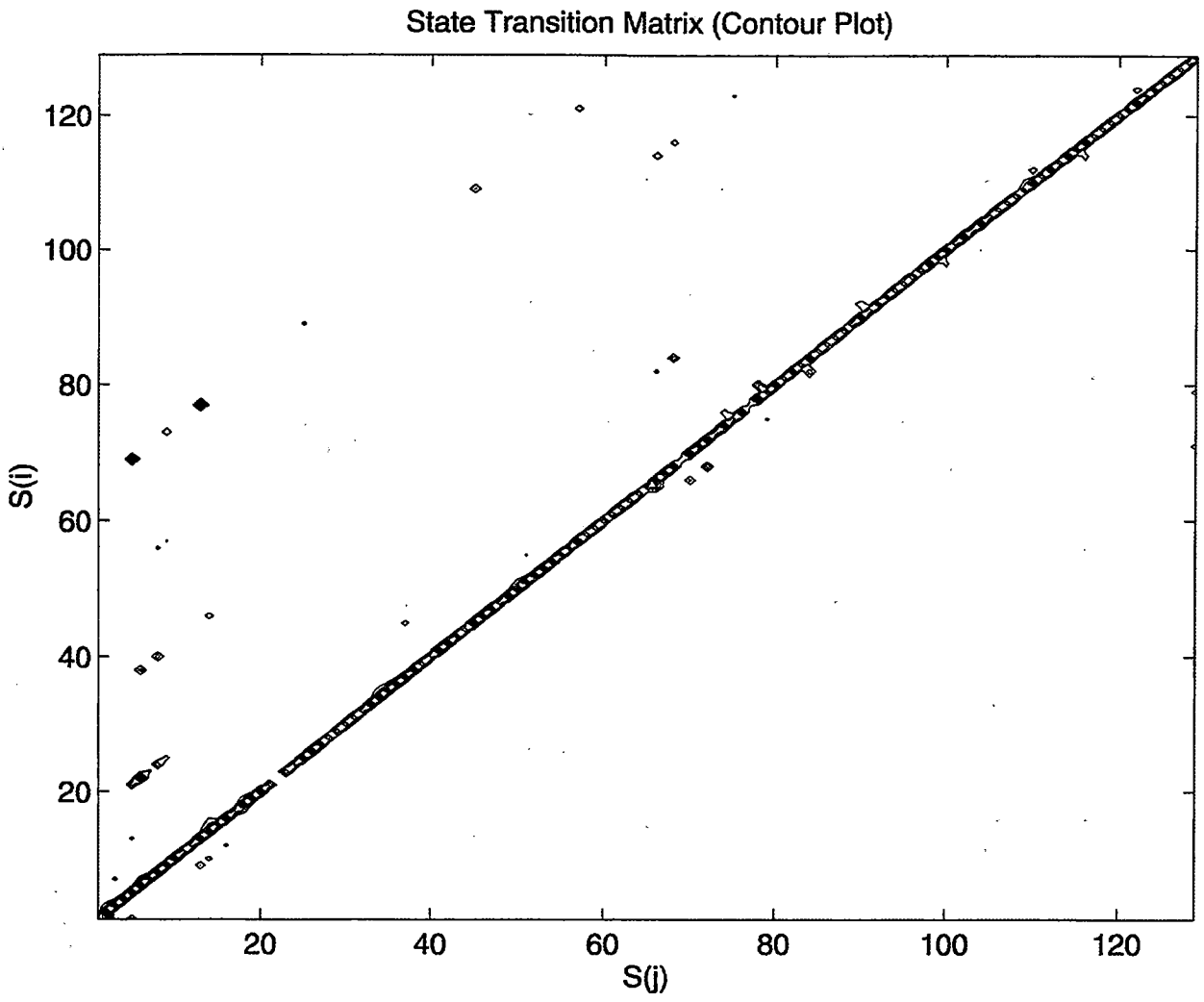


Figure 4-2 — State Transition Matrix (Contour View)

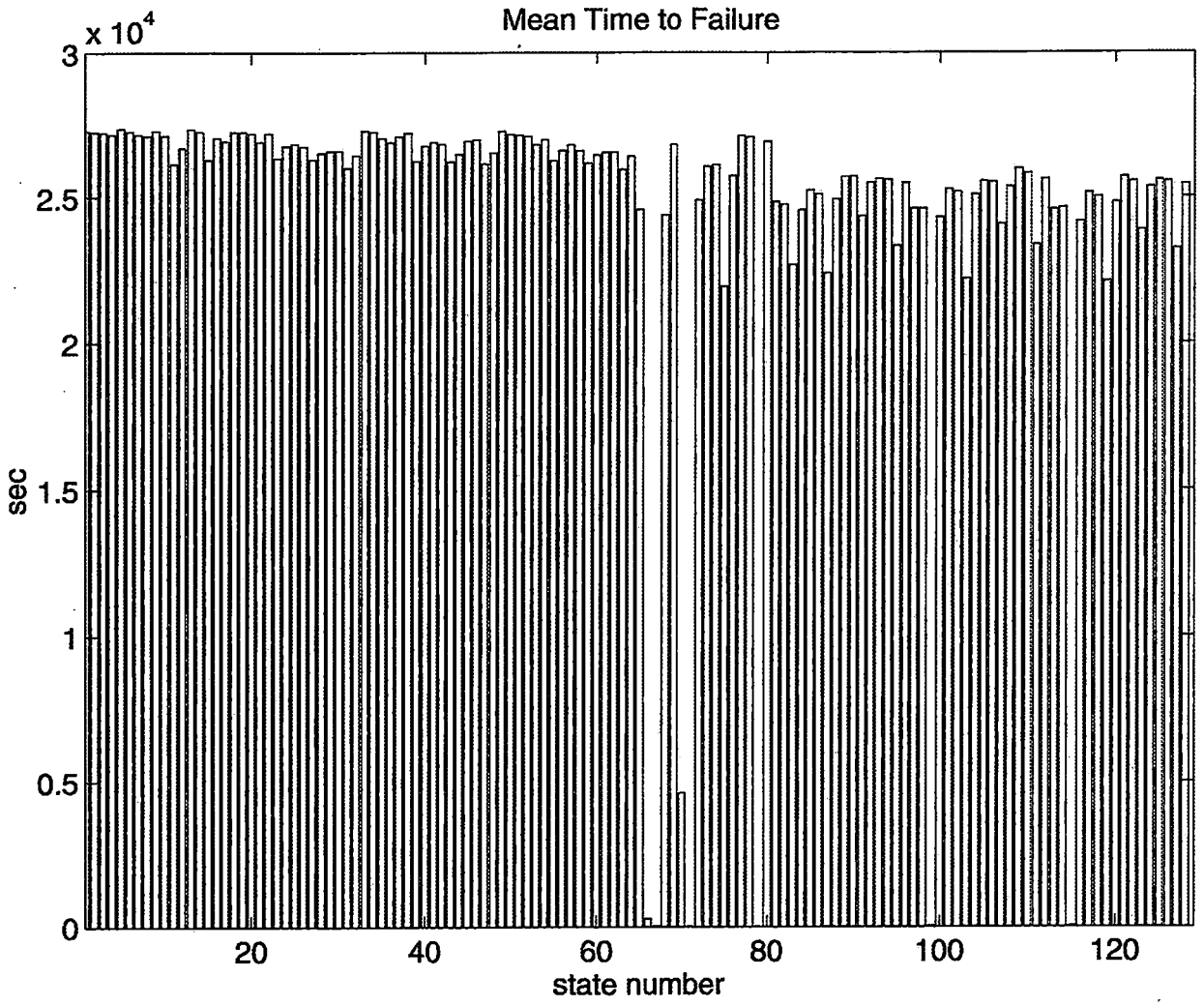


Figure 4-3 — MTTF as a Function of Safety State

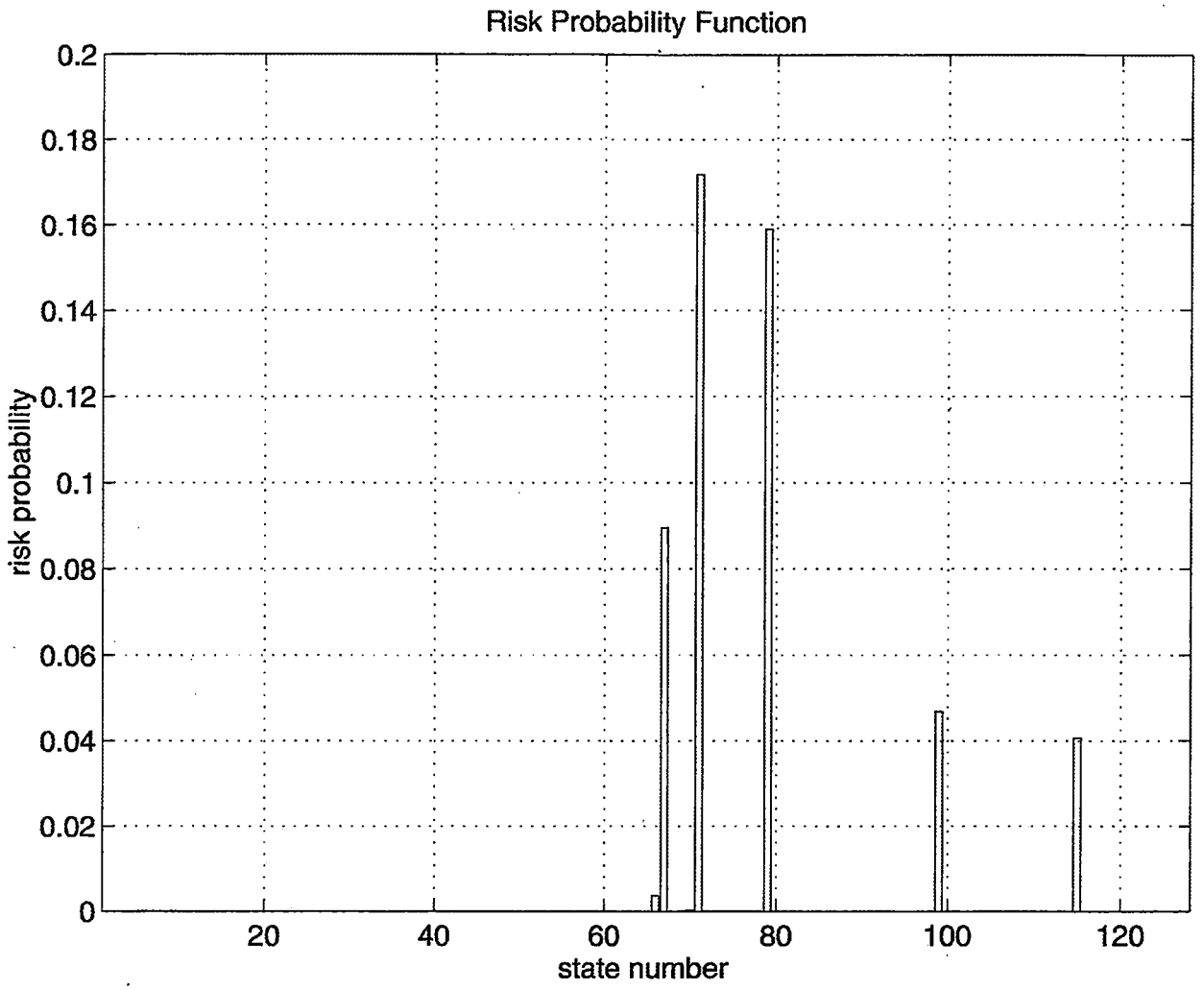


Figure 4-4 — Risk Probability as a Function of Safety State

Table 4-3a. — Risk Function Values

state	risk probability
0	3.665911e-05
1	3.670811e-05
2	3.673025e-05
3	3.683592e-05
4	3.656192e-05
5	3.668415e-05
6	3.682628e-05
7	3.691074e-05
8	3.666303e-05
9	3.688520e-05
10	3.826712e-05
11	3.746290e-05
12	3.658067e-05
13	3.668879e-05
14	3.805374e-05
15	3.699488e-05
16	3.712783e-05
17	3.669579e-05
18	3.670769e-05
19	3.676927e-05
20	3.717917e-05
21	3.677300e-05
22	3.798726e-05
23	3.737859e-05
24	3.726584e-05
25	3.739357e-05
26	3.803250e-05
27	3.771626e-05
28	3.763017e-05
29	3.761168e-05
30	3.847551e-05
31	3.783688e-05

Table 4-3b. Risk Function Values (continued)

state	risk probability
32	3.666176e-05
33	3.669719e-05
34	3.700730e-05
35	3.718579e-05
36	3.691381e-05
37	3.675101e-05
38	3.811438e-05
39	3.737176e-05
40	3.718822e-05
41	3.727720e-05
42	3.813179e-05
43	3.775556e-05
44	3.714476e-05
45	3.707438e-05
46	3.824729e-05
47	3.769418e-05
48	3.667969e-05
49	3.683916e-05
50	3.685009e-05
51	3.691703e-05
52	3.731020e-05
53	3.708427e-05
54	3.810838e-05
55	3.759305e-05
56	3.731806e-05
57	3.760678e-05
58	3.822687e-05
59	3.780334e-05
60	3.767245e-05
61	3.768800e-05
62	3.853966e-05
63	3.789223e-05

Table 4-3c. — Risk Function Values (continued)

state	risk probability
64	4.072072e-05
65	3.670344e-03
66	8.954155e-02
67	4.102894e-05
68	3.729218e-05
69	2.172839e-04
70	1.718398e-01
71	4.018481e-05
72	3.839946e-05
73	3.831102e-05
74	4.563649e-05
75	3.885834e-05
76	3.689360e-05
77	3.694418e-05
78	1.589488e-01
79	3.716552e-05
80	4.026902e-05
81	4.043406e-05
82	4.415102e-05
83	4.074997e-05
84	3.966327e-05
85	3.986338e-05
86	4.470954e-05
87	4.014169e-05
88	3.892607e-05
89	3.891547e-05
90	4.110291e-05
91	3.924490e-05
92	3.904216e-05
93	3.909455e-05
94	4.288274e-05
95	3.924964e-05

Table 4-3d. — Risk Function Values (continued)

state	risk probability
96	4.065515e-05
97	4.067023e-05
98	4.682744e-02
99	4.117748e-05
100	3.960947e-05
101	3.973881e-05
102	4.510500e-05
103	3.989938e-05
104	3.916737e-05
105	3.921421e-05
106	4.156458e-05
107	3.946551e-05
108	3.851417e-05
109	3.873631e-05
110	4.280711e-05
111	3.904190e-05
112	4.069965e-05
113	4.061480e-05
114	4.057289e-02
115	4.141718e-05
116	3.979895e-05
117	4.000007e-05
118	4.527933e-05
119	4.031103e-05
120	3.893162e-05
121	3.916019e-05
122	4.193459e-05
123	3.946246e-05
124	3.908867e-05
125	3.916637e-05
126	4.305445e-05
127	3.931972e-05

4.5 PHASE 4 RESULTS—RISK ESTIMATION

The risk estimation phase is the phase in which the risk probability function is applied to the safety state trajectories. A safety state trajectory output from the system observation phase is a summary of the safety state values (as a function of time) of one test session. The risk probability function transforms each safety state into a dynamic risk probability trajectory. A dynamic risk probability can also be thought of as an *instantaneous risk probability trajectory*, as it summarizes the dynamic risk probability as it changes with time.

Samples of instantaneous risk trajectories is shown in Figures 4-5 and 4-6. These figures show the instantaneous risk probability trajectories for some of the experimental sessions that were run as part of the control automation experiment. Each figure plots several trajectories for a single operator, representing several test sessions for that operator. The y-axis label indicates the automation level of the test. The formal test included three separate test sessions for each operator: *manual mode* (where the operator was responsible for all throttle and braking input), *partial automation* (in which the operator used cruise control and programmed stop systems), and *full automation* (in which a full autopilot was employed). In addition, the *random* test variant corresponds to the second training session, where the failure scenarios were generated by a random process and the operator was free to select from the available automation modes.

Figures 4-5 and 4-6 are two representative samples of the test data collected. Figures 4-5a and 4-6a show the *instantaneous risk trajectories*, representing the equivalent risk probability as a function of time (via the safety state trajectory). In these figures, four plots are shown, correlating to the four separate test sessions for each subject. Figures 4-5b and 4-6b show the *cumulative risk trajectories*, which are time-integrated functions of the instantaneous trajectories. In the cumulative plots, the solid line corresponds to the manual mode test session, the dash-dot-dot line with the partial automation variant, the dashed line with the full automation test, and the dash-dot line with the random test session. Figures 4-5c and 4-6c show the *average risk trajectories*, where the cumulative risk is divided by the accrued time.

These three types of figures show three different ways of interpreting the safety state analysis data. The instantaneous plots allow identification of specific points in time where high risk activity occurs, allowing rapid identification of near-collision situations. The cumulative plots estimate the overall likelihood that a failure will occur. An alternate way to consider the

cumulative plot is as a predictor of the number of accidents that the operator should have had. The steadily increasing trend in the cumulative plots underscores the notion of risk exposure, discussed in chapter 2. Finally, the averaged plots provide a sense of how well an operator recovers from high risk situations and provides an estimate of average risk level for a given operator and session.

Figures 4-5 and 4-6 show the relative performance of two test subjects in the control automation experiment. Subject 18 (Figure 4-5) exhibited behavior that was on the high end of the risk spectrum. This subject experienced two collisions, one each in the full automation and random tests. The collision events are easily identified by inspecting the instantaneous risk trajectories and noting the high risk spikes that are evident. These spikes immediately preceded the actual collisions, indicating that the subject entered an extremely high risk state immediately prior to each collision. In the manual mode test session, the subject experienced one period of moderately high risk, while in the partial automation test we see maximum risk levels which are much lower in magnitude than the other test sessions. The cumulative and average risk trajectories reflect the high risk states as expected.

Subject 7, by contrast, had a total of five moderately high-level risk situations, distributed throughout the four test sessions. Each of these situations could be interpreted as a near collision event. However, none of these situations resulted in collision, indicating that subject 7 displayed good response to potentially risky situations. Inspection of the cumulative risk trajectories shows that subject 7 displayed very consistent performance relative to risk level; the average risk trajectories support this assessment as well (note the scale of the vertical axis).

Using the safety state analysis to compare the risk-related performance of subjects 18 and 7, we can make the following conclusions: The test results of subject 18 show evidence of ability to complete a test session with extremely low risk behavior. However, subject 18 is more likely to have a collision when higher risk situations emerge. Subject 7, on the other hand, while experiencing a greater number of near collision situations, was able to effectively control the vehicle through those situations and avert any higher levels of risk.

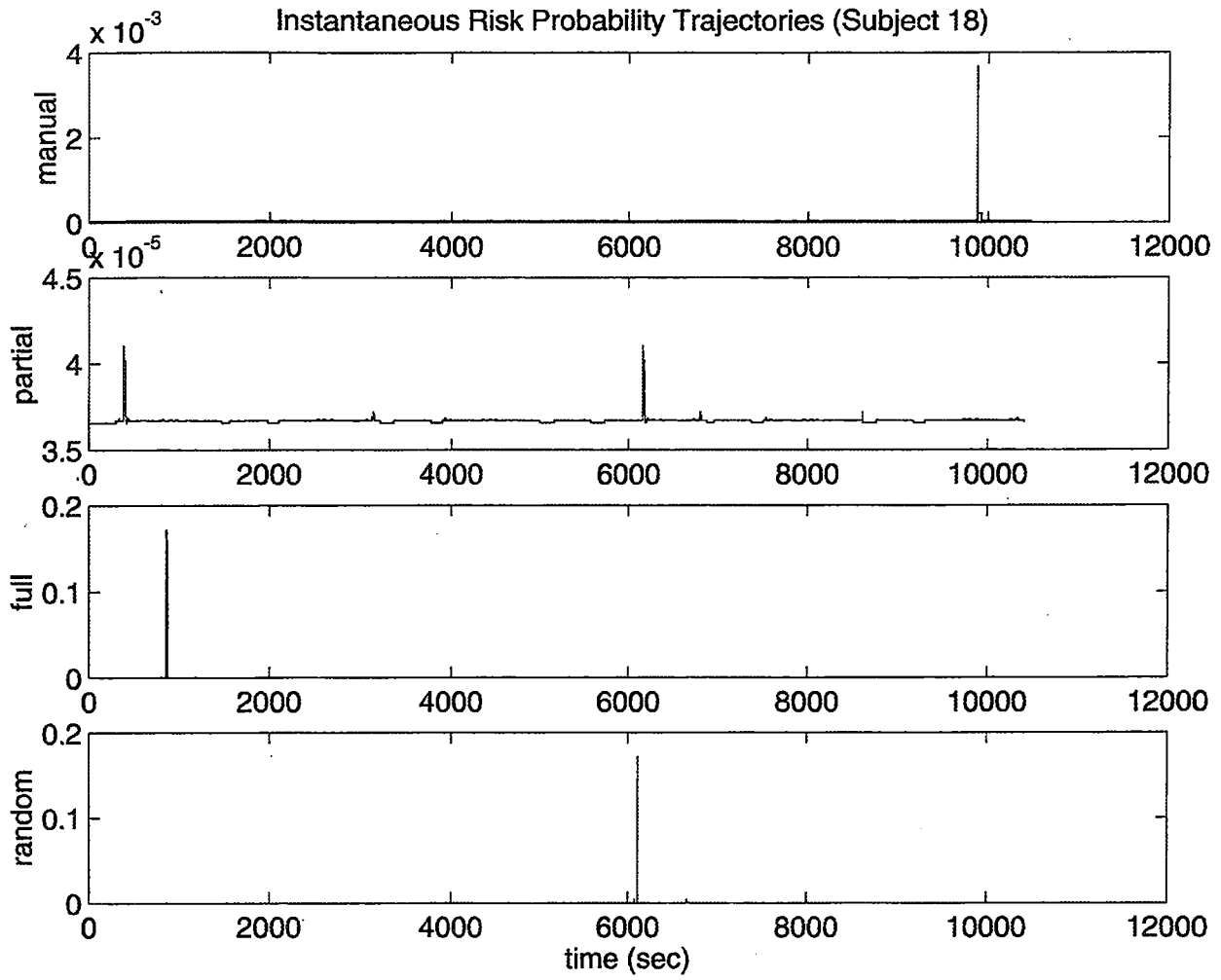


Figure 4-5a. — Instantaneous Risk Trajectories—Subject 18

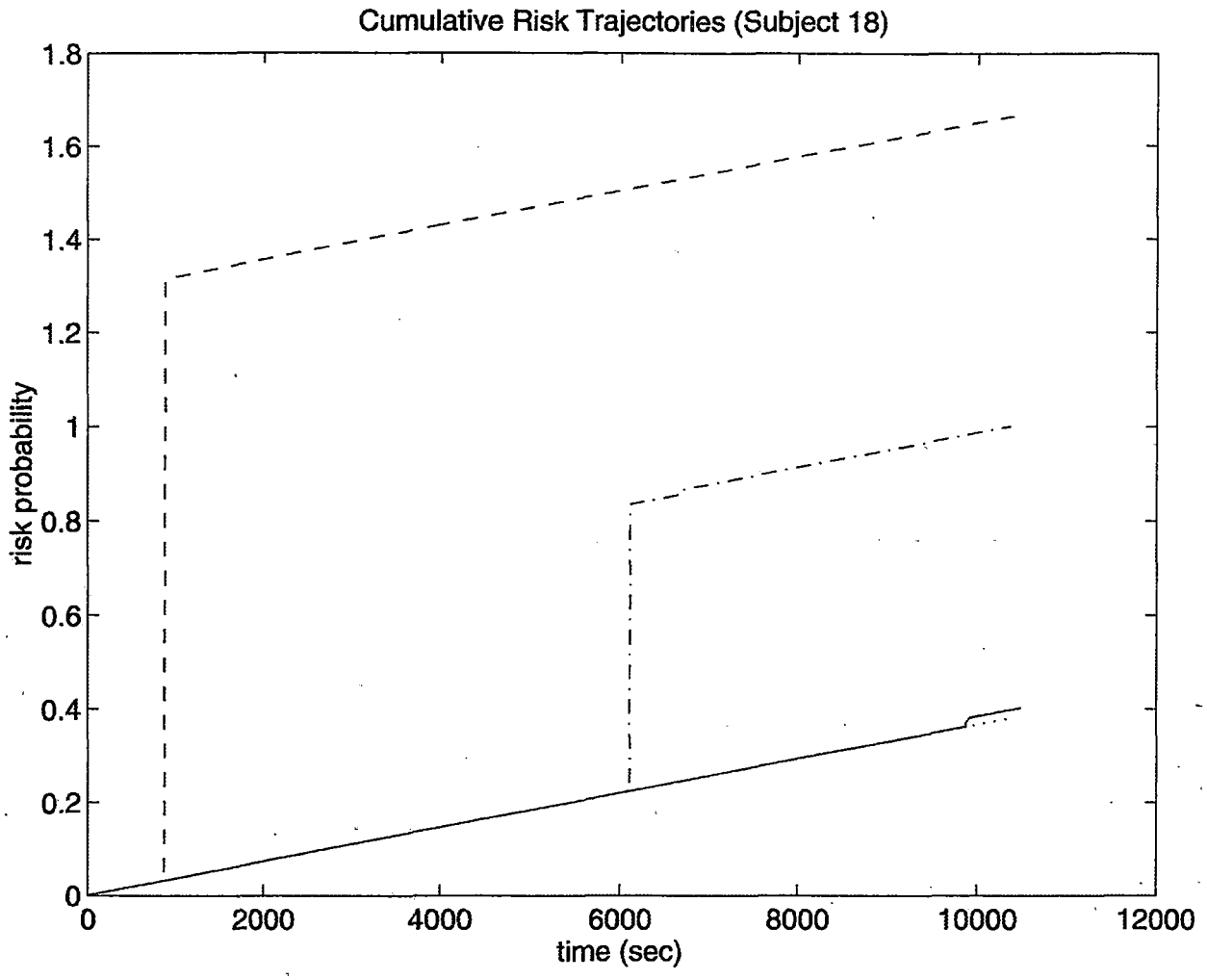


Figure 4-5b. — Cumulative Risk Trajectories—Subject 18

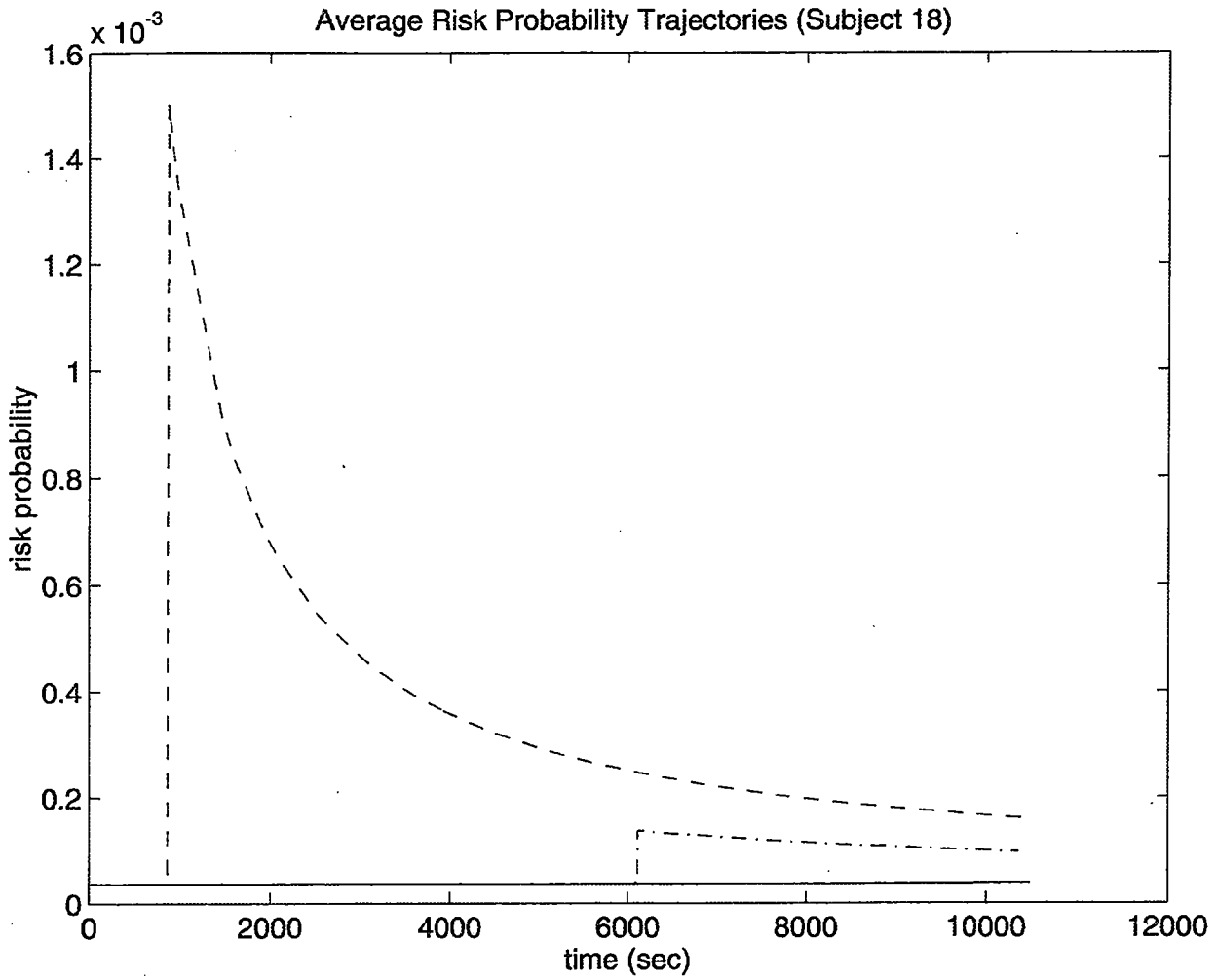


Figure 4-5c — Average Risk Trajectories—Subject 18

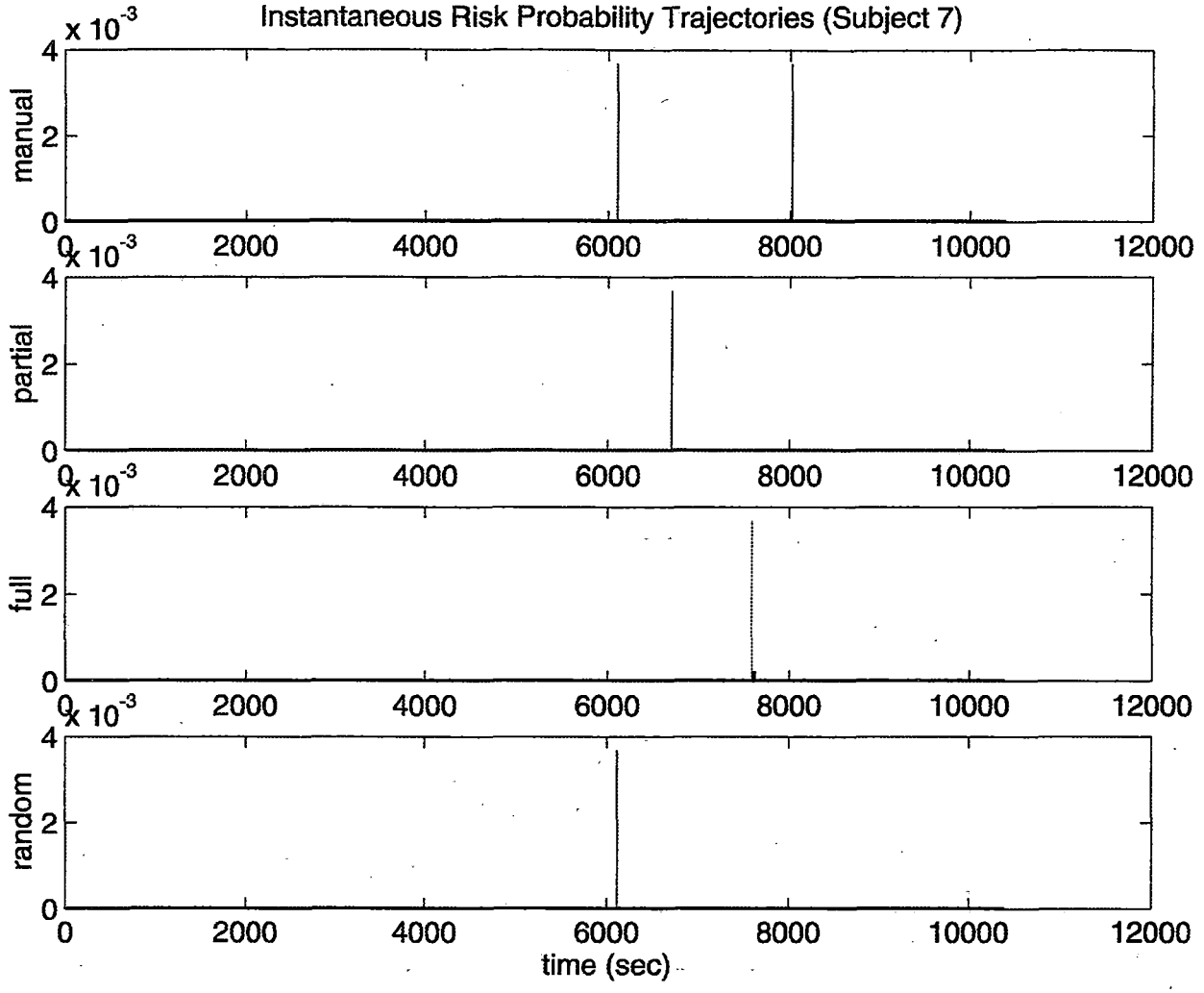


Figure 4-6a — Instantaneous Risk Trajectories—Subject 7

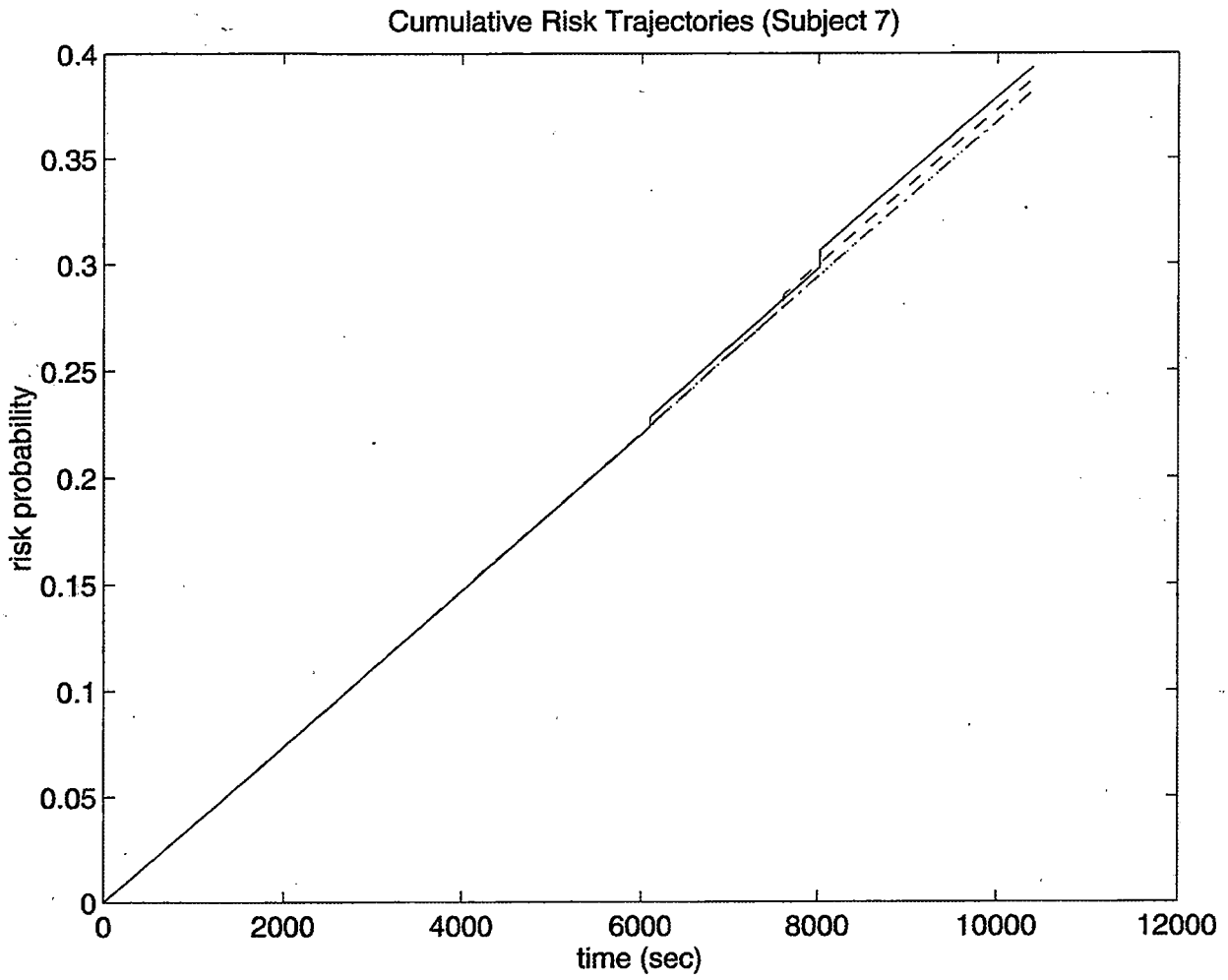


Figure 4-6b — Cumulative Risk Trajectories—Subject 7

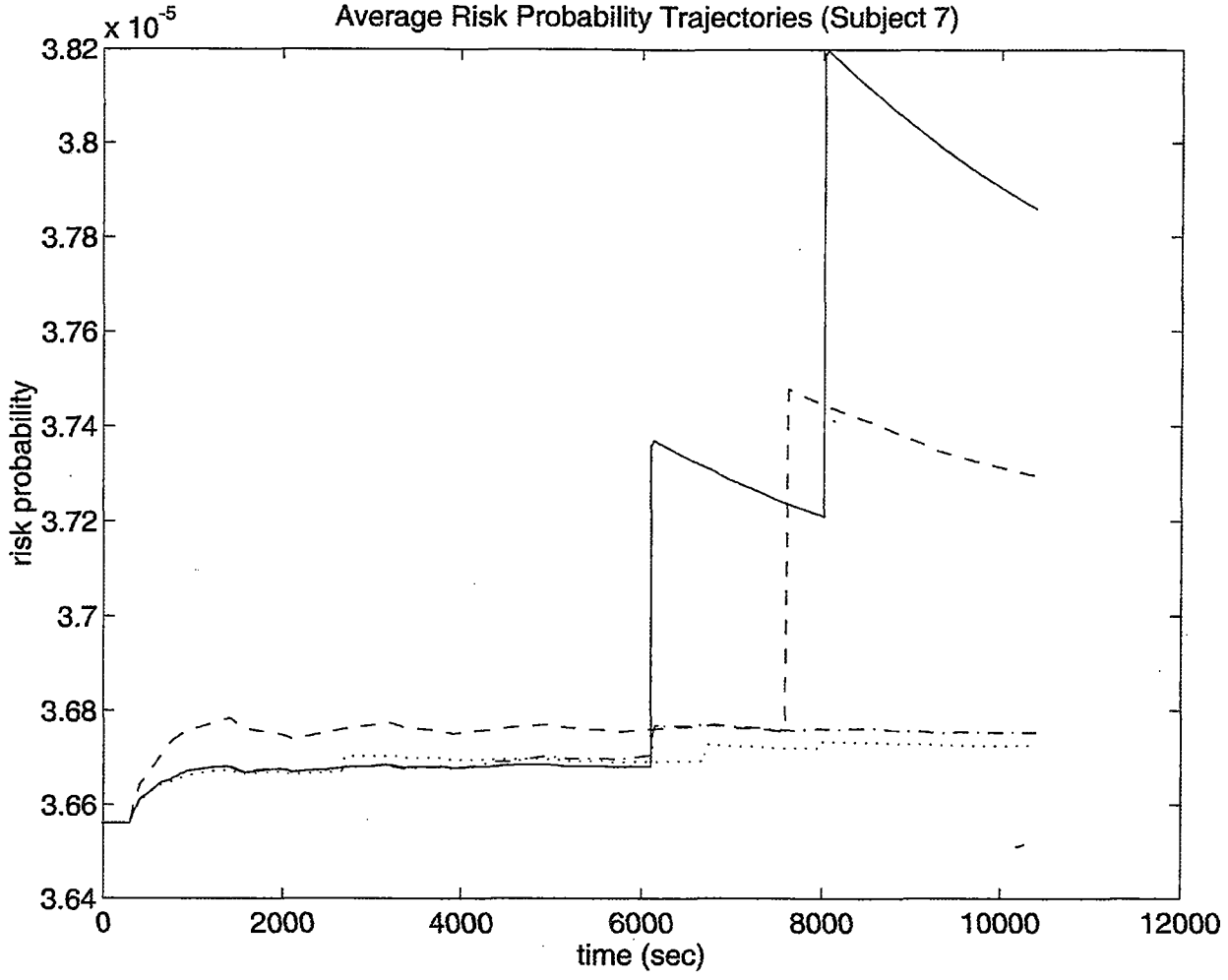


Figure 4-6c — Average Risk Trajectories—Subject 7

4.6 DISCUSSION

As a brief summary, the goal of the safety state model is to identify the risk probability of an unrecoverable failure event in a complex system which includes human operator input. The fundamental notion is that actual failures are preceded by system states that could be described as near-collisions, and that near-collisions occur much more frequently than actual failures. If we can identify the system states that constitute near-collisions, we can take steps to avoid these states and thus reduce the number of actual failures.

A number of interesting details become apparent during application of the safety state model. First, there is the state transition matrix. Because the state transition matrix is a two-dimensional matrix, the values of state transition probabilities can be shown as a surface in a three-dimensional space. In this type of representation, the row and column indices of the transition matrix constitute the x-axis and y-axis, respectively, of the three-dimensional space, and the transition probabilities are represented as the height of the surface (Figure 4-1).

As shown in Figure 4-1, there is a row of "mountains" down the main diagonal axis, which corresponds to the main diagonal of the state transition matrix. The values of these elements are generally the largest in the matrix, with the last element being the largest overall. The diagonal elements represent the probabilities of remaining in the same state during a transition period. We note that, provided the measurement period for the state transitions is sufficiently small compared to the holding times for the states, this is an expected condition, and it can be used as a qualitative judge of the sampling period. If the holding times were on the same order as the time slice interval, then there is a possibility that state occupancies might not have been captured by the model during the system observation phase, suggesting that the specified time slice interval was too large. The last element (i.e., where S(i)=128 and S(j)=128) is the probability of holding in the trapping state, which is (by definition) equal to one—this is the largest possible value in the state transition matrix, and should only occur in this matrix element.

There is also a distinct pattern shown in the remaining "mountains" in the figure. The "mountains" tend to form lines that are parallel to the main diagonal, and they tend to be grouped into distinct "sub-matrix" patterns. These patterns correspond to state changes that occur as a result of only one binary condition changing state. (In other words, the numerical relationship between the two numbers is that the binary representations differ by only one bit.) As a general

rule, the patterns as observed are desirable, as they indicate that the sampling period is short enough to separately identify changes in individual binary conditions.

Let us also consider the evolution of the risk trajectory. Data from 96 test and training sessions was used in the model calibration phase. After each safety state trajectory was added to the statistics matrix, the state transition matrix, MTTF function, and risk function were calculated. In Figure 4-7, the evolution of the risk probability function is shown. The safety state is shown along the x-axis. The z-axis represents the risk probability. The y-axis represents the number of files that have been included to that point. If we were to slice a plane parallel to the x-z plane (perpendicular to the y-axis), the intersection with the surface would be the risk function after y files were incorporated.

The resultant surface is quite interesting. When a smaller number of files are included (i.e., if we were to isolate an x-z plane at a small value of y), the risk probability function is relatively flat, indicating that, as far as we know from the available data, there is no significant difference in risk probability between the states. The notion correlates with our intuitive sense, because until there is a collision, we do not assume that one will happen (and hence P_f is an array of zeros). As the number of trajectories increases, and the number of recorded collisions rises, safety state trajectory paths with collisions are incorporated into the state transition matrix, and peaks start to appear in the risk function. By the tail end of the evolution, that is, after many of the safety state trajectories have been incorporated, the patterns have leveled out along the y-axis. This trend indicates that there exists a point where additional data does not substantially change the shape of the risk function (Figure 4-7). At that point, the average behavior of the system has been learned.

Finally, to calibrate our results, we compare the collision data from the tests to the expected number of collisions that are predicted by the model. During the practice and test sessions, there were a total of 10 collisions (Table 4-2). In addition, there were 20 training sessions, during which there were a total of 22 collisions. Thus, there were a total of 32 collisions that occurred in the data used for model calibration. From the safety state model data, we add the final values of the cumulative risk trajectories across the complete set of trajectories to arrive at a predicted value for the total number of collisions. The predicted value, from the available data, was calculated to be 36.7 expected collisions, which compares well with the actual number of collisions.

Based on the overall results found, we believe that the safety state model is a useful tool for estimating the dynamic risk probability.

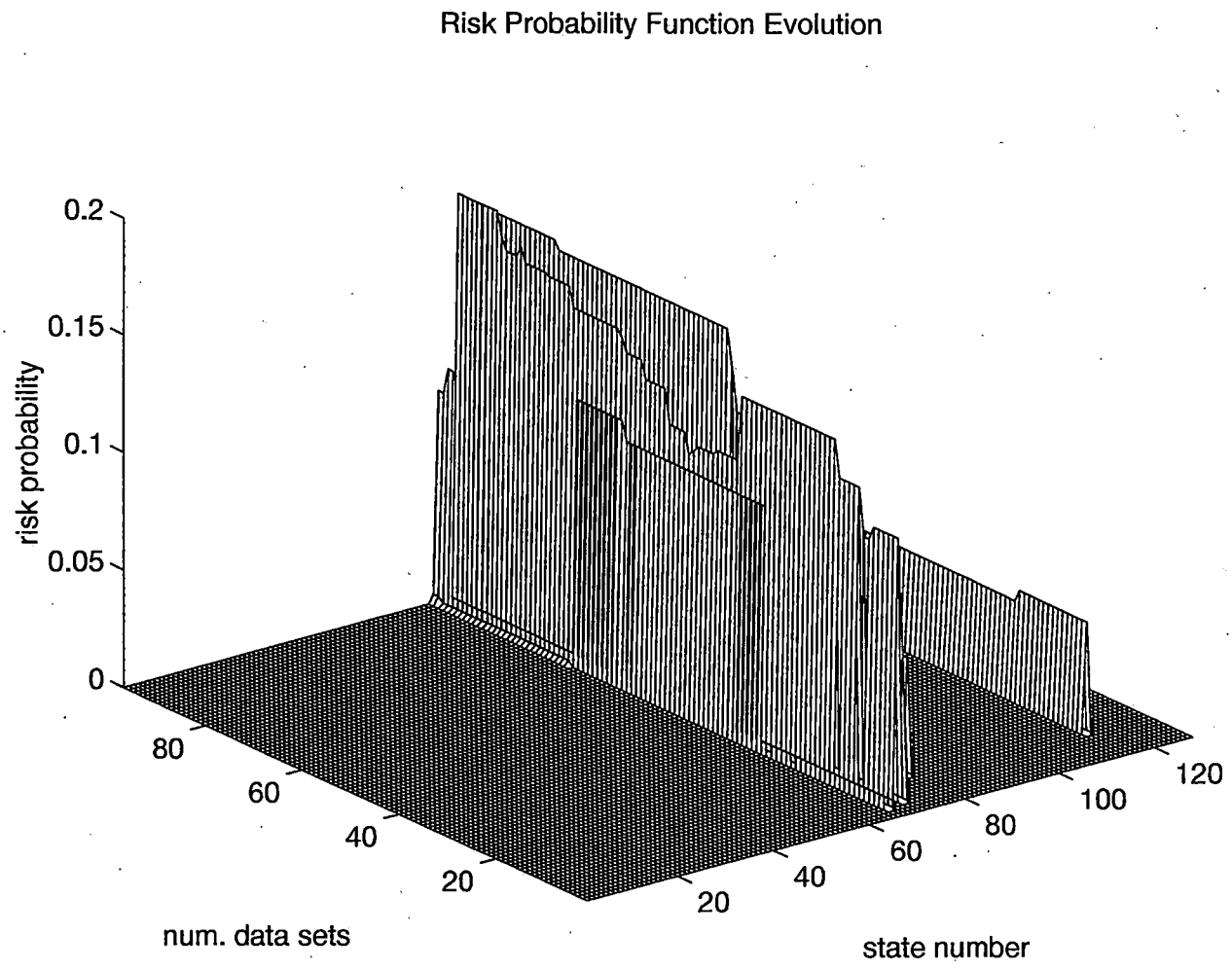


Figure 4-7 — Evolution of Risk Probability Function

5. SUMMARY AND CONCLUSIONS

The overall objective of this research was the development of a method for estimating the dynamic risk probability in complex dynamic systems. An underlying assumption is the existence of dynamic risk probability as a function of system state. It is intuitively clear that the probability of occurrence of a "bad event" depends on the state of the system. This is supported by the accepted notion of a near collision. The research detailed here was focused on finding a method for estimating dynamic risk probability, thus providing a quantitative measure for comparing the risk-related performance of human operators in complex human-machine systems.

The work can be considered in terms of three major components. The first, contained in chapter 2, covered a top-level review of safety analysis and a discussion of the interrelationship between safety, risk, and system reliability. This section also included a brief review of existing risk assessment methods and evaluation of the applicability of those methods for estimating dynamic risk probability. This work led to the following conclusions:

1. The pursuit of safety is the minimization of risk.

2. Risk is the relative level of chance that the health and well-being of a person is threatened.

3. Risk is a function of both the chance that a bad event (an accident) will occur and the magnitude of the outcome, relative to the health and well-being of human beings, when an accident occurs.

4. Risk probability can be considered to be a dynamic quantity, as a function of dynamic system state.

The second component, covered in chapter 3, focused on development of a theoretical method for estimating the risk probability. The approach developed was based on discrete finite Markov process analysis. The system state (*safety state*) was defined as the exhaustive combination of a finite set of binary conditions, each of which may be contributory to the occurrence of a particular accident. The accident itself is modeled as a trapping state, as it is not reversible. The resultant collection of safety states is considered to be a Markov process. Markov process analysis is used to determine the mean time to failure (MTTF) for each individual operational

state. The MTTF estimate is then converted into an equivalent risk probability, thus creating a risk probability function which is used to transform system state into risk probability.

The third component, in chapter 4, was experimental demonstration of the safety state model. Demonstration was accomplished through application of the safety state model for evaluation of operator performance in high-speed rail. The goal was demonstration of viability of the developed method. Data was obtained in conjunction with a human factors experiment evaluating control automation in high-speed rail. A five-condition seven-bit safety state model was defined. The risk event evaluated was a grade crossing collision. Data obtained was processed using the safety state model.

The experimental results demonstrate that the safety state model is a useful tool for evaluating dynamic risk probability. This conclusion is based on the following:

1. The method for generating the Markov process state transition matrix from raw collected data proved successful. The form of the resultant state transition matrix closely matched expectations.

2. The method for transforming the state transition matrix to estimates of MTTF, as function of system state, was successful. This was verified by performing the calculations using both the direct method (equation 3.20), which was derived using two independent approaches, and the iterative method (equation 3.12) as verification.

3. The method for transforming the MTTF function to the equivalent risk probability function was successful. Inspection of the resultant risk probability function allowed rapid identification of system states having relatively high risk.

4. The method of applying the risk probability function to create instantaneous risk probability trajectories was successful. Inspection of the resultant instantaneous risk trajectories allowed rapid identification of the causal chain of system states leading to accidents. In addition, inspection of the instantaneous risk trajectories allowed rapid quantitative identification of near collision situations.

5. The technique of summarizing the instantaneous risk probability trajectories into cumulative and average risk trajectories was successful. The cumulative risk trajectories provide a

reasonable means for predicting the overall number of accidents, and is an objective measure of risk exposure. The average risk trajectories provide a means of objective comparison of individual operator performance.

Several issues have been raised, which present opportunity for future development of the method. The primary limitation of the method is the exponential growth of the system state number with an increasing number of binary conditions. An appropriate approach to mitigate this shortcoming is through exploration of parallel processing computer technologies. In addition, there are issues with sensor requirements and data storage and processing which could be addressed. Finally, on the theoretical front, the model should be extended to address multiple failure events with a single safety state model.

In addition, further experimental application of the safety state model is crucial to further development. Salient characteristics of systems well-suited to safety state model analysis include:

1. distributed system control,
2. significant component of human interaction relative to system control,
3. highly dynamic system,
4. closed system,
5. moderate to high level of risk in the event of an accident, and
6. reasonable capability for instrumentation.

Two examples in transportation that are particularly well-suited for experimental application of the safety state model are high-speed passenger rail and auto racing. While an experimental application to high-speed rail would provide a more direct path for general application, there is a significant shortage of implemented systems available for evaluation. Auto racing, on the other hand, provides the advantages of availability, as well as providing a highly dynamic and risky environment which would allow rapid generation of pertinent data. Additionally, many forms of auto racing already include a high level of sensor data and telemetry for communicating the sensor data with ground stations.

In conclusion, the safety state model is shown to be a viable approach for evaluating dynamic risk probability in complex systems. It is recommended that experimental research on the safety state model be continued.

REFERENCES

Askey, Shumei Y. (1995) *Design and Evaluation of Decision Aids for Control of High-Speed Trains: Experiments and Model.* Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA

Askey, Shumei Y., and Sheridan, Thomas (1995). *Safety of High-Speed Guided Ground Transportation Systems, Phase II: Design and Evaluation of Decision Aids for Control of High-Speed Trains: Experiments and Model.* Final Report to DOT/FRA (U.S. Department of Transportation/Federal Rail Administration) September 1995.

Babcock, Philip (1986). *An Introduction to Reliability Modeling of Fault-Tolerant Systems.* Technical Report CSDL-R-1899, Charles Stark Draper Laboratory, Cambridge, MA.

Endsley, M. (1987) SAGAT: A methodology for the measurement of situation awareness (NOR DOC 87-83). Hawthorne, CA: Northrup Corp.

Endsley, M. (1988) Situation Awareness Global Assessment Technique (SAGAT). In *Proceedings of the National Aerospace and Electronics Conference.* New York: IEEE.

Endsley, M. (1995) Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors*, (37) 1

Fitts, Paul M. (1951) *Human Engineering for an Effective Air Navigation and Traffic Control System.* Technical Report, National Research Council, Washington, D.C.

Gaba, David M., Howard, Steven K., and Small, Stephen D. (1995) Situation Awareness in Anesthesiology. *Human Factors* (37) 1

Gratt, L.B. (1987) Risk Analysis or Risk Assessment: A Proposal for Consistent Definitions. In *Uncertainty in Risk Assessment, Risk Management, and Decision Making.* Plenum Press, New York.

Hald, A. (1952) *Statistical Theory with Engineering Applications*, John Wiley & Sons

Hamburg, Morris, and Young, Peg (1994) *Statistical Analysis for Decision Making* (6th Edition), Dryden Press

Hendy, K.C. (1995) Situation Awareness and Workload: Birds of a Feather? AGARD AMP Symposium on "Situational Awareness: Limitations and Enhancements in the Aviation Environment"

Howard, Ronald A. (1971) *Dynamic Probabilistic Systems (Volume 1).* John Wiley and Sons.

Lanzilotta, Edward J. (1995) Analysis of Driver Safety Performance Using the Safety State Model. In *Transportation Research Record*, No. 1485, Safety and Human Performance: Human Performance and Safety in Highway, Traffic, and ITS Systems. Transportation Research Board, National Research Council.

Lanzilotta, Edward J. (1995) Using the Safety State Model to Measure Driver Performance. SAE Technical Paper 950968. Also in SAE publication SP-1088.

Lanzilotta, Edward J., and Sheridan, Thomas B. (1998). *Safety of High-Speed Guided Ground Transportation Systems, Human Factors Phase III: Effects of Control Automation on Operator Performance.* Final Report to DOT/FRA (U.S. Department of Transportation/Federal Rail Administration) December 1995.

Lewis, Elmer E. (1987) *Introduction to Reliability Engineering.* John Wiley and Sons.

Lowrance, William W. (1976) *Of Acceptable Risk.* William Kaufmann, Inc., Los Altos, CA.

Rescher, N. (1983) *Risk: A Philosophical Introduction to the Theory of Risk Evaluation and Management.* University Press of America.

Rowe, W.D. (1977) *An Anatomy of Risk.* John Wiley and Sons.

Sarter, Nadine B., and Woods, David D. (1995) How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control. *Human Factors* (37) 1

Sheridan, Thomas B., Lanzilotta, Edward J., and Askey, Shumei Y. (1994). *Safety of High-Speed Guided Ground Transportation Systems, Phase I: Human Factors.* Final Report to DOT/FRA (U.S. Department of Transportation/Federal Rail Administration) October 1994.

Shinar, David. (1978) *Psychology on the Road: The Human Factor in Traffic Safety.* Wiley.

Smith, Kip, and Hancock, P.A. (1995) Situation Awareness is Adaptive, Externally Directed Consciousness. *Human Factors* (37) 1

Wharton, F. (1992)Risk Management: Basic Concepts and General Principles. In Ansell, J., and Wharton, F. (ed.) *Risk Analysis, Assessment, and Management.* John Wiley and Sons.

APPENDIX A — DESIGN AND IMPLEMENTATION: HIGH-SPEED RAIL SIMULATION SYSTEM

An important component of this research has been the development of a high-speed rail simulation system. This system was developed for the purpose of performing laboratory experiments, and is categorized as a distributed interactive simulation system. It is interactive in the sense that it is designed to be used by human operators in real-time—a "virtual reality" system for high-speed rail operation. The simulation system is considered a distributed system because it operates on multiple computers, which are interconnected by a local area network.

This chapter describes the design and implementation of the simulation system, from a systems perspective. The following sections discuss the goals and motivations of the simulation system, the system architecture and related design issues, the active elements of the simulation, the support elements of the system, some software engineering issues, and a summary of the configurations used for the experiments.

A.1 GOALS AND OBJECTIVES

The overall objective of this simulation system is to provide a virtual environment for high-speed rail operations, with the intent of using the system for laboratory-based human factors studies. As a result, the overall performance requirements of the system are to provide a sufficiently realistic environment for human-in-the-loop operation, such that the task objectives and constraints of the real operational tasks can be met.

In rail operation, there are two primary classes of operating personnel: locomotive and dispatchers. The locomotive engineers perform their duties in the vehicle, generally in the cab and while the vehicle is in motion. Their primary task is speed and position control of the vehicles, which includes detection and reaction to emergency situations. The dispatchers perform their duties from the wayside. Their primary tasks include control of the switching points and overall system coordination.

The high-speed rail simulation system has provision for supporting complex rail system operations. The system supports multiple system elements, each implemented on one or more computer system, operating in conjunction with one another and communicating over a local area network.

A.2 SYSTEM ARCHITECTURE ISSUES

During the development of the simulation system, considerable effort was spent addressing issues relating to the system architecture. Broadly speaking, the term "system architecture" refers to the organization, interconnection, and functionality of the individual elements which comprise the system. In the case of a human-interactive simulation system, these issues include design of storage representations for the various databases, specifying the mechanisms for intercommunication between the computers which form the system, and identifying appropriate user interfaces.

A.2.1 Road Database Representation

A primary issue in designing a transportation simulation system is selection of a database representation of the environment. Such a database must contain all the pertinent information needed to describe the environment, yet should also be stored in the smallest possible data space.

One characteristic of rail systems is that they cover large distances over paths of very small width. Because these vehicles operate at ground level and along narrow paths, the view from a rail vehicle is quite limited relative to the distance traveled. This is in stark contrast to the view from an air vehicle, which is unobstructed over a wide area. In addition, an air vehicle is not constrained to narrow corridors, like a rail vehicle. As a result, air vehicle simulators must be able to reproduce a wide area of visual field to a reasonable resolution. For the purposes of a rail vehicle simulation, however, the topology can be reduced to a network of interconnected paths.

An explicit design objective was to incorporate dynamic elements in the environment. Pertinent to rail operation, these include signal states, switch states, weather conditions, and hazards. In order to achieve this, the design of the database must include variables for these states. In addition, a mechanism must be provided for communicating state changes among the simulation elements.

Two distinct database formats were designed to address these issues. One was designed to capture the topology of the road itself (the road database), while the second was designed for representation of the objects that appear along the road (the object database). Although the two are interrelated (through reference to the same inertial coordinate system), they are implemented

as separate entities. This is because only a subset of the active simulator elements require the information contained in the object database, while all of the simulation elements require the road database.

The road database contains all of the information required to describe the network of guideway paths which comprise the road system. In the abstract form, the network is composed of nodes and links. Each link is a path between two nodes, and each node is a point where links interconnect. The network is a collection of nodes and links.

To describe these elements, four data structures are defined: the road segment, the road unit, the connection unit, and the network header. A road database is a binary file containing all of these data structures, organized with the network header first, followed by arrays of the other three data structures (Figure A-1). The data structures are organized as multiply linked lists, using indices into the respective arrays as the linking mechanism.

The most basic element is a road segment. A road segment is defined as a piece of road which has a fixed start point (represented in X-Y-Z coordinates, as well as heading, grade, and banking), a fixed length, a constant curvature in heading, and a constant curvature in pitch. The data structure for a road segment contains the preceding data, as well as dynamic state information (for signals, hazards, and weather) and a link to the parent road. A C implementation of the road element data structure is shown in Figure A-2.

A road unit is defined as a collection of road segments which collectively form an uninterrupted path. The data structure for the road contains links to the segments which define the start and end of the road, as well as the connection points which are used to connect this road to other roads. These links are specified as indices into the arrays which contain the road element and connection unit data structures. The data structure also contains a name for the road. A C implementation of the road unit data structure is shown in Figure A-3.

A connection point is defined as a point at which two or more roads interconnect. In a rail system, these are used to represent switches, stations, and entry points. The data structure for a connection point (known as a connection unit) contains the links to the roads which connect at this point. It also contains a state variable, used by connection points which have a dynamic state (such as a switch), and a name. As with the road unit, the connection unit references the road

unit links via indices into the road unit structure array. A C implementation of the road unit data structure is shown in Figure A-4.

The highest level data structure is the network header. This data structure contains all of the information which organizes the individual arrays into a system, including the locations and sizes of the individual arrays, as well as the name and type revision of the system. A C implementation of the network header data structure is shown in Figure A-5.

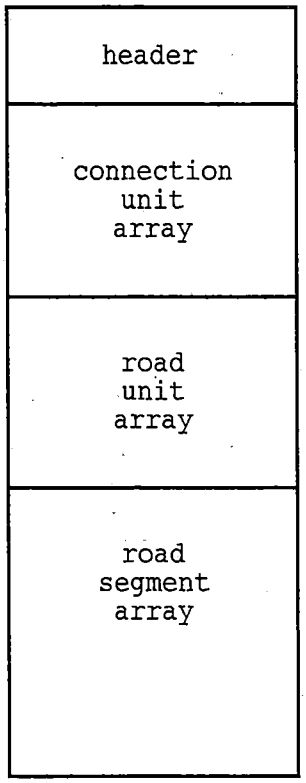


Figure A-1. Road Database File Organization

```
/* definition of the road segment structure (and fields) */
struct road_element  {          /* one for each individual road piece */
  int road;                     /* index of parent road unit */
  float pitch;                  /* aka grade (radians) */
  float curvature;              /* inverse turning radius (1/meters) */
  float banking;                /* tilt of roadway (radians) */
  float x_position;             /* distance east of origin (m) */
  float y_position;             /* distance north of origin (m) */
  float elevation;              /* above sea level (meters) */
  float orientation;            /* inertial yaw angle (radians) */
  unsigned int hazard_class;    /* potential hazard class (bit-encoded) */
  unsigned int hazard_state;    /* hazard state (bit-encoded) */
  float hazard_position;        /* relative to start of segment */
  unsigned int weather_state; /* weather state (value-encoded) */
  /* lower 8 bits are reserved for temperature information (deg C) */
#define WEATHER_CODE_MASK  0xffffff00
#define WEATHER_TEMP_MASK  0x000000ff
  unsigned int signal_class_up;  /* signal class (upstream) */
  unsigned int signal_state_up;  /* signal state (upstream) */
  unsigned int signal_class_dwn; /* signal class (downstream) */
  unsigned int signal_state_dwn; /* signal state (downstream) */
#define NO_SIGNAL          0x00000000
#define RAIL_CLEAR         0x00000001
#define RAIL_APP_START     0x00000002
#define RAIL_APP_MED       0x00000004
#define RAIL_APPROACH      0x00000008
#define RAIL_CAUTION       0x00000010
#define RAIL_RESTRICT      0x00000020
#define RAIL_STOP          0x00000040
  unsigned int speed_limit;      /* speed limit (value-encoded, kph) */
  float segment_length;          /* length of this segment */
};
```

Figure A-2. Road Segment Data Structure

```
/* definition of the road unit structure */
struct road_unit  {     /* for each roadway */
  char road_id[16];     /* ID string */
  int start_connect;    /* index of start connection */
  int end_connect;      /* index of end connection */
  int start_segment;    /* index of begin segment */
  int end_segment;      /* index of end segment */
};
```

Figure A-3. Road Unit Data Structure

```
/* definition of the connect unit structure (and fields) */
struct connect_unit  {              /* for each connection */
  char connect_id[16];              /* identifier string */
  unsigned int connect_type;        /* type code */
#define RAIL_ENTRY       0x0001     /*  termination point */
#define RAIL_SWITCH      0x0002     /*  switch */
#define RAIL_STATION     0x0004     /*  station */
#define RAIL_INTERSECT   0x0008     /*  track crossing */
#define HWY_ENTRY        0x0010     /*  end of the road */
#define HWY_RAMP         0x0020     /*  on-ramp/off-ramp */
#define HWY_INTERSECT    0x0080     /*  intersection */
  int connect_state;                /* connection state */
  /* definition of connection state is TBD */
#define CU_MAX   6              /*  maximum is 6-way intersection (?) */
  int connect_point[CU_MAX];  /* road unit index array */
  /* (organization depends on type of connect) */
};
```

Figure A-4. Connection Unit Data Structure

```
/* definition of the header structure */
struct road_network  {          /* header, one per system */
  char network_id[16];          /* identification string */
  unsigned int id_code;         /* unique database identification code */
  unsigned int pathdata_rev;    /* revision of database format used */
  unsigned long num_connects;   /* number of connection points */
  unsigned long num_roads;      /* number of road units */
  unsigned long num_segments;   /* total number of segments in database
*/
  struct connect_unit *connect_list;  /* ptr to connect unit array */
  struct road_unit *road_list;        /* ptr to road unit array */
  struct road_element *segment_list;  /* ptr to road segment array */
  unsigned int checksum;        /* for stored database integrity check */
};
```

Figure A-5. Network Header Data Structure

A graphical representation of the linked list interconnections is shown in Figure A-6. The data is stored in a binary disk file. Creation and modification of these binary files is accomplished with an off-line tool called `Pathnet` (described in section A.4.1).

The object database is used for representation of the visual environment. This database is independent of the road environment database, in that they are contained in separate disk files and there are no references or links between them. However, they are related in that they share the same "physical" space. The visual environment database contains all of the information necessary to specify the objects in the visual field. In effect, the visual environment database is a comprehensive list of objects that exist in the visual field.

Like the road database, the object database is implemented as a set of arrays of data structures, which contain links between different levels of data structures. This database is different from the road environment database in that it is more hierarchical (Figure A-7).

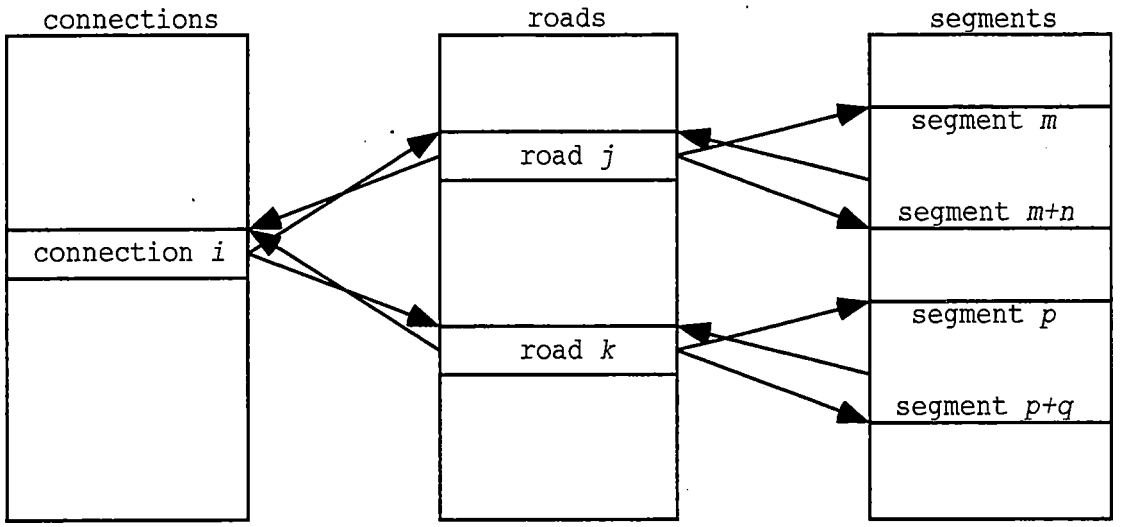


Figure A-6. Linked List Interconnections in Road Database File

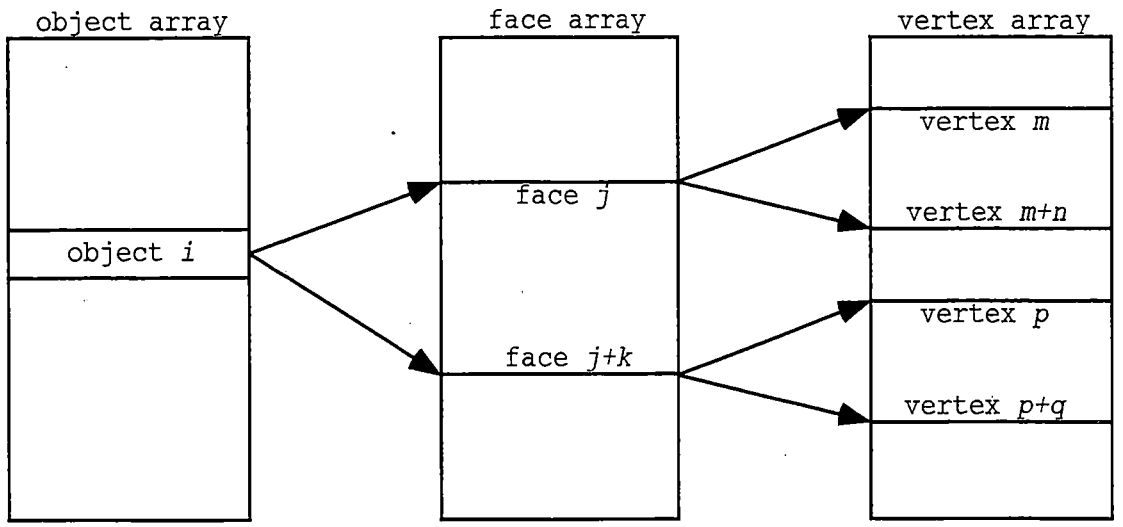


Figure A-7. Object Database Hierarchy

At the highest level is the header structure, containing references to the individual arrays. The next level is the list of objects. An object has the properties of object type, size, position in inertial space, and orientation in inertial space. The data structure also knows the number of faces that comprise the object, and has reference to the list of these faces.

A face is defined as a single polygon that is part of the object. A face has the property of color, and the data structure includes the number of vertices that form the face and reference to that list of vertices. A vertex is a point in inertial space, and all of the vertices that form the objects in the database are contained in a single list. The data structure for each vertex contains specification of the three-dimensional point that locates the vertex.

The objects defined by the visual environment database are static. That is, they have no dynamic state, and their parameters cannot change in time. The data structures that are used for specification of the linked list interconnections are shown in Figures A-8 through A-11. The data is stored in a binary disk file. Creation and modification of these binary files is accomplished using `Pathnet`.

```
/* define the database header for the entire terrain */
struct terrain_hdr  {
  char network_id[16];  /* should match that of path database file */
  unsigned int id_code; /* should match that of path database file */
  unsigned int otwdata_rev;  /* revision of OTW database format */
  float rcolor;        /* color coding for base terrain */
  float gcolor;
  float bcolor;
  int x_offset;        /* smallest X value */
  int y_offset;        /* smallest Y value */
  int x_cnt;           /* number of blocks in the X direction */
  int y_cnt;           /* number of blocks in the Y direction */
  int subblk_cnt;      /* number of sub-blocks in a block */
  int subblk_sz;       /* edge dimension of sub-block (m) */
  float **terr_zlist; /* ptr to terrain elevation row list (2D matrix) */
  float *terr_zdata;  /* ptr to terrain elevation data */
  int olist_cnt;       /* number of objects in the database */
  struct obj_header *obj_list;
  int flist_cnt;       /* number of object faces in the database */
  struct obj_face *face_list;
  int vlist_cnt;       /* number of face vertices in the database */
  struct vpoint *vrtx_list;
  int spare1, spare2, spare3;
};
```

Figure A-8. Object Database, Header Data Structure

```
/* define the representation of an object */
struct obj_header  {
  int obj_id;      /* coded type of object (determines base dimensions) */
  float xlate_x;   /* location of object center */
  float xlate_y;
  float xlate_z;
  float rotate;    /* horizontal planar rotation of object */
  float hscale;    /* height scale factor */
  float wscale;    /* width scale factor */
  int num_faces;   /* count of face polygons */
  int face1_indx;
  struct obj_face *face1_optr;
  int active;      /* use for lighting cues */
};
```

Figure A-9. Object Database, Object Data Structure

```
/* define the representation of an object face polygon */
struct obj_face  {
  float rcolor;       /* color coding */
  float gcolor;
  float bcolor;
  int vertex_cnt;     /* number of vertices in the face */
  int vertex1_indx;
  struct vpoint *vertex1_vptr;
  float oriented;     /* outward direction of face (wrt E) (horizontal) */
};
```

Figure A-10. Object Database, Face Data Structure

```
/* define the representation of a polygon vertex point */
struct vpoint  {
  float y_pt;   /* north of origin */
  float z_pt;   /* elevation above origin */
  float x_pt;   /* east of origin */
};
```

Figure A-11. Object Database, Vertex Data Structure

A.2.2 Network Interconnection

The individual simulation elements are programs that run on high-performance graphics workstations. These workstations are interconnected via a local area network. The interconnection protocol used for these connections is generically TCP/IP (Transmission Control Protocol, Internet Protocol). Specifically, the UDP (User Datagram Protocol) protocol is used, which is layered above IP (Internet Protocol).

The selection of this protocol is a significant element of the system architecture. The alternative protocol, TCP (Transmission Control Protocol), is connection-oriented and has guaranteed

reliability, while UDP is a connectionless datagram protocol. In other words, using TCP, each data packet sent from one machine to another is guaranteed to be delivered, regardless of the time required. If a network failure prevents a packet from reaching its destination, it will be retransmitted until successfully received. The UDP protocol, on the other hand, is a datagram-based protocol, without guaranteed reliability. This means that once a packet has been sent out on the network, there are no mechanisms in place to check for receipt or to retransmit the packet if it does not reach its destination.

In the case of a distributed interactive simulation system, the information being broadcast over the network is real-time state data, which is more sensitive to transmission delay than reliability. The vehicle state data, which includes current vehicle position and speed, is time-sensitive and is transmitted fairly frequently (about once every half-second). The implication is that, if a datagram packet was not properly received, an updated packet (with more recent information) would be sent in a timeframe which is comparable to that which would be required for retransmission of the original packet.

A.2.3 OTW View

The vehicle simulation includes an out-the-window (OTW) viewport for the locomotive engineer. Through this view, the operator can look out into the physical environment and see objects that exist in that environment. These objects are rendered as true three-dimensional objects, and maintain true perspective as the viewer moves through the environment.

True three-dimensional graphics require an immense amount of computational power, and are best implemented on specialized graphics workstations. The platform used for this implementation is an Indigo-2 Extreme workstation, manufactured by Silicon Graphics Inc. (SGI). This machine was considered a mid-line platform at the time of purchase (January 1995), and boasts computational performance of approximately 100 Megaflops and graphics performance of approximately 150,000 polygons per second.

Despite the specialized hardware, there remain limitations with synthesizing the OTW view. It was determined by the project team that, to maintain a jitter-free view, the frame update rate should be 20 frames per second or greater. In order to maintain this rate, and to enhance the perception of motion, several steps were taken in the design of the OTW view.

A night view was selected to reduce the number of required background objects. The view of objects up the track diminishes in light intensity, giving the perception that the objects are lighted by the vehicle headlights. To support that perception, the intensity of the block signal lights does not diminish with distance, since those signals are sources of light emission instead of light reflectors.

To limit the drawing requirements, the objects along the wayside are drawn as wire-frame objects. That is, only the outlines of the objects are drawn, and it is possible to see through the objects. In addition, objects are constructed from a small collection of fundamental shapes. This collection includes a cube, a pyramid, and an octagonal column. More complex objects are constructed from these basic shapes, such as a building or a pedestrian bridge.

Certain objects in the view are filled as solid shapes. These include the rails, the roadbed, the grade crossing roadway, and cars going across the grade crossing. Also painted as solids are the block signs and the kilometer posts, as they provide information via numerals on the signs.

In summary, the design of the OTW view represents a compromise between the desire to provide a realistic visual environment and the available computational resources. The resultant implementation provides a limited fidelity view with a medium fidelity frame rate. Despite the limitations, the view provides sufficient quality and task cues to allow immersion into the task of vehicle operation.

A.3 ACTIVE SIMULATION ELEMENTS

In the high-speed rail simulation system, the executable programs described as active simulation elements are those software modules that execute when the simulator is in operation. The current implementation of the system has two distinct types of active simulation elements: vehicle simulations and central traffic control (CTC) simulations. A vehicle simulation provides a human operator with a user interface which reasonably resembles an actual vehicle. These generally require two workstations for operation. A CTC simulation approximates the interface used by a human CTC operator.

The system architecture has been designed to support multiple vehicles coexisting in a common virtual environment, as well as multiple CTC interfaces. However, to date, the current implementation has been tested with only one each of the vehicle and CTC simulation elements.

The following subsections give a brief overview of the active simulation elements. As these programs total approximately 50,000 lines of C source code, a detailed description is beyond the scope of this document. Instead, the intent is to highlight the major feature of each. Figures are provided where appropriate, to convey a sense of perspective about the displays developed for the simulator. In all cases, the actual displays are color, and tend to feature lighter features on a dark background. However, to simplify the publication process, these displays are reproduced here in monochrome (black-and-white) and in reverse video (dark objects on a light background). Unless otherwise specified, the software described was designed and developed by this author.

A.3.1 CTC Simulation

The CTC simulation element provides an interface at which a human CTC operator can control a rail system. The primary display is a two-dimensional plan display of the rail system. This display is geometrically correct, providing the operator with an accurate view of the road curves and interconnections.

The individual block segments are identified with white marks at the block boundaries. Each block segment is color-coded, indicating the most restrictive signal level in that block. At the highest level of resolution (zoom), the block segment is shown with two lines, representing the signal indications in both directions. The actual color coding of the signals varies with the implementation.

The stations are depicted as magenta rectangles, and are identified by name alongside the icon. The switches are identified with magenta circles. At the highest level of resolution, a text display of detailed block information is available, which includes signal status.

The primary input mechanism for the CTC operator is the computer mouse. Using this device, the operator is able to zoom up and down (i.e., decrease or increase the field of view), as well as pan (i.e., move left or right) and tilt (i.e., move up or down). The operator also uses the mouse to

change the state of a switch. Through the use of select function keys on the keyboard, the operator can alter the state of a signal, either setting or clearing a signal in either direction.

In general, the signals are controlled automatically by the CTC simulation. The CTC simulation element takes the role of an automatic switch system, sensing the position of vehicles in the system and setting the signals accordingly. In this capacity, the CTC simulation acts as the responsible agent for the state of the system signals and switches. The change of any signal or switch state is initiated by the CTC simulation, and it is the responsibility of the CTC simulation to apprise all active vehicle simulations of the changes in signal and switch states as they occur.

To date, there are two versions of the CTC simulation. These differ only in the implementation of the signal control system—one version is set up to provide a seven-aspect signal system (in support of the display-aiding experiment), while the other provides a five-aspect signaling system (for the control automation experiment, described herein).

A.3.2 Vehicle Simulation—Display-Aided Version

For the purposes of the display aiding experiment (Askey 1995, Askey and Sheridan 1995), a train simulation was developed which featured advanced display technology. At the core of the simulation is a real-time process loop which includes the vehicle dynamics and user interface I/O processing. The primary user input devices are a throttle lever, mounted to the table near the computer display, the computer keyboard, and the computer mouse. The sole output device is the computer display.

The vehicle operation simulates that of an actual rail vehicle. This includes accurate vehicle dynamics, as well as realistic safety systems. The safety systems include an alerter system, which is used to ensure that the operator remains active in system operation, and an advanced train protection (ATP) system, which automatically enforces speed limit compliance.

A significant feature of this simulation is the capability for providing advanced displays to the user. These are enabled via command line options when execution of the program is initiated. In the basic mode, the fundamental instruments are provided for the operator. These include a speedometer, odometer, vehicle system instruments (such as brake pressure, trolley voltage, and

door status), in-cab signal indicator, system status and warning lights (such as emergency brake status, alerter warning, and ATP warning), and throttle position indicator.

The next level of display aiding includes a preview display. The purpose of the preview display is to provide the operator with information about the state of the roadway beyond that visible through the out-the-window view. The additional information includes block boundaries, kilometer boundaries, civil and signal speed limits, multi-block signal preview, and position of stations (among others). The preview distance may be adjusted by the operator, allowing trade-off between range and resolution.

The next level of display aiding includes a predictor display. This display adds three curves that project from the current position and speed indicator. The top-most line provides a prediction of the speed trajectory, assuming that the throttle is maintained at the current level. Modulating the throttle will cause this prediction to change. The middle line indicates the stopping speed trajectory when full service braking is used. The bottom line indicates a similar trajectory for the emergency brake. Both of the braking trajectories are a function of speed alone. This display allows an operator to improve the strategic planning of throttle and brake application to suit the conditions that are shown in the preview display.

The highest level of aiding is includes the advisor display. The advisor display shows a pre-computed speed trajectory which has been optimized for various higher-order performance factors, such as fuel consumption and passenger comfort. This speed trajectory is overlaid on the preview display. The operator then attempts to manipulate the throttle and brakes so that the predictor display matches the advised speed trajectory.

The train simulation has the option of providing an OTW viewport in the upper third of the display area. In addition, the simulation can be configured to drive an external OTW server, as described in section A.3.4.

The train simulation provides a recording system for storing state vehicle data which occurs during system operation. The data are stored in the form of ASCII data records, each of which contains a time stamp, an event code, the position and speed of the vehicle, and additional information as appropriate.

A.3.3 Vehicle Simulation—Control Automation Version

To suit the needs of the control automation experiment (described in detail in (Lanzilotta & Sheridan, 1998)), a second train simulation was implemented. Because the advanced displays, as used in the display-aiding experiment, were not relevant for the control automation experiment, the instrument panel was redesigned to better utilize the available space for the basic instrument set. In addition, the instruments themselves were redesigned to more closely approximate instruments currently in use. The primary input devices are the throttle lever and the computer keyboard. The output device is the computer screen.

The overall functional operation of this train simulation is similar to that of the train described in section A.3.2. The vehicle dynamics approximate those of an actual vehicle, and the same safety systems (alerter and ATP) are included.

The instrument panel display provides the basic instruments for vehicle operation. At the center is a large round speedometer, with a red pointer for the current speed and a smaller yellow pointer behind for use with the automation systems. The in-cab signaling system is located above the speedometer, and contains indications for both the current block (the signal most recently passed) and the next block (the upcoming signal). There are four small round gauges for system state monitoring, including brake pressure, bearing temperature, and trolley voltage. The vertical LED bars are ammeters to indicate the current applied to each of the four traction motors. There are status and warning indicator lights for emergency brake status, ATP warning, alerter warning, and door status, as well as a clock and indicator lights for the control automation modes.

There are three control automation modes: cruise control, programmed stop, and autopilot. The cruise control system operates much like an automotive cruise control—the operator selects a desired cruise speed, and the system maintains that speed, applying throttle as required. The operator also has provision for making fine adjustments of the set speed, via keyboard button inputs. The control loop for the cruise control is based on a proportional-integral (PI) control loop for speed.

The programmed stop system is designed to stop the vehicle at designated stopping points. This implementation stops the vehicle at the end of the current block. (Stations are always located at a

block junction.) The programmed stop system uses a look-up table, based on the stopping curves for the vehicle, to modulate the brakes such that the vehicle stops at the appropriate point. There are several safeguards to prevent dangerous situations from occurring. The first is an overspeed protector—the programmed stop system cannot be properly engaged if the vehicle speed is above 80 km per hour. Another safety system guards against late application—if the speed is too great at the position of application for the vehicle to be stopped using the service brakes, the system detects a fault. In both cases, the emergency brake is applied.

The autopilot system utilizes a pre-programmed speed trajectory, and uses both the traction motors and the brake system to maintain that speed. In effect, the pre-programmed speed trajectory takes the place of the operator command to the cruise control system. In addition, the autopilot makes use of the brake system to slow the vehicle for low speed sections. When approaching station stops, the autopilot invokes the programmed stop system automatically. Using the autopilot, normal speed control of the vehicle becomes a "set and forget" operation—once engaged, no further input is required.

This version of the train simulation can be configured to display either the instrument panel or the OTW view. When configured as the instrument panel, the OTW server (section A.3.4) may be optionally invoked on a separate machine to provide the OTW view. Similarly, when the train simulation is displaying the OTW view, the dashboard server (section A.3.5) may be optionally invoked on a separate machine to provide the instrument panel.

This version of train simulation also provides a recording system for storing state vehicle data which occurs during system operation. As with the previous train simulation, the data are stored in the form of ASCII data records, each containing a time stamp, an event code, the position and speed of the vehicle, and additional information as appropriate.

A.3.4 Vehicle Simulation—OTW Server

The OTW server is an independent module that is used for secondary display of the OTW viewport. Its function is to display the OTW view only, as a slave to a primary train simulation.

Communication between the primary train simulation and the OTW server is accomplished via the local area network. The primary simulation element sends state information to the OTW

server, via the local area network, which then updates its internal estimation of the vehicle state. To maintain smooth graphics output in the absence of new state information, the OTW server performs position estimation using dead reckoning, based on the last available state information.

A.3.5 Vehicle Simulation—Dashboard Server

Similar to the OTW server, the dashboard server provides a dashboard display on a secondary machine. The dashboard format used in this process is identical to that from the control automation train simulation. Like the OTW server, the dashboard server is in communication with the primary process via the local area network. In this case, the data transmitted from the primary process to the server is only the data required for the instrument panel display.

A.4 SUPPORT SOFTWARE

In addition to the active simulation elements, outlined in section A.3, there are a number of programs that exist to support the overall simulation system. One of these is used for creating and modifying the road and object databases used by the train simulation system. The other are data analysis programs, used for post-processing the data obtained from the simulation system.

A.4.1 `Pathnet`

The off-line program `Pathnet` is an important support element of the simulation system. The virtual environment is contained in two types of files, known as the road and object databases. These files are used by the active simulation elements during operation. `Pathnet` is a tool used for the creation and modification of these databases.

A.4.2 Additional Data Analysis Tools

Additional programs have been implemented as required, to provide support tools for the various experiments that are performed. This section provides a brief introduction to the programs used in support of the control automation experiment.

The program `transform` was developed to transform the raw data, generated by the train simulation, into the failure response times required by the control automation experiment. This program reads through an entire raw data file, identifies the failures that have occurred during the

session, and identifies the operator response to those failures. The output of the program is a new file, containing the summary of the failure responses.

The program `bonus_process` was developed to calculate the bonus point performance of the subjects in the control automation experiment. As with `transform`, `bonus_process` reads through the entire raw data file, and selects those data points that required for calculation of the bonus points. These include response to failures, as well as station stopping performance and schedule maintenance. The output of the program is a text-based summary which is directed to the terminal output (computer screen).

The program `ss_process` was developed to convert the raw data file into a safety state trajectory. Once again, raw data files are provided as input, and the resultant output is a text-based summary of the safety state values, as a function of time.

The program `mtbf_calc` was developed to convert the safety state trajectories into a risk probability function. The input to this program is a set of safety state trajectories (generated by `ss_process`), and the output is a set of risk trajectories that result from transforming the safety state trajectories with the risk function. These risk trajectories are stored as disk files.

The program `risk_stats` was developed to perform statistical analysis of the risk trajectory output from `mtbf_calc`. For each of the risk trajectory files, a corresponding statistics file is generated which summarizes the state occupancy for that trajectory, as well as the average risk probability over the trajectory.

A.5 SOFTWARE ENGINEERING ISSUES

In the development of any substantial body of software like the high-speed rail simulation system, issues of software engineering must be addressed to ensure the ongoing viability of the project. These issues are especially critical if the project is to be developed and sustained by several software engineers. The following sections outline some of the software engineering issues that were addressed.

A.5.1 Shared Libraries

In an effort to modularize the code to the greatest degree possible, the concept of shared libraries is used wherever possible. This concept forms a basis for sharing and re-use of code among separate elements in the system.

There are two distinct scenarios where libraries are especially prudent. In the first, there are a number of different pieces of software that need one or more functions that are related. Development of a single library for these functions allows several modules to share the code, lessening the development load and unifying the interface.

The second scenario occurs in the case where two separate processes need to communicate using a common protocol. By incorporating all of the functions related to that protocol in a single library, it is easier to ensure consistency throughout the function set.

The high-speed rail simulation system has four shared libraries: the database interface library (`libdb`), the network interface library (`libnet`), the OTW interface library (`libotw`), and the schedule library (`libsched`). The database interface library contains those functions used for loading and interpreting a road database. The network interface library contains those functions used for inter-process communication over the local area network. The OTW interface library contains those functions used for displaying an OTW viewport. The schedule library contains those functions for loading and interpreting a schedule database.

A.5.2 Development File Hierarchy

It was recognized early in the software development phase that there would be a substantial amount of software development required. In order to partition the project into manageable chunks, a development file tree was created.

The root of the tree exists at a level separate from and parallel to the users personal directories. Located at `/usr/projects/rail-sim`, this root directory contains distinct directories for vehicle simulation code (`vehicles`), CTC simulation code (`ctc`), shared libraries (`lib`), database information (`database`), experiment-specific scripts (`exp`), and data recording (`data`).

In the `vehicles` directory, there are four subdirectories (`veh-1`, `veh-2`, `veh-otw`, and `veh-dash`), each of which contains the source code modules for an executable program. In addition, this directory includes subdirectories for common code (`common`), local libraries (`lib`), and local include files (`include`).

Similarly, the `ctc` directory contains two subdirectories (`ctc-1` and `ctc-2`) for source code, as well as the subdirectories `common`, `lib`, and `include`.

The `lib` directory contains four subdirectories for source code: `libdb` for the database interface library, `libnet` for the network interface library, `libotw` for the OTW interface library, and `libsched` for the schedule library.

The `exp` directory contains subdirectories for the display aiding experiment (`exp-disp`) and the control automation experiment (`exp-auto`). These subdirectories also appear under the `data` directory, to allow segregation of the data obtained from the two experiments.

A graphical depiction of the file system hierarchy is shown in Figure A-12. The file system hierarchy is replicated on all the machines.

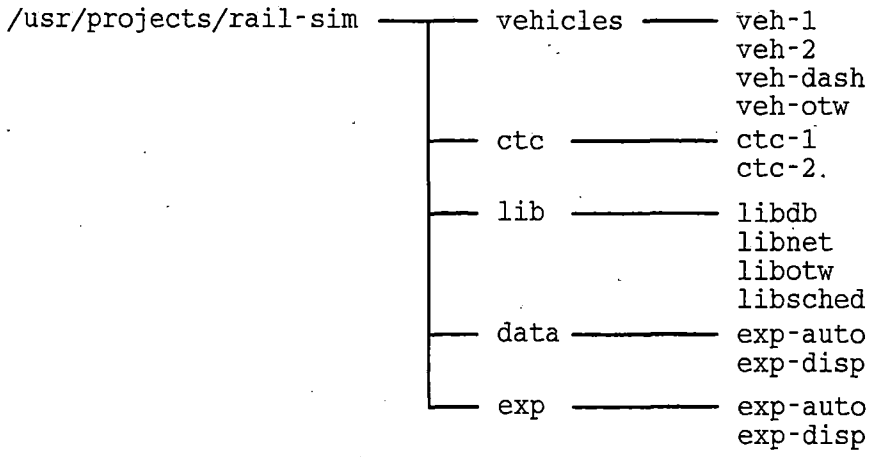


Figure A-12. File System Hierarchy

A.5.3 Software Build Engineering

The primary tool used to support software build engineering is `make`, which is a standard Unix tool. The build specifications for each of the executable modules in the system is contained in a file named `Makefile`, located in the local directory. In addition, there is a master `Makefile`, located in `/usr/projects/rail-sim`, which will rebuild the entire system from scratch.

A.5.4 Revision Control

The primary tool used to support software revision control is `RCS`, which is a publicly-available and widely-used tool. It is supplied as part of the SGI development environment, along with the compiler, linker, and source-code debugger. `RCS` stores multiple versions of a source code file in a separate archive file. Each time the code is changed and "checked-in" to the archive file, the differences between the new and old versions are recorded, and comments are inserted to provide a "paper trail." The comments include the date and time of revision, as well as the person that was responsible for those changes.

At any point in time, any of the revisions that are stored in the archive file may be retrieved without risk to any of the other versions. Thus, it is possible (and easy) to revert back to an earlier version of the software, without losing any of the subsequent changes.

A majority of the source code modules also include "markers" for storing `RCS` header information. These "markers" allow the `RCS` data to be included in the executable module. It is then possible to identify the source code modules that comprise an executable module, even if the executable has been separated from the source code directories.

A.6 SYSTEM CONFIGURATIONS TO SUPPORT EXPERIMENTS

To date, two experiments have been conducted using the high-speed simulation system. The first was an exploration into the effects of display aiding on operator performance. The second was focused on identifying the effects of control automation on operator performance. The configuration of the simulation system was tailored in each case to the objectives of the experiment.

In the case of the control automation experiment, the alternate CTC simulation (five-aspect signaling system) was run on a Personal Iris. The control automation train simulation was run on the Indigo-2 machine, configured to display the OTW view, while the second Personal Iris machine was executing the dashboard server program. A system schematic of this configuration is shown in Figure A-13.

For the display aiding experiment, the CTC simulation used was the version that supported the seven-aspect signal system. This was run on a Personal Iris machine. The train simulation was run using two machines. The Indigo-2 was used as the primary train simulation machine, which executed the display aiding train simulation. The second machine was a Personal Iris, executing the OTW server. A system schematic of this configuration is shown in figure A-14.

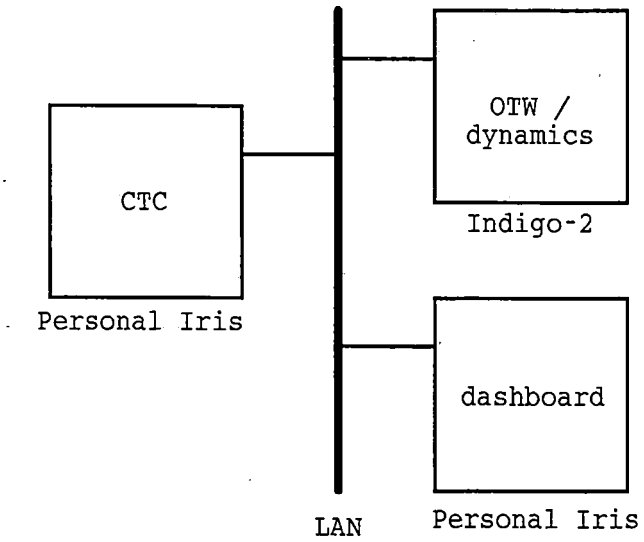


Figure A-13. System Schematic of Control Automation Experiment Configuration

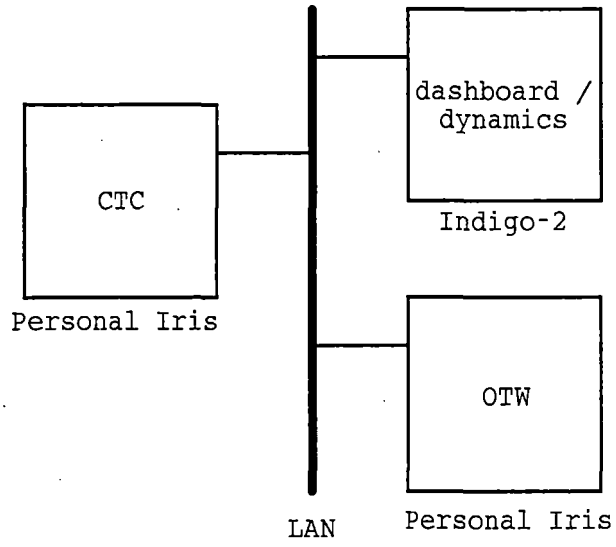


Figure A-14. System Schematic of Display Aiding Experiment Configuration