Proceedings of the 1998 Annual Reliability and Maintainability Symposium, Anaheim, California, Jan. 19-22, 1998.

MEADEP and Its Applications in Evaluating Dependability for Air Traffic Control Systems

Dong Tang • SoHaR Incorporated • Beverly Hills
Myron Hecht • SoHaR Incorporated • Beverly Hills
Jady Handal • Federal Aviation Administration • Washington D.C.
Loni Czekalski • Federal Aviation Administration • Washington D.C.

Keywords: availability evaluation, failure measurement, Markov model, parameter estimation, software tool

SUMMARY & CONCLUSIONS

MEADEP is a user-friendly dependability[1] evaluation tool for measurement-based analysis of computing systems including both hardware and software. Features of MEADEP are: a data processor for converting data in various formats (records with a number of fields stored in a commercial database format) to the MEADEP format, a statistical analysis module for graphical data presentation and parameter estimation, a graphical modeling interface for constructing reliability block and Markov diagrams, and a model solution module for availability/reliability calculation with graphical parametric analysis. Use of the tool on failure data from measurements can provide quantitative assessments of dependability for critical systems, while greatly reducing requirements for specialized skills in data processing, analysis, and modeling from the user. MEADEP has been applied to evaluate dependability for several air traffic control systems (ATC) and results produced by MEADEP have provided valuable feedback to the program management of these critical systems.

1. INTRODUCTION

Automation of dependability evaluation has been realized by the computer engineering community for over 15 years during which many dependability modeling tools were developed [1] [9]. Some of representative tools are SAVE [2], SHARPE [11], and UltraSAN [12]. The emergence of these tools has given impetus to the applications of advanced modeling and evaluation techniques. These tools were developed for building models based on parameters, with emphasis on model solution techniques. Although many theoretical modeling and solution issues such as non-exponential failure arrival/recovery times and numerical stiffness[2] have been addressed, practical issues such as data analysis, parameter estimation, and graphical user interface (GUI) have rarely been considered in these tools. An exception in the GUI aspect is UltraSAN.

From the research viewpoint, a capable tool should be able to handle any probability distribution. However, this capability is rarely used in practical engineering projects that require field measurements for the following reasons: First, it is very difficult to measure and identify failure arrival distributions for each component. When failures are rare and distributed over multiple replicas, this identification is impossible. Second, errors caused by using the exponential approximation of actual distributions may not be as significant as errors introduced in other evaluation steps such as measurement, parameter estimation and model construction. In our experience, a dependability assessment that is accurate to the right order of magnitude is good enough in practice, taking account of various possible errors. It is thus questionable, if not impossible, to make costly efforts to identify all actual distributions to correct minor errors. Third, in order to have a conservative evaluation, upper bounds based approaches (failure rate upper bounds, instead of means) are usually used in engineering. Official standards (e.g., Military Handbooks) for estimating failure rate confidence intervals typically assume the exponential distribution, and these standards have been followed by reliability engineers for many years.

Thus, from the engineering viewpoint, it is desirable to have software tools which integrate data processing, statistical analysis, reasonable dependability modeling and evaluation,[3] and a user-friendly interface to provide non-expert users with an easy-to-operate environment for producing quantitative dependability evaluations for real systems. MEADEP (MEAsure DEPendability), introduced in this paper, is such a tool. The purpose of developing MEADEP was to facilitate the

[1]The *dependability* concept was proposed in the *15th International Symposium on Fault-Tolerant Computing* (FTCS-15) [7] and revised in FTCS-25 [8]. Dependability is defined as the "property of a computer system such that reliance can justifiably be placed on the service it delivers." Major measures of dependability include *reliability*, *availability*, *safety*, and *maintainability*.

[2]In a Markov dependability model, failure rates tend to be very small and recovery rates tend to be much larger. Stiffness means the technical difficulty in model solution caused by the difference between the largest and the smallest parameters in the model.

[3]The exponential distribution is typically assumed in the model evaluation.

use of measurement-based dependability analysis methods [5] [13] and to reduce the cost of such analyses so that it can become an integral part of engineering projects where dependability is an important consideration.

2. OVERVIEW OF MEADEP

MEADEP is a failure data based dependability analysis and modeling tool. Dependability measures generated by MEADEP are either directly obtained from data, such as failure rate and event distribution, or evaluated by combined use of failure data and dependability models, such as system level reliability and availability. Thus, two basic types of input to MEADEP are:

- Data — structured failure reports containing information on failure time, location, type, impact and other failure characteristics
- Models — graphical specifications of dependability models including reliability blocks and Markov chains

The output of MEADEP consists of results obtained from data and results evaluated from models where model parameters are either estimated from data or specified by users. Results obtained from data include:

- Pie charts for event distribution
- Progressive curves over time for Mean Time Between Events (MTBE) and its confidence interval
- Histograms for Time Between Events (TBE) and for Time To Recovery (TTR) distributions, with super-plotting of typical analytical functions, accompanied by the results of their goodness-of-fit tests
- The mean, lower and upper bounds for failure rate, recovery rate, and coverage
- Clustering analysis[4] statistics

Results evaluated from models include:

- Mean Time Between Failures (MTBF)
- Reliability for a given time period
- Steady-state availability

The functions of MEADEP include: data processing and editing, parameter estimation, graphical data analysis, graphical model generation, and model solution. MEADEP has the following features:

Support for data conversion: Structured data in a variety of formats (ASCII Delimited Text, Access, dBASE, Paradox, etc.) can be converted to the MEADEP data format.

Estimation of parameters from data: Typically used parameters (failure rate, coverage, etc.) and their upper and lower bounds at a certain level of confidence are estimated by statistical routines taken from mature numerical libraries.

Graphical presentation of data: A number of graphical formats are provided to display dependability characteristics for data (e.g., pie charts and histograms).

Graphical Input of models: A graphical "drag and drop" interface allows the user to create models hierarchically, out of reliability block diagrams (including the k-out-of-n block) and Markov reward models [3].

Parametric analysis in solution: The model solution portion of MEADEP allows a model to be run with a range of user-specified values for a selected parameter, and the results can be displayed graphically.

A library of dependability models: A library of dependability models, including primitive models for typical fault-tolerant architectures and complex models for real critical systems are included in MEADEP for reuse by users.

User friendly interface: For all of its functions, MEADEP provides a user-friendly GUI featuring menus, dialogs, pictures, printing previews, and extensive on-line help information.

3. DETAILS OF MEADEP

Figure 1 is a layout of MEADEP. In the figure, rectangles represent software modules and ellipses represent input or output files. The *Data Pre-Processor* (DPP) module, interacts with the user to convert source data to the MEADEP internal data. The source data can be manually generated structured trouble reports or computer generated event logs. The *Data Editor and Analyzer* (DEA) module is used to edit internal data and to perform statistical analysis on the data. Parameter values estimated from the data by this module can be inserted into the text modeling file generated by another module, the *Model Generator* (MG). The MG module provides a graphical user interface for the user to draw model diagrams and then to generate, from the diagrams, a text modeling file that contains model specifications suitable for solution. Model diagrams can be imported from library files containing predefined models to reduce development time. The *Model Evaluator* (ME) module produces results based on the model specifications and

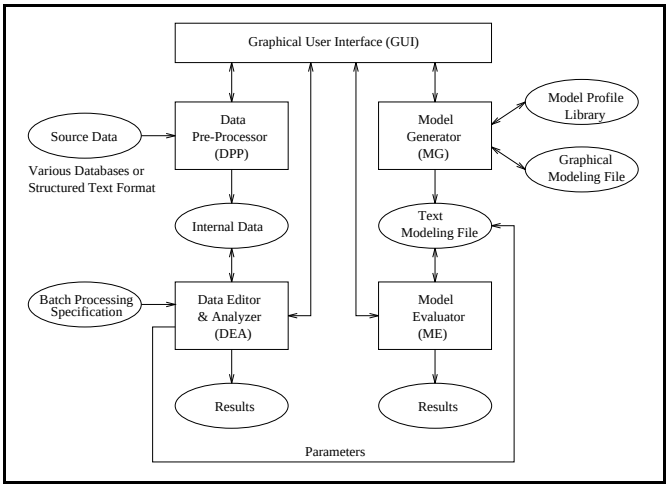

Figure 1 Layout of MEADEP

[4]Clustering analysis is a method to identify related events. The method merges multiple events into a single cluster if time between any two neighbor events is less than a specified interval.

parameters in the text modeling file. All modules are integrated with the *Graphical User Interface* (GUI).

Source data formats supported by MEADEP include ASCII delimited text and a variety of databases such as Access, dBASE and Paradox. The MEADEP data are composed of records representing events and stored in the Access format. The data conversion performed by the DPP module is guided by a mapping, supplied by the user, between source data fields and the MEADEP data fields (an example is shown in Figure 2). The user is also allowed to generate internal data manually by typing in each record with the DEA module. This option is useful when the source data are hand-written event logs. The MEADEP data schema consists of the following 13 fields:

- Event ID (identification of the event)
- Date (date on which the event occurred)
- Time (time at which the event occurred)
- Duration (duration of the event)
- Type (type of the event)
- Location (where the event occurred)
- Subsystem (subsystem in which the event occurred)
- Component (component in which the event occurred)
- Cause (cause of the event)
- Criticality (criticality of the event)
- Coverage (indication if the failure event has been covered by redundancy)
- Count (number of the occurrences of the event)
- Notes (description for or comments on the event)

The DEA module works on data with the above format and performs statistical analysis. It has three major categories of functions: data editing, graphical analysis, and parameter estimation. The data editing functions include: correctness checking for the data format, searching records, sorting records, adding records, deleting records, modifying records, and other data editing functions. The graphical analysis can generate: pie charts for event distributions, histograms for TBE and TTR distributions, and progressive curves for MTBF and its confidence interval over the time axis. The parameter estimation category provides the mean and upper and lower bounds at a specified confidence level for: MTBF, Mean Time To Recovery (MTTR), failure rate, recovery rate, and fault-tolerance coverage (estimates are also given even if failures are rare). These estimates can then be inserted into a text modeling file (discussed later) to be bound to model parameters. Parameter estimation can be specified by the user through multiple window dialogs and can also be specified by a predefined batch command file. The latter can save time significantly for sophisticated and repeated analysis.

MEADEP allows the user to super-plot, over a histogram, five different analytical probability distribution functions (pdf) determined by the sample mean and sample variance: exponential, gamma, Weibull, normal and lognormal. Meanwhile, the estimated parameters for these functions as well as the results of the Chi-Square and Kolmogorov-Smirnov goodness-of-fit tests [5] are displayed.

The MG module is a graphical "drag and drop" interface for constructing dependability models. A model is developed hierarchically, from the top level to the bottom level, forming a tree-structure. Each node in the tree is a diagram of serial or parallel reliability blocks (block diagram), a k-out-of-n model (block diagram), or a Markov chain (Markov diagram). The user can navigate from one diagram to another to build models. For a block diagram, the user can draw blocks and links between blocks. Each elemental block (which has no submodels) is associated either with a failure rate and recovery rate or with a constant representing availability. For a Markov diagram, the user can draw states and transition arcs between states and specify: the reward for each state, the rate for each transition arc, the initial state and the failure state for the model. When the model construction is completed, the diagrams can be saved in a graphical modeling file for reuse.

The model can also be translated into a text modeling file which contains model specifications for directing the ME module to evaluate the model for results. In the process of generating the text modeling file, MG will ask the user to define values for each parameter. Parameter values can be either entered manually at this time or inserted by the DEA module at a later time when they are estimated from data. The generated text modeling file consists of the following four sections:

- Parameter Initialization (parameter names and their bound values)
- Evaluation Sequence (a sequence of expressions)
- Markov Definition (description of states, transitions and rewards for all Markov chains)
- Output Specification (names of the model/submodels for which results will be generated)

A library of model profiles is provided with MEADEP. A model profile (or *library file*) is a graphical modeling file that defines the structure of a dependability model for a particular system or subsystem, but does not contain parameter values. It can be read into a screen diagram during the modeling process. Thus the user can make use of these files in developing his own model. The user can also save frequently used model diagrams as library files for reuse.

The ME module has two major functions: editing the text modeling file (editor) and evaluating the model (evaluator). The editor allows the user to revise models and parameters and then to see the effects of the revisions on results immediately. The evaluator provides regular results and parametric analysis. For the regular results, the modeling file is evaluated once and the results for all models (or submodels) listed in the output specification section of the file are generated. In the parametric analysis, the user specifies a loop and multiple sets of results are generated graphically. One of the following four loop types can be chosen by the user:

- *Loop by Increment*: The user specifies a parameter, an initial value, an upper bound, and an increment. In each loop cycle, the selected parameter is increased by the increment and results are recalculated.

- *Loop by Value Set*: The user specifies a parameter and a set of values. In each loop cycle, a different value in the value set is assigned to the selected parameter and results are recalculated.
- *Loop by Time Increment*: The user specifies an initial time, an end time, and a time increment. In each loop cycle, the Time variable is increased by the increment and reliability is calculated for the new Time value.
- *Loop by Time Set*: The user specifies a set of values for the Time variable. In each loop cycle, a different value in the set is assigned to the Time variable and reliability is calculated for the new Time value.

A text file containing a list of parameters and their initial values (*parameter file*) can be included in the model evaluation process. When this option is selected, the parameter file is processed before the text modeling file. Any parameter used in the model can be initialized in the parameter file, modeling file, or both. If a parameter is initialized in both files, the value in the modeling file will override that in the parameter file. The parameter file may contain parameters other than those defined in the modeling file. This provision allows a standard parameter list to be reused for multiple models, without having to input these parameter values into each model in the modeling process.

All interactions between the user and the software modules discussed above are through a graphical user interface. The interface provides convenient menus, dialogs, pictures, printing previews, and extensive help information. One of the useful MEADEP features is its ability to convert a model diagram or a graphical output (histogram, curve, etc.) to the popular Windows metafile format (wmf). The format allows these diagrams or graphs to be imported to Windows-based word processors such as Microsoft Word and WordPerfect (Figures 4 and 5 in this paper were generated in this way).

MEADEP was developed on Windows 95 using Microsoft Visual C++, the Open Database Connectivity interface, the IMSL Numerical libraries, and the Olectra Chart graphical package. The parameter estimation methods used were based on [6] [14], and the model solution methods used were based on [10] [15]. For several test cases, including the complex VSCS availability model to be discussed in the next section and a Markov model that shows a certain degree of stiffness (having a failure rate of 10^{-8} and a recovery rate of 10^{2}), the steady-state and transient results produced by MEADEP were the same as those produced by SHARPE [11].

4. APPLICATIONS OF MEADEP ON ATC SYSTEMS

MEADEP has been used to evaluate dependability for two major air traffic control systems: the Voice Switching and Control System (VSCS) and the Air Route Traffic Control Center (ARTCC). The VSCS is a digital communication system responsible for voice switching between pilots and air traffic controllers. The modeled ARTCC components include a set of radars, radar data processing and display subsystems. Both evaluations were based on measurements from field systems and provided valuable feedback to the project management.

For the VSCS, an operational availability model which is a hierarchy of 23 reliability block and Markov model diagrams, was developed to represent the system. The Program Trouble Report (PTR) data from the first 12 VSCS systems installed in major U.S. air traffic control centers were used to estimate model parameters. Figure 2 shows a data conversion screen which maps the VSCS PTR data fields that are useful for parameter estimation to the MEADEP data fields. Figure 3 shows a parameter estimation screen in which a failure rate is to be estimated from the converted VSCS data.

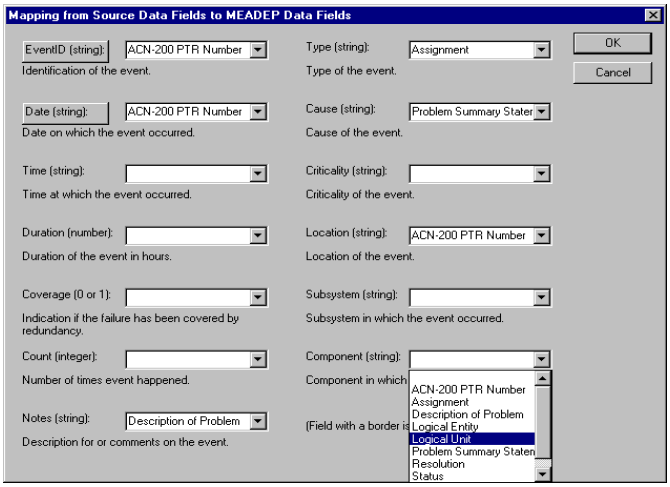

Figure 2 Data Conversion for VSCS PTRs

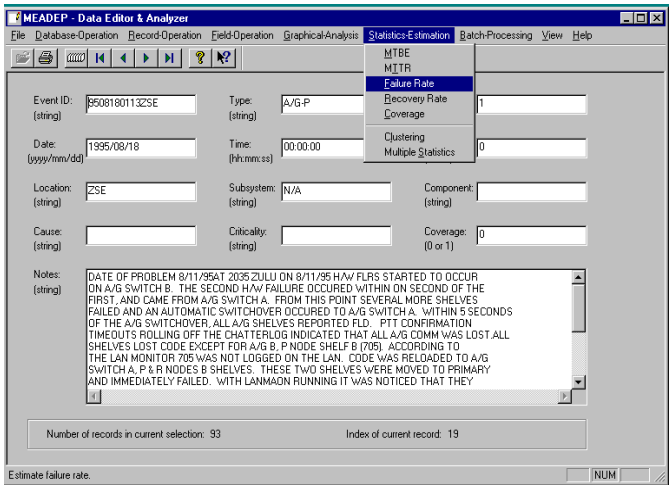

Figure 3 Parameter Estimation from Converted VSCS Data

Figure 4 shows a submodel of the VSCS Operational Availability Model: the Air to Ground Switch Subsystem Model. The subsystem has two redundant air to ground channels (AGC) and a pair of redundancy control units (DMC). In the diagram, the two heavy square blocks marked as AGC and DMC represent two sub-diagrams (submodels) of this diagram (the two blocks can be expanded to view the lower level diagrams). Parameters λ_{agc} and μ_{agc}, which are used in this model and are also placed in the AGC block, will be evaluated

from the lower level diagram AGC. Similarly, parameters λ_{dmc} and μ_{dmc} will be evaluated from diagram DMC. The data used in this evaluation represented unexpected failures and outages in a cumulated period of 2,212 system operational days. The system availability evaluated from the data as of June 30, 1996 was on the level of five 9's (0.99999) and was dominated by software failures.

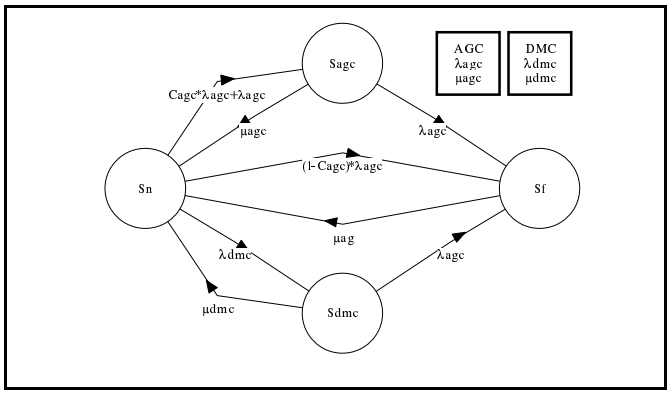

Figure 4 VSCS Air to Ground Switch Subsystem Model

Two lessons can be learned from this evaluation: First, if no major failures occur in the future, it would take 15 years of normal operation for all of the 21 sites to demonstrate an availability of the required seven 9's at the 80% confidence level, using the upper bound based evaluation method discussed in [Tang95]. Therefore, in order to quantify availability for systems with such high requirements, it is necessary to introduce accelerated testing and assessment methods. Second, the only VSCS system-level failure occurring in the monitored period was the Seattle VSCS Type I failure (a problem that precludes the primary air traffic control system mission objective of controlling aircraft) of August 11, 1995 when all of the eight air to ground telephony switch shelves (for simplicity, AG shelf) failed. The event was widely reported by the media. The problem diagnosis provided by the manufacturer identified the likely root cause as an undetected fault in a memory chip. The fault corrupted the length field (set to 0-length) of a message broadcast by an A/G shelf. All other A/G shelves (including primaries and standbys), upon receiving this invalid message, reset and cleared all application code from their processors simultaneously due to a general protection fault caused by the 0-length. This catastrophic failure was the result of three rare conditions [4]: a memory chip fault, hardware memory error detection function disabled, and software defects in data consistency checking. This event indicates that data errors can be caused by hardware faults, and that software should prevent catastrophic failures from these errors. To target this issue, testing with random data error injection may be a solution.

For the ARTCC system, a hierarchical model consisting of 14 reliability block and Markov model diagrams was developed. One of the two major parts modeled is the Radar System. A submodel in the Radar System is the Radar Coverage Markov Reward Model shown in Figure 5. This model represents 12 radar sites which provide surveillance coverage for a certain airspace. The 12 circular areas covered by these radar sites intersect with each other. If all radar sites are operational (State S_0), there is 100% airspace surveillance coverage and the reward rate is thus 1 (shown in the circle of S_0). As the number of failed radars increases, the surveillance coverage, or reward rate, decreases. In the figure, the reward rate for a single radar site failure (S_1) is 0.992, and for two radar site failures (S_2) it is 0.98, etc. Each radar site is modeled by a lower level diagram, Rsite. The failure rate (λ_{rsite}) and recovery rate (μ_{rsite}) of the radar site is evaluated by this lower level diagram, as shown by the square block (which can be expanded to view the lower level diagram) in the figure. The expected airspace surveillance coverage is the availability evaluated from this Markov reward model.

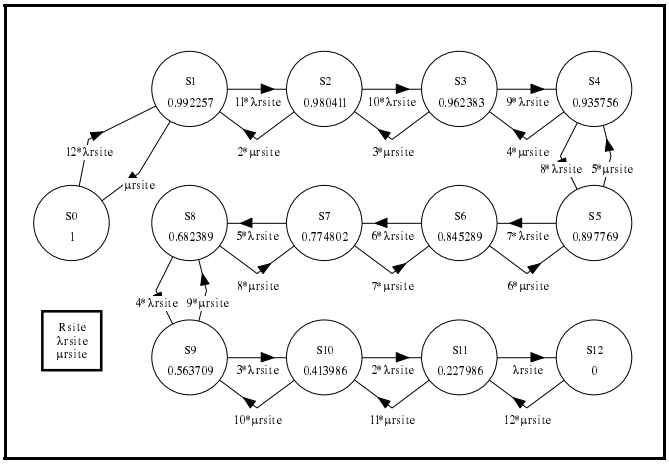

Figure 5 Radar Coverage Markov Reward Model

In addition to the 12 radar sites, the Radar System also includes the Automation Subsystem, which processes information from the radars, and the Display Channel, which displays objects detected by the radars. Figure 6 shows the relative contributions of these three subsystems to the Radar System unavailability. The radar sites are a major availability bottleneck (54%)[5]. However, all of the three subsystems have the same order of availability of three 9's, based on the data that represented both scheduled and unscheduled outages.

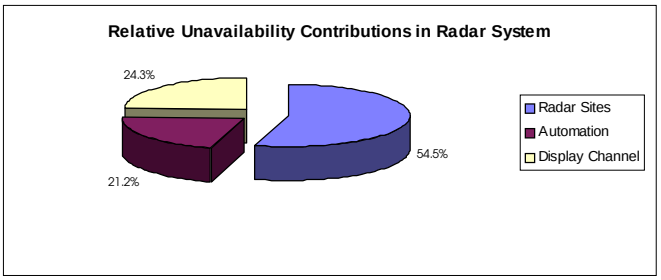

Figure 6 Radar System Unavailability Distribution

Figure 7 shows the distribution histogram of time between outage events for a particular radar site, generated by MEADEP. The curves plotted on the screen are the

[5]The Radar Sites availability is a performance-related availability. It represents the average space coverage for the addressed area.

exponential, gamma, and Weibull functions. All three function passed both Chi-Square and Kolmogorov-Smirnov goodness-of-fit tests at the 0.1 significance level.

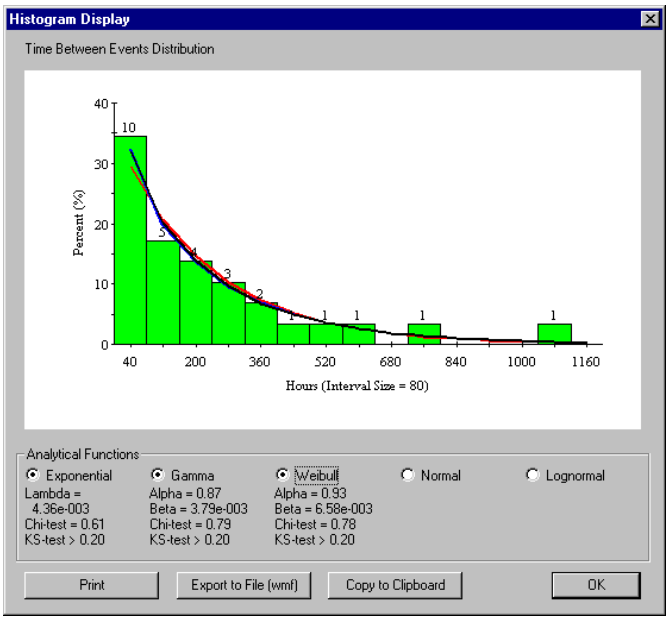

Figure 7 Distribution of Time Between Outages for a Radar Site

ACKNOWLEDGMENT

This work was partially supported by the U.S. Nuclear Regulatory Commission (NRC) under Contract NRC-04-95-081 and partially supported by the Federal Aviation Administration (FAA) under Contract DTFA01-93-Y-0169, Subcontract 96-107.

REFERENCES

[1] R. Geist and K. S. Trivedi, "Reliability Estimation of Fault-Tolerant Systems: Tools and Techniques," *IEEE Computer*, July 1990, pp. 52-61.

[2] A. Goyal, et al., "The System Availability Estimator," *Proceedings of the 16th International Symposium on Fault-Tolerant Computing*, June 1986, pp. 84-89.

[3] A. Goyal, S. S. Lavenberg and K.S. Trivedi, "Probabilistic Modeling of Computer System Availability," *Annals of Operations Research*, No. 8, March 1987, pp. 285-306.

[4] H. Hecht, "Rare Conditions — An Important Cause of Failures," *Proceedings of the 8th Annual Conference on Computer Assurance*, Gaithersburg, MD, June 1993, pp. 81-85.

[5] R. K. Iyer and D. Tang, "Experimental Analysis of Computer System Dependability," *Fault-Tolerant Computer System Design*, D. K. Pradhan (Ed.), Prentice Hall PTR, Upper Saddle River, NJ, 1996, pp. 282-392.

[6] D. Kececioglu, *Reliability and Life Testing Handbook*, Vol. 1 & 2, PTR Prentice Hall, Englewood Cliffs, NJ, 1993.

[7] J. C. Laprie, "Dependable Computing and Fault Tolerance: Concepts and Terminology," *Proceedings of the 15th International Symposium on Fault-Tolerant Computing*, June 1985, pp. 2-11.

[8] J. C. Laprie, "Dependable Computing: Concepts, Limits, Challenges," *Special Issue of the 25th International Symposium on Fault-Tolerant Computing*, June 1995, pp. 42-54.

[9] J. F. Meyer, "Performability: A Retrospective and Some Points to the Future," *Performance Evaluation*, Vol. 14, Feb. 1992, pp. 139-156.

[10] A. Reibman and K. S. Trivedi, "Numerical Transient Analysis of Markov Models," *Computational Operations Research*, Vol. 15, No. 1, 1988, pp. 19-36.

[11] R. A. Sahner and K. S. Trivedi, "Reliability Modeling Using SHARPE", *IEEE Transactions on Reliability*, vol 36, February 1987, pp. 186-193.

[12] W. H. Sanders, W. D. Obal II, M. A. Qureshi, and F. K. Widjanarko, "The UltraSAN Modeling Environment," *Performance Evaluation*, Vol. 24, No. 1, Oct/Nov 1995, pp. 89-115.

[13] D. P. Siewiorek and R. W. Swarz, *Reliable Computer Systems: Design and Evaluation*, *Digital Press*, Bedford, Mass., 1992.

[14] D. Tang and M. Hecht, "Evaluation of Software Dependability Based on Stability Test Data," *Proceedings of the 25th International Symposium on Fault-Tolerant Computing*, Pasadena, CA, June 1995, pp. 434-443.

[15] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

BIOGRAPHIES

Dong Tang, Ph.D.
SoHaR Incorporated
8421 Wilshire Blvd., Suite 201
Beverly Hills, CA 90211-3204
Phone: (213) 653-4718 ext. 104
Email: tang@sohar.com

Dong Tang received the PhD degree in computer science from University of Illinois at Urbana-Champaign (1992). He has been a Senior Research Engineer at SoHaR since 1993. He was a Visiting Research Assistant Professor at University of Illinois from 1992 to 1993 and a Research Scientist at the Institute of Computing Technology, Academia Sinica from 1983 to 1986. His research interests include dependability measurement, modeling, and evaluation for computer hardware

and software systems, software reliability engineering, and evaluation tool development. He has published over 20 technical papers in refereed journals, conference proceedings and books. He has served on several international conference program committees. He was Program Chair of the *International Workshop on Computer-Aided Design, Test, and Evaluation for Dependability*, Beijing, China, July 1996. He is a member of IEEE.

Myron Hecht
SoHaR Incorporated
8421 Wilshire Blvd., Suite 201
Beverly Hills, CA 90211-3204
Phone: (213) 653-4717 ext. 111
Email: myron@sohar.com

Myron Hecht, co-founder and President of SoHaR, has worked in the areas of software and systems reliability. He directs SoHaR's support of the FAA Advanced Automation System which is to replace the current air traffic control systems. His activities in basic research and development at SoHaR have resulted in new architectures for real time distributed systems, methodologies for the development and verification of fault tolerant software, and he has also developed designs for highly reliable distributed systems for both process control and C^3I.

Jady Handal, Ph.D.
Federal Aviation Administration
800 Independence Ave. SW
Washington D.C. 20591
Phone: (202) 267-3214
Email: jady_handal@mail.hq.faa.gov

Dr. Jady Handal is the Special Assistant to the Director for Communications, Navigation and Surveillance Systems for the Federal Aviation Administration (FAA). He is responsible for the development of a full context cost benefit analysis system that allows the FAA to make availability centered risk based investment decisions.

Loni Czekalski
Federal Aviation Administration
800 Independence Ave. SW
Washington D.C. 20591
Phone: (202) 267-9467

Loni Czekalski, Director for Communications, Navigation and Surveillance Systems for the FAA, is responsible for research, development, acquisition and deployment of communication, navigation, landing and surveillance systems into the National Airspace System (NAS). Ms. Czekalski manages approximately 260 employees, about 1000 contractors, approximately 100 programs and an annual budget in excess of $300 Million. Her organization is responsible for delivering a total of 10,414 systems to an aging NAS in two years, which accounted for 99% of all new systems delivered by the FAA's Acquisition Organization.