Reliability Estimation through Computer Simulation using ASAP Data

By

Michael Victor Warthon

A project submitted in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING
in
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGUEZ CAMPUS
2014

Approved by:

_____	_____
Manuel Rodriguez, PhD. President, Graduate Committee	Date
_____	_____
Jose Fernando Vega, PhD. Member, Graduate Committee	Date
_____	_____
Pedro Resto, PhD. Member, Graduate Committee	Date
_____	_____
Pedro Rivera, PhD. Chairperson of the Department	Date
_____	_____
Edgar Acuna, PhD. Graduate School Representative	Date
_____	_____

Abstract

This work covers the design and implementation of a Microsoft Excel-based simulation program intended to estimate helicopter reliability. The simulation program is written in Visual Basic for Applications (VBA) and will be used by the US Army Reliability and Maintainability (RAM) Engineering and Assessment group in order to improve the scheduling of planned interventions and thus decrease the occurrence of unexpected failures while in service.

Currently, the Aviation and Missile Research, Development, and Engineering Center (AMRDEC) uses a block diagram model to estimate helicopter reliability, which assumes that planned and unplanned interventions as well as repair durations can be modeled using particular (e.g. Weibull, log-normal, etc) probability density functions (pdf's). Data extracted from the Aviation System Assessment Program (ASAP), the maintenance log in which all helicopter interventions are documented, showed that such pdf assumptions are not appropriate. Thus, empirical distributions were used in order to allow the simulation model to mimic better actual reliability performance.

The structure of the VBA simulation program is built around the events that happen through the life of the helicopter, such as failures while in use, repair and maintenance interventions, idle periods, and others. Input data is extracted from ASAP into an Excel worksheet from which the empirical distributions are defined. The results of the simulation run are then delivered to the same Excel worksheet for additional analysis and graph preparation.

Resumen

Este trabajo cubre el diseño e implantación de un programa de simulación construido en la plataforma de Microsoft Excel y será de utilidad para estimar la confiabilidad de helicópteros. El programa de simulación está escrito en lenguaje Básico Visual para Aplicaciones (VBA) y será usado por el Grupo de Ingeniería y Evaluación de Confiabilidad y Mantenimiento (RAM) de las Fuerzas Armadas de EEUU para mejorar la calendarización de las intervenciones planificadas, lo cual reduce la ocurrencia de fallas inesperadas mientras el helicóptero ofrece servicio.

Al presente, el Centro de Investigación, Desarrollo e Ingeniería para Aviación y Misiles (AMRDEC) usa modelos de diagramas de bloque para estimar confiabilidad en helicópteros, lo cual presume que las intervenciones planificadas y no planificadas, al igual que las reparaciones se pueden modelar usando funciones de densidad específicas (e.g. Weibull, log-normal, etc). Datos extraídos del Programa de Evaluación del Sistema de Aviación (ASAP), la bitácora de mantenimiento en la cual se documentan todas las intervenciones de mantenimiento de cada helicóptero, demostró que esas presunciones en los modelos probabilísticos no son apropiadas. Por tanto, se enfatizó el uso de distribuciones empíricas para facilitar que el modelo de simulación imite mejor la ejecutoria en confiabilidad del helicóptero.

La estructura del programa de simulación en VBA está construida alrededor de los eventos que ocurren a través de la vida del helicóptero, tales como las fallas en momentos de uso, las actividades de reparación y mantenimiento, los periodos de ocio, y otras. Los datos de entrada son extraídos de ASAP hacia una plantilla de Excel, donde se definen las distribuciones empíricas. Los resultados de la corrida de simulación son entonces llevados a la misma plantilla de Excel para análisis adicional y preparación de gráficas.

Table of Contents

List of Figures

Figures **Page**

List of Tables

Tables **Page**

1 Introduction

The Aviation and Missile Command, located at Redstone Arsenal in Alabama has responsibility over a large number of helicopters. The Aviation and Missile Research, Development, and Engineering Center (AMRDEC) has been working with the collection and processing of helicopter maintenance data. The Aviation System Assessment Program (ASAP), a large SQL database, serves as repository for the maintenance activity data.

The current method used for helicopter reliability estimation are reliability block diagrams (RBD's), which display the parts that work together and their dependence upon one another. RBD's requires the selection of known probability models which should describe the time to failure of each system component. Given the probability models by component, the RBD approach calculates a reliability estimate dependent on the serial or parallel linkage assessed between components.

The use of computer simulation allows for the use of historical data collected from helicopter unplanned and planned stoppages and their repair durations which are collected in the ASAP database throughout the life of the helicopter. The fact that the actual helicopter behavior, captured in ASAP, is being used as input to the simulation model allows for more accurate reliability estimates. The VBA-based model, allows for results to be presented in the universally used worksheet environment provided by Microsoft Excel.

1.1 Justification

The current method of using reliability block diagrams to simulate part failure in helicopters is a sound method but it does have its shortcomings; one of them being that RBD's require the use of a minimal set of probability models, which is used to describe by component the behavior of unplanned (i.e. failures) and planned (i.e. maintenance activities) stoppages. However, a given component, used in different locations of the helicopter, might be subjected to different stresses from the environment. Therefore, using the same distribution for the different applications to the same component does not reflect reality accurately, for which the reliability estimates provided with the RBD modeling approach might be off from the actual helicopter performance.

This simulation program incorporates historical helicopter data stored in ASAP. This includes unplanned and planned stoppages, as well as repair or maintenance durations. The possible use of probability models is limited to components that in a specific application comply with the proposed model. When the probability model is not justified, an empirical distribution will be created. This will allow for a helicopter to be accurately simulated with results that show the frequency of part failures, helicopter uptime, helicopter downtime, and the helicopter maintenance time.

Moreover, commercial simulation languages, such as Rockwell Software's ARENA, require significant license fees. By constructing a simulation program using VBA in Microsoft Excel, there is no cost concern with respect to new software or purchasing licenses.

1.2 Literature Review

Simulation can be used for a variety of situations. One project performed by Chao Yang and Zhengfu Zhu analyzed the reliability of an ad hoc network [1]. This ad hoc network is a mobile network that does not have a fixed infrastructure. Also, the nodes in the network can form the network topology randomly. A mathematical model was used in order to create the simulation and a minimal availability test was done to observe how often the network was available. This test had two phases. Phase 1 consisted of making four consecutive call set-ups across the network and Phase 2 required maintaining the call for five minutes. There were five criteria to determine whether the test succeeded or failed, which were:

- The test fails in Phase I if all 4 call set-up attempts result in either call set-up error or call set-up failure.
- The test fails in Phase II if the total reset events plus reset stimuli is five or greater.
- The test fails in Phase II if the throughput is less than 80 bit/s.
- The test fails in Phase II if the residual error ratio is greater than 10^{-3}.
- The test fails in Phase II if the call and subsequent reestablishments of that call are cleared two or more times due to premature disconnects and/or premature disconnect stimuli.

Estimating the reliability of such a network using mathematical modeling would require making a lot of assumptions. However, this team was able to use the simulation software OMNEST to model the nodes in the network, their mobility, their failures, and the network's overall availability in order to simulate the reliability of the network.

Daniel Sasso and William E. Biles, performed a simulation through the use of object oriented programming [2]. This approach was taken for a data driven model where the data was taken from the Geographic Information System (GIS). Geographic data was captured for the Panama Canal and stored in a database where it would be queued in order to retrieve the data needed for the simulation. The goal was to simulate the arrival of ships to the Panama Canal, the channels they used to move through the system, the locks they encountered in the channel, and their final destination after they exit the system. The simulation software used was ARENA. Both GIS and ARENA had VBA integrated into them and was used to create the simulation models and events. The distributions used in the simulation were obtained by using the Input Analyzer that comes with ARENA. Various code modules were developed in VBA to handle different aspects of the simulation, such as the initialization of the model. The team was able to simulate the behavior of the system and obtain results that could then be analyzed.

This project shares some similar aspects to the author's project as both simulation models are data driven with data being taken from a database. The database used by the helicopter simulation program is connected to the ASAP website and uses data extracted from this website. Furthermore, VBA driven events were used with ARENA in order to simulate the system. In the case of the simulation program, VBA events are still used but without the visual aid of the simulation's progress through time provided by ARENA. Code modules were also developed to handle different aspects of the simulation, such as data input, displaying extracted data, and printing the results. The project performed by Sasso and Biles shows that a simulation using VBA is possible. The major difference between our approaches is that they used VBA in conjunction with ARENA while the helicopter simulation program will be completely in Microsoft Excel.

There have been cases where databases have been used with maintenance data in order to regulate or predict maintenance activity. One group has pushed the use of a database further through the use of a Condition Based Maintenance (CBM) database [3]. The problem found with this database is that there would be some data that would be missing or inaccurate. This could be the result of people entering the data incorrectly or poor calibration of the measuring device. They corrected this through the use of different approaches to calculate or replace the missing data.

Another case involved minimizing as much as possible the amount of maintenance performed on aircraft in order to get planes in the air and ensure that they land safely for the Aircraft Launch and Recovery Equipment (ALRE) [4]. Since high operational availability is crucial for ALRE operations, data mining was used from several different systems in order to obtain supply data, maintenance action forms (MAFs), performance, and availability. They used two approaches in order to find the parts that failed together or relatively close together over time. The analysis from their results allowed them to not only indicate what types of maintenance should be done but also with what frequency and how to do maintenance. This will allow them to make changes to technical manuals in order to better identify problem components or implementing more or less preventative maintenance.

Their effort was focused on extracting data from a database and manipulating that data in order to find the relationships between part failures, operational availability, and preventive maintenance. When compared with the helicopter simulation, this ALRE initiative pursues the reduction of planned maintenance activities while the helicopter reliability data analysis shows intense planned maintenance interventions to prevent the occurrence of unplanned maintenance (i.e. failure incidents) while helicopters are in use.

1.3 Summary of Following Chapters

Chapter 1 introduced the project, its importance, and made comparisons with similar projects. Chapter 2 focuses on the methods and processes used in developing the new simulation model. Chapter 3 details the work involved in creating the simulation logic, the input data required, and the results obtained from running various simulations. Chapter 4 talks about the conclusions

drawn from the results shown in Chapter 3 and the work that can be done to further improve the program.

2 Background

2.1 VBA

Visual Basic for Applications (VBA) is an implementation of Visual Basic, which is an event-driven programming language. It is used to modify or control aspects of the host application. This includes manipulating the features of the user interface (like menus and toolbars), working with custom user forms, or using dialog boxes.

The data extracted from the ASAP database is already saved as an excel worksheet and VBA can perform the calculations and procedures needed to perform a simulation all from one workbook. Therefore, VBA was a prime candidate as the programming language to use in building the simulation program.

2.2 Data Analysis

The analysis process can be broken down into several steps as shown by Figure 1 below. As seen in the figure, an excel file is obtained from the ASAP database. This excel file will then be used to extract the time to failures, time to repairs, time to maintenances, and No Test events for each part. It should be noted that No Test does not fall into any of the reliability measures but is still part of the helicopter's behavior and is therefore examined separately. It is only when these values are known that the simulation can be started.

The data that was extracted from ASAP consists of the part maintenance information for various systems and helicopters for Blackhawk. The file that we have consists of 43 columns and over 5000 rows of data. The table below lists all of the column headings along with their description.

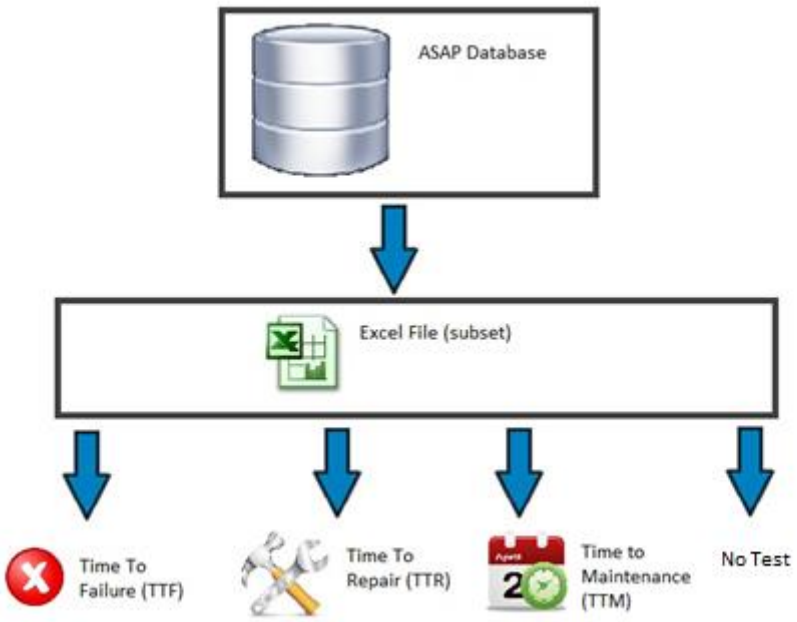

Figure 1. Analysis Procedure

Table 1. Column Definitions

Column Heading	Description
KEY13	Combination of aircraft model, serial number, fault date, and fault number
MODEL	Aircraft Model
SERNO	Aircraft Serial Number
FDATE	Date the fault occurred.
FTIME	Time the fault occurred.
SYS	System
FNO	Fault Number
EI_ID	Refers to the aircraft model's id.
<UIC>	
UNIT	The unit that the helicopter belongs to.
STAT	Aircraft Status
ACHRS	Number of aircraft flight hours when fault occurred.
<RACHRS>	
CLOSED	
WDISC	When the fault was discovered.
HREC	How the fault was recognized.
MEF	Malfunction Effect
TYPE	Failure Type
ACTCD	Action Code
CDATE	Date the correction occurred.
CTIME	Time the correction occurred.
CACHRS	Number of aircraft flight hours when correction occurred.
CWUC	Suggested WUC by maintainer.
<RFG>	WUC given by the scorer.

<CNAME>	Component Name
FAULT	Fault
ACTION	Action
TMEN	Total Maintainers
TMMH	Total Maintenance Man Hours
TIMH	TI Man Hours
DISC_PID	
<SCD1> - <SCD9>	Score Definition 1 through 9
<SCORED>	Indicates whether the data has been scored or not.
IMPORT_DATE	The data when the data was imported into ASAP.
<PRIMARY_EVENT>	Primary Event for a continuation of maintenance

Not all of the columns were used when analyzing the data. Columns identifying the helicopter such as IDs were not included in the analysis. Since the analysis relies on the dates when the failure occurred, when the maintenance occurred, how long the repairs took, and the type of maintenance action taken, any columns providing information about the helicopter or specific failure or maintenance information is not needed.

2.2.1 Helicopters

Based on the helicopter serial number field we were able to segregate the data by helicopter. The figure below shows some helicopters (rows) and the variety of work unit codes (WUC's) typical of helicopters (columns). WUC's are multi-level identifiers (from system all the way to specific parts) for components that are intervened within the helicopter, as defined in MIL-STD-780.

Table 2. Number of WUCs for each helicopter

Count of STAT Column Labels																										
Row Labels	'02	'05	'04	'19	'11	'06	'07	'09	'03	'08	'18	'15	'00	'12	'76	'10	'17	'52	'16	'13	'14	'99	'36	'83	'80	Grand Total
'8624514	2350	3769	458	304	335	356	326	140	159	188	376	122	28	79	4	54	38	39	16	10	2	7				9160
'8624575	2125	3582	543	236	500	352	260	143	197	144		84	42	55	4	46	26	41	19	9	1	25				8434
'8524472	2051	3581	406	242	576	396	261	130	261	145		90	25	91	3	58	27	44	14	8	4	6				8419
'8624510	2335	3167	302	183	480	296	234	117	165	181	104	93	15	65	2	43	28	42	16	9	1	5				7883
'8524396	878	226	179	178	56	73	42	115	33	59	30	40	21	17	40	15	26	20	12	4	9					2073
'8524447	657	280	175	210	57	162	46	73	27	56	3	7	22	30	44	15	10	1	4	17						1896
'7923296	920	113	186	118	68	130	16	42	34	29	23	27	14	16	28	21	11	8	8	9						1821
'8023445	656	225	124	173	40	50	7	91	34	36	1	28	14	21	10	11	77	5	11	1	4	2				1621
'8724646	530	280	123	164	22	76	29	34	39	13	9	15	21	21	18	16	5	6	3	3						1427
'8423942	388	111	68	106	29	66	9	24	22	19	14	8	15	24	23	17	7	7	20	4	5					986

From this table, it became easy to select the helicopters that would be used for the simulation. The rows with the highlighted cells were the ones that were selected as they have the most data to work with. In summary, the simulation was performed using helicopter with serial number 8624514, since it had the largest amount of data.

2.2.2 Part Parameters

It appeared that the easiest way to obtain a part's time to failure is to look at the date and time that the failure was recorded. Figure 2 below demonstrates the first problem with that approach.

	A	B	C	D	E	G	H	I	J	K	L	M	N	O	P	Q	R	
1														2				
2	KEY13	MODEL	SERNO	FDATE	FDateTime	FTIME		SYS	FNO	EI_ID	<UIC>	UNIT	STAT	ACHRS	<RACHRS>	CLOSED	WDISC	H
3	UH-60L-0126883-20081028-A-0081	UH-60L	'0126883	28-Oct-08	10/28/2008 12:59	12:59		A	81	UH0307	WYKKC0	HIARNG(H	-	1133		1	H	G
4	UH-60L-0126883-20081028-A-0082	UH-60L	'0126883	28-Oct-08	10/28/2008 13:00	13:00		A	83	UH0307	WYKKC0	HIARNG(H	-	1133		1	H	G
5	UH-60L-0126883-20081028-A-0083	UH-60L	'0126883	28-Oct-08	10/28/2008 13:01	13:01		A	84	UH0307	WYKKC0	HIARNG(H	-	1133		1	H	G
6	UH-60L-0126883-20081028-A-0084	UH-60L	'0126883	28-Oct-08	10/28/2008 13:02	13:02		A	82	UH0307	WYKKC0	HIARNG(H	-	1133		1	H	G
7	UH-60L-0126883-20081110-A-0001	UH-60L	'0126883	10-Nov-08	11/10/2008 6:20	6:20		A	1	UH0307	WYKKC0	HIARNG(H	-	1136.2		1	K	G
8	UH-60L-0126883-20081113-A-0001	UH-60L	'0126883	13-Nov-08	11/13/2008 7:00	7:00		A	1	UH0307	WYKKC0	HIARNG(H	-	1136.2		1	O	O
9	UH-60L-0126883-20090224-A-0001	UH-60L	'0126883	24-Feb-09	2/24/2009 7:44	7:44		A	1	UH0307	WYKKC0	HIARNG(H	-	1157.5		1	O	G
10	UH-60L-0126883-20090226-A-0002	UH-60L	'0126883	26-Feb-09	2/26/2009 15:50	15:50		A	2	UH0307	WYKKC0	HIARNG(H	-	1157.5		1	O	G
11	UH-60L-0126883-20090407-A-0005	UH-60L	'0126883	7-Apr-09	4/7/2009 8:26	8:26		A	5	UH0307	WYKKC0	HIARNG(H	-	1168.7		1	O	G
12	UH-60L-0126883-20090516-A-0002	UH-60L	'0126883	16-May-09	5/16/2009 8:57	8:57		A	2	UH0307	WYKKC0	HIARNG(H	+	1177		1	S	G
13	UH-60L-0126883-20090625-A-0001	UH-60L	'0126883	25-Jun-09	6/25/2009 9:46	9:46		A	3	UH0307	WYKKC0	HIARNG(H	/	1213.8		1	K	J
14	UH-60L-0126883-20090625-A-0003	UH-60L	'0126883	25-Jun-09	6/25/2009 12:00	12:00		A	1	UH0307	WYKKC0	HIARNG(H	-	1213.8		1	O	G
15	UH-60L-0126883-20090711-A-0005	UH-60L	'0126883	11-Jul-09	7/11/2009 9:49	9:49		A	5	UH0307	WYKKC0	HIARNG(H	+	1219.3		1	O	O
16	UH-60L-0126883-20090716-A-0003	UH-60L	'0126883	16-Jul-09	7/16/2009 11:20	11:20		A	3	UH0307	WYKKC0	HIARNG(H	-	1221.1		1	O	G
17	UH-60L-0126883-20090716-A-0004	UH-60L	'0126883	16-Jul-09	7/16/2009 11:20	11:20		A	4	UH0307	WYHMC0	C CO., 1-1(	-	1221.1		0	O	G
18	UH-60L-0126883-20090716-A-0005	UH-60L	'0126883	16-Jul-09	7/16/2009 11:20	11:20		A	5	UH0307	WYHMC0	C CO., 1-1(	-	1221.1		0	O	G
19	UH-60L-0126883-20090811-A-0005	UH-60L	'0126883	11-Aug-09	8/11/2009 9:20	9:20		A	5	UH0307	WYHMC0	C CO., 1-1(	X	1229.3		1	O	G
20	UH-60L-0126883-20090826-A-0001	UH-60L	'0126883	26-Aug-09	8/26/2009 12:04	12:04		A	1	UH0307	WYHMC0	C CO., 1-1(	-	1238.6		1	O	O
21	UH-60L-0126883-20090827-A-0002	UH-60L	'0126883	27-Aug-09	8/27/2009 9:47	9:47		A	2	UH0307	WYHMC0	C CO., 1-1(	-	1242.4		1	O	G
22	UH-60L-0126883-20090915-A-0001	UH-60L	'0126883	15-Sep-09	9/15/2009 8:37	8:37		A	1	UH0307	WYHMC0	C CO., 1-1(	-	1248.3		1	O	O
23	UH-60L-0126883-20091001-A-0006	UH-60L	'0126883	1-Oct-09	10/1/2009 10:49	10:49		A	6	UH0307	WYHMC0	C CO., 1-1(	-	1253		1	O	O

Figure 2. Helicopter Data with a focus on the date and time for part 00.

The columns highlighted in red in the Figure contain the date and time that a failure occurred for a part with a WUC of 00. Notice all of the failures that occur within a minute of each other. This obviously does not make any sense because this indicates that this part cannot last even two minutes without breaking down again. After looking at the data again, it was found that not each failure is the part breaking down but might possibly be a maintenance action. The score set for the row of data signals whether the part failed or if it was undergoing maintenance. The table below shows the various score identifiers and their meanings.

Table 3. Score IDs

scd1	P-score	FAILURES vs MAINTENANCE vs OTHER
A	MA	mission abort
B	MAF	mission affecting failure
P	CCMA w UM	crew correctable maintenance action with unscheduled maintenance
E	EMA w SM	essential maintenance action with scheduled maintenance
S	SMA	scheduled maintenance action
Q	CCMA wo UM	crew correctable maintenance action without unscheduled maintenance
C	COM	continuation of maintenance
D	DM	dependent malfunction
U	EMA wo SM	essential maintenance action without scheduled maintenance
N	NO TEST	no test

Table 3 shows the different possibilities that each row could refer to. There are three options: the part has failed, the part has undergone maintenance, or no test. The scores highlighted in yellow indicate that a failure has occurred. These were classified as failures because the score indicates that the mission failed or was aborted as a result of what happened. The score "CCMA w UM" was also included as a failure due to the maintenance action being unscheduled. The rest of the scores were labeled as maintenance actions because each action was a scheduled maintenance action or was a maintenance action that did not result in unscheduled maintenance.

Using this information, the data was sorted by the date and time of an event along with the score that the event received. This was done for the data for only one part in order to try to obtain it's time to failure empirical distribution. A probability model could have been used but in many instanced results from goodness-of-fit tests demonstrated that such models were not adequate (i.e. the test statistic failed). Thus, it was felt that an empirical distribution would provide a more realistic view of the time to failure and the maintenance actions.

2.2.3 Empirical Distributions

Empirical distributions were to be developed for a helicopter's time to failure (TTF), time to repair (TTR), and time to maintenance (TTM). For the TTR of a helicopter, the empirical distribution was obtained by looking at the man-hours column. The values in the column were converted to minutes and sorted from smallest to largest. Another column was added to keep track of the row number of the values with zero being set at the first value. Comparing the last position number of a number with the last row provided the probability for that number. The process for obtaining the empirical distribution is shown below.

	A	B
1	Position	ServT(min)
2	1	6
3	2	6
4	3	6
5	4	6
6	5	6
7	6	6
8	7	6
9	8	6
10	9	6
11	10	6
12	11	6
13	12	12
14	13	12
15	14	12
16	15	12
17	16	12
18	17	12
19	18	12
20	19	12
21	20	15
22	21	18
23	22	18
24	23	18
25	24	18

D	E	F	G
Position	ServT(min)		Empirical Distribution
11	6	11/66 =	0.166666667
19	12	19/66 =	0.287878788
20	15	20/66 =	0.303030303
28	18	28/66 =	0.424242424
30	24	30/66 =	0.454545455
36	30	36/66 =	0.545454545
40	36	40/66 =	0.606060606
41	42	41/66 =	0.621212121
54	60	54/66 =	0.818181818
55	84	55/66 =	0.833333333
58	90	58/66 =	0.878787879
59	108	59/66 =	0.893939394
61	120	61/66 =	0.924242424
62	162	62/66 =	0.939393939
63	165	63/66 =	0.954545455
64	180	64/66 =	0.96969697
65	198	65/66 =	0.984848485
66	240	66/66 =	1

Figure 3. TTR Empirical Distribution example

This probability reflects how often that value occurred and the chance that the helicopter will be repaired at that time. Figure 4 below shows the resulting graph for the empirical distribution provided in Table 4. The final time value (3240) might be seen as an outlier (value too far away from the rest), but such an intervention really happened.

The man hours column can refer to a repair event or a maintenance event occurring from a failure and therefore the empirical distribution developed from this data can be seen as related to both the time to repair and time to maintenance. Moreover, the amount of events in the data for maintenance and failures leading to repairs are one sided. Figure 4 below shows the amount of data available for each event.

Table 4. Time to Repair Empirical Distribution

ServT(min)	Cprob 2
3	0.00022
30	0.65852
36	0.702114
42	0.7107
48	0.721048
49.5	0.721158
50	0.721268
51	0.721708
60	0.814399
114	0.89531
174	0.933289
204	0.970828
3240	1

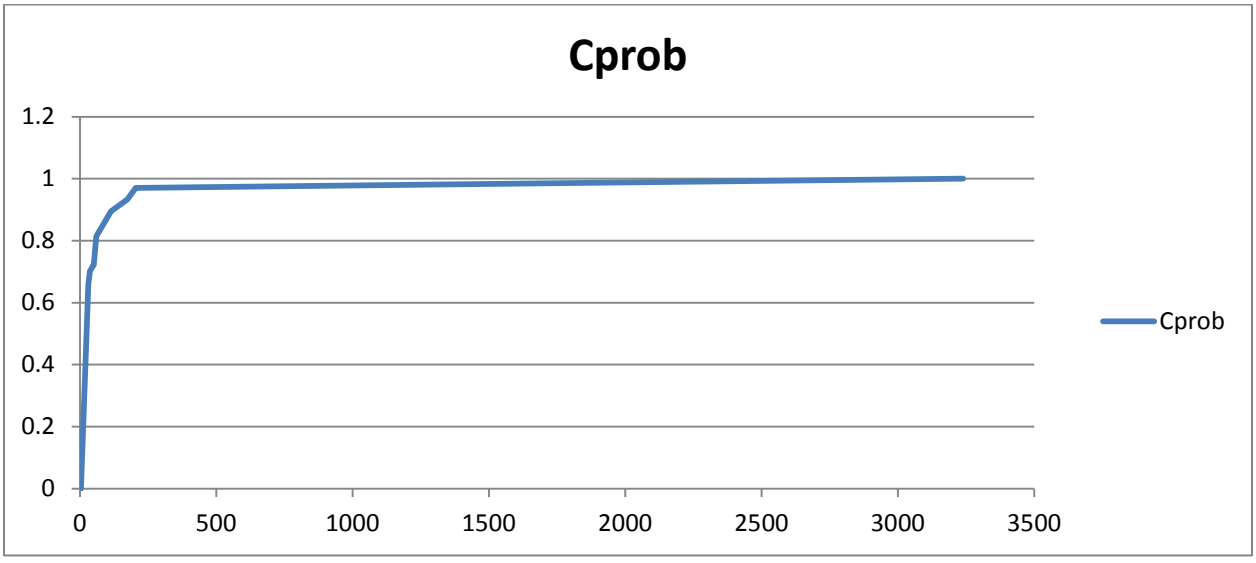

Figure 4. Time to Repair Empirical Distribution

Row Labels	Count of P-score	
01ma	51	0.005614
02maf	9	0.000991
03ccma w um	6	0.000661
04ema w sm	550	0.060546
05sma	4960	0.546015
06ccma wo um	50	0.005504
07com	1008	0.110964
08dm	1	0.00011
09ema wo sm	894	0.098415
10no test	1555	0.17118
Grand Total	**9084**	1

total % of failure events:	0.007266
total % of maint events:	0.992734

Figure 5. Event percentages for helicopter

The values in Figure 4 above show that the number of failure events is insignificantly small when compared to the number of maintenance events. This reflects the emphasis on prevention or planned maintenance to avoid risky incidents caused by malfunctions while the helicopter is in service.

To further emphasize the need for an empirical distribution, ExpertFit (a statistical analysis program) was used to perform goodness-of-fit tests on helicopter data in order to select the most adequate probability model. The Figures below show an overview of the data used to perform the tests for time to failure (TTF) data. Figure 6 below shows a summary of the data entered into ExpertFit, which automatically calculates the min, max, and mean value of the data. A summary of the data for TTM, TTR, and No Test as shown in Table 5 below.

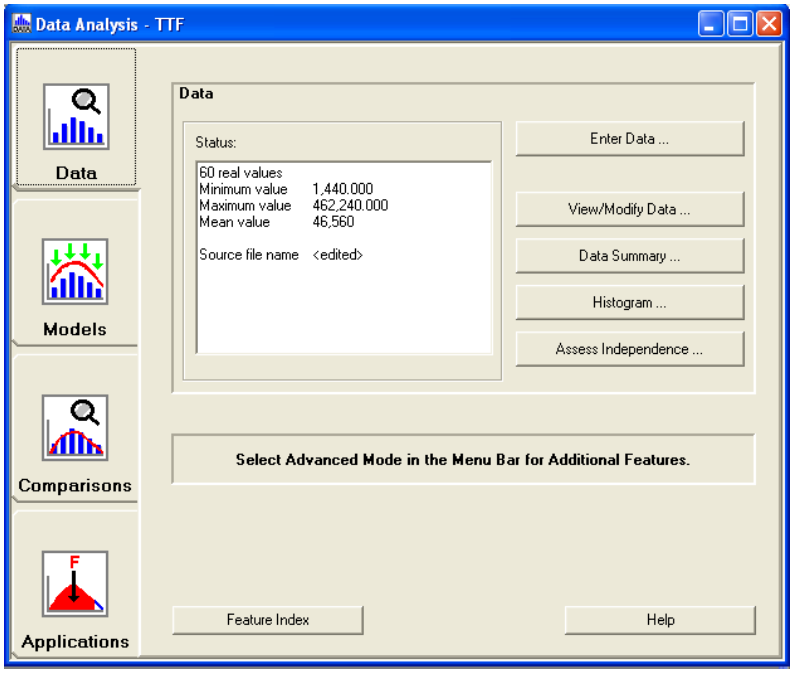

Figure 6. Data Analysis Overview for TTF

Table 5. Data Overview

	Number of Values	Min Value	Max Value	Mean Value
TTF	60	1,440	462,240	46,580
TTM	7,447	3,000	3,240	49.58
TTR	66	6	240	47.91
No Test	1,556	6	1,440	33.98

ExpertFit took the data and calculated a list of probability models that could be used with the data. The models are each given a value indicating whether they should be rejected or not. Normally, a Weibull distribution or Lognormal probability distribution are used to model data. Therefore, the following Figures show the results of these two probability distributions for the TTF, TTM, TTR, and No Test parameters.

From Figures 7 through 14 we can see that for some data sets it is just not possible to use a probability distribution. For No Test and TTM, the two probability distributions were both rejected. Therefore, it is preferable to use an empirical distribution for these two parameters. For both TTF and TTR, a probability distribution can be used to model the data. However, when

looking at both the Lognormal and Weibull distributions, it is seen that these are not the first preferred choices for the parameters. For TTF, Lognormal is the fourth model that can be used while the Weibull distribution is the third model that can be used. For TTR, Lognormal is the eighth model that can be used while the Weibull distribution is the third model that can be used.

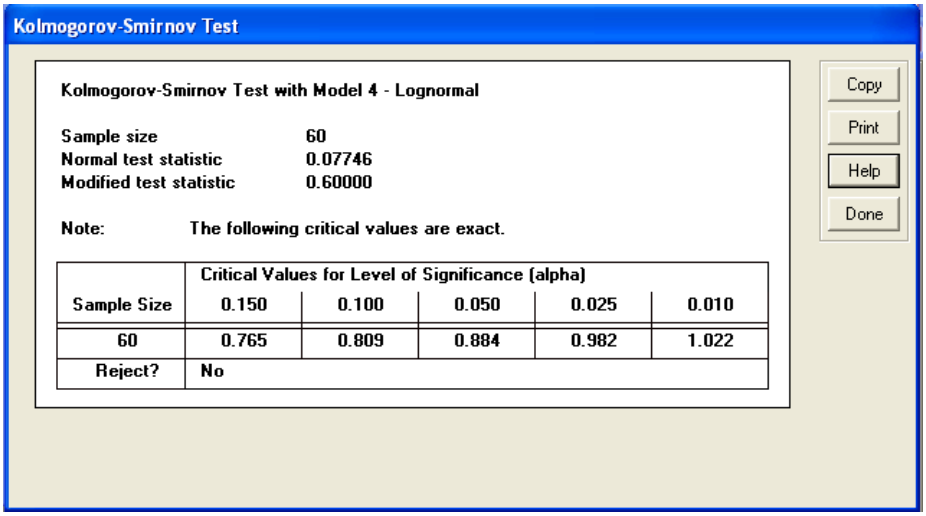

Figure 7. TTF Goodness of Fit Test Results (Lognormal)

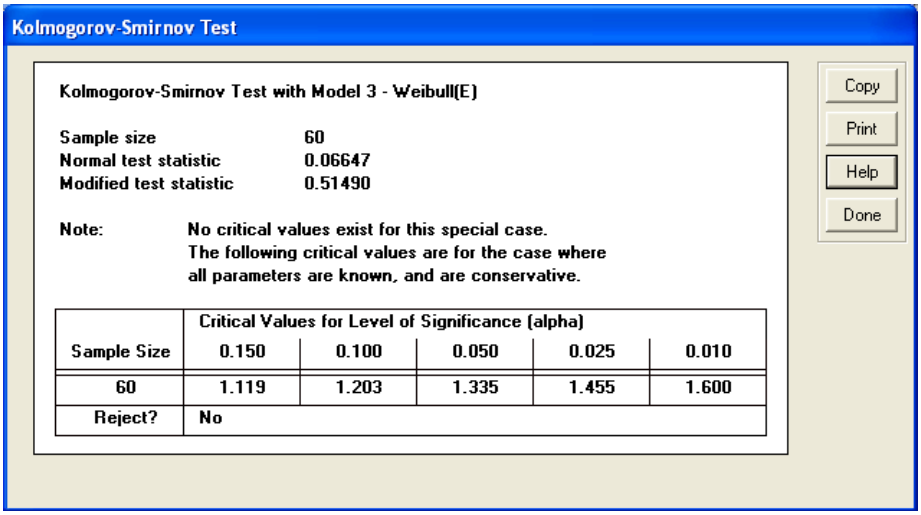

Figure 8. TTF Goodness of Fit Test Results (Weibull)

Other probability distributions such as Pearson are preferred to be used before Weibull or Lognormal. The positions of both Weibull and Lognormal reinforce the notion that an empirical distribution should be used over probability distributions.

2.2.4 Event Tables

As the analysis of the helicopter proceeded, an interesting detail was detected when observing the behavior of the events of a helicopter. The time between events was calculated again but without regard as to whether the event was a failure or maintenance action.

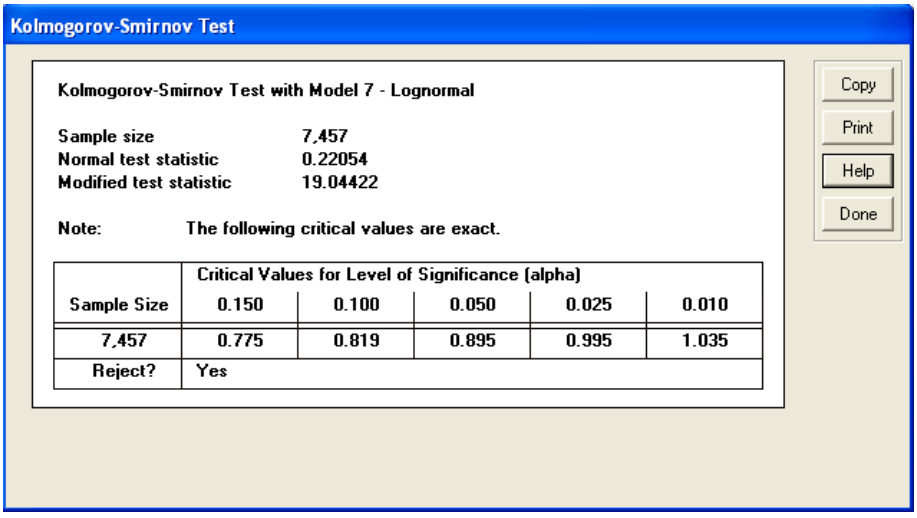

Figure 9. TTM Goodness of Fit Test Results (Lognormal)

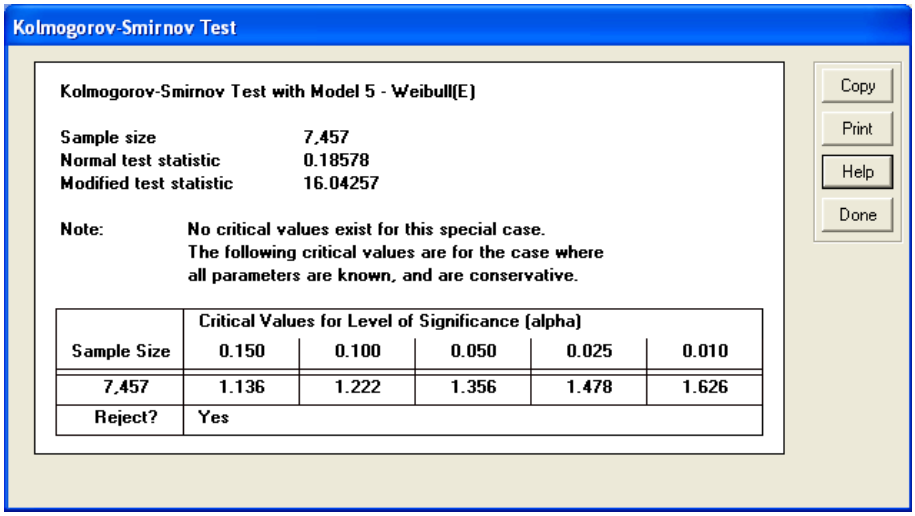

Figure 10.TTM Goodness of Fit Test Results (Weibull)

Figure 15 below shows a pivot table displaying the results of the time between occurrences; it was found that each event occurs in increments of days or 1440 minutes. Usually events occur immediately after the previous event. It was an unusual development as this says that all events, failure and maintenance, do not occur randomly or according to a probability distribution but in increments of 24 hours.

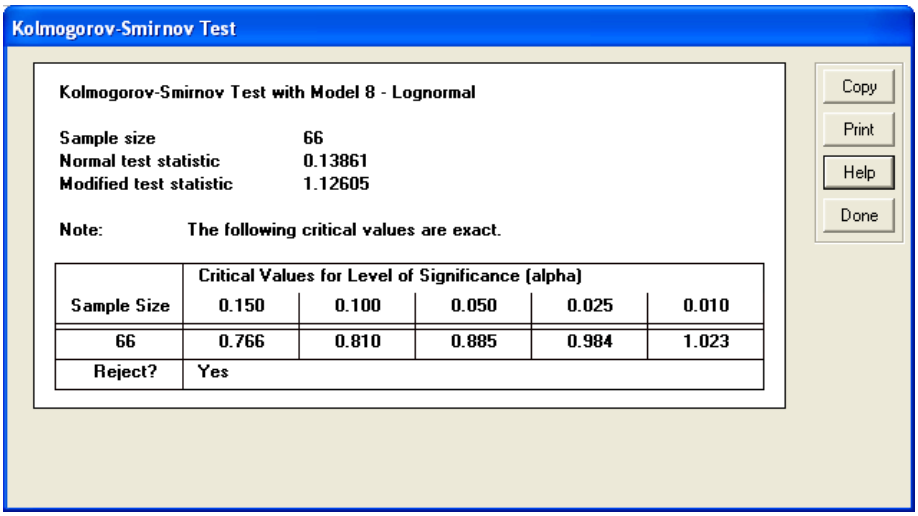

Figure 11. TTR Goodness of Fit Test Results (Lognormal)

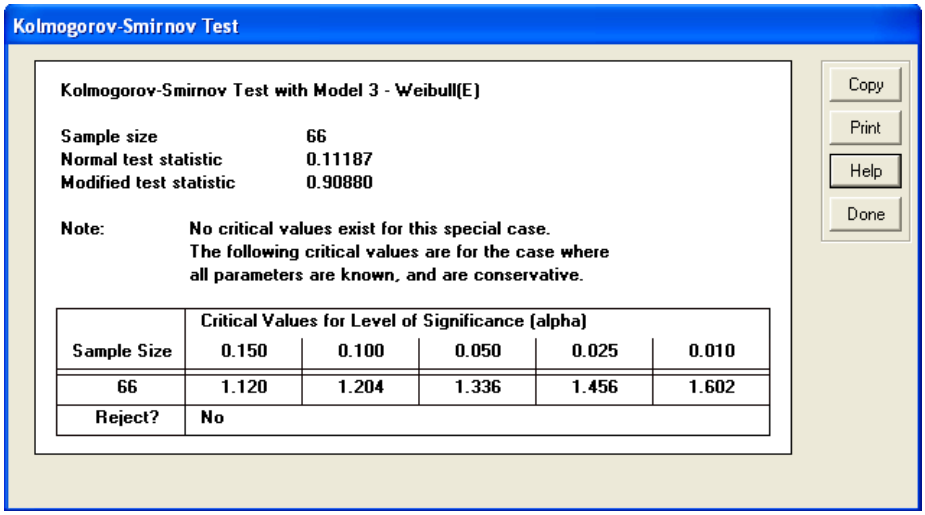

Figure 12. TTR Goodness of Fit Test Results (Weibull)

To accurately portray the behavior of the helicopter and maintenance process in the simulation, it was decided that the behavior shown by the time between events should be taken into account. The data reveals a lot about how the current maintenance process works as most events are handled immediately and, as mentioned earlier, most of the events are maintenance events. Therefore, the helicopters are shown to be maintained often and it could be almost daily.

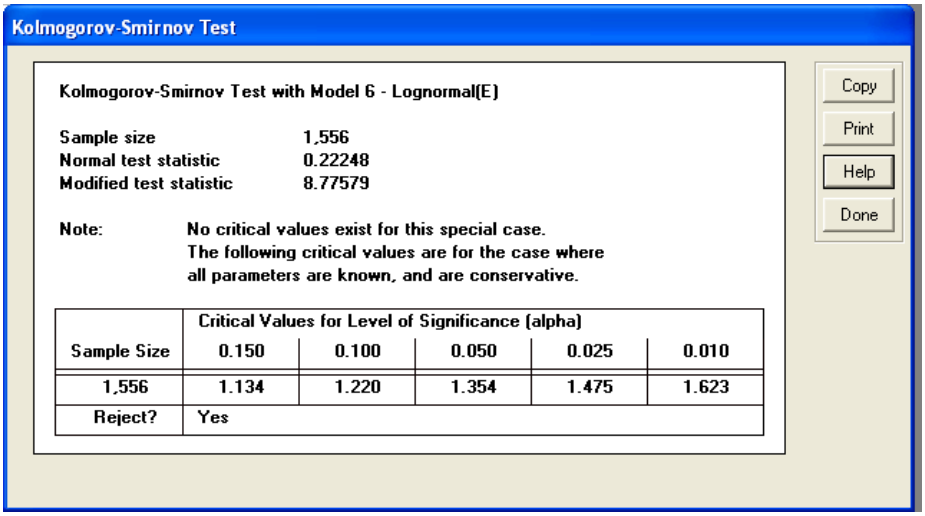

Figure 13. No Test Goodness of Fit Test Results (Lognormal)

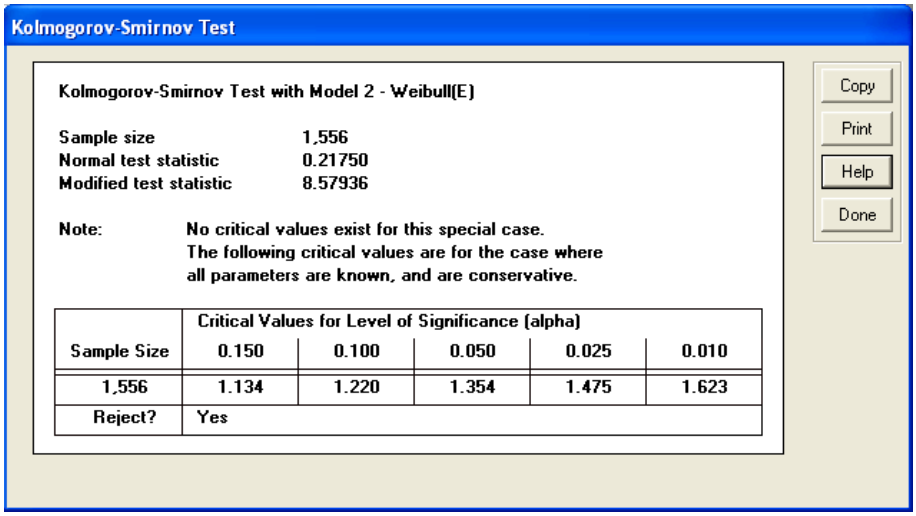

Figure 14. No Test Goodness of Fit Test Results (Weibull)

helicop	A																
	0	1	2	3	4	5	6	7	12	13	14	16	17	18	31	45	
Count of tboccur	**Column Labels**																
Row Labels	**0**	**1440**	**2880**	**4320**	**5760**	**7200**	**8640**	**10080**	**17280**	**18720**	**20160**	**23040**	**24480**	**25920**	**44640**	**64800**	**Grand Total**
'00	18	5		2	1									1			27
'02	1781	438	55	26	13	7	4	1				2			1	1	2329
'03	144	12	1	1	1												159
'04	395	46	9	1	3		1										455
'05	3611	91	25	8	3		2										3740
'06	247	72	17	6	4	1		1					1				349
'07	296	26	1	2	1												326
'08	153	26	2	2													183
'09	102	22	6	4	1	1	1		1								138
'10	40	9	1		3						1						54
'11	283	36	8	5	1												333
'12	60	12	2	3					1								78
'13	7	2								1							10
'14	1		1														2
'15	95	20	3	1		1											120
'16	11	4		1													16
'17	21	11	1	2		1			1								37
'18	252	97	13	9	3		1										375
'19	244	41	11	5		1		1									303
'52	34	5															39
'76	3		1														4
'99	3	4															7
Grand Total	**7801**	**979**	**157**	**78**	**34**	**12**	**9**	**3**	**3**	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**9084**

Figure 15. Time of event occurrence for each part

It was not enough to just obtain a distribution to model the time between event occurrences. The event to follow is dependent on the current one; and this is the best way to describe how the maintenance sequence behaves. Therefore, two transition tables are needed: one table shows the possibility of transitioning from one event to another (Figure 16), and another table shows the probabilities on how far into the future (in days) the next maintenance activity will have to wait (Figure 17).

		1	2	3	4	5	6
	(10,4)	**04ema w sm**	**05sma**	**06ccma wo um**	**07com**	**08ema wo sm**	**09no test**
04ema w sm	**1**	0.24214	0.44732	0.45841	0.71349	0.86322	1.00000
05sma	**2**	0.03068	0.82080	0.82426	0.85738	0.92056	1.00000
06ccma wo um	**3**	0.08333	0.52083	0.66667	0.72917	0.91667	1.00000
07com	**4**	0.11411	0.32533	0.32933	0.80280	0.90090	1.00000
08ema wo sm	**5**	0.08390	0.38776	0.39909	0.53855	0.86848	1.00000
09no test	**6**	0.04266	0.32644	0.33032	0.38720	0.44796	1.00000

Figure 16. From-To Probability Table

There are a few events (01 through 03) missing from Figure 16. The reason for this is that a maintenance event can never go straight into a failure event; maintenance events can only go from one maintenance event to another and so Figure 17 helps in defining when the next maintenance event should happen: right now (zero minutes), one day (1440 minutes), two days (2880 minutes), and up to six days (8640 minutes).

			1	2	3	4	5	6
		Minutes						
Activity	(22,3)	0	1440	2880	4320	5760	7200	8640
01ma	1	1	1	1	1	1	1	1
	2	-1						
	3	-1						
	4	0.75	1	1	1	1	1	1
	5	0.9	1	1	1	1	1	1
	6	-1						
	7	0.733333333	0.933333333	1	1	1	1	1
	8	0.833333333	0.833333333	1	1	1	1	1
	9	0.777777778	1	1	1	1	1	1
02maf	10	-1						
	11	-1						
	12	-1						
	13	-1						
	14	0.666666667	0.666666667	0.666666667	1	1	1	1
	15	-1						
	16	-1						
	17	0	1	1	1	1	1	1
	18	1	1	1	1	1	1	1
03ccma w um	19	-1						
	20	-1						
	21	0	0	1	1	1	1	1
	22	-1						
	23	-1						
	24	-1						
	25	0.75	0.75	1	1	1	1	1
	26	0	0	1	1	1	1	1
	27	-1						

Figure 17. Time between Transitions Table

Figure 17 shows some of the values for events 1 through 3. A negative value is used whenever there are no values found for an event transition and the data should ensure that these transitions never occur. The gap between values represent the probability of being chosen at random. In Figure 14, there were values greater than six days but Figure 17 only goes up to six days. The reason those values were omitted is that most of the probability is accumulated between zero and two days, and there are very few values beyond six.

It is important to mention that the failure, maintenance, and repair realities for the selected helicopter were extracted from the ASP database. However, the only data that was never available was the flight or service realities of helicopters. For this reason it was decided that in-

service helicopters would perform one or two flights per day; if only one flight is conducted it lasts between four and six hours, while in the case of two flights within the day, each trip lasts between two and four hours.

2.3 VBA Simulation

Since the simulation will be done in Microsoft Excel, there are no objects, servers, resources, sinks, sources, or any visual representation of the system, like you would see in a simulation software such as SIMIO. Therefore, the simulation will be event-driven with each act of the helicopter representing an event being called in the program. These events would cover failures, repairs, maintenance, helicopter startup, helicopter shutdown, beginning of a new day, and reliability estimation.

In order to determine when an event occurs or will occur, a variable that holds the current simulation time will be required. This variable, referred to as "clocktime", will be used to hold the current time in the simulation in minutes. For example, a clock time of 420 would be 7am on the first day of the simulation while a clock time of 73,020 would be 5pm on the fiftieth day of the simulation. With this, events such as helicopter maintenance, repairs, and takeoffs can be scheduled as they would in real life.

The general steps involved in the simulation are as follows:

1) Initialize the program;
2) Go into the timing subroutine in order to acquire which event comes next;
3) Go back to the main subroutine and call forth the appropriate event subroutine;
4) The event subroutine runs to completion;
5) Repeat steps 2-4 until the simulation run length is reached.

3 Simulation Program

The simulation program is on the worksheet labeled “hel_8624514”. This worksheet contains all of the information needed to run the simulation, which include the transition tables from figures 16 and 17, empirical distributions, simulation run length, and the reliability estimation frequency. The results of the simulation are also shown on this page. The steps involved in the simulation program are:

1) Initialize the program
    a. Setup all of the arrays, event times, the first maintenance event, and the first day.
2) Go into the timing subroutine in order to check which event is scheduled next.
3) Maintenance Events
    a. If no maintenance event occurs, then schedule how many runs the helicopter will perform that day.
    b. If a maintenance event does occur, then reschedule the helicopter activities for the next day.
        i. Schedule the next maintenance event.
4) Helicopter Runs
    a. Check whether a failure occurs.
        i. If a failure does occur, then schedule a repair event. Otherwise, the helicopter will simply complete its run or runs for the day.
5) When a new day starts repeat steps 2-4.

The main simulation subroutine will start off with the initialize subroutine and then direct which even subroutine will be started. This subroutine acts as the main hub and the program will always keep coming back to it until the simulation run length has been reached.

3.1 Initialization Subroutine

The initialization subroutine starts up the whole program by setting variables to the values that were entered into the spreadsheet. The length of the simulation and the frequency of the reliability estimations are set here as well as all of the arrays used throughout the program. The TTF, TTR, and TTM arrays are all read from the worksheet using constant values. Therefore, all arrays are of fixed lengths and are taken from specific locations in the worksheet. The issue of

fixed length arrays will be addressed in future work in order to create a simulation program that will accept any size of empirical distributions calculated from the ASAP data.

Aside from collecting data and setting the values for all variables and arrays, this subroutine also schedules several events as well. The first maintenance event is scheduled following from the assumption that the helicopter's last event was a no test event. The event times are scheduled for the first maintenance event and for the first day. All other event times are set to infinity in order to ensure that all other events will not fire. The first failure day is also set in the initialization subroutine from the TTF empirical distribution collected earlier in the subroutine. Lastly, all data collection variables used in the reliability estimation subroutine are set to zero.

3.2 Timing Subroutine

The timing subroutine is tasked with finding out which event is scheduled to come next. It loops through all of the events and checks their times in order to find the one with the lowest value. It is for this reason that all event times are set to infinity in the initialization subroutine. Once it has found the event with the lowest time, the clocktime variable is moved forward and set to this event time.

3.3 Beginning of the Day Subroutine

The beginning of the day subroutine handles all of the preparation that is needed for a new day in the simulation. This subroutine handles various areas of the behavior of the helicopter such as the number of runs it will make that day and if the helicopter will encounter a failure that day.

The subroutine starts off by scheduling the next beginning of the day event 1440 minutes later. A counter holding the number of days that have passed in the simulation is incremented and is used in checking whether that day is one where a failure will occur. The status of the helicopter is first checked to see if a maintenance activity is scheduled for that day. If not, the statistics for the last idle period is collected. If there is no maintenance activity and no failure occurring that day, then the number of runs that the helicopter will make is set randomly to either one or two runs and the next idle period is scheduled.

If the day counter falls on a failure day, then the type of failure (MA, MAF, or CCMA w UM) is checked and the helicopter status is reset to reflect the fact that a failure will occur. The event

time for the failure event is set to the current clocktime. The event time for the next idle period is set to infinity and the service flag is set to zero. Also, the scheduled service completion time is reset to infinity. Once everything has been reset, the next failure day is scheduled. If the next failure day will occur right now, then it is rescheduled to happen tomorrow because only one failure can occur each day. This behavior is attributed to the fact that when a failure occurs, other failures that happen at that time will most likely be handled along with the failure that triggered the event. Figure 18 below shows part of the code for the whole beginning of the day subroutine.

```
Sub beg_of_day()
    'schedule the beginning of the next day (1 day = 1440 minutes)
    eventime(0) = clocktime + 1440#
    'increase the day_counter by 1
    day_counter = day_counter + 1
    '
    'determine the number of service trips for the day if helicopter has NO maintenance activity
    If helicop_status = 10 Then
        'collect statistics for last idle period
        idle_accum = idle_accum + 1# * (clocktime - idle_clock)
        idle_counter = idle_counter + 1
    End If
    '
    'check if there is a failure scheduled for this day (1-ma, 2-maf, 3-ccma w um)
    If day_counter = failure_day Then
        'determine the failure type
        u1 = Rnd()
        For i = 1 To 3
            If u1 <= cfprob(i) Then
                'set the event_flag
                event_flag(i) = 1
                'set the failure flag
                failure_flag = i
                'schedule the failure for now
                eventime(i) = clocktime
                Exit For
            End If
        Next i
        'reset the helicopter status
        helicop_status = failure_flag
        'reset the idle period initiation time to infinity
        eventime(10) = duration + 1#
        'reset the service flag to OFF
        event_flag(10) = 0
        'reset the service completion time to infinity
        eventime(11) = duration + 1#
```

Figure 18. Beginning of the Day Subroutine

3.4 Helicopter Service Period Subroutine

This subroutine is split into two areas. The first area handles the case for when the helicopter starts to provide service. A flag variable was created in order to check whether the helicopter is starting a run or completing a run.

In the case for when the helicopter is just starting its service, a service clock is set in order to provide the amount of time the helicopter has been out in service. The status of the helicopter is set accordingly and the end of the service is scheduled. The time that the service run ends is dependent on the number of runs the helicopter is scheduled to perform that day. If there is only one run, then the helicopter will go out for between four and six hours on assignment. If there are

two runs scheduled for that day, then the helicopter will go out for between two and four hours per trip. Lastly, the failures are set up so that all event times are set to infinity and their associated event flags are set to zero. The figure below shows the first half of the helicopter service period subroutine.

```
If event_flag(11) = 1 Then
    'service period starts
    service_clock = clocktime
    'reassure helicop_status
    helicop_status = 10
    'schedule the end of the service period for 1 or 2 trips/day
    u = Rnd()
    If num_trips = 1 And which_trip = 1 Then
        eventime(11) = clocktime + (240# + 120# * u)
    ElseIf num_trips = 2 And which_trip <= 2 Then
        eventime(11) = clocktime + (120# + 120# * u)
    Else
        Debug.Print "helicop service with flag=1 has error in logic"
        Stop
    End If
    'the service flag is ON
    event_flag(11) = -1
    'assure the idle flag is OFF
    event_flag(10) = 0
    '
    'set failure_flag to zero
    failure_flag = 0
    'set the ttf to infinity and the event flag to zero for all failures
    For i = 1 To 3
        eventime(i) = duration + 1#
        event_flag(i) = 0
    Next i
    '
Else
    'service period ends; i.e. event_flag(11) = -1
```

Figure 19. Helicopter Service Period Subroutine - Service Starts

The second half of the subroutine handles the case for when the helicopter finishes its run and is about to become idle again. The event time for the service time event is set to infinity and the flag for the service time event is set to zero. Statistics are collected at this point with one variable holding the accumulated total time that the helicopter has been in service and another variable containing the total amount of times the helicopter has been out in service. Moreover, a counter is incremented that is used to check the amount of runs the helicopter has performed for that day. Since the helicopter has just finished its run, the idle period event is set to the current clocktime. The idle flag is also set to one in order to indicate this transition. Figure 20 below shows the other half of this subroutine.

```
Else
    'service period ends; i.e. event_flag(11) = -1
    'set the occurrence of the next service period to infinity
    eventime(11) = duration + 1#
    'service flag is turned OFF
    event_flag(11) = 0
    'collect statistics
    service_accum = service_accum + (clocktime - service_clock)
    service_counter = service_counter + 1
    '
    'increase the which_trip counter
    which_trip = which_trip + 1
    '
    'schedule the beginning of the next idle period
    eventime(10) = clocktime
    'idle flag is turned ON
    event_flag(10) = 1
End If

Exit Sub
```

Figure 20. Helicopter Service Period Subroutine - Service Ends

3.5 Helicopter Idle Period Subroutine

This subroutine, as with the helicopter service period subroutine, is split into multiple parts. The first part handles the case for when the helicopter is starting its idle period. The second part collects the statistics of idle period and is used for when the helicopter is moving between runs. The third part takes care of the case of when a failure has occurred and the helicopter is entering the idle period from that maintenance event. The fourth part of the subroutine is the completion of the idle period that was started following a maintenance event. Regardless of whether the helicopter is entering or leaving its idle period, the beginning of the next idle period is set to infinity.

The first part of the subroutine starts off by checking what behavior the helicopter is currently performing. The idle period can occur any time during the day between runs and also at the end of the day after the helicopter has finished its last run. The idle period lasts for as long as needed until another event subroutine is scheduled to run, which means anytime when the helicopter is in service or in repair. The morning is already handled and so the cases for when the helicopter has started operating are taken into account in the code. Thus, the idle period subroutine will run between service runs in the afternoon and all during the night until the start of the next day. Depending on these various situations, the helicopter must have an idle period of a certain amount of hours or else it will not perform the set amount of runs for that day or its activities will cross over into the next day. The first part of the subroutine is shown in the figure below.

```
eventime(10) = duration + 1#
'
If event_flag(10) = 1 Then
    'idle period starts
    idle_clock = clocktime
    'schedule the end of the idle period for 1 or 2 trips/day
    u = Rnd()
    If num_trips = 1 And which_trip = 1 Then
        k = 1
        eventime(10) = clocktime + (480# + 360# * u)
    ElseIf num_trips = 1 And which_trip = 2 Then
        k = 2
        'idle period until the beginning of the next day
        eventime(10) = 1440# * CDbl(day_counter)
    ElseIf num_trips = 2 And which_trip = 1 Then
        k = 3
        'first trip
        eventime(10) = clocktime + (360# + 180# * u)
    ElseIf num_trips = 2 And which_trip = 2 Then
        k = 4
        'second trip
        x = 1440# * CDbl(day_counter) - 600#
        eventime(10) = x + 180# * u
    ElseIf num_trips = 2 And which_trip = 3 Then
        k = 5
        'idle period until the beginning of the next day
        eventime(10) = 1440# * CDbl(day_counter)
    Else
        Stop
    End If
    event_flag(10) = -1
    'redefine the value of prev_interv
    prev_interv = 0
```

Figure 21. Helicopter Idle Period Subroutine – Starting Idle Period Between Runs

The second part of the subroutine occurs when the helicopter is functioning normally and no failures have occurred. Statistics are collected at this point, which include the total accumulated time the helicopter has been idle and the total number of times the helicopter has been idle. Once those have been collected, the next service time period is scheduled unless the helicopter has already completed the required number of runs for the day. If the helicopter is done for the day then another idle period is set for the end of the day. The service flag and idle flag are then set to be on and off or one and zero respectively. The code for the second part of the subroutine is shown in Figure 22 below.

The third part of the subroutine handles the cases for when the helicopter has encountered a failure during its run. After the failure has been handled, the subroutine prepares the helicopter for the idle period by first setting the idle clock variable to the current clocktime, which will help record the helicopter's accumulated idle time later. The idle period is then set to end at the beginning of the next day. Lastly, the idle flag is set to on and the service flag is set to off. The figure below shows the section of code handling this case.

```
ElseIf event_flag(10) = -1 Then
    'collect statistics
    idle_accum = idle_accum + 1# * (clocktime - idle_clock)
    idle_counter = idle_counter + 1
    idle_clock = clocktime
    '
    'schedule the beginning of the service period
    If k <> 2 And k <> 5 Then
        eventime(11) = clocktime    'another service period
        'turn ON the service flag
        event_flag(11) = 1
        'turn OFF the idle flag
        event_flag(10) = 0
    Else
        eventime(10) = clocktime  'idle until the beginning of the next day
        'turn ON the idle flag
        event_flag(10) = 1
        'turn OFF the service flag
        event_flag(11) = 0
    End If
```

Figure 22. Helicopter Idle Period Subroutine - Idle Period Ends Between Runs

```
ElseIf event_flag(10) = 2 Then
    'coming out of an unplanned or planned maintenance period
    'idle period starts
    idle_clock = clocktime
    '
    'idle period until the beginning of the next day
    eventime(10) = 1440# * CDbl(day_counter)
    'turn ON the idle flag
    event_flag(10) = 3
    'turn OFF the service flag
    event_flag(11) = 0
    '
```

Figure 23.Helicopter Idle Period Subroutine - Idle Period Starts After Repair

The last part of the subroutine deals with the helicopter coming out of the idle period when that idle period came after a repair event. As with the second part, the statistics concerning the idle period are collected, the flag for the idle period is set to zero, and the event time for the idle period event is set to infinity. Figure 24 below shows the code for the last part of the subroutine.

```
Else
    'event_flag(10) = 3; complete the idle period following an unplanned maint period
    'collect statistics
    idle_accum = idle_accum + 1# * (clocktime - idle_clock)
    idle_counter = idle_counter + 1
    idle_clock = clocktime
    '
    'set the beginning of the next idle period to infinity
    eventime(10) = duration + 1#
    'turn OFF the idle flag
    event_flag(10) = 0
End If
```

Figure 24.Helicopter Idle Period Subroutine - Idle Period Ends. (Idle period started after a repair event)

3.6 Scheduled Maintenance Event Subroutine

As with the helicopter event subroutines (idle and service periods), this subroutine is split into various parts to handle when the helicopter is entering a maintenance event and when it is leaving a maintenance event. Before the subroutine checks whether the helicopter is entering or leaving a maintenance event, the maintenance flag is set to the current event in order to know for which maintenance event the subroutine is handling (EMA, SMA, CCMA wo UM, COM, EMA wo SM, or NO TEST).

For the case of helicopter entering a maintenance event, the time the maintenance event began is recorded and the counter for the number of times that event has occurred is increased. The subroutine then checks whether the maintenance being performed is either a regular maintenance activity or a no test activity. No test is a special case and therefore has its own empirical distribution that the code uses to determine the duration that the event will last. To obtain the duration of the event, a random number is generated between 0 and 1. This random number will represent a percentage and where it falls in the empirical distribution will determine the amount of time the event will take. Therefore using the empirical distribution for either no test or TTM, the end of the scheduled maintenance event is set. The code for this part of the subroutine is shown in the Figure 25 below.

The second half of the program handles when the maintenance has been completed for the helicopter. First, the statistics for the planned maintenance period is collected. As with previous statistics, the total accumulated time in planned maintenance and the total number of times a planned maintenance period occurs is recorded. The event time for the completion of the maintenance activity is set to infinity and the maintenance flag is set to zero. The next planned maintenance activity is then scheduled using the two transition tables. Two random numbers are generated to be used to select which maintenance activity is next and how many days later the maintenance activity will occur. There is a possibility that the next maintenance event will occur right away. If the next event does occur immediately, then the helicopter status is set to the maintenance flag and the counter for the maintenance event is incremented. A check is also done if the next maintenance event happens in less than a day. If this occurs, then the helicopter status is set to reflect that it is able to provide service before the maintenance activity occurs.

```
If event_flag(maint_flag) = 1 Then
    '
    'beginning of the planned maint period
    interv_clock(maint_flag) = clocktime
    '
    'add to the state transition table
    state_trans_table(prev_interv, maint_flag) = state_trans_table(prev_interv, maint_flag) + 1
    '
    'redefine the value of prev_interv
    prev_interv = maint_flag
    '
    'determine the planned maint duration
    u1 = Rnd()
    'check if (4 <= maint_flag <= 8) or (maint_flag = 9)
    If maint_flag < 9 Then
        For i = 2 To 7
            If u1 <= maint_probs(i) Then
                eventime(maint_flag) = clocktime + maint_times(i - 1) + (maint_times(i) - maint_times(i - 1)) * _
                                       (u1 - maint_probs(i - 1)) / (maint_probs(i) - maint_probs(i - 1))
                event_flag(maint_flag) = -1
                Exit For        'i
            End If
        Next i
    Else
        For i = 2 To 7
            If u1 <= notest_probs(i) Then
                eventime(maint_flag) = clocktime + notest_times(i - 1) + (notest_times(i) - notest_times(i - 1)) * _
                                       (u1 - notest_probs(i - 1)) / (notest_probs(i) - notest_probs(i - 1))
                event_flag(maint_flag) = -1
                Exit For        'i
            End If
        Next i
    End If
'
Else
```

Figure 25. Scheduled Maintenance Event Subroutine - Entering Maintenance Activity

The event time for the idle period is set to the current clocktime because the helicopter will be idle before the next maintenance event occurs. The event flag for the idle period event is set to two so that the idle period event knows that the helicopter just came out of a planned maintenance activity. The last check that is performed is to see whether the next maintenance event will occur after one day. The actions performed for this scenario is similar to the previous one. The helicopter status is set so as to indicate that the helicopter can provide service now. The event time for the idle period event is set to the current clocktime with the event flag for that event set to two. Figure 26 below shows some of the second part of the subroutine.

```
Else
    'event_flag(maint_flag) = -1; end of the planned maintenance period
    '
    'collect statistics on the unplanned maint period
    interv_accum(maint_flag) = interv_accum(maint_flag) + (clocktime - interv_clock(maint_flag))
    interv_counter(maint_flag) = interv_counter(maint_flag) + 1
    'set the end of the previous planned maint period to infinity
    eventime(maint_flag) = duration + 1#
    'set maint_flag to zero
    event_flag(maint_flag) = 0
    'determine the next planned maintenance activity
    '(4-ema w sm, 5-sma, 6-ccma wo um, 7-com, 8-ema wo sm, 9-no test)
    u = Rnd()
    For i = 1 To 6
        If u <= from_to_probs(maint_flag - 3, i) Then
            'k is the new value for maint_flag
            k = 3 + i
            Exit For
        End If
    Next i
    'determine the time to the next maintenance activity
    'j points to (prev_interv,k) unplanned maint distribution
    j = 9 * (maint_flag - 1) + k
    'check if data is available from state maint_flag to state j on time to maintenance (ttm)
    If interv_dur_probs(j, 1) = -1# Then
        Debug.Print "sched main event: problem with selected planned maint distribution"
        Stop
    End If
    u1 = Rnd()
    For i = 1 To 7
        '
        If u1 <= interv_dur_probs(j, i) Then
            'set the prev_interv to maint_flag
            prev_interv = maint_flag
            'time to next maintenance period determined
            eventime(k) = clocktime + 1440# * (CDbl(i) - 1#)
```

Figure 26. Scheduled Maintenance Event Subroutine - Completion of Maintenance Event

3.7 Unscheduled Maintenance Event Subroutine

The unscheduled maintenance event subroutine is split into two parts like the previous event subroutines. The first half deals with the helicopter entering into the event and the second half handles the case when the helicopter is finished with the event. The events classified as unscheduled maintenance generally indicate that a failure has happened and so the helicopter has had to stop providing service before it has completed its run. The subroutine first has to check which failure event has occurred by going into the event array and checking to see which failure event has a value of one. Then it moves on to either the first or second half of the code.

The first half of the subroutine starts off by collecting statistics on the service period that was cut short by the failure. The total accumulated time is incremented by the time spent in the service period. A counter for the total number of times the helicopter has gone out on service is also incremented by one. A variable called “interv_clock” is set to the current clocktime in order to

later record the amount of time spent in the unplanned maintenance event. Since a failure has occurred, the event time for the service completion event is set to infinity as the service can no longer be completed. The counter for the previous event is incremented by one and the previous intervention is set to the failure that is happening. Next, the duration of the unplanned maintenance event is set using the TTR empirical distribution. A random number is generated from zero to one to provide a probability to locate in the distribution. The duration of the event depends on where the random probability is found in the distribution. Figure 27 below shows the code for the first half of the subroutine.

```
If event_flag(failure_flag) = 1 Then
    '
    'collect statistics on the service period
    service_accum = service_accum + (clocktime - service_clock)
    service_counter = service_counter + 1
    '
    'beginning of the unplanned maint period
    interv_clock(failure_flag) = clocktime
    '
    'set the end of the service period to infinity because the unplanned maint is beginning
    eventime(11) = duration + 1#
    '
    'add to the state transition table
    state_trans_table(prev_event, failure_flag) = state_trans_table(prev_event, failure_flag) + 1
    '
    'redefine the value of prev_interv
    prev_interv = failure_flag
    '
    'determine the unplanned maint (repair) duration
    u1 = Rnd()
    For i = 2 To 18
        '
        If u1 <= repair_probs(i) Then
            '
            eventime(failure_flag) = clocktime + repair_times(i - 1) + (repair_times(i) - repair_times(i - 1)) * _
                                     (u1 - repair_probs(i - 1)) / (repair_probs(i) - repair_probs(i - 1))
            event_flag(failure_flag) = -1
            'capture the beginning of the unplanned main period
            interv_clock(failure_flag) = clocktime
            '
            Exit For
        End If
        '
    Next i
```

Figure 27. Unscheduled Maintenance Event Subroutine - Entering Unscheduled Maintenance

When the helicopter is finished with the unplanned maintenance event, the first thing that occurs is that the statistics are collected. The total accumulated time spent in that failure event is recorded along with a counter indicating the total amount of times a certain failure occurred. The variables associated with the unplanned maintenance event are reset to not interfere with the program. The event time for the failure is set to infinity and the flags for that event are set back to zero. The helicopter status is set back to 10, which indicates that the helicopter is able to provide service again. Next, the idle period event is set to the current clocktime along with a flag

indicating that the helicopter is entering into the idle period after it has encountered a failure. Lastly, a counter for the failure event is incremented by one. Figure 28 below shows the other half of the code for the unplanned maintenance event subroutine.

```
Else
    'event_flag(failure_flag) = -1; end of the unplanned maintenance period
    '
    'collect statistics on the unplanned maint period
    interv_accum(failure_flag) = interv_accum(failure_flag) + (clocktime - interv_clock(failure_flag))
    interv_counter(failure_flag) = interv_counter(failure_flag) + 1
    '
    'Debug.Print "unplanned maint period type "; failure_flag; " completed at clocktime="; clocktime
    '
    'set the unplanned maintenance period to infinity
    eventime(failure_flag) = duration + 1#
    'set the unplanned maintenance flag to +1
    event_flag(failure_flag) = 0
    '
    'set helicop_status to 10 (able to provide service)
    helicop_status = 10
    '
    'schedule an idle period to begin immediately (until the beginning of the next day)
    eventime(10) = clocktime
    'flag set to two for idle period until the beginning of the next day (coming out of unplanned maint)
    event_flag(10) = 2
    '
    'add to the state transition table
    state_trans_table(failure_flag, 10) = state_trans_table(failure_flag, 10) + 1
    '
    'set the failure_flag to zero
    failure_flag = 0
    '
End If
'
Exit Sub
```

Figure 28. Unscheduled Maintenance Event Subroutine - Completion of Unscheduled Maintenance Event

3.8 Reliability Estimation Event Subroutine

The reliability estimation event is the analysis of the current behavior of the helicopter. The subroutine starts by observing the current value for all event flags. For failures, if the sum of the failure event flags is greater than one, then that indicates that there is more than one failure active at that time and so an error has occurred in the logic. For maintenance event flags, if the sum of the flags does not equal to one, then that indicates that some problem has occurred in the maintenance subroutine. If the sum of the service period flag and the idle period flag are greater than three, then the helicopter has gone outside the bounds of the program. The value that should result from those flags is -1, 1, or 2 (idle or service event completion, idle or service event starting, idle service coming from unplanned maintenance). To further ensure that the helicopter is behaving as it should, the status of the helicopter is also checked to see if it is either less one or

greater than ten. If it is, then another logic failure has occurred as it can only be a value between one and ten (From helicopter failures to helicopter in service).

The last piece of information that this subroutine collects are the statistics for the entire helicopter. There are already counters and variables holding the accumulated time the helicopter is in an event. The subroutine looks at the helicopter as a whole and counts the number of times the helicopter is in a failure event for all of the different types of failures and the same goes for the maintenance events. The first value the subroutine counts is the number of times the helicopter's status states that it is able to be in service and therefore reflects the number of times the helicopter is idle or in service. The second value looks at the number of times the helicopter is in service or is idle along with the number of times the helicopter has been in a maintenance event. The second value excludes the no test maintenance event as it is different from the other maintenance events. The third value takes the no test event into account along with the aspects counted in the second value. The fourth value looks at the helicopter's downtime and counts the number of times the helicopter has been in an unplanned maintenance event.

4 Experiments and Results

The helicopter chosen to simulate was the one with serial number 8624514, which had the most data rows. The simulation duration time was set to five years or 2628000 minutes. The number of maintenance states was set to 9 and are labeled as follows: 01ma, 02maf, 03ccma w um, 04ema w sm, 05sma, 06ccma wo um, 07com, 09ema wo sm, and 10no test. The number of systems found in the data for the helicopter was two: systems A and E; however, the vast majority of the data available pertained to system A incidents. The reliability estimation frequency was set to 10, which will have the program perform reliability estimation every ten minutes of simulation. The results of the simulation are shown in Figure 29 below.

Figure 29 shows all the statistical counters and variables which are printed onto the spreadsheet. The total accumulated times are converted into percentages to show just how much of the helicopter's time is spent in a certain event. The counters also help identify which event or events the helicopter is most often in; we can see that the helicopter is idle for long periods of time and actually is idle for about 72% of the simulation. The majority of the time left in the simulation is spent working as the helicopter is in service for about 20% of the simulation. Roughly, the other 8% of the simulation is spent in either failure or maintenance events. Surprisingly, the total percentage of the failure events or downtime that occurs is about .1%.

SIMULATION RESULTS

		(6,18)			(7,21)				
		as % ot run length	counter			04ema w sm	05sma	06ccma wo um	07com
helicop_status					04ema w sm	210	178	10	240
10	idle	0.711344592	3519		05sma	254	6610	32	260
10	in service	0.201826588	1841		06ccma wo um	8	28	20	8
					07com	160	306	14	680
4	ema w sm	0.009166566	482		08ema wo sm	106	382	20	160
5	sma	0.09722566	4594		09no test	92	538	10	118
6	ccma wo um	0.00950302	63			830	8042	106	1466
7	com	0.018637501	830						
8	ema wo sm	0.014432262	723						
9	no test	0.022593547	1335						
1	ma	0.001526356	77						
2	maf	0.000169837	5						
3	ccma w um	7.10397E-05	7						
	rel 1=i+s	0.880224	0.001818881	downtime	43800	hrs in five years of simulation			
	rel 2=i+s+sm	0.983390			79.67	hrs devoted to failure repairs in 5 yrs			
	rel 3=i+s+sm+nt	0.9981811							

Figure 29. Simulation Results

There is also a transition table towards the upper right on Figure 29 that details the number of times one event transitioned to another one. The maintenance event CCMA wo UM row in the

table has the least amount of transitions. The event CCMA wo UM stands for a crew correctible maintenance action without the need for unplanned maintenance. This event seems to describe circumstances where a failure occurs on a run but the crew is able to fix the failure successfully without having to stop the service. The results show that it is likely that when failures do occur during a service period that the crew is not able to fix the failure so as to continue running and instead must stop the service in order to repair the helicopter. The transition that occurred most often was from event SMA to event SMA with 6610 occurrences. What is interesting is that no other transition comes even close to this value. Event SMA stands for scheduled maintenance action and indicates just how much of a role that preventive maintenance plays in the current maintenance process. These values are further emphasized when we look at the percentage of time the helicopter spent in these two maintenance events. The maintenance event with the most amount of time and occurrences is the SMA event while the one with the least amount of time and occurrences is event CCMA wo UM.

5 Conclusions

5.1 Data Analysis and the Usefulness of Empirical Distributions

The analysis of the data provided us with the actual behavior of the helicopter instead of predictions that would have been made by probability distributions. There were helicopters with very little amount of data rows while others had several thousand rows of data. The simulation effort was focused on the helicopters with the most data rows as they would provide us with data that would enable us to provide the most realistic behavior in the simulation.

At first, the work was focused on simulating each piece of the helicopter but was then changed to simulating the helicopter as a whole. This was due to the data showing that the systems of the helicopter affected one another and so to create distributions for areas of a helicopter would not work. Also, it was found that System A was the dominant system in the data. Most of the data rows pertained to that system of the helicopter instead of being spread out across all of the systems in the helicopter.

Lastly when observing the events recorded in the excel spreadsheet, it was found that maintenance events were the most frequently occurring event in the data. Less than 1% of the total amount of events in the data was failures, which makes about 99% of the data planned maintenance activities.

5.2 Simulation Results

A serious effort was made in generating output that aligned with the historical data seen in ASAP. Some data was still missing, such as actual flight time or flight hours that could have been used to verify that the simulation results truly did reflect the actual behavior of the helicopter in the real world. Using the data that was received from ASAP, the reliability was measured in the simulation using three aspects of the helicopter's behavior. These aspects were failures, planned maintenance, and no test.

From the simulation performed with the data for helicopter 8624514, several conclusions can be made for the maintenance process currently in place. Both the simulation and data confirm that maintenance events are the most frequent occurring events. More specifically, it is the scheduled maintenance event that occurs most often and the results show how effective the current

maintenance process is. The helicopter was either idle or in service for about 80% of the simulation time and so the scheduled preventive maintenance does accomplish its job of keeping the helicopter able to provide service. While failures do occur, they only take up about .1% of the simulation time and so while the current process does contain an enormous amount of preventative maintenance events (6630), it does stop failures from happening too often.

6 Future Work

6.1 Opportunities Regarding the Swift Download of ASAP Data for Reliability Assessment

Currently, the simulation runs for only helicopter 8624514. This is a good start for providing feedback on the reliability of a helicopter and how the maintenance activities affect the helicopter's uptime and downtime. The simulation was not able to be verified though as the flight duration data was missing. Therefore, the amount of hours that the helicopter normally spends performing missions during the day is needed in order to observe whether the simulation results accurately reflect reality. It would be great to also obtain data for various common places where helicopters are put to work along with the common maintenance tactics that are used with them. This would provide a bigger picture as to what is going on with the helicopters and how different areas affect their performance as opposed to other areas.

6.2 Extraction of Knowledge from the ASAP Data for the Improvement of the Planned versus Unplanned Maintenance Activities

Analysis on the results of the simulation and from the data analysis could be done to observe other aspects of planned maintenance intense activities. There was a column in the ASAP data regarding the number of people working on maintenance. This column was used in the simulation for the aspect of Time to Repair and Time to Maintenance as the more workers working on a problem, the shorter the amount of time needed to finish it. This could be looked at further to see the need and effects of these activities with regards to people safety, failure prevention, mechanic/technician requirements, helicopter inventory, and overall cost. All of these are a factor to consider when looking at maintenance and it does affect the downtime of a helicopter. The results of the simulation could also be entered into another program that will analyze the data and parse it to provide results that could be used with the existing FMECAs.

Since the results should reflect what is actually happening with the helicopter, the FMECA for that helicopter could be updated with what is being seen in the simulation to provide a more accurate FMECA.

6.3 Simulation Program

While the simulation does work, the effort was focused the data for helicopter 8624514. It will not be able to analyze data for a different helicopter that has a different number of data values. Therefore, the future work that can be done for the simulation program is to alter the code in order to enable the program to use data based on how much is available rather than relying on a fixed length of data for each aspect of the helicopter (TTF, TTR, TTM, and No Test). By running simulations for various helicopters, it will be possible to compare the results and observe whether helicopters will share similar end results. To further gauge the behavior of helicopters using this maintenance process, further collaboration could be done with AMRDEC in order to obtain the location of where the helicopters are operating. This could be done in order to compare the results of different locations and see the various results in places like Iraq versus Alabama. Lastly, a data extraction program has been worked on but has not been completed successfully. This program is designed to help with the analysis of the helicopter data in order to quickly provide the user with the data used in the simulation. This will be combined with the simulation program so that the user can enter the extracted data into the simulation and modify it as needed before running the simulation. All of these extensions to the program will enable more analysis and observations to be performed on the helicopters in the ASAP database. Doing so will provide more feedback on the current maintenance processes being used and the results could enable the current process to be improved.

References

[1] Yang, Chao and Zhengfu Zhu, “Study on Reliability Estimation Approach Based on Simulation of Ad Hoc Network” in 2011 9th International Conference on Reliability, Maintainability and Safety (ICRMS) (Guiyang 2011)

[2] Sasso, Daniel and William Biles, “An Object-Oriented Programming Approach for a GIS Data-Driven Simulation Model of Traffic on an Inland Waterway” in Simulation Conference, 2008. WSC 2008. Winter (Austin, TX2008)

[3] Bennane , Abderrazak and Soumaya Yacout, “Processing Missing and Inaccurate Data in a Condition Based Maintenance Database” in 2010 40th International Conference Computers and Industrial Engineering (CIE) (Awaji 2010)

[4] Young ,Thomas , Matthew Fehskens, Paavan Pujara, et. al, “Utilizing Data Mining to Influence Maintenance Actions” in 2010 IEEE AUTOTESTCON (Orlando, FL 2010)