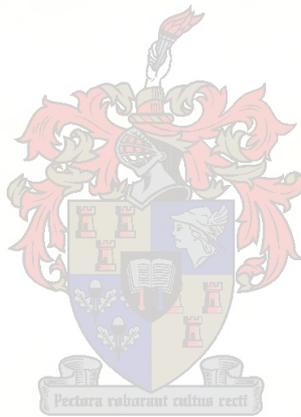


# **THE DEVELOPMENT OF TECHNIQUES TO SELECT A CONTROL POLICY DURING PROACTIVE ON-LINE PLANNING AND CONTROL**

**J. W. MORRIS**



Thesis presented in partial fulfilment of the requirements for the degree of Masters of Industrial Engineering at the University of Stellenbosch.

Study leader: Mr J. Bekker

December 2001

## DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

**Signature:**

**Date:**

## SYNOPSIS

The worldwide trend for systems is to become more complex. This leads to the need for new ways to control these complex systems. A relatively new approach for controlling systems, called on-line planning and control, poses many potential benefits to a variety of end-users, especially in the manufacturing environment. Davis [3] developed a framework for on-line planning and control that is currently incomplete. This project aims to fill one of the gaps in the framework by automating one of the functions, eliminating the need for a human observer. This function, the real-time compromise analysis function, does the comparison of the statistical performance estimates to select a control policy for implementation in the system being controlled (the real-world system) at the current moment in time.

In this project, two techniques were developed to automate the function. The first technique is based on a common technique for statistically comparing two systems, the paired-t confidence interval technique. The paired-t confidence interval technique is used to compare the control policies by building confidence intervals of the expected differences for the respective performance criteria and testing the hypothesis that the statistical performance estimates of the one control policy are better than those of the other control policy. The results of these comparisons are then consolidated into a compromise function that is used to determine the control policy to be implemented currently in the real-world system.

The second developed technique is derived, but differs greatly, from Davis's [3] dominance probability density function approach, and it includes principles of the paired-t confidence interval technique. It compares the control policies by determining the probability (confidence level) with which one can assume that the performance criterion of the one control policy will provide a performance value that is better than the other's and *vice versa*. These confidence levels are then aggregated into a single compromise function that is used to determine the control policy to be implemented currently in the real-world system.

After the techniques were developed, it was not possible to determine their efficiency mathematically, because their statistical base is suspect. The techniques needed to be implemented before they could be evaluated and it was decided to develop an emulator of the on-line planning and control process in accordance with the framework given by Davis [3] to implement them. This Emulator is in essence a Visual Basic<sup>®</sup> program that uses Arena<sup>®</sup> models. However, this Emulator needed certain deviations from the framework to make it possible. Firstly, while the systems that will be controlled with the on-line planning and control process will be complex systems, the system controlled in the Emulator is only a straightforward

M/M/1/FIFO/ $\infty$ / $\infty$  system. This allowed for the conditions that have not been addressed sufficiently, e.g. the initialising of the system models, to be bypassed. Secondly, the Emulator does not include all parts of the framework, and parts for which the technology does not currently exist have been excluded. Thirdly, the real-world system is replaced with a model, because a real-world system was not available for the study. Finally, concurrent operations are actually done sequentially, but in a way that makes it seem that they were done concurrently, as not to influence the results.

This Emulator was used to analyse both techniques for two different traffic intensities. The first part of the analysis consisted of an off-line non-terminating analysis of the individual control policies of the system. This was used as a base line against which the on-line planning and control process of the Emulator was evaluated.

The findings of the evaluations were that, at the traffic intensities evaluated, the techniques provided results that were very similar to the results of the best individual control. From these results, it was speculated that at different traffic intensities, different control policies would be better than the techniques themselves, while the techniques will only give slightly worse results. In addition, because the on-line planning and control process attempts to respond to changing conditions, it can be assumed that the techniques will excel in those conditions where the input distribution is changing continuously. It is also speculated that the techniques may be advantageous in cases where it is not possible to determine beforehand which of the individual control policies to use because it is impossible to predict the input distribution that will occur. It is expected that the techniques will give good (but unfortunately, not necessarily the best) results for any input distribution, while an individual control policy that may give the best results for one input distribution, may prove disastrous for another input distribution.

Three important conclusions can be made from the project. Firstly, it is possible to automate the real-time compromise analysis function. Secondly, an emulator can be developed to evaluate the techniques for the real-time compromise analysis. The greatest advantage of this Emulator is that it can run significantly faster than real-time, enabling the generation of enough data to make the significant statistical comparisons needed to evaluate the techniques. The final conclusion is that while initial evaluations are inconclusive, it can be shown that the techniques warrant further study.

Three important recommendations can be made from the project. Firstly, the techniques need to be studied further, because they cannot be claimed to be perfect, or that they are the only possible techniques that will work. In fact, they are merely techniques that may work and other

techniques may still prove to be better. Secondly, because it would be foolhardy to assume that the Emulator is complete, the Emulator needs to be improved with the most critical need to develop the Emulator in a programming language and simulation package that allows concurrent operations and effortless initialisation. This will enable the Emulator to be much faster and a lot more flexible. The final recommendation is that the techniques need to be evaluated with other parameters in other increasingly complex systems, culminating in the evaluation of the on-line planning and control process with the techniques included in a real-world flexible manufacturing system. Only then can there be decided conclusively on whether the techniques are efficient or not.

It is hoped that this project will form a valuable building block that will facilitate making on-line planning and control a viable alternative to controlling complex systems, enabling them to respond better to changing conditions that are currently becoming the norm.

## OPSOMMING

Wêreldwyd is stelsels besig om meer ingewikkeld te raak. Dit bring mee dat nuwe metodes benodig word om hierdie ingewikkelde stelsels te beheer. Gekoppelde beplanning en beheer ("On-line planning and control") is 'n relatiewe nuwe metode om stelsels te beheer en het baie moontlike voordele vir 'n verskeidenheid van gebruikers, veral in die vervaardigingsomgewing. Davis [3] het 'n raamwerk ontwikkel vir gekoppelde beplanning en beheer, maar die raamwerk is tans onvolledig. Hierdie projek het gepoog om een van die gapings in die raamwerk te vul deur een van die funksies te outomatiseer en sodoende die behoefte vir 'n menslike waarnemer te elimineer. Hierdie funksie, die intydse-kompromie-analise-funksie ("real-time compromise analysis function"), is verantwoordelik vir die vergelyking van die statistiese prestasieskattings om 'n beheerbeleid te kies wat geïmplementeer moet word in die stelsel wat beheer word (die regte-wêreld-stelsel).

Die projek het twee tegnieke ontwikkel om die funksie te outomatiseer. Die eerste tegniek is gebaseer op 'n algemene tegniek om twee stelsels statisties met mekaar te vergelyk, naamlik die gepaarde-t vertrouensinterval-tegniek. Die gepaarde-t vertrouensinterval-tegniek word gebruik om die beheerbeleide te vergelyk deur vertrouensintervalle te bou van die verwagte verskille vir die verskillende vertoningskriteria en om die hipotese te toets dat die statistiese prestasieskattings van die een beheerbeleid beter is as dié van 'n ander beheerbeleid. Die resultate van hierdie vergelykings word dan gekonsolideer in 'n kompromiefunksie wat gebruik word om te bepaal watter beheerbeleid tans geïmplementeer moet word in die regte-wêreld-stelsel.

Die tweede ontwikkelde tegniek is afgelei, maar verskil baie, van Davis [3] se oorheersende waarskynlikheidsdigtheid-funksie ("dominance probability density function") -benadering en gebruik ook idees van die gepaarde-t vertrouensinterval-tegniek. Dit vergelyk die beheerbeleide deur die waarskynlikheid (vertrouensvlak) te bereken waarmee aanvaar kan word dat die vertoningskriterium van een van die beheerbeleide 'n beter vertoningswaarde sal hê as die ander, en omgekeerd. Hierdie vertrouensvlakke word dan gekonsolideer in 'n kompromiefunksie wat gebruik word om te bepaal watter beheerbeleid tans geïmplementeer moet word in die regte wêreld stelsel.

Nadat die tegnieke ontwikkel is, was dit nie moontlik om hulle effektiwiteit wiskundig te evalueer nie, want hulle statistiese basis is verdag. Dus moes die tegnieke geïmplementeer word voordat hulle geëvalueer kon word. Daar is besluit om 'n emuleerder van die proses van gekoppelde beplanning en beheer te ontwikkel volgens die raamwerk wat deur Davis [3] ontwikkel is sodat die tegnieke geïmplementeer kan word. Hierdie Emuleerder is 'n Visual Basic<sup>®</sup> program wat

Arena<sup>®</sup> modelle gebruik. Om die Emuleerder moontlik te maak, was sekere afwykings van die raamwerk nodig. Die eerste hiervan is dat die stelsels wat beheer word met gekoppelde beplanning en beheer, komplekse stelsels is, maar dat die stelsel wat deur die Emuleerder beheer word, slegs 'n eenvoudige M/M/1/EIEB/ $\infty$ / $\infty$  sisteem is. Dit maak dit moontlik om aspekte wat nog nie genoegsaam aangespreek is nie, byvoorbeeld die inisiëring van die stelselmodelle, te omseil. Tweedens bevat die Emuleerder nie al die dele van die raamwerk nie en dele waarvoor die tegnologie tans nog nie bestaan nie, is uitgelaat. Derdens, die regte wêreld stelsel is vervang met 'n model, want 'n regte wêreld stelsel was nie beskikbaar nie. Laastens is operasies wat eintlik gelyktydig gedoen moes word, sekwensteel gedoen, maar op so 'n manier dat dit lyk asof hulle gelyktydig gedoen is, sodat die resultate nie beïnvloed word nie.

Die Emuleerder is gebruik om beide tegnieke te analiseer vir twee verskillende verkeersdigthede. Die eerste deel van die analise het bestaan uit 'n nie-terminerende analise van die individuele beheerbeide van die stelsel. Dit is gebruik as 'n basislyn waarteen die Emuleerder se proses van gekoppelde beplanning en beheer geëvalueer is.

Die bevindinge van die evaluasie was dat vir die verkeersdigthede wat geëvalueer is, die tegnieke resultate lewer wat vergelykbaar is met die van die beste individuele beheerbeide. Oor hierdie resultate is daar gespekuleer dat by verskillende verkeersdigthede, verskillende beheerbeide beter sal vaar as die tegnieke, terwyl die tegnieke slegs marginale swakker resultate sal lewer. En omdat gekoppelde beplanning en beheer poog om te reageer op veranderende omstandighede, kan dit aanvaar word dat die tegnieke sal presteer in omstandighede waar die toevoerverdeling die heelyd verander. Dit word ook beweer dat die tegnieke tot voordeel sal wees in gevalle waar dit nie moontlik is om vooraf te bepaal watter van die individuele beheerbeide om te gebruik nie, omdat dit onmoontlik is om te voorspel watter toevoerverdeling gerealiseer gaan word. Dit word verwag dat die tegnieke goeie (maar ongelukkig nie noodwendig die beste nie) resultate sal lewer vir enige toevoerverdeling, terwyl 'n individuele beheerbeid wat moontlik die beste resultate vir die een toevoerverdeling sal gee, katastrofies kan wees vir 'n ander toevoerverdeling.

Drie belangrike gevolgtrekkings kan gemaak word van die projek. Eerstens, dit is moontlik om die intydse-kompromie-analise-funksie te outomatiseer. Tweedens, 'n emuleerder kan ontwikkel word om die tegnieke vir die intydse-kompromie-analise te evalueer. Die grootste voordeel van die Emuleerder is dat dit heelwat vinniger as reële tyd kan opereer, wat dit moontlik maak om genoeg data te genereer om die betekenisvolle statistiese vergelykings te maak wat benodig word om die tegnieke te evalueer. Die laaste gevolgtrekking is dat, alhoewel die aanvanklike evaluasie nie beslissend is nie, dit gewys kan word dat die tegnieke verdere studie verdien.

Drie belangrike aanbevelings kan gemaak word vanuit die projek. Eerstens, die tegnieke moet nog verder bestudeer word, omdat daar nie beweer kan word dat hulle perfek is of dat hulle die enigste tegnieke is wat kan werk nie. Om die waarheid te sê, hulle is slegs tegnieke wat moontlik kan werk en ander tegnieke kan steeds bewys word om beter te wees. Tweedens sou dit onsinnig wees om te beweer dat die Emuleerder volledig is, en moet die Emuleerder nog verbeter word. Die mees kritiese vereiste is om die Emuleerder te ontwikkel in 'n programmeringstaal en simulasiepakket wat gelyktydige operasies en moeitelose inisiëring toelaat. Dit sal die Emuleerder toelaat om baie vinniger en meer buigsaam te wees. Die laaste aanbeveling is dat die tegnieke geëvalueer moet word met ander parameters in ander stelsels van stygende kompleksiteit, wat die hoogtepunt bereik in die evaluasie van die proses van gekoppelde beplanning en beheer met die tegnieke ingesluit in 'n regte-wêreld buigbare vervaardigingstelsel ("flexible manufacturing system"). Slegs dan sal dit moontlik wees om onomwonde te sê of die tegnieke effektief is of nie.

Daar word gehoop dat hierdie projek 'n waardevolle boublok sal vorm wat sal bydra om gekoppelde beplanning en beheer 'n uitvoerbare alternatief te maak vir die beheer van komplekse stelsels, omdat dit hulle sal toelaat om beter te reageer op die veranderende omstandighede wat deesdae die norm is.

## ACKNOWLEDGEMENTS

I would like to acknowledge the following people for their contributions. Without their help this project would not have been possible:

- ⇒ My study leader, James Bekker, for his assistance and advice, but also for his example of excellence in his work.
- ⇒ The department's technical personnel, Dewald Stander and Willemjan Amoraal, for their help with getting the Emulator running.
- ⇒ Lindie Cloete, for the meticulous proof-reading of this thesis.
- ⇒ My bursary company, SASOL, for sponsoring my studies and giving me time off to complete my studies.
- ⇒ My family and friends, for keeping me going when the going got tough.

## TERMS OF REFERENCE

This project is the result of a chapter (Davis [3]) by Professor W. J. Davis of the University of Illinois in the Handbook of Simulation by Banks [1], which stated that a lot of research is still needed in this field. Paragraphs 13.1, 13.3 and 13.4 of the above-mentioned chapter should be read together with this document.

# TABLE OF CONTENTS

SYNOPSIS .....	I
OPSOMMING .....	IV
ACKNOWLEDGEMENTS .....	VII
TERMS OF REFERENCE .....	VIII
TABLE OF CONTENTS.....	IX
LIST OF FIGURES.....	XII
LIST OF TABLES.....	XIV
GLOSSARY.....	XVII
<b>1 INTRODUCTION TO THE PROJECT .....</b>	<b>1</b>
<b>2 LITERATURE STUDY .....</b>	<b>3</b>
2.1 INTRODUCTION TO PROACTIVE ON-LINE PLANNING AND CONTROL.....	3
2.2 DETAILED DESCRIPTION OF PROACTIVE ON-LINE PLANNING AND CONTROL.....	5
2.3 ASPECTS REGARDING CURRENT ON-LINE SIMULATION ANALYSIS .....	11
2.4 ASPECTS REGARDING OFF-LINE SIMULATION ANALYSIS .....	14
2.5 ASPECTS REGARDING VARIANCE REDUCTION.....	19
2.6 ASPECTS REGARDING THE EMULATOR .....	21
2.7 ASPECTS REGARDING SAMPLE SIZE.....	23
2.8 CONCLUSIONS .....	27
2.9 SUMMARY .....	28
<b>3 THE DEVELOPMENT OF THE TECHNIQUES TO SELECT A CONTROL POLICY</b> .....	<b>29</b>
3.1 THE FIRST TECHNIQUE .....	30
3.2 THE SECOND TECHNIQUE.....	45
3.3 SUMMARY .....	59
<b>4 THE DEVELOPMENT OF THE EMULATOR.....</b>	<b>62</b>
4.1 INTRODUCTION TO THE EMULATOR .....	62
4.2 THE ARENA® MODELS.....	68
4.3 THE EMULATOR PROGRAMMED IN VISUAL BASIC® .....	77
4.4 VALIDATING AND VERIFYING THE EMULATOR .....	80
4.5 SUMMARY .....	81
<b>5 EVALUATING THE NEW TECHNIQUES .....</b>	<b>84</b>
5.1 OFF-LINE NON-TERMINATING ANALYSIS FOR A TRAFFIC INTENSITY OF 0.7 .....	84
5.2 ON-LINE PLANNING AND CONTROL FOR A TRAFFIC INTENSITY OF 0.7 .....	96

5.3	COMBINING THE RESULTS FOR A TRAFFIC INTENSITY OF 0.7 .....	104
5.4	OFF-LINE NON-TERMINATING ANALYSIS FOR A TRAFFIC INTENSITY OF 0.9 .....	107
5.5	ON-LINE PLANNING AND CONTROL FOR A TRAFFIC INTENSITY OF 0.9 .....	115
5.6	COMBINING THE RESULTS FOR A TRAFFIC INTENSITY OF 0.9 .....	119
5.7	DISCUSSION OF THE RESULTS OF THE EVALUATION OF THE TECHNIQUES .....	121
5.8	SUMMARY .....	126
6	CONCLUSIONS.....	128
6.1	IT IS POSSIBLE TO AUTOMATE THE REAL-TIME COMPROMISE ANALYSIS FUNCTION .....	128
6.2	AN EMULATOR CAN BE DEVELOPED TO EVALUATE THE TECHNIQUES THAT AUTOMATE THE REAL-TIME COMPROMISE ANALYSIS FUNCTION .....	128
6.3	THE TECHNIQUES DEVELOPED IN THIS PROJECT TO AUTOMATE THE REAL-TIME COMPROMISE ANALYSIS FUNCTION WARRANT FURTHER STUDY .....	129
7	RECOMMENDATIONS .....	130
7.1	THE DEVELOPED TECHNIQUES NEED TO BE STUDIED FURTHER .....	130
7.2	THE DEVELOPED EMULATOR NEEDS TO BE IMPROVED .....	131
7.3	THE DEVELOPED TECHNIQUES NEED TO BE EVALUATED FURTHER .....	132
8	SUMMARY .....	134
9	REFERENCES .....	135
10	BIBLIOGRAPHY .....	136
	APPENDICES .....	138
A	IMPLEMENTATION OF AND SAMPLE CALCULATIONS FOR ANTONACCI'S METHOD.....	A-1
B	THE EMULATOR IMPLEMENTED IN ARENA® AND VISUAL BASIC® .....	B-1
B.1	ARENA® MODEL LOGIC .....	B-1
B.2	VISUAL BASIC® CODE .....	B-10
B.3	SUMMARY OF THE SUBPROGRAMS AND FUNCTIONS USED IN THE EMULATOR .....	B-55
B.4	VISUAL BASIC® FORM .....	B-62
C	OPERATING THE EMULATOR.....	C-1
D	VISUAL BASIC® PROGRAM "FILES" .....	D-1
D.1	VISUAL BASIC® CODE .....	D-1
D.2	VISUAL BASIC® FORM .....	D-5
D.3	VERIFICATION .....	D-6
E	THE EVOLUTION OF THE EMULATOR.....	E-1
E.1	VISUAL BASIC®, ARENA® AND MICROSOFT® EXCEL® .....	E-1
E.2	VISUAL BASIC® AND ARENA® .....	E-1

E.3	ARENA®'S VISUAL BASIC® AND ARENA® .....	E-2
<b>F</b>	<b>OUTPUT RESULTS.....</b>	<b>F-1</b>
F.1	OUTPUT RESULTS FOR A TRAFFIC DENSITY OF 0.7 .....	F-1
F.2	OUTPUT RESULTS FOR A TRAFFIC DENSITY OF 0.9 .....	F-5

E.3	ARENA**s VISUAL BASIC* AND ARENA*	E-2
F	OUTPUT RESULTS	F-1
F.1	OUTPUT RESULTS FOR A TRAFFIC DENSITY OF 0.7	F-1
F.2	OUTPUT RESULTS FOR A TRAFFIC DENSITY OF 0.9	F-5

## LIST OF FIGURES

Figure 2.1 Davis [3]'s schematic representation of the proactive on-line planning and control process.....	5
Figure 2.2 A schematic representation of the Emulator .....	22
Figure 2.3 Schematic representation of Antonacci's method .....	24
Figure 3.1 Schematic representation of the first developed technique .....	31
Figure 3.2 The First technique implemented .....	37
Figure 3.3 The different parts of the First technique.....	38
Figure 3.4 Part A of the First technique.....	39
Figure 3.5 Part B of the First technique .....	39
Figure 3.6 Part C.1 of the First technique .....	40
Figure 3.7 Part 1.1 of the First technique .....	41
Figure 3.8 Part 2.1.1 of the First technique .....	42
Figure 3.9 Part 3.1 of the First technique .....	43
Figure 3.10 Part 4 of the First technique.....	44
Figure 3.11 Schematic representation of the second developed technique.....	46
Figure 3.12 The Second technique implemented .....	50
Figure 3.13 The different parts of the Second technique.....	51
Figure 3.14 Part A of the Second technique .....	52
Figure 3.15 Part B of the Second technique .....	52
Figure 3.16 Part C.1 of the Second technique .....	53
Figure 3.17 Part 1.1 of the Second technique .....	54
Figure 3.18 Part 2.1.1 of the Second technique .....	55
Figure 3.19 Part 3.1 of the Second technique .....	57
Figure 3.20 Part 4 of the Second technique.....	58
Figure 4.1 Schematic representation of the Emulator with the different parts distinguished .....	63
Figure 4.2 Schematic representation of the Emulator operations.....	64
Figure 4.3 The initial Visual Basic® form of the Emulator .....	65
Figure 4.4 The pseudo-concurrent operation of the Emulator illustrated .....	68
Figure 4.5 The concept model for the Arena® models .....	69
Figure 4.6 The concept model of the model logic of the creation of entities .....	77
Figure 4.7 Schematic block diagram of the Visual Basic® routines .....	78
Figure 4.8 The parts of Davis [3]'s on-line planning and control process represented in the Emulator.....	82
Figure 5.1 Moving average plot of <b>Time in system</b> for <u>Longest service time</u> .....	86
Figure 5.2 Correlogram for the performance criterion <b>Lateness</b> for <u>Last in first out</u> .....	87
Figure 5.3 Correlogram for the performance criterion <b>Lateness</b> for <u>Last in first out</u> batched in batches of size 800 .....	88
Figure 5.4 Correlogram for the performance criterion <b>Job productivity</b> for the First technique .....	99

Figure 5.5 Correlogram for the performance criterion **Job productivity** for the First technique with batch size 350 ..... 100

Figure 5.6 Moving average plot of **Length of service queue** for First in first out ..... 108

Figure A. 1 Antonacci’s method implemented in Excel® .....A-2

Figure A. 2 Parts of Excel® spreadsheet corresponding to Antonacci’s method discussed together .....A-3

Figure A. 3 Part A of Antonacci’s method .....A-4

Figure A. 4 Part B of Antonacci’s method .....A-4

Figure A. 5 Part C.1 of Antonacci’s method.....A-4

Figure A. 6 Part 1 of Antonacci’s method.....A-5

Figure A. 7 Part 2.1 of Antonacci’s method .....A-7

Figure A. 8 Part 3 of Antonacci’s method.....A-7

Figure A. 9 Part 4 of Antonacci’s method.....A-7

Figure B. 1 The overall model logic for the real-world system model..... B-2

Figure B. 2 The overall model logic for the alternative system model ..... B-3

Figure B. 3 The model logic of the creation of entities for the real-world system or the alternative system model..... B-5

Figure B. 4 The model logic of the server for the real-world system or alternative system model ..... B-6

Figure B. 5 The model logic of the tallies and the disposal of the entities of the real-world system modelB-7

Figure B. 6 The model logic of the tallies and the disposal of the entities of the alternative system modelB-7

Figure B. 7 The definition of the experimental parameters for the real-world system model ..... B-8

Figure B. 8 The definition of the experimental parameters for the alternative system model ..... B-8

Figure B. 9 The model animation ..... B-9

Figure B. 10 Code for the writing of discrete change variables to file..... B-9

Figure B. 11 *FrmInitial* of the Emulator ..... B-62

Figure D. 1 The Visual Basic® form used by Visual Basic® program “Files” ..... D-5

Figure D. 2 Data for *ControlPolicyOne* ..... D-6

Figure D. 3 Data for *ControlPolicyTwo*..... D-7

Figure D. 4 Data for *ControlPolicyThree* ..... D-7

Figure D. 5 Data for *ControlPolicyCombined* ..... D-8

Figure D. 6 Results of calculations on *ControlPolicyCombined* ..... D-9

## LIST OF TABLES

Table G. 1 Terms used in the project .....	xviii
Table G. 2 Variables used in the project .....	xxi
Table G. 3 Symbols used in the project .....	xxiii
Table G. 4 Abbreviations used in the project .....	xxiv
Table G. 5 Prefixes used in the project .....	xxv
Table 2.1 The conclusions from the literature study .....	28
Table 3.1 The properties of on-line simulation and their special assumptions.....	30
Table 3.2 The differences between the two techniques .....	60
Table 3.3 Subjects that require further study, as identified during the development of the techniques.....	61
Table 4.1 The main subprograms of the Emulator .....	79
Table 4.2 Subjects that require further study, as identified during the development of the Emulator .....	83
Table 5.1 Variable settings for off-line non-terminating analysis for a traffic intensity of 0.7 .....	85
Table 5.2 Data for pilot runs for a traffic intensity of 0.7 .....	89
Table 5.3 Batch size information for <u>First in first out</u> for a traffic intensity of 0.7 .....	89
Table 5.4 Results for <u>First in first out</u> pilot run for a traffic intensity of 0.7 .....	90
Table 5.5 Batch size information for <u>Last in first out</u> for a traffic intensity of 0.7 .....	90
Table 5.6 Results for <u>Last in first out</u> pilot run for a traffic intensity of 0.7 .....	91
Table 5.7 Batch size information for <u>Latest job</u> for a traffic intensity of 0.7 .....	91
Table 5.8 Results for <u>Latest job</u> pilot run for a traffic intensity of 0.7 .....	92
Table 5.9 Batch size information for <u>Longest service time</u> for a traffic intensity of 0.7 .....	92
Table 5.10 Results for <u>Longest service time</u> pilot run for a traffic intensity of 0.7 .....	94
Table 5.11 Specific data for <u>Longest service time</u> production run for a traffic intensity of 0.7 .....	94
Table 5.12 Results for <u>Longest service time</u> production run for a traffic intensity of 0.7 .....	95
Table 5.13 Batch size information for <u>Shortest service time</u> .....	95
Table 5.14 Results for <u>Shortest service time</u> pilot run for a traffic intensity of 0.7 .....	96
Table 5.15 Emulator variable settings for a traffic intensity of 0.7 .....	97
Table 5.16 Specific data for First technique pilot run for a traffic intensity of 0.7 .....	101
Table 5.17 Batch size information for First technique pilot run for a traffic intensity of 0.7 .....	101
Table 5.18 Results for First technique pilot run for a traffic intensity of 0.7 .....	102
Table 5.19 Control policy switching information for First technique pilot run for a traffic intensity of 0.7 .....	102
Table 5.20 Specific data for Second technique pilot run for a traffic intensity of 0.7 .....	103
Table 5.21 Batch size information for Second technique pilot run for a traffic intensity of 0.7 .....	103
Table 5.22 Results for Second technique pilot run for a traffic intensity of 0.7 .....	104
Table 5.23 Control policy switching information for Second technique pilot run for a traffic intensity of 0.7 .....	104

Table 5.24 Combined results for a traffic intensity of 0.7 .....	106
Table 5.25 Variable settings for off-line non-terminating analysis for a traffic intensity of 0.9 .....	107
Table 5.26 Data for pilot runs for a traffic intensity of 0.9 .....	108
Table 5.27 Batch size information for <u>First in first out</u> for a traffic intensity of 0.9 .....	109
Table 5.28 Results for <u>First in first out</u> pilot run for a traffic intensity of 0.9 .....	110
Table 5.29 Batch size information for <u>Last in first out</u> for a traffic intensity of 0.9 .....	110
Table 5.30 Results for <u>Last in first out</u> pilot run for a traffic intensity of 0.9 .....	111
Table 5.31 Batch size information for <u>Latest job</u> for a traffic intensity of 0.9 .....	111
Table 5.32 Results for <u>Latest job</u> pilot run for a traffic intensity of 0.9 .....	112
Table 5.33 Batch size information for <u>Longest service time</u> for a traffic intensity of 0.9 .....	112
Table 5.34 Results for <u>Longest service time</u> pilot run for a traffic intensity of 0.9 .....	113
Table 5.35 Specific data for <u>Longest service time</u> production run for a traffic intensity of 0.9 .....	113
Table 5.36 Results for <u>Longest service time</u> production run for a traffic intensity of 0.9 .....	114
Table 5.37 Batch size information for <u>Shortest service time</u> for a traffic intensity of 0.9 .....	114
Table 5.38 Results for <u>Shortest service time</u> pilot run for a traffic intensity of 0.9 .....	115
Table 5.39 Emulator variable settings for a traffic intensity of 0.9 .....	115
Table 5.40 Specific data for First technique pilot run for a traffic intensity of 0.9 .....	116
Table 5.41 Batch size information for First technique pilot run for a traffic intensity of 0.9 .....	116
Table 5.42 Results for First technique pilot run for a traffic intensity of 0.9 .....	117
Table 5.43 Control policy switching information for First technique pilot run for a traffic intensity of 0.9 .....	117
Table 5.44 Specific data for Second technique pilot run for a traffic intensity of 0.9 .....	118
Table 5.45 Batch size information for Second technique pilot run for a traffic intensity of 0.9 .....	118
Table 5.46 Results for Second technique pilot run for a traffic intensity of 0.9 .....	119
Table 5.47 Control policy switching information for Second technique pilot run for a traffic intensity of 0.9 .....	119
Table 5.48 Combined results for a traffic intensity of 0.9 .....	120
Table 5.49 Combined results for both traffic intensities .....	124
Table 5.50 Combined usage and switching information for both traffic intensities and both techniques	125
Table 5.51 Subjects that require further study in increasingly complex systems, as identified during the evaluation of the techniques .....	127
Table B. 1 The subprograms in the <i>This Document</i> part of the real-world system model .....	B-55
Table B. 2 The secondary subprograms of the Visual Basic® form <i>fmbInitial</i> .....	B-56
Table B. 3 The functions of the Visual Basic® module, <i>Module 2</i> .....	B-57
Table B. 4 The subprograms in the Visual Basic® module, <i>Module 2</i> .....	B-60
Table D. 1 Calculated values for percentage of time spend under control policy .....	D-8

Table D. 2 Calculated values for average number of breaks between switches..... D-9

## GLOSSARY

The glossary is divided into five parts, each in its own table. Table G. 1 gives the terms used in the study, Table G. 2 the variables, Table G. 3 the symbols, Table G. 4 the abbreviations and Table G. 5 the prefixes.

Table G. 1 Terms used in the project

Term	Explanation
Alternative system	System where an alternative control policy is implemented.
Alternative system model	Model used to evaluate the different alternative systems.
Arena <sup>®</sup>	A simulation package from Rockwell Software.
Breaktime	The time at which the real-world system model must be stopped to evaluate a control policy with the alternative system model.
Central limit theorem	Theorem stating that the distribution of a large number of averages approaches the standard normal distribution as the sample size increases, regardless of the distribution of individual observations.
Confidence interval	Interval in which performances criterion values are expected to be for a specific confidence level.
Confidence level	Level of confidence with which statistical inferences about observations are made.
Control policy	The set of operational parameters that comprises an alternative system.
Current model	Model with which the current system is evaluated.
Current system	System in which the current control policy is implemented.
Design parameter	Both the real-world system and its model are characterized by a set of design parameters that effect both the state transition and the output functions during design.
Emulator	A system replicating the results of another system, while doing it in a different way. When emulator is written with a capital, it refers to the specific emulator used in this project.
Excel <sup>®</sup>	A spreadsheet package from Microsoft <sup>®</sup> Corporation.
Independent and identically distributed	Observations are said to be independent and identically distributed when there is no correlation between them and they are from the same probability distribution.
Method	Particular and systematic way of doing something (same as technique).
Model	A representation of the system in a form other than the system itself.
Off-line planning process	The off-line planning process tries to define the optimal set of values for the design parameters and is detached from the real-world system.
Off-line simulation	Refers to conventional discrete-event simulation and is detached from the real-world system.
Off-line simulation analysis	Refers to conventional discrete-event simulation analysis and is detached from the real-world system.
On-line planning and control process	Planning and control done in real-time while attached to the real-world system.

Term	Explanation
On-line simulation	A new simulation paradigm that tries to bring simulation on-line by attaching it to the real-world system.
On-line simulation analysis	The analysis of on-line simulation output, that includes both real-time output analysis and real-time compromise analysis, while attached to the real-world system.
Operator	The person overseeing the controlling of the real-world system.
Operational parameter	Both the real-world system and its model are characterized by a set of operational parameters that effect both the state transition and the output functions during operation.
Performance criterion	Measure of the performance of the system.
Performance criterion value	Specific instance of a performance criterion.
Proactive on-line planning and control	On-line planning and control that operates concurrently with the real-world system and is constantly seeking an improved control policy.
Reactive on-line planning and control	On-line planning and control that can be viewed as performing off-line planning more frequently over shorter planning horizons.
Real-time operations	Operations that are fast enough to influence decisions.
Real-world system	System undergoing planning and control.
Real-world system model	Model of the system undergoing planning and control.
Replication	A specific simulation of a system.
Run	A single replication of the Emulator.
Runtime	Period emulated during a run of the Emulator.
Stack	Set of the last $W$ performance criterion values.
Steady state analysis	Analysis of a system, with IID observations, as averaged over a long time.
System	Combination of parts to achieve a common goal.
Technique	Particular and systematic way of doing something (same as method).
Traffic intensity	Ratio between the expected service time and the expected time between arrivals for a queuing system.
Transient analysis	Analysis of the start-up response of a system.
Trial	A specific observation generated by breaking the real-world system model at a specific time and making a replication from that state.
Visual Basic <sup>®</sup>	Programming package from Microsoft <sup>®</sup> Corporation.

Term	Explanation
Word®	A word processing package from Microsoft® Corporation.
WordPad®	A word processing package from Microsoft® Corporation.

Table G. 2 Variables used in the project

(Capital letters refer to a specific instance, e.g. the total number and uncapitalized letters refer to the general use, e.g. a counter value. Variables in subscripts have the same meaning as normal variables.)

Variable	Explanation
$a$	Constant for Arena <sup>®</sup> 's random number generator.
$c$	Constant for Arena <sup>®</sup> 's random number generator.
$CF_c$	Compromise function for the current system.
$CF_p$	Compromise function for alternative system $p$ .
$CL$	Confidence level between 0 and 100, that is the level of confidence with which statistical inferences about observations is made.
$CL_a$	Confidence level used for Antonacci's method
$CL_g$	Confidence level for which it can be assumed that the alternative system's performance criterion $g$ is better than the current's or <i>vice versa</i> .
$D$	Number of confidence intervals for Bonferroni inequality.
$G$	Total number of performance criteria considered.
$g$	Counter for performance criteria.
$h$	Confidence interval half width.
$h^*$	Desired confidence interval half width.
$H_c$	Indicator used in compromise function of the current system.
$H_p$	Indicator used in compromise function of the alternative system.
$H_0$	The null hypothesis.
$H_1$	The alternative hypothesis.
$I_{gp}$	Confidence interval for the expected value of a measure of performance $\mu_{gp}$ . That is a $100(1-\alpha_{gp})$ percent confidence interval for a measure of performance $w$ , performance criterion $g$ and alternative system $p$ .
$J$	Total number of entities considered in a replication evaluating an alternative system (same as planning horizon).
$\overline{JP}_{wp}$	Average job productivity for replication $w$ for alternative system $p$ .
$j$	Counter for the number of entities being serviced.
$\overline{LN}_{wp}$	Average lateness for replication $w$ for alternative system $p$ .
$\overline{LSQ}_{wp}$	Average length of server queue for replication $w$ for alternative system $p$ .
$M$	Subset size for grouping best alternative systems.

Variable	Explanation
$m$	Constant for Arena <sup>®</sup> 's random number generator.
$n$	Number of batches.
$n^*$	Required number of batches.
$P$	Total number of alternative systems compared (including current system).
$\overline{PP}_{wp}$	Average process productivity for replication $w$ for alternative system $p$ .
$p$	Counter for alternative systems.
$PT_{jwp}$	Process time for entity $j$ for replication $w$ for alternative system $p$ .
$R_{expo}$	Random sample value from the exponential distribution.
$S$	Value added to the top of the stack when the oldest projected value is removed from the bottom of the stack.
$s^2( )$	Unbiased estimator of sample variance $\sigma^2$ .
$TIS_{jwp}$	Time in system for entity $j$ for replication $w$ for alternative system $p$ .
$\overline{TIS}_{wp}$	Average time in system for replication $w$ of alternative system $p$ .
$t$	Counter for techniques.
$t_{n-1,1-\alpha/2}$	Upper $1-\alpha/2$ critical point on the Student t-distribution with $n-1$ degrees of freedom. In Microsoft <sup>®</sup> Excel <sup>®</sup> the value is given by the function TINV ( $\alpha$ , $v$ ) with $v = n-1$ .
$U(0,1)$	Random number between 0 and 1 from the uniform distribution.
$v_g$	Importance value for performance criterion $g$ .
$W$	Total number of most recent performance criteria values used for statistical analysis.
$w$	Counter for the performance criteria values.
$\overline{\overline{X}}_p$	Average of criterion $G$ 's averages for alternative system $p$ .
$\overline{\overline{X}}_{sp}$	Average of replication $w$ 's averages for performance criterion $g$ for alternative system $p$ .
$\overline{X}_{wgp}$	Average of entity $J$ 's performance criterion values for the $w^{\text{th}}$ replication of performance criterion $g$ for alternative system $p$ .
$X_{jwgp}$	Entity $j$ 's performance criterion value for the $w^{\text{th}}$ replication of performance criterion $g$ for alternative system $p$ .
$Y_i$	Seed number $i$ .
$\overline{Z}_g$	Sample mean of the $w$ differences for performance criterion $g$ .
$Z_{wg}$	Difference between the $w^{\text{th}}$ respective observation of two alternative systems for performance criterion $g$ .

Table G. 3 Symbols used in the project

Symbol	Explanation
$\alpha$	Two-tail significance level (alpha). This is the degree of uncertainty about the statistical statement under specified conditions.
$\alpha_a$	Alpha value used for Antonacci's method.
$\alpha_c$	Alpha value used for comparison.
$\alpha_g$	Alpha value for Bonferroni approach for performance criterion $g$ .
$\alpha_{gp}$	Alpha value for Bonferroni approach for performance criterion $g$ and alternative system $p$ .
$\alpha_p$	Alpha value for Bonferroni approach for alternative system $p$ .
$\beta$	The mean of the exponential distribution given by $f(x) = \frac{1}{\beta} e^{-\frac{x}{\beta}}, \beta > 0, x > 0$ .
$\delta$	Difference between one control policy's performance and another's for dominance probability density function approach.
$\mu_{gp}$	The expected response of performance criterion $g$ for alternative system $p$ , equal to $E(\bar{X}_{wgp})$ .
$\rho$	Traffic intensity.
$\sigma^2$	Sample variance.
$\zeta_g$	Difference between the expected responses of performance criterion $g$ for two alternative systems, equal to $\mu_{g1} - \mu_{g2}$ .

Table G. 4 Abbreviations used in the project

Abbreviation	Explanation
CIM	Computer integrated manufacturing.
CRN	Common random numbers.
e.c.d.f.	Empirical cumulative density function.
Eqn.	Equation.
FIFO	First in first out (Alternative system 1).
FMS	Flexible manufacturing system.
IID	Independent and identically distributed.
JP	Job productivity (Performance criterion 4).
LIFO	Last in first out (Alternative system 2).
LJ	Latest job (Alternative system 3).
LN	Lateness (Performance criterion 3).
LSQ	Length of service queue (Performance criterion 5).
LST	Longest service time (Alternative system 4).
n.a.	Not applicable.
PP	Process productivity (Performance criterion 2).
SST	Shortest service time (Alternative system 5).
TIS	Time in system (Performance criterion 1).
VBA	Visual Basic* for Applications.

Table G. 5 Prefixes used in the project

Prefix	Explanation
att	Attribute.
cll	Cell.
cmd	Command button.
cnt	Counter.
dir	Directory list box.
drv	Drive list box.
fil	File list box.
fnc	Function.
fra	Frame.
frm	Form.
g_	Global variable.
ind	Index.
lbl	Label.
lst	List box.
opt	Option button.
pic	Picture box.
rng	Range.
see	Seed variable.
set	Set.
sub	Subprogram.
tal	Tally.
txt	Textbox.
var	Variable.

## 1 INTRODUCTION TO THE PROJECT

The worldwide trend is for systems to become more complex. This is especially true in the manufacturing environment with the emergence of computer integrated manufacturing (CIM) and flexible manufacturing systems (FMSs), but is also applicable to other systems like air traffic control and vehicle routing. This leads to the need for new ways to control these systems, e.g. the need of manufacturers to change the way they plan and control their processes, so that they can respond better to changing conditions on the shop floor. Rodgers and Gordon [13] named a range of techniques from the disciplines of control theory, operations research and artificial intelligence that are currently being used to address the problem of planning and control. A relatively new approach, on-line planning and control, poses many potential benefits to a variety of end-users, especially in a manufacturing organization. Drake and Smith [7] state that this is possible because on-line planning and control processes can predict the future behaviour of the system reliably given its current state, and because it has the ability to emulate and / or dictate the control logic of a system.

Davis [3] states that on-line planning and control is still a barren field and lots of research needs to be done. On-line planning and control in simple systems such as robotics has been addressed with significant advances in the intelligent control technologies. However, these are not suitable to large-scale discrete-event systems because it only addresses the management of a single continuous subsystem without coordination. Large-scale systems will require the coordinated operation of many different subsystems, each with its own sophisticated intelligent controller to address its own planning and control in real-time. A new modelling approach is needed to enable this. It should be controller-based to allow for systems using a control architecture, because the ability to assess the impact that the controller interactions have on the system becomes crucial. Davis [3] developed this and showed that it is especially useful in the on-line planning and control situation where both planning and control must be distributed across a hierarchy of intelligent controllers, also called coordinators. This new paradigm of modelling controller interactions will have the advantages of enhanced maintainability and re-usability because most of the existing code will be re-usable. The modelling aspects are not part of this project and the reader should see Davis [3] in this regard. Davis [3] cautions against believing that modelling controller interactions will solve every issue pertaining to the design and management of large-scale discrete-event systems. It does, however, represent a useful alternative to the conventional entity flows through stochastic queuing networks and is imperative to the development of proactive on-line planning and control. Drake and Smith [7] give a framework for on-line

planning and control, but it is for reactive on-line planning and control. The framework Davis [3] developed for on-line planning and control is proactive and will be used for this project.

This project will only focus on a small part of the framework, i.e. the development of techniques to select a control policy for implementation during proactive on-line planning and control. It is important to keep in mind that there are still many other pieces of the puzzle missing.

The project starts with a literature study in Chapter 2 on page 3 by looking at proactive on-line planning and control as well as other aspects that may be used later in the project. The project will then develop two techniques and describe them in Chapter 3 on page 29. These two techniques need to be evaluated and the on-line planning and control emulator that makes it possible to evaluate them is described in Chapter 4 on page 61. In Chapter 5 on page 84, the data generated with the on-line planning and control emulator is analysed and the results interpreted. The project ends with conclusions in Chapter 6 on page 121, recommendations in Chapter 7 on page 130 and a summary in Chapter 8 on page 134.

## 2 LITERATURE STUDY

The literature study discusses aspects that provide a background to the project and explains concepts that will be used in the project. The project focuses on a small part of the framework developed for proactive on-line planning and control, so the literature study starts by introducing proactive on-line planning and control and discussing Davis [3]'s framework. The rest of the literature study comprises of concepts that will be used in the project. These include current on-line and off-line simulation analysis techniques, variance reduction techniques, emulations and the determination of sample size.

### 2.1 Introduction to proactive on-line planning and control

Proactive on-line planning and control is an advanced and specialised use of simulation. In order to understand where it fits into the simulation family, it is first necessary to distinguish and discuss the three major uses for simulation. Davis [4] groups the studies done by simulationists into three main categories:

- a) Systems design.
- b) Training systems.
- c) On-line planning and control systems.

The historic, and most widely used, application of discrete-event simulation is the design of systems. This usually entails either the off-line analysis of a system to be implemented or the operation of an existing system in a new manner. The goal is to specify ideal values for a set of design parameters that will be implemented when the system is brought into operation. Once the system is implemented, the model becomes antiquated. Off-line simulation can be used for off-line planning. The purpose of simulation within the off-line planning scenario is to assess the performance of the system while it operates under a given set of values for the design parameters. The entire off-line planning process tries to define the optimal set of values for these design parameters. This can seldom be achieved for real-world systems, because the model is always an approximation of the real-world system and it is virtually impossible to explore all the feasible values for the design parameters. To understand the role on-line simulation plays in on-line planning and control, the reader must be familiar with the process of off-line simulation as well as off-line planning. If this is not the case, the reader should familiarise himself with Chapters 3, 6, 7, 9 and 10 from Banks [1].

The use of simulation for training tries to bring the human into the picture. It attempts to let people interact with simulations in a way that replicates the way they interact with the actual system. The models employed must consider the control inputs that can be managed by a person to influence the behaviour of the actual systems. Moreover, there is a need to render the environment in which the managed system operates effectively in order to provide the external stimulus to which the trainee must respond.

The use we are interested in, is on-line planning and control, which refers to the process of managing a system. The planning part comprises of the determination of the optimal values of the parameters of the system. Examples of these optimal values for the parameters may be to minimise work in progress, maximise throughput or any other combination of desirable outcomes. The control part is concerned with the execution of the plan. This may be to remove a bottleneck to reduce work in progress or to increase the workload at some workstations to increase throughput.

The use of on-line simulation models during the on-line planning and control process is a relatively new concept. It evolved from a critical need for a validated model to make realistic projections of the future performance of the system in an on-line manner, but because today's systems are complex, time-variant and stochastic, deterministic models are not adequate. Therefore, on-line planning and control was born along with a completely new set of problems. Examples of these problems include that the models need to be updated constantly, that they have to be validated and that they must consider the same control input that the manager uses to influence the operation of the actual system. The simulation analysis employed in on-line planning and control requires the on-line projection of the system operating under one or more alternative strategies. This requires the models to be executed much faster than real-time and the employed model to be initialised to the current state of the system continuously.

The on-line planning and control process discussed up to now has included both reactive and proactive planning and control. Most systems discussed in the literature refer to reactive planning and control, e.g. those by Drake and Smith [7], Holter, *et al* [10] and ElMaraghy, *et al* [8]. Rodgers and Gordon [13] look at non-simulation, simulation and hybrid approaches to planning and control, but they are all reactive. Davis [3] states that reactive planning and control algorithms are not true on-line algorithms. This is because they begin by choosing a control policy to be implemented in an off-line manner. The system is monitored until the planned response and the realised response have deviated in such a manner that the current control policy is no longer valid. A new control policy is then determined by off-line planning. This process is inefficient, because it can be viewed as only performing off-line planning more frequently over shorter

planning horizons. Davis [3] explains that it is inefficient, but easier to implement, because less new technologies are needed than for proactive on-line planning and control.

Proactive on-line planning and control proposes to eliminate this inefficiency. It operates concurrently with the real-world system and is constantly seeking an improved control policy. Alternative control policies are always being generated and a new control policy can be implemented whenever it is beneficial to do so. From now on any reference to on-line planning and control will refer to proactive on-line planning and control. A schematic representation for the proactive on-line planning and control process as proposed by Davis [3] is shown in Figure 2.1 and will be discussed in the next paragraph.

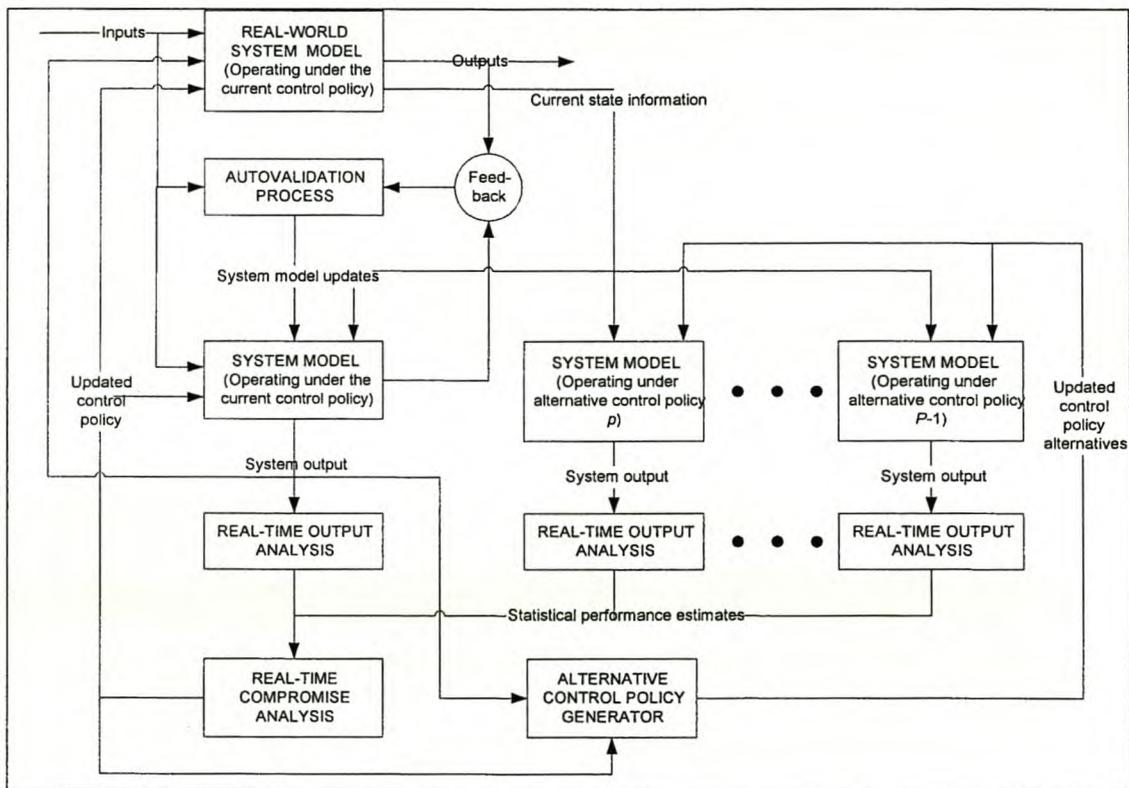


Figure 2.1 Davis [3]'s schematic representation of the proactive on-line planning and control process

## 2.2 Detailed description of proactive on-line planning and control

Proactive on-line planning and control is a young field of study, and Prof. W. J. Davis, of the University of Illinois, has done a lot of work in the field. In Davis [3], he explains proactive on-line planning and control with the use of Figure 2.1 and says while it may seem complex, it is only a simplification of the true on-line planning and control process. The framework he developed will be discussed in detail in this paragraph because this project uses it as a backbone.

Two aspects deserve a special mention. The first is the control policy. The selected control policy is an important link between most of the blocks. Together with the state transition function, it determines the state of the system as a function of time. This response is not deterministic due to stochastic system elements and inputs outside its control.

The second important aspect is that there is no mention of planning in Figure 2.1. It is explained by acknowledging that the control policy that has been selected for implementation is a critical element of both the real-world system and its model. In the real-time operation of the system, a selected plan must be implemented immediately. This implies that choosing a plan necessarily requires specification of the control policy that will implement the selected plan. Therefore, the plan selected and its control policy become intrinsically linked, and the implementation of the control policy is the only element that needs to be considered.

The on-line planning and control process being discussed has the following assumptions.

- a) The simulation model for the real-world system exists and has passed the validation process.
- b) Off-line planning has been performed and the optimum set of design parameters for the operation of the system has been determined.

To explain the on-line planning and control process, the elements as given by Figure 2.1 are now discussed.

### *2.2.1 Real-world system operating under the current control policy*

In the top left-hand corner of Figure 2.1 is the block representing the real-world system operating under the current control policy. The real-world system receives both exogenous and endogenous input. The system has no control over the exogenous input (which is random), but the endogenous control input (that depends on the selected control policy) is forwarded from the real-time compromise analysis function. It also receives the system output from the system model operating under the current control policy as feedforward information. As output, it gives the current state of the real-world system to the different system models, as well as the state of the system as a function of time to the autovalidation process.

### *2.2.2 Autovalidation process*

The block beneath the real-world system depicts the autovalidation process. The system model had been validated before the on-line planning and control process started, but must be validated continuously during on-line planning and control. This is because the operating characteristics of

a real-world system are seldom stationary and change constantly with time. Typically, these changes are slow, but they can also be abrupt. The system model must be updated continuously to reflect these changes. The process receives the exogenous input, as well as the state of the system model as a function of time, as inputs. These projections influence the autovalidation process, because when the realised system performance is significantly outside the confidence interval for the projected response, the system model may no longer be valid and must be determined again. The autovalidation process compares the output projected by the model against the measured output from the system and updates the model to improve its accuracy. This updated system model is then forwarded to the system models operating under the current and alternative control policies. It has been determined that autovalidation is essential for on-line planning and control, but the technology to construct the autovalidation capability does not currently exist. Autovalidation is not necessary for off-line planning until the system is modified, because it does not consider the real-time operation of the system.

### *2.2.3 System model operating under the current control policy*

Below the block representing the autovalidation process is the representation of the system model operating under the current control policy. An on-line simulation of the implemented control policy must be conducted concurrently with the implementation of the control policy by the real-world system because the real-world system is stochastic. As the real-world system evolves over time, while operating under a selected control policy, it will realise only one of the potential-state trajectories that could occur. Thus, on-line simulation must be done to determine the statistics that characterise the future response of the real-world system given the current system state. Thus, two important needs of on-line simulation become apparent. The first is the need to initialise the on-line simulation to the current system state. The second is that on-line simulations must be executed significantly faster than real-time to provide the essential number of simulation trials to quantify future performance of the real system statistically. The system model receives as inputs the same exogenous input as the real-world system, system model updates from the autovalidation process and the updated control policies. As outputs, it provides the projected state of the system model as a function of time for the autovalidation process and statistical estimates of the future performance of the real system under the current control law with the following uses:

- a) The projections provide another input for the real system that can be viewed as feedforward information to be employed by the controller of the real system.

- b) The projected response of the system operating under the control policy can assist in the generation of alternative control policies for possible implementation.
- c) These projections provide a reference level of performance against which the predicted performance of the other potential control policies can be measured in the real-time compromise analysis function.

#### *2.2.4 System model operating under alternative control policies*

The blocks to the right of the representation of the system model operating under the current control policy block are the blocks depicting the system model operating under alternative control policies. Any number of additional possibilities of the system model can be included in the on-line planning and control process. Each of them uses on-line simulation to generate trials that are used by the real-time output analysis functions to compute statistical estimates of their future performance. These on-line simulations receive as input the current state of the real-world system, any update to the system model derived from the autovalidation process and any updated control policy alternatives from the alternative control policy generator. It provides system output to the real-time output analysis function. Again, these on-line simulations generate trials as quickly as possible to characterise the future performance of the real system statistically as it operates under the specific alternative control policy.

#### *2.2.5 Real-time output analysis functions*

Below the blocks representing the system models are the blocks representing the real-time output analysis functions. These take the system output from each on-line simulation trial for every system model and estimate the statistical performance of every system model. There is some correspondence between the on-line output analysis function and the off-line output analysis function, but there is a marked difference in the statistical techniques needed to implement the respective functions. While off-line analysis incorporates explicit measures to prevent the consideration of transient effects, on-line simulations focus solely upon the transient phenomena. The output of the analysis is estimates of the statistical performance of every system model and is passed to the real-time compromise analysis function.

The method given by Davis [3] will be used for the on-line statistical characterisation of the different performance criteria for each of the system models. It entails specifying performance criteria for the real-world model and evaluating each projected trajectory generated by the system models operating under different control policies for each of these performance criteria. This

method is neither complete nor proven, and is the best method available only because it is the only method available.

The major concern with this approach is that it initialises each new simulation trial to the most recent recorded state to employ all known information. This gives rise to statistical concerns. While each simulation trial is generated, the system continues to evolve and the initial conditions of consecutive trials are therefore not identical anymore. This invalidates any conventional statistical analysis, but Davis [3] states that this is better than losing the information that would be lost when the system state is frozen while the system actually continues to evolve. It is interesting to note that while Davis [3] proposes initialising each new simulation to the most recent state, his demonstration keeps his initial state constant. He reads the current state and then generates 10 simulation trials with their accompanying performance criteria for the next 100 jobs. Davis, *et al* [5] also describes an updating procedure that would allow predicted information to be retained while any known prediction errors could be removed. This may be necessary because it could be invalid to include any simulation trial that does not pass through a known state. This is, however, not the concern of this project and is still not proved to be worth the effort.

Another concern is the number of simulation trials that are used to perform the output analysis. Davis [3] solves it in a manner developed by Antonacci. It is discussed in detail in paragraph 2.7 on page 23.

#### 2.2.6 *The real-time compromise analysis function*

The block representing the real-time compromise analysis function in the bottom left-hand corner comprises of the actual comparison of the statistical performance estimates of the real-world system operating under the current and alternative control policies. These statistical performance estimates are generated by the real-time output analysis functions that determine it from the system models operating under the current and alternative control policies. This means that the real-time compromise analysis function is intrinsically linked to the real-time output analysis functions, because the technique used by the real-time compromise analysis function determines what statistical performance estimates need to be generated by the real-time output analysis functions. Therefore, while the focus of this project is the real-time compromise analysis function, the statistical performance estimates generated by the real-time output analysis functions are also specified.

The implementation of a given control policy can be viewed as a distinct system configuration and the purpose of this function is to select the system configuration to be implemented currently. A compromise solution is sought because it is assumed that multiple performance

criteria must be considered. Even though he gives some ideas (discussed in paragraph 2.3 on page 11), Davis [3] states that the procedures for the compromise analysis are not currently known. He cautions (Davis [2] and Davis, *et al* [5]) that it is easy to generate the data, but that the true test is in the ability of the decision-maker to assimilate the generated data and make a decision from it. He proposes that algorithms must be explored to assist the analyst or to automate the required analysis. This project proposes to develop some techniques that will perform this role.

If an alternative control policy is demonstrated to provide an improved performance over the control policy that is currently being used by the real-world system, the new control policy is transmitted to the real-world system for immediate implementation. It is also transmitted to the system model operating under the current control policy, which then begins to estimate the future performance of the system under the new control policy, and to the alternative control policy generator to provide the starting point for generating new control policy alternatives.

#### *2.2.7 Alternative control policy generator*

The final block is the block representing the alternative control policy generator. It receives the system output from the system model under the current control policy as well as the updated control policy from the real-time compromise analysis function. This information is used to produce updated control policy alternatives. Not much research has been done on the procedures for generating these alternative control policies.

#### *2.2.8 An example of the on-line planning and control process*

Davis, *et al* [6] has shown that the on-line planning and control process is viable by constructing a demonstration of a proactive on-line planning and control system. Even though he states that it will be possible to implement an operational on-line planning and control process for a real manufacturing cell within a few years, his demonstration still has the following simplifications:

- a) Their system did not include a real-time compromise analysis function, but relied on a human observer to decide upon the control policy to be used by interpreting the statistical performance criteria generated by the real-time output analysis.
- b) They used an emulator in the place of the real-world system even though they would have preferred a real-world system to show their modelling approach.
- c) They did not have an autovalidation process or an alternative control policy generator.

- d) The on-line simulation of the system models as well as the real-time output analysis are done on a separate computer to save time.

### 2.3 Aspects regarding current on-line simulation analysis

This section follows on the previous section's detailed discussion of the proactive on-line planning and control process by examining some aspects that were stumbled upon while studying the proactive on-line planning and control process. While these aspects enlighten certain subjects, they will not necessarily be used in the project.

#### 2.3.1 Graphical display

Davis, *et al* [5] states that a graphical display is a good starting point for real-time compromise analysis and gives a template that is also used in Davis [3] and Davis, *et al* [6]. The main part of the template comprises of an area where the set of  $W$  most recent performance criteria of the two performance criteria being compared are plotted against each other for different control policies. The rest of the graphical display is individual empirical cumulative density functions (e.c.d.f.s) for the different control policies for the two performance criteria considered. While this graphic interface would be helpful, a display is needed for every possible combination of any two criteria. They will also be changing constantly, so a human cannot monitor all of them continuously. The ideal would be an expert solution that will determine the best compromise and display the graphical views that influenced the decision.

#### 2.3.2 Correlation

The graphical display discussed in the previous paragraph enables the identification of correlation between the two performance criteria. If there is a positive covariance, there is a high likelihood of sampling a high value for the one performance criterion when sampling a high value of the other performance criterion and *vice versa*. Davis, *et al* [5] proposes that one of the criteria can then be dropped. However, it is unclear how to decide which is the one to be dropped and why it is necessary to drop one of the performance criteria. Would it not strengthen the utility function to have two performance criteria that promote the same alternative systems? These questions on positive correlation are currently unanswerable.

Negative correlation implies that a high value of one criterion would tend to generate a low value for the other. If maximising both criteria, a statistical trade-off between the criteria over which there would be no control, would exist.

To get an idea of the number of calculations necessary for correlation analysis, consider the following. For  $G$  criteria considered,  $G \times (G - 1) / 2$  pairwise correlations between each possible pair for every control policy are needed. Thus for  $P$  control policies,  $P \times G \times (G - 1) / 2$  pairwise correlations for each control policy must be investigated independently because performance criteria may be correlated under one control policy but not under another. In some articles, Davis discusses correlation (Davis, *et al* [5]), but in others (Davis [3] and Davis, *et al* [6]), he only shows that it is computed, but does not state the reason for it.

### 2.3.3 *Davis's technique*

Davis [3] proposes the development of a utility function for the real-time performance criteria by aggregating the individual performance criteria in order to provide a single, aggregate performance criterion that can be used for comparison. This utility function must then be evaluated for each simulation trial and a graphic generated to compare the utility function against each individual performance function that is considered within the utility function. He claims that only then can it be attempted to use the principle of stochastic dominance to assert the control policy that should be employed based on the statistics for the utility function. He does not explain why it is necessary to compare the utility function against each individual performance function.

There are limitations to the utility function approach. Some of these limitations are:

- a) The contributions of the individual criteria become obscured in the aggregation process.
- b) Holter, *et al* [10] mentions that the single rule can be ineffective if there are negatively correlated performance criteria, because then no rule can optimise all objectives simultaneously.
- c) Davis, *et al* [5] claims that there is no single static utility function. Their research has shown that for a given performance criteria, the sense of optimisation may actually change from a maximisation to a minimisation given the current state of the system. An example is that while the purpose usually is to maximise the process utilisation, in some cases control policies can change so that it requires less process utilisation for the same throughput, and then the ideal would be to minimise process utilisation.
- d) The definition of the appropriate utility function poses a problem. The statistics computed for each performance criterion under each alternative control strategy are time-variant. From this it follows that the statistical characterisations governing the compromise among the performance criteria are also time-variant. Hence in the on-line

planning and control scenario, it is virtually impossible to define a utility function on an *a priori* basis for the entire period in which on-line planning and control is to be addressed.

Davis then mentions that Whitt [15] states that stochastic entities can be compared by the notion of stochastic dominance. Let  $F_1(x)$  and  $F_2(x)$  represent the cumulative probability density function for a performance criterion or the utility function as computed for control policies 1 and 2 respectively. Then the principle of stochastic dominance states that for a maximisation, control policy 1 would dominate control policy 2 if it could be demonstrated that  $F_1(x) > F_2(x)$  for every  $x$ . For a minimisation, control policy 1 would dominate control policy 2 if it could be demonstrated that  $F_1(x) < F_2(x)$  for every  $x$ .

There are some limitations here as well. They are as follows:

- a) It is difficult to demonstrate that one entity is stochastically dominant to another because the requirements are very restrictive. The cumulative density function for the performance criterion arising from the operation of the system under one alternative control policy must be consistently greater than the cumulative density function for the same performance criterion operating under another control policy. To assert complete dominance in the on-line planning and control process, it must be shown that one control policy stochastically dominates all the others with respect to all performance criteria. This is very rarely possible and trade-offs must be considered for the development of a compromise solution.
- b) Even if a utility function could be defined, despite the problems discussed previously, it is unlikely that one control policy would dominate all the others stochastically.
- c) The demonstration of stochastic dominance is not a complete solution, because it provides no information about the extent to which one control policy dominates another with respect to a given performance criterion, nor does it guarantee that the dominating control policy will generate better performance criteria when implemented.

#### 2.3.4 *Dominance probability density function*

Davis [3] states that they are experimenting with new ways of computing dominance probabilities. This entails computing, in real-time, the probability that a control policy will provide an overall performance value that is  $\delta$  greater than the value generated by another control policy. This dominance probability function is calculated for all possible combinations of control policies for every value of  $\delta$ .

The limitations of the approach are:

- a) This operation is computationally intensive.  $P$  control alternatives and  $G$  performance criteria require a total of  $(G \times (P - 1) \times P) / 2$  dominance probability distributions in real-time for every  $\delta$ .
- b) The probabilities are dependent on the current state of the system and must be recomputed constantly.
- c) Even when these probabilistic dominance distributions are computed, it is unknown how to employ them efficiently to obtain the best compromise solution.

### 2.3.5 Other approaches

According to Davis [3], some other approaches have been considered in the operations research literature, but these do not extend to consider the stochastic performance criteria that are evaluated using simulation.

## 2.4 Aspects regarding off-line simulation analysis

There are similarities between on-line simulation analysis and off-line simulation analysis. Examples of these similarities include the need to make comparisons and the calculations of means, variances, etc. Aspects from off-line simulation analysis were studied to determine whether or not they could be used, or adapted for use, later on in the project for the on-line simulation analysis process. While these aspects enlighten certain subjects, they will not necessarily be used in the project.

### 2.4.1 Overview of techniques used for off-line simulation analysis

In order to see whether or not any of the techniques used for off-line simulation analysis could be adapted for on-line use, the techniques described by two acknowledged leaders in the simulation field were studied (Law, *et al* [12] and Banks [1]).

Law, *et al* [12] groups his techniques as follows:

- a) Comparing two systems.
- b) Comparing more than two systems. This includes comparisons with a standard, and pairwise comparisons.
- c) Ranking and selecting. This includes the selection of the best of  $P$  systems, selecting a subset of size  $M$  containing the best of  $P$  systems and selecting the  $M$  best of  $P$  systems.

Banks [1] groups his techniques as follows:

- d) Screening problems.
- e) Selecting the best.
- f) Comparisons with a standard.
- g) Comparisons with a default.
- h) Estimating functional relationships.

Intuitively the following looked promising:

- a) Comparing two systems from Law, *et al* [12].
- b) Comparisons with a standard from Law, *et al* [12].
- c) Pairwise comparisons from Law, *et al* [12].
- d) Selection of the best of  $P$  systems from Law, *et al* [12].
- e) Selecting the best from Banks [1].
- f) Comparisons with a default from Banks [1]

The techniques given by Law, *et al* [12] and Banks [1] contain adaptations to be valid for both terminating and steady state analysis, but in this case the emphasis is on terminating analysis because of the fixed planning horizon. Steady state analysis would make it impossible to pick up transient effects. Comparisons with a standard and pairwise comparisons from Law, *et al* [12] both have the problem that they only show which alternative systems differ from the standard, and not which single one is the best and should be implemented. In addition, the Bonferroni inequality (see the discussion in 2.4.3 on page 18) requires that the confidence levels of the individual alternative systems be increased to ensure that the overall confidence level is high enough. This would make confidence intervals quite wide for small sample sizes and make it difficult to show an improvement. The selection of the best of  $P$  systems from Law, *et al* [12] requires a two-stage sampling approach, which is not possible in on-line simulation analysis, that requires the sample size to be given.

Of the five techniques given by Banks [1] for selecting the best, three require a two-stage approach and the other require predetermined sample sizes. Banks [1] gives two techniques for comparisons with a default system. The first again only shows which alternative systems differ from the standard, and not which single one is best. The second (by Paulson) looks very

promising. It determines which single alternative system is best in a single stage, but unfortunately requires a predetermined sample size.

That only leaves us with comparing two systems from Law, *et al* [12], for which he describes two techniques. The first, a modified two-sample t confidence interval, does not pair up the observations of the two systems and does not require that the sample sizes of the two alternative systems be the same. It does not, however, allow the use of common random numbers (CRN) to reduce the variation between alternative systems. The use of common random numbers may prove to be of critical importance for variance reduction in on-line simulation, which only leaves us with the second possibility, a paired-t confidence interval technique. It allows common random numbers using single stage sampling and indicates whether the alternative system is an improvement on the default system or not. It seems suitable and is discussed briefly.

#### 2.4.2 *The paired-t confidence interval technique*

Of all the techniques studied in the previous paragraph only the paired-t confidence interval technique by Law, *et al* [12] looks promising for use in on-line simulation analysis. It takes the following approach to the comparison of two systems based on some performance criterion. A confidence interval is formed for the difference between the two expectations. This gives information as to whether the interval misses or contains zero respectively and thus results in a “reject” or “fail to reject” hypothesis test conclusion. It also quantifies how much the measures differ, if at all. Kelton, *et al* [11] only compares the two alternative systems by building confidence intervals of the expected differences and testing the null hypothesis that there is no difference between the two expectations.

The technique has the following assumptions:

- a) A normal-theory approach is followed.
- b) Observations from the same alternative system are independent and identically distributed (IID).
- c) The  $\bar{X}_{wgp}$  's are random variables defined over an entire replication.

It can be explained in the following manner. For  $p = 1$  and  $2$ , let  $\bar{X}_{1gp}, \bar{X}_{2gp} \dots \bar{X}_{wgp}$  be a sample of  $W$  IID observations of performance criterion  $g$  from system  $p$ , and let  $\mu_{gp} = E(\bar{X}_{wgp})$  be the expected response of interest. The objective is then to construct a confidence interval for  $\zeta_g = \mu_{g1} - \mu_{g2}$ . Whether or not  $\bar{X}_{wg1}$  and  $\bar{X}_{wg2}$  are independent depends on how the

simulations are executed. When common random numbers are used, the  $\bar{X}_{wg1}$  and  $\bar{X}_{wg2}$  are dependent and the paired-t confidence interval technique should be used and not the modified two-sample t confidence interval technique. The two systems need to have the same number of observations, so it may be necessary to discharge some of the observations of the system having more observations. The respective observations of the alternative systems ( $\bar{X}_{wg1}$  and  $\bar{X}_{wg2}$ ) are paired and their difference forms IID random variables ( $Z_{wg} = \bar{X}_{wg1} - \bar{X}_{wg2}$  for  $w = 1, 2, \dots, W$ ). This enables the formation of a confidence interval, called the paired-t confidence interval, for  $E(Z_{wg}) = \zeta_g$ .

The sample mean is given by:

$$\bar{Z}_g = \frac{\sum_{w=1}^W Z_{wg}}{W} \quad \text{Equation 2.1}$$

The estimated variance of the sample is given by:

$$s^2(\bar{Z}_g) = \frac{\sum_{w=1}^W [Z_{wg} - \bar{Z}_g]^2}{(W - 1)} \quad \text{Equation 2.2}$$

The approximate 100 (1- $\alpha_g$ ) percent confidence interval lower bound can be formed by:

$$LB = \bar{Z}_g - t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.3}$$

The approximate 100 (1- $\alpha_g$ ) percent confidence interval upper bound can be formed by:

$$UB = \bar{Z}_g + t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.4}$$

The confidence interval enclosed by the lower and upper bound is exact (i.e. it covers  $\zeta_g$  with probability 1- $\alpha_g$ ) if the difference random variables are normally distributed. Otherwise, the central limit theorem implies that the coverage will be nearly exact for a large number of observations.

The advantages of this technique are as follows:

- a) The normal-theory approach is simple and robust in this context, since the troublesome skewness in the underlying distributions of the output random variables should be

improved upon subtraction. That is assuming the two output distributions are skewed in the same direction.

- b) It is unnecessary to assume that the paired observations are independent. Allowing this dependence is important, because it leads to a reduction in the variance of the difference random variables and thus to a smaller confidence interval.
- c) Common random numbers can be used to induce this positive correlation between observations of different systems.
- d) It is unnecessary to assume that the variances of the paired observations are equal.

The disadvantages of this technique are as follows:

- a) The sample size is halved. This results in a loss of degrees of freedom, which in turn increases the confidence interval half width. Even though it results in a loss of degrees of freedom, the reduction in variance from CRN often offsets the loss, resulting in a tighter interval.
- b) Banks [1] explains when there are more than two systems, difficulties arise in extending the analysis above to all the  $D = (P \times (P - 1) / 2)$  differences  $\mu_{gp} - \mu_{g1}$  for all  $p \neq g$ . A standard approach is to form each confidence interval at level  $1 - \alpha_{gp} / D$ , rather than  $g - \alpha_{gp}$ , which guarantees that the overall confidence level for all  $D$  intervals is at least  $g - \alpha_g$  by the Bonferroni inequality (discussed in paragraph 2.4.3). Unfortunately, this procedure results in confidence levels that are so high when  $P$  is large that the confidence intervals may be too wide to detect differences in expected performance. Therefore, procedures that are more complex are required for comparing more than two systems at a time.

### 2.4.3 Bonferroni inequality

The paired-t confidence interval technique discussed above does not consider multiple performance criteria as would be needed for on-line simulation analysis. However, Law, *et al* [12] considers multiple performance criteria for the terminating analysis of a single system. If  $I_{gp}$  is a  $100 \times (1 - \alpha_{gp})$  percent confidence interval for the measure of performance  $g$ , for a terminating or nonterminating simulation,  $\mu_{gp}$  (where  $g = 1, 2, \dots, G$ ), then the probability that all  $G$  confidence intervals simultaneously contain their respective true measures satisfies

$$P(\mu_{gp} \in I_{gp} \text{ for all } g = 1, 2, \dots, G) \geq 1 - \sum_{g=1}^G \alpha_{gp} \tag{Equation 2.5}$$

whether or not the  $I_{gp}$ 's are independent. This result is known as the Bonferroni inequality and has serious implications for a simulation study. For example, suppose that 90 percent confidence intervals (i.e.  $\alpha_{gp} = 0.1$  for all  $g$ ) are constructed for 10 different measures of performance. Then the probability that each of the ten confidence intervals contains its true measure can only be claimed to be greater than or equal to zero.

A solution exists for this problem when the value of  $G$  is small (preferable smaller than 10). To ensure that the overall confidence level associated with  $G$  confidence intervals is at least  $100 \times (1 - \alpha_p)$  percent, choose the  $\alpha_{gp}$ 's so that  $\sum_{g=1}^G \alpha_{gp} = \alpha_p$ . The  $\alpha_{gp}$ 's do not have to be equal (the  $\alpha_{gp}$ 's corresponding to the more important measures could be chosen smaller). This enables the construction of ten 99 percent confidence intervals with the overall confidence level at least 90 percent. The difficulty with this solution is that the confidence intervals will be larger than originally if a fixed sample size procedure is used.

Banks [1] claims Bonferroni's inequality applies in very general circumstances. No conditions or restrictions are placed on the sample, parameters or the techniques of computing the intervals  $I_{gp}$ . Bonferroni's inequality thus not only applies to multiple performance criteria, but also to the comparison of multiple systems.

## 2.5 Aspects regarding variance reduction

When random variates from probability distributions are used as input for a stochastic simulation, they produce variance in the output. This variance is unfavourable because it leads to less precise results and wider confidence intervals. Thus, it would be beneficial to reduce the output variance. The most common way of reducing variance is to simulate more. This includes more replications for both terminating analysis and the truncated-replications approach to steady state analysis and making the single replication longer for the batch-means approach to steady state analysis. In the case of on-line simulation analysis, it is not possible to simulate more because the sample size depends on whether the system is in transient or steady state.

### 2.5.1 Overview of variance reduction techniques

In situations where it is not possible to simulate more, or when the sample size is determined beforehand, it is possible to reduce variance by controlling the randomness in the simulation experiment by controlling the random-number generator. This enables the induction of certain kinds of correlations that can be exploited to reduce the variance and thus increase the precision of the output. Kelton, *et al* [11] mentions four possible techniques:

- a) Antithetic variates.
- b) Control variates.
- c) Indirect estimation.
- d) Common random numbers (CRN).

The technique of antithetic variates attempts to induce negative correlation between the results of one replication and another, and uses this correlation to reduce variance. Within an antithetic pair, the negative correlation will cause the average of the two to snap towards the true expectation more closely than when the two are independent.

Control variates use an ancillary “controlling” random variable to adjust the results of the simulation up or down, as warranted by the control variates. In a given model, there are many potential control variates. There are different ways of selecting them and different ways of specifying the direction and magnitude of the adjustment to the simulation output.

Indirect estimation estimates something related to the required estimate and then transforms the estimate by a fixed formula. In essence, it is possible to replace some estimate that will have some variance in it with a known estimate that has no variance. While this idea of variance reduction seems quite intuitive, it also turns out to work in some not-so-obvious settings.

The three techniques are discussed in more detail in Law, *et al* [12]. Unfortunately, all three of them are only applicable to one model at a time. On-line simulation analysis requires the reduction of variance when comparing two different models. However, Kelton, *et al* [11]’s fourth technique, common random numbers, allows variance reduction across alternative systems and is discussed in the next paragraph.

### 2.5.2 Common random numbers (CRN)

A variance-reduction technique, called common random numbers (CRN), uses the same random numbers, synchronised in some way, across simulated alternative systems. Other names for this technique includes matched pairs, matched streams or correlated sampling.

To estimate the difference between alternative systems, it makes sense to simulate the alternative systems under conditions as similar as possible, so that the differences between the alternative systems’ results are because of the differences between them and not due to the random numbers behaving differently for the different alternative systems. This necessitates the synchronisation of the random numbers across the alternative systems as far as the model logic and differences between the alternative systems allow.

One possible approach is to dedicate a stream of random numbers to the different places where the variates are generated, but in complex models, it might be difficult to ensure that everything is matched up between alternative systems. In some models it may be possible to calculate how far apart the set of random numbers of the different streams must be chosen to ensure that there is no overlap of random-number usage within a stream across different replications. If the model allows it, determine how many random numbers will be needed for every replication and ensure that the spacing of the streams in the random number table is larger. Unfortunately, this method becomes less viable as the size and complexity of the model increases.

A method that may work better in some models, is to assign to the entity immediately upon arrival the attribute values for all the delays, decisions etc. that it may need on its entire path through the model. When the entity needs one of these values during its life in the model, it is read from the appropriate attribute and not generated on the spot. This could require a lot of computer memory if there are many attributes with many entities at the same time. It could also lead to a significant increase in execution time, making it more useful to do the more simulations one tried to avoid initially, but may still prove helpful if more simulation is not possible. To make CRN work one needs the output from the alternative systems to be positively correlated, the stronger the better. While it is possible to build examples where the correlation is negative, causing the CRN to backfire, they are not common. This method delivers good results in models that allow it.

## 2.6 Aspects regarding the Emulator

An emulator provides a means of evaluating any part of the on-line planning and control process. Davis [3] explains this by stating that their approaches and conclusions are empirical. Consequently, it is not possible to evaluate any proposed aspects mathematically. To determine whether the new aspects are viable or not, they must be implemented and evaluated. Davis, *et al* [6] has shown that it is possible to implement a distributed, on-line planning and control system with the World Wide Web and it is thus not necessary to implement a true on-line planning and control system to evaluate aspects. (This would be a drawback because these have their own problems emerging from the distributed nature of on-line planning and control.) An emulator of the on-line planning and control process would generate the same output as an on-line planning and control process, but could be implemented by using Arena<sup>®</sup> and Visual Basic<sup>®</sup> for Applications (VBA). See Sadowski [14] for an introduction to its uses. This project will use two different emulators.

The first, which is shown schematically in Figure 2.2, is needed because Visual Basic® does not allow for concurrent operations as a language like Java® would. This means that to emulate concurrent operations, a computer program (or emulator) has to be used to organise the sequential operations so that it is not possible to distinguish its output from the output of concurrent operations. In this case, the concurrent operations are the evaluation of the alternative systems with the alternative system models. Another advantage of using an emulator is that it is possible to run it significantly faster than real-time. This enables the generation of enough data to make significant statistical comparisons. This emulator will henceforth be called the Emulator.

The second is the emulator of the system being studied. This is a pilot project that is testing aspects and evaluating them to see whether they are viable or not. This means that it is not practical to implement the techniques on a real-world system (e.g. a FMS) and an Arena® model is used to emulate the real-world system. This emulator is integrated into the Emulator and is represented as the block named “Real-world system model” in Figure 2.2.

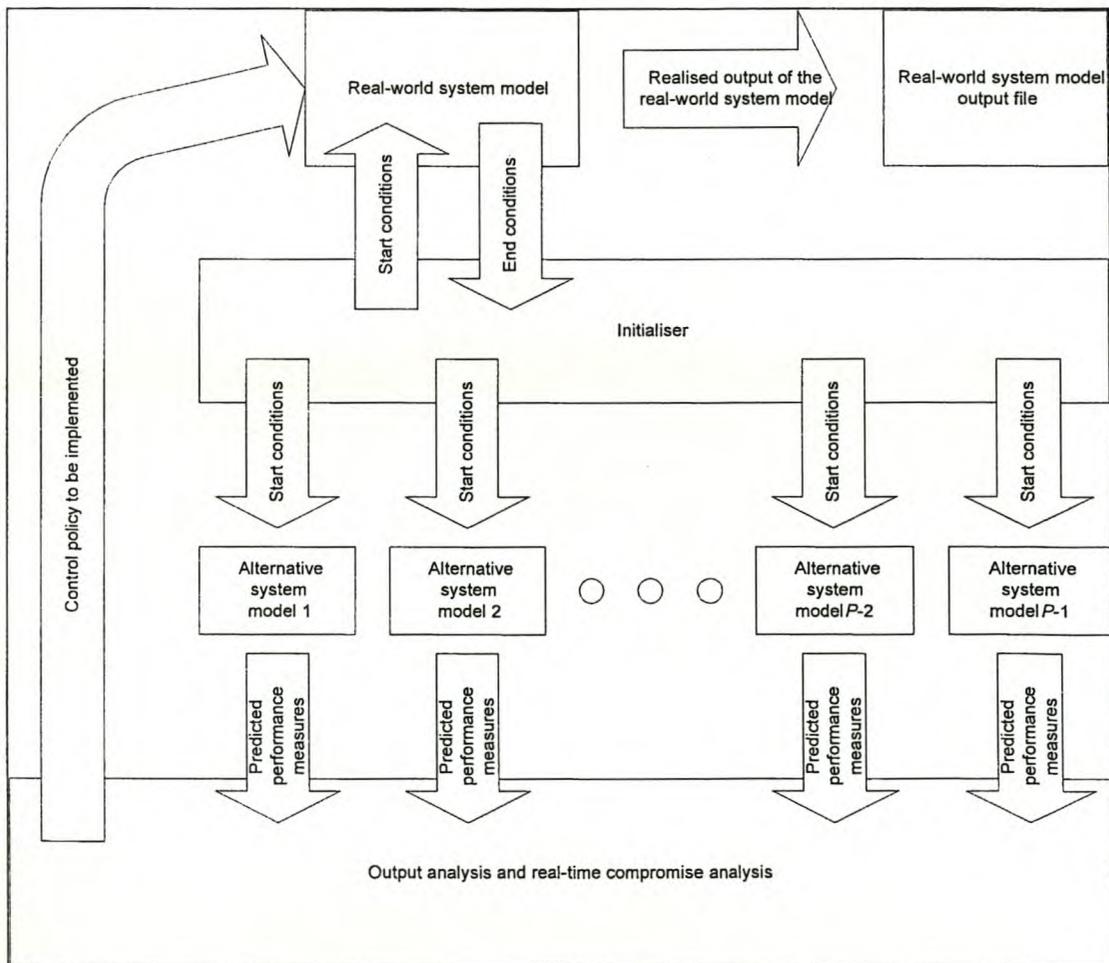


Figure 2.2 A schematic representation of the Emulator

## 2.7 Aspects regarding sample size

Sample size plays an important role when comparing values for performance criteria for alternative systems. Davis [2] explains that it is important to compromise between the better statistical confidence in larger sample sizes and the need to pick up transient behaviour. When the system is operating with nearly steady state dynamics, the desire is to increase the number of simulation trials employed in order to acquire more confidence in the statistical estimates. On the other hand, when system behaviour is extremely transient, many old trials may provide undesirable bias by utilising trials that were initialised to the starting state that are no longer relevant. This means that it is necessary to determine the optimal sample size before the start of the on-line simulation analysis.

### 2.7.1 Antonacci's method for determining the sample size

Davis [3] advocates a method developed by Antonacci, shown in Figure 2.3, to determine the sample size over which the specific performance criteria for two alternative systems are to be compared. This adaptive sample size algorithm method results in larger sample sizes if the system is operating in steady state for more confidence in the results, but for transient behaviour, it results in small sample sizes to pick up the transient behaviour. This is achieved by maintaining a fixed number of computed performance criteria from prior simulation trials in a pushdown stack. As a new trial provides a new projected performance value, the new value is added to the top of the stack and the oldest projected value is removed from the bottom of the stack. The algorithm begins by taking the 50, 100, 250, 500, 1 000 and then 2 000 most recent projected values for a given performance criterion and computes its sample mean and associated confidence interval. Using the properties of statistical estimates, the confidence interval at a given confidence level for the 50 most recent projections should be larger than that of the 100 most recent at the same confidence level if the system is operating in nearly steady state. The same should be the case for the 100 most recent samples, as opposed to the case of the 250 most recent samples, and so forth. However, if the system is operating in transient mode, the sample variances can increase to a point where the confidence interval is actually larger for the larger sample sizes. Hence the smallest sample size is used, for which the next larger sample size's confidence interval is not contained within the smaller sample size's confidence interval. This maximises the sample size, but reduces it when transients occur. When more than one performance criterion is considered, the smallest sample size determined for any performance criterion is used for estimating all the performance criteria.

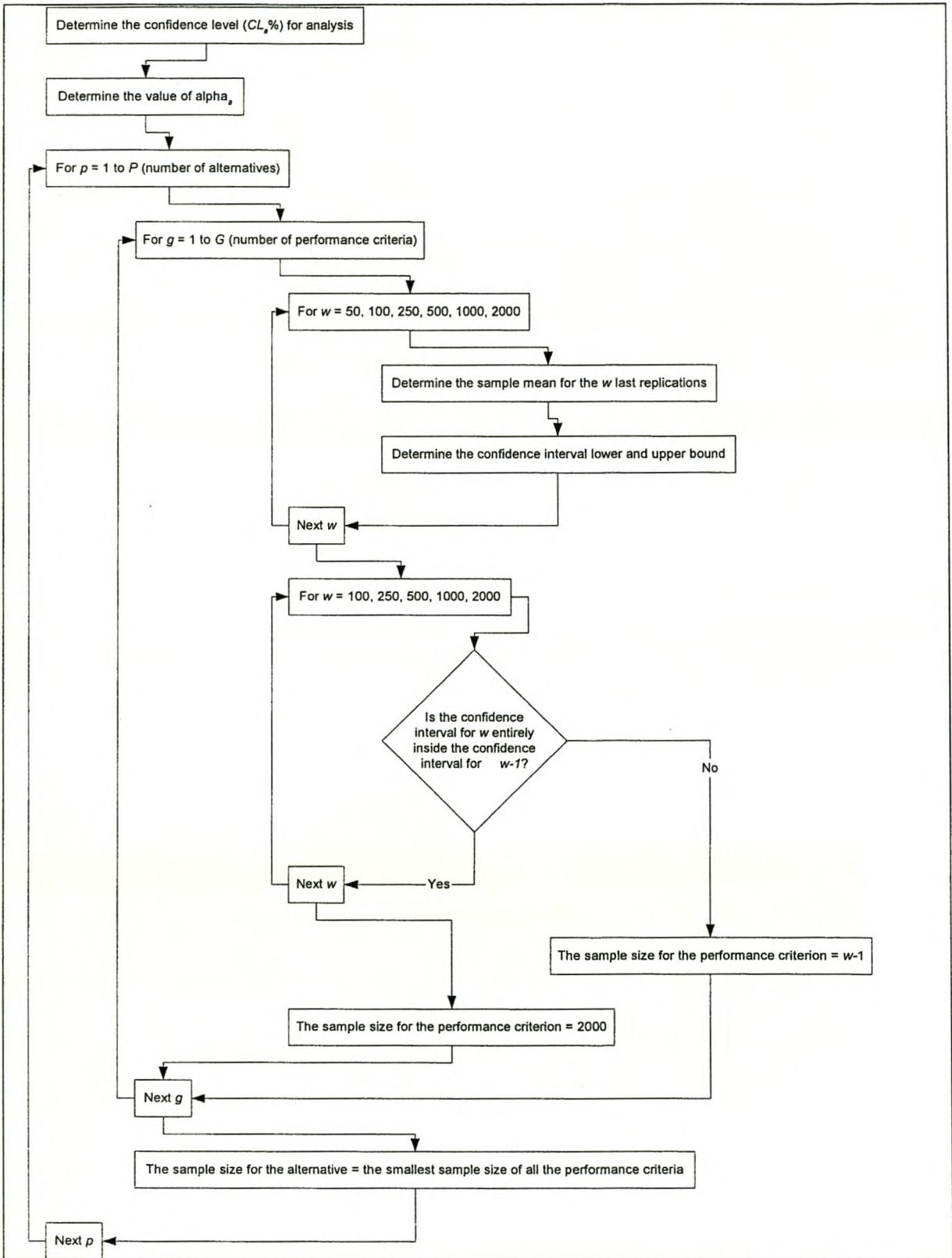


Figure 2.3 Schematic representation of Antonacci's method

### 2.7.2 Mathematical formulation of Antonacci's method

Figure 2.3 shows that the method starts with the choice of a confidence level ( $CL_a\%$ ) to be used for the analysis. This percentage value is usually taken as 95%. The value of alpha ( $\alpha_a$ ) is then determined from the confidence level by the following equation:

$$\alpha_a = 1 - \left( \frac{CL_a}{100} \right) \quad \text{Equation 2.6}$$

For the  $G$  performance criteria of the  $P$  alternatives, the upper and lower bound of the confidence interval are determined for the  $w$  most recent replications in the stack. Antonacci proposed the values 50, 100, 250, 500, 1 000 and 2 000 for  $w$ . The central limit theorem states that if  $w$  is large enough, these performance criteria values will be approximately normally distributed. That is, random variable  $\bar{X}_{wgp}$  is normally distributed with mean:

$$\bar{\bar{X}}_{gp} = \frac{\sum_{w=1}^W \bar{X}_{wgp}}{W} \quad \text{Equation 2.7}$$

and variance  $\sigma^2$ . The unbiased estimator of the population variance is given by:

$$s^2(\bar{X}_{wgp}) = \frac{\sum_{w=1}^w (\bar{X}_{wgp} - \bar{\bar{X}}_{gp})^2}{W - 1} \quad \text{Equation 2.8}$$

When studying performance criteria values other than the mean, for instance the total number of entities serviced, the performance criteria values are no longer averages. They are the sum of a random number of occurrences within a replication. This sum is also approximately normally distributed and the equations are similar to the above. However, for proportions, percentiles, minimums and maximums this is not the case, and an entirely different approach has to be followed.

The confidence interval is an estimator of the performance criteria value that is more descriptive than a point estimator. It is an interval estimator and specifies the range in which the true mean of the specific performance criteria is likely to be expected. The interpretation of the confidence interval is as follows: if 100 confidence intervals are constructed for a given performance criterion, then  $(1-\alpha_a) \times 100$  of these intervals should include the true mean of the performance criterion. In this case, the confidence interval is computed using the t-distribution, because the population variance is not known and needs to be estimated from the sample by  $s^2(\bar{X}_{wgp})$ . The t-distribution is a small sample distribution, making it suitable for changing sample sizes. The

unbiased estimator of the population variance is used to compute the upper and lower bounds of the confidence interval.

The lower bound is given by:

$$LB = \bar{\bar{X}}_{gp} - t_{n-1, 1-\alpha/2} \sqrt{\frac{s(\bar{X}_{wgp})^2}{W}} \quad \text{Equation 2.9}$$

and the upper bound by:

$$UB = \bar{\bar{X}}_{gp} + t_{n-1, 1-\alpha/2} \sqrt{\frac{s(\bar{X}_{wgp})^2}{W}} \quad \text{Equation 2.10}$$

Starting at  $w = 100$ , and for all intervals up to 2 000, determine whether the confidence interval falls entirely inside the previous smaller sample size's confidence interval or not. If it does not, the sample size of the previous smaller sample size is used as the sample size for this specific performance criterion. If all the confidence intervals up to and including  $w = 2\,000$  lie within the next smaller sample size's confidence level, steady state operation can be assumed and a sample size of 2 000 is used. This value of 2 000 may seem arbitrary, and it is. Davis [3] gives no reason for the use of 2 000 as the upper limit or for any value of  $w$  used. The sample sizes for all the different performance criteria of a specific alternative system are determined and the smallest sample size is used for all the performance criteria.

The interested reader is referred to Appendix A for a implementation of and sample calculations for Antonacci's method.

### 2.7.3 *The sample size used in the Emulator*

The initial idea was to use Antonacci's method to determine the sample size to be used for the on-line simulation analysis. However, after Antonacci's method was implemented in the Emulator, it was noted that it always proposed a sample size of 50. The reason for this can be explained as follows: to enable the evaluation of the techniques developed, the Emulator's settings was set to propagate switching between the different control policies. This was done by maximising performance criteria that would intuitively be minimised and *vice versa*. The continuous switching between control policies kept the Emulator in a transient state and the transient state resulted in a sample size of 50.

In a real life application where it would be preferable to keep the system in the steady state as long as possible, Antonacci's method may prove to be very useful, but in this case it was desired to set the sample size to a predetermined value to gain more confidence in the results. Thus, to

have some confidence in the results, without losing too much of the transient behaviour, this value was chosen as 500. It is a ten times larger than 50, but still not near 2 000, which is supposed to indicate steady state. When Davis demonstrates Web-based simulation (Davis, *et al* [6]), he uses a fixed sample size (1 000) as well.

It would be a disadvantage if the starting control policy had a large influence on the results. To remedy this, the Emulator is set to start switching as soon as possible. Consequently, when the Emulator arrives at the predetermined required sample size, it is already at the optimal control policy for that state. When the data is analysed, all observations made before the predetermined sample size has been reached are truncated because they were not made under the required sample size. Therefore, while a sample size is specified, during the operation of the Emulator it is increased incrementally. At the beginning, the Emulator waits until it has 50 observations for all alternative systems and then uses those 50 observations to determine whether the control policy should switch or not. When all the alternative systems have 100 observations, it will use the last 100 observations to determine whether the control policy should switch or not and so forth until the required sample size is reached.

## 2.8 Conclusions

The literature study has enabled the formulation of the conclusions shown in Table 2.1. The paragraph where it is discussed and the paragraph's page are also shown. These are not the only conclusions possible, but merely the most revealing.

Table 2.1 The conclusions from the literature study

Conclusion	Paragraph	Page
Techniques are needed to automate the real-time compromise analysis function.	2.2.6	9
The real-time compromise analysis function is intrinsically linked to the real-time output analysis functions.	2.2.6	9
A utility function can aggregate the individual performance criteria in order to provide a single, aggregate performance criterion that can be used for comparison.	2.3.3	12
The paired-t confidence interval technique looks promising for use in on-line simulation analysis.	2.4.2	16
If the systems allow it, common random numbers (CRN) can be used to reduce variation when comparing two systems.	2.5.2	20
An emulator is required to evaluate parts of the on-line planning and control process.	2.6	21
To have some confidence in the results, without losing too much of the transient behaviour, the sample size can be set to a predetermined value.	2.7.3	26

## 2.9 Summary

The literature study started with the description of proactive on-line planning and control, emphasizing where this project fits into the overall picture. Following this, aspects regarding the current on-line and off-line simulation analysis techniques were discussed. Then variance reduction, emulations and sample size were discussed in detail. The literature study ended with the conclusions from the literature study.

### 3 THE DEVELOPMENT OF THE TECHNIQUES TO SELECT A CONTROL POLICY

The real-time compromise analysis function that does the actual comparison of the statistical performance estimates of the real-world system operating under the current and alternative control policies was described in paragraph 2.2.6 on page 9. The purpose of this function was determined as the selection of the control policy to be implemented currently in the real-world system model. The development of techniques to decide on the control policy to be implemented is a critical need. When Davis, *et al* [6] implemented an on-line planning and control process (described in paragraph 2.2.8 on page 10), their system did not include a real-time compromise analysis function, because the techniques needed to execute the function did not exist. They relied upon a human observer to decide upon the control policy to be implemented by interpreting the statistical performance criteria generated by the real-time output analysis. To eliminate the need for a human observer, techniques need to be developed to automate the real-time compromise analysis function.

The link between the real-time output analysis functions and the real-time compromise analysis function was shown in paragraph 2.2.6 on page 9. The technique used by the real-time compromise analysis function determines what statistical performance estimates need to be generated by the real-time output analysis functions. Therefore, while the technique is developed for the real-time compromise analysis function, it includes parts that are components of the real-time output analysis functions. The parts that are actually part of the real-time output analysis functions include the determination of the sample means and variances, etc.

In paragraph 2.3.3 on page 12 the need for a utility function that can aggregate the individual performance criteria to provide a single, aggregate performance criterion was highlighted. Thus, the techniques need to compare the corresponding performance criteria of the different performance criteria and then aggregate the results of the comparison into a single compromise function. This compromise function is used for the final comparison to determine the control policy to be implemented currently in the real-world system model.

On-line simulation used during the on-line planning and control process has properties that necessitate special assumptions when analysing its results as done by the techniques that were developed. The properties of on-line simulation and the special assumptions they require are given in Table 3.1.

This chapter discusses two techniques that were developed to automate the real-time compromise analysis function. The chapter introduces the techniques, follows with a detailed

mathematical formulation discussion and finishes with a schematic implementation with sample calculations for both techniques.

Table 3.1 The properties of on-line simulation and their special assumptions

Property	Assumption
On-line simulation continuously initialises the system model to the current state of the real-world system whenever its previous trial is finished. This leads to different initial conditions for every replication.	It is assumed that this does not invalidate the independence assumption.
The real-world system's control policy changes with time, resulting in consecutive trials of the system models not being initialised to the same initial conditions.	It is assumed that this does not invalidate the assumption of IID replications.
The autovalidation process may change the system model resulting in the output of consecutive trials of the system models not being identically distributed.	It is assumed that this does not invalidate the assumption of IID replications.

### 3.1 The First technique

The first developed technique is based on the paired-t confidence interval technique discussed in paragraph 2.4.2 on page 16. The paired-t confidence interval technique is a common technique for comparing two systems statistically.

In this case the paired-t confidence interval technique is used to compare the alternative system with the current system by building confidence intervals of the expected differences for the respective performance criteria and testing the hypothesis that the alternative system's expectations of the performance criteria are better than those of the current system. The results of the comparisons made with the paired-t confidence interval technique are then consolidated into a compromise function that is used to determine the control policy to be implemented currently in the real-world system model. The following mathematical formulation and discussion represent only one evaluation, but the technique will be executed every time the system models

complete a new replication of any alternative system. An algorithm describing the first technique is shown in Figure 3.1.

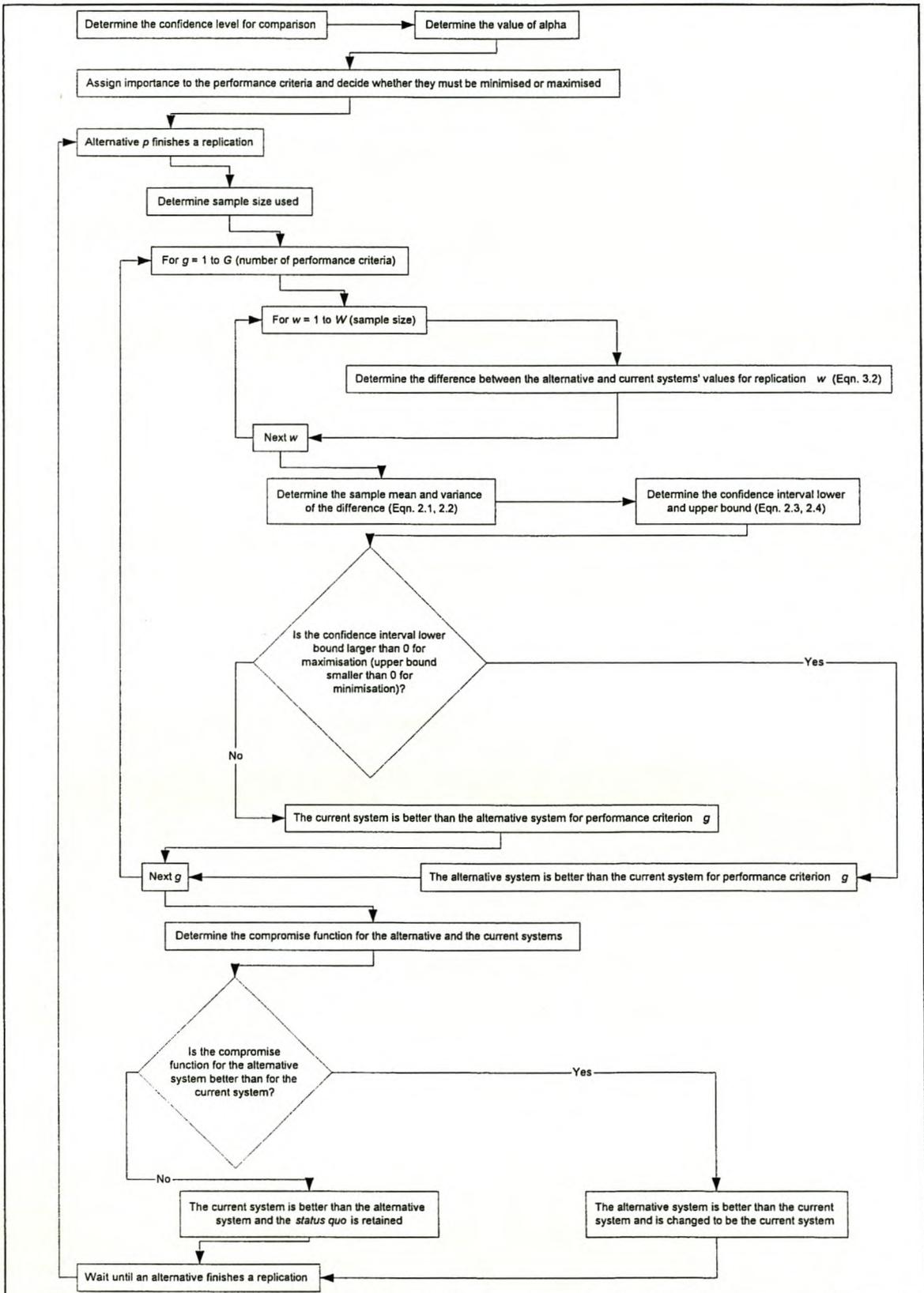


Figure 3.1 Schematic representation of the first developed technique

### 3.1.1 Mathematical formulation of the First technique

The technique for comparing alternatives, as illustrated in Figure 3.1, starts with the determination of the confidence level ( $CL_g\%$ ) for the comparison. This is a user-assigned value and is usually chosen as 95%. The effect of the confidence level on the efficiency of the technique is a subject that requires further study. Many different confidence levels need to be evaluated for the same conditions to determine what effect a change of confidence level has on the technique. For the purpose of this project, the confidence level will be kept constant at 95%. The confidence level is then used to determine alpha ( $\alpha_c$ ) with the following equation:

$$\alpha_c = 1 - \left( \frac{CL_g}{100} \right) \quad \text{Equation 3.1}$$

The operator has control over the technique by assigning importances to the  $G$  different performance criteria. The  $G$  different performance criteria are each assigned a value  $v_g$  between and including one and ten, depending on their importance to the system (higher values correspond to more important performance criteria). More than one performance criterion can be assigned the same value, should they have the same importance. It is also specified whether the performance criterion should be minimised or maximised. In a true on-line planning and control environment, the importances of the performance criteria and whether the performance criteria should be minimised or maximised will be changeable during the operation of the real-world system. This will allow the operator to adjust the real-world system to major changes in the operating requirements, while the technique will adapt to the other requirements. This can be explained as follows: a warehouse may become temporarily unavailable. The performance criterion referring to the minimisation of the inventory in that warehouse can be changed to be a great deal more important than the other performance criteria, leading to the virtual elimination of inventory in that warehouse.

Figure 3.1 shows that whenever one of the  $P$  system models has completed a new replication, the alternative system  $p$  needs to be compared with the current system. The way the technique is implemented makes no distinction between the current system and the  $P-1$  alternative systems. Consequently, when the current system has completed a new replication, it is also compared with itself, but the results of the comparison are ignored. It is recommended that this inefficiency be eliminated in further studies to save computing time.

The next step is the determination of the sample size to be used for the comparison. In this case, the incremental increasing of the sample size up to a fixed sample size procedure, discussed in paragraph 2.7.3 on page 26, is used, but the effect of the sample size on the calculations is a

subject that requires further study. Possible cases that should be implemented and their results examined are different fixed sample sizes, Antonacci's method and even newly developed methods.

This project will only focus on performance criteria values that are estimating means, e.g. average time in system, average queue length, etc. When studying performance criteria values other than means, for instance the total number of entities serviced, the performance criteria values are no longer averages. They are the sum of a random number of occurrences within a replication. This sum is also approximately normally distributed and the equations are similar to the ones shown. However, for proportions, percentiles, minimums and maximums this is not the case and an entirely different approach has to be followed. The approach for proportions, percentiles, minimums and maximums is beyond the scope of this project, but it requires further study.

The  $G$  performance criteria used to compare the alternative system with the current system are all evaluated sequentially. The first step is to determine the difference between the alternative system's and current system's respective values for the  $W$  (sample size determined in previous step) most recent samples. The rest of the technique is dependent on the order of the subtraction, so it is important to deduct the current system's value from the alternative system's value and not *vice versa*. Thus, for every performance criterion  $g$  the difference

$$Z_{wg} = \bar{X}_{wg1} - \bar{X}_{wg2} \tag{Equation 3.2}$$

between the alternative and current systems' values ( $\bar{X}_{wg1}$  and  $\bar{X}_{wg2}$ ) for a specific replication  $w$  is computed for the last  $W$  replications.

These differences form a new sample, also with sample size  $W$ . When discussing the paired-t confidence interval technique in paragraph 2.4.2 on page 16, it was shown that the confidence interval lower and upper bound are determined from the sample mean and variance. The sample mean of the new sample is given by:

$$\bar{Z}_g = \frac{\sum_{w=1}^W Z_{wg}}{W} \tag{Equation 2.1}$$

and the sample variance of the new sample by:

$$s^2(\bar{Z}_g) = \frac{\sum_{w=1}^W [Z_{wg} - \bar{Z}_g]^2}{(W - 1)} \tag{Equation 2.2}$$

This enables the formation of the (approximate)  $100(1-\alpha_c)$  percent lower bound of the confidence interval:

$$LB = \bar{Z}_g - t_{n-1, 1-\alpha_c/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.3}$$

as well as the (approximate)  $100(1-\alpha_c)$  percent confidence interval upper bound of the confidence interval:

$$UB = \bar{Z}_g + t_{n-1, 1-\alpha_c/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.4}$$

These upper and lower bounds are used to test the hypothesis that the alternative system's expectations are better than those of the current system. The hypothesis is formulated as follows:

$H_0$ : The alternative system's performance criterion  $g$  is better than that of the current system.

$H_1$ : The alternative system's performance criterion  $g$  is not better than that of the current system.

That is, if the performance criterion needs to be maximised, the hypothesis is rejected if the confidence interval lower bound is equal to or less than zero, and it can be assumed with confidence equal to  $CL_g\%$  that the current system's expectations are better or at least the same than the alternative system's for the specific performance criterion  $g$ . However, if the confidence interval lower bound is larger than zero, there are no grounds to reject the hypothesis and can it be assumed with confidence equal to  $CL_g\%$  that the alternative system's expectations are better than those of the current system for the specific performance criterion  $g$ .

If the performance criterion needs to be minimised, the hypothesis is rejected if the confidence interval upper bound is equal to or larger than zero, and it can be assumed with confidence equal to  $CL_g\%$  that the current system's expectations are better (or at least the same) than those of the alternative system for the specific performance criterion  $g$ . However, if the confidence interval upper bound is smaller than zero, there are no grounds to reject the hypothesis and can it be assumed with confidence equal to  $CL_g\%$  that the alternative system's expectations are better than those of the current system for the specific performance criterion  $g$ .

The case where the lower or upper bound is equal to zero requires attention. In the minimisation case, if the confidence interval lower bound is equal to zero, it is assumed with confidence equal to  $CL_g\%$  that the current system's expectations are better than the alternative system's for the specific performance criterion  $g$ . In the maximisation case, if the confidence interval upper bound is equal to zero, it is assumed with confidence equal to  $CL_g\%$  that the current system's

expectations are better than the alternative system's for the specific performance criterion  $g$ . This is necessary because it is beneficial to maintain the current control policy in the real-world system, because it is desired to minimise the number of control policy changes in the real-world system.

The decision to swap is also dependent on the variance, because the confidence intervals are sensitive to the sample variance that differs from sample to sample.

After the  $G$  performance criteria have been compared, the compromise function needs to be determined. The compromise functions are defined for the alternative system ( $CF_p$ ) and the current system ( $CF_c$ ) with the following equations:

$$CF_p = \sum_{l=1}^l (H_p \times v_l) \quad \text{Equation 3.3}$$

with  $H_p = 0$  if the hypothesis is rejected and  $H_p = 1$  if the hypothesis cannot be rejected, and

$$CF_c = \sum_{g=1}^g (H_c \times v_g) \quad \text{Equation 3.4}$$

with  $H_c = 1$  if the hypothesis is rejected and  $H_c = 0$  if the hypothesis cannot be rejected.

The definition of the compromise functions requires further study. Different compromise functions need to be implemented and tested. A possible approach that warrants further study is to multiply all the  $v_g$ 's that are not equal to 0 to determine  $CF_p$  and  $CF_c$ , respectively.

The final step is to compare  $CF_p$  and  $CF_c$  to determine whether the alternative system's expectations are better than those of the current system or not. If  $CF_p$  is larger than  $CF_c$ , it is assumed that the alternative system is better and it is swapped with the current system in the real-world system. If  $CF_p$  is smaller or equal to  $CF_c$ , we assume that the current system is still the best and retain the *status quo*.

The entire process, from the determination of the sample size to be used for the comparison to the determination of whether the alternative system's expectations are better than those of the current system or not, is repeated whenever an alternative finishes a replication.

### 3.1.2 Schematic implementation with sample calculations for the First technique

To illustrate the First technique, a complete schematic implementation with sample calculations is shown in this paragraph. Figure 3.2 shows an overview of the first developed technique, with the parts corresponding to the technique coloured in. The darker colour corresponds to information supplied by the user or to results from the replications of alternative systems, while the lighter

colour indicates cells that are calculated. Figure 3.3 shows the different parts, with their annotations, that will be discussed as units in the following sections. When the first digit of the annotation is a letter, it refers to information supplied by the user or to results from the replications of alternative systems, while a number refers to calculations. Annotations with the same prefixes are nearly similar, differing only with the digit that is not the same.

First technique				th most recent trial					th most recent trial					Additional performance criteria				
				th most recent trial	Trial number	Mean for planning horizon J	Alternative-Current	Statistical analysis of A-C			th most recent trial	Trial number	Mean for planning horizon J	Alternative-Current	Statistical analysis of A-C			
11	Given input from "Start off":			Performance criterion 1	1	5433	7.099	-0.463	Sample mean=	w=	50	1	5433	3.999	3.936	w=	50	
12	Confidence level (%) =	95		Sample mean=	2	5438	6.836	5.526	Sample mean=	2.22		2	5438	3.424	4.390	Sample mean=	1.96	
13	Importance (1-10)			Sample variance=	3	5427	7.969	8.437	Sample variance=	97.04		3	5439	2.494	0.254	Sample variance=	88.45	
14	Performance criterion 1=	8	MAX	Confidence interval LB=	4	5436	9.341	13.771	Confidence interval LB=	-0.58		4	5438	1.145	6.162	Confidence interval LB=	-0.39	
15	Performance criterion 2=	3	MAX	Confidence interval UB=	5	5425	8.195	-4.677	Confidence interval UB=	5.02		5	5429	2.433	2.028	Confidence interval UB=	4.31	
16	Performance criterion 3=	4	MIN	Alternative better than current?	6	5434	6.708	10.975	w=	100		6	5434	4.252	-3.366	w=	100	
17	Performance criterion 4=	5	MAX	Alternative points=	7	5433	2.224	-10.217	Sample mean=	3.44		7	5433	2.247	-11.561	Sample mean=	1.48	
18	Performance criterion 5=	7	MIN	Current points=	8	5432	1.099	-6.122	Sample variance=	84.96		8	5432	2.933	-10.491	Sample variance=	94.62	
19				Performance criterion 2	9	5431	5.777	-3.644	Confidence interval LB=	1.62		9	5431	1.464	1.530	Confidence interval LB=	-0.11	
20	Calculated			Sample mean=	10	5430	2.667	-9.796	Confidence interval UB=	5.27		10	5430	3.924	-3.012	Confidence interval UB=	3.08	
21				Sample variance=	11	5429	4.999	-15.681	w=	250		11	5429	2.912	-8.699	w=	250	
22				Confidence interval LB=	12	5429	4.162	-10.668	Sample mean=	3.08		12	5428	3.066	12.393	Sample mean=	2.93	
23				Confidence interval UB=	13	5427	2.343	3.994	Sample variance=	79.69		13	5427	3.236	-1.706	Sample variance=	54.84	
24				Alternative better than current?	14	5426	2.299	-2.629	Confidence interval LB=	1.97		14	5426	1.093	4.644	Confidence interval LB=	2.01	
25				Alternative points=	15	5425	8.994	4.948	Confidence interval UB=	4.19		15	5425	2.969	-8.357	Confidence interval UB=	3.85	
26	Sample size used=	50		Current points=	16	5424	5.069	-9.463	w=	500		16	5424	2.519	-2.472	w=	500	
27	Output			Performance criterion 3	17	5423	1.099	0.491	Sample mean=	3.07		17	5423	3.409	5.968	Sample mean=	2.99	
28	Compromise function total			Sample mean=	18	5422	2.996	-10.463	Sample variance=	75.67		18	5422	3.499	4.061	Sample variance=	94.57	
29	Alternative=	7.00		Sample variance=	19	5421	0.844	-15.374	Confidence interval LB=	2.31		19	5421	3.620	5.945	Confidence interval LB=	2.34	
30	Current=	20.00		Confidence interval LB=	20	5420	4.573	2.743	Confidence interval UB=	3.83		20	5420	3.676	25.960	Confidence interval UB=	3.64	
31	Alternative better than current?	NO		Confidence interval UB=	21	5419	3.672	-1.337	w=	1000		21	5419	3.669	-11.374	w=	1000	
32				Alternative better than current?	22	5418	4.423	-9.698	Sample mean=	3.99		22	5418	3.894	3.887	Sample mean=	3.17	
33				Alternative points=	23	5417	6.822	6.174	Sample variance=	74.73		23	5417	3.610	-8.286	Sample variance=	52.72	
34				Current points=	24	5416	7.989	-7.327	Confidence interval LB=	-3.05		24	5416	3.800	-3.261	Confidence interval LB=	2.72	
35				Performance criterion 4	25	5415	6.176	10.424	Confidence interval UB=	4.12		25	5415	3.228	4.867	Confidence interval UB=	3.62	
36				Sample mean=	26	5414	8.750	4.137	w=	2000		26	5414	3.406	-7.014	w=	2000	
37				Sample variance=	27	5413	6.166	-8.137	Sample mean=	3.86		27	5413	4.002	-6.147	Sample mean=	3.22	
38				Confidence interval LB=	28	5412	4.700	-8.876	Sample variance=	77.54		28	5412	4.709	10.059	Sample variance=	51.20	
39				Confidence interval UB=	29	5411	6.536	7.341	Confidence interval LB=	3.48		29	5411	2.926	-2.042	Confidence interval LB=	2.90	
40				Alternative better than current?	30	5410	6.911	0.443	Confidence interval UB=	4.25		30	5410	3.913	8.060	Confidence interval UB=	3.53	
41				Alternative points=	31	5409	5.709	19.014				31	5409	3.736	-14.778			
42				Current points=	32	5408	6.406	6.726				32	5408	3.730	-5.832			
43				Performance criterion 5	33	5407	4.123	-3.880				33	5407	3.473	-0.412			
44				Sample mean=	34	5406	6.677	15.719				34	5406	2.814	2.009			
45				Sample variance=	35	5406	7.292	5.713				35	5406	1.999	1.503			
46				Confidence interval LB=	36	5404	7.993	13.637				36	5404	4.069	13.840			
47				Confidence interval UB=	37	5403	4.398	22.433				37	5403	3.861	3.396			
48				Alternative better than current?	38	5402	5.938	-7.038				38	5402	4.391	-1.031			
49				Alternative points=	39	5401	6.124	20.936				39	5401	4.621	4.052			
50				Current points=	40	5400	6.296	9.700				40	5400	1.191	-2.773			
51					41	5399	6.394	15.677				41	5399	4.674	13.905			
52					42	5398	9.273	10.788				42	5398	1.693	8.189			
53					43	5397	1.973	2.406				43	5397	4.802	12.200			
54					44	5396	3.388	5.962				44	5396	2.708	4.614			

Figure 3.2 The First technique implemented

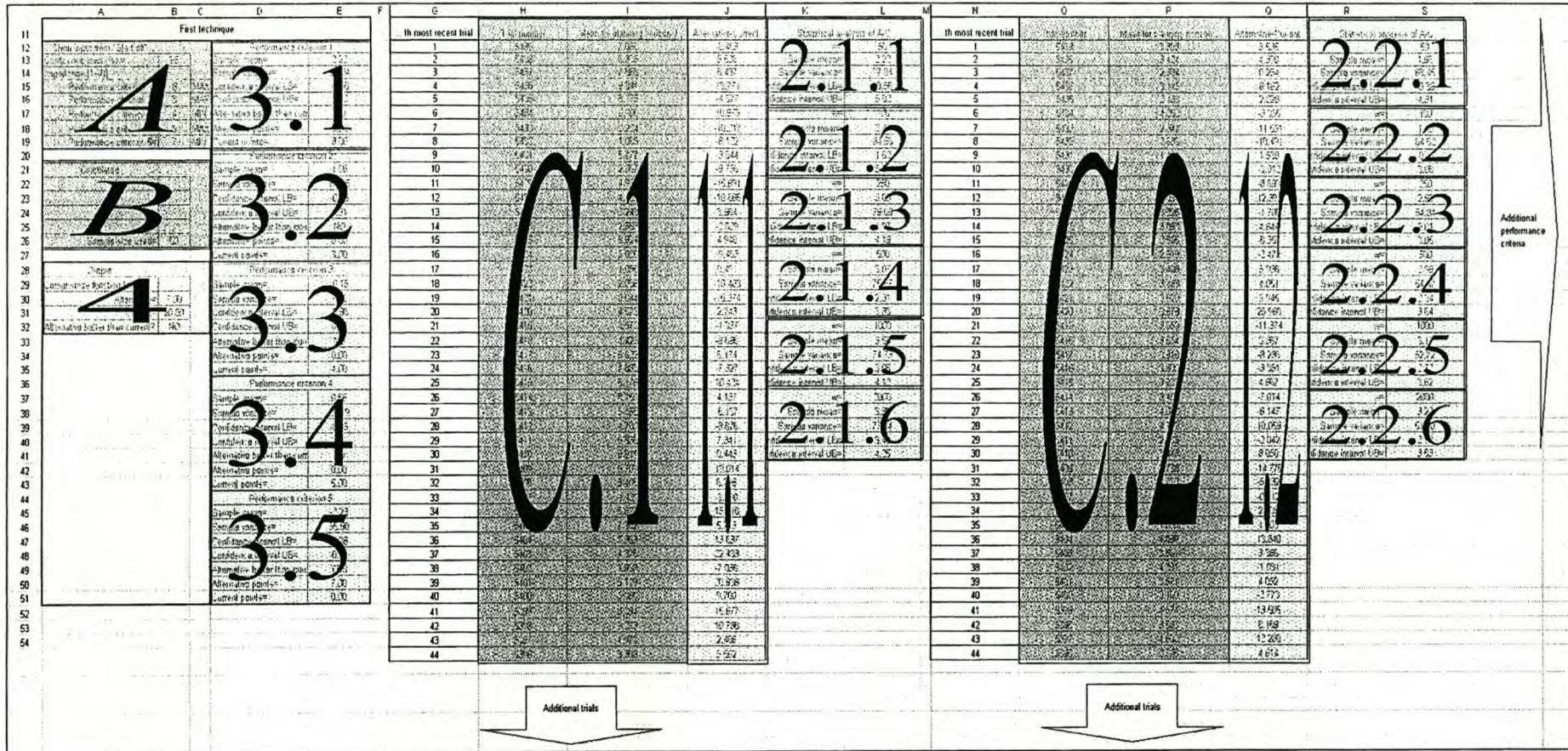


Figure 3.3 The different parts of the First technique

	A	B	C
13	Confidence level (%) =	95	
14	Importance [1-10]		
15	Performance criterion 1 =	8	MAX
16	Performance criterion 2 =	3	MAX
17	Performance criterion 3 =	4	MIN
18	Performance criterion 4 =	5	MAX
19	Performance criterion 5 =	7	MIN

Figure 3.4 Part A of the First technique

Part A, shown in Figure 3.4, comprises of the information supplied by the user. This includes the confidence level for the comparison, the importance of the different performance criteria ( $v_g$ ) and whether the performance criteria should be maximised or minimised. Cell B13 gives the confidence level, while cells B15: B19 gives the relevant importance of the performance criteria (on a scale of 1 to 10 with 1 the least important). Cells C15: C19 indicates whether the performance criteria should be minimised or maximised.

	A	B
21	Calculated:	
22		
23		
24		
25		
26	Sample size used =	50

Figure 3.5 Part B of the First technique

Part B, shown in Figure 3.5, comprises of the fixed sample size discussed in paragraph 2.7.3 on page 26. In this case, the sample size was set to 50 to make the sample calculations easier. Cell B26 gives the sample size to be used in the comparison.

Part C.1 to C.G gives the trial number and the mean for planning horizon  $J$  for the  $G$  different performance criteria. Figure 3.6 shows the first 44 entries of performance criterion 1 only.

	H	I
11	Trial number	Mean for planning horizon J
12	5439	7.088
13	5438	6.835
14	5437	7.868
15	5436	9.341
16	5435	8.175
17	5434	6.708
18	5433	2.224
19	5432	1.085
20	5431	5.777
21	5430	2.667
22	5429	4.933
23	5428	4.162
24	5427	2.243
25	5426	2.558
26	5425	8.924
27	5424	5.028
28	5423	1.058
29	5422	2.036
30	5421	0.844
31	5420	4.573
32	5419	3.572
33	5418	4.423
34	5417	5.622
35	5416	7.883
36	5415	6.176
37	5414	8.750
38	5413	5.155
39	5412	4.700
40	5411	6.538
41	5410	6.911
42	5409	5.709
43	5408	5.405
44	5407	-1.123
45	5406	6.677
46	5405	7.392
47	5404	7.883
48	5403	4.376
49	5402	3.938
50	5401	5.129
51	5400	5.365
52	5399	8.394
53	5398	9.273
54	5397	1.973
55	5396	3.388

Figure 3.6 Part C.1 of the First technique

The trial numbers are given in cells H12: H2012 and the means for planning horizon  $J$  in cells I12: I2012.

Part 1.1 to 1.G shows the calculation of the difference between the alternative system's mean for planning horizon  $J$  and the current system's mean for planning horizon  $J$  for the  $G$  different performance criteria. Figure 3.7 shows the first 44 entries of performance criterion 1 only.

	J
11	Alternative-Current
12	-0.463
13	5.526
14	8.437
15	13.771
16	-4.677
17	10.975
18	-10.217
19	-6.122
20	-3.644
21	-9.756
22	-15.691
23	-10.868
24	3.984
25	-2.629
26	4.948
27	-9.463
28	0.491
29	-10.483
30	-15.374
31	2.743
32	-1.337
33	-9.698
34	6.174
35	-7.327
36	10.424
37	4.137
38	-8.137
39	-8.876
40	7.341
41	0.443
42	19.014
43	6.726
44	-3.890
45	15.718
46	5.713
47	13.637
48	22.433
49	-7.038
50	20.936
51	9.700
52	15.677
53	10.788
54	2.406
55	5.562

Figure 3.7 Part 1.1 of the First technique

For every performance criterion  $g$ , the difference between the alternative system's and the current system's values for a specific replication  $w$  is computed for the last  $W$  replications. If  $\bar{X}_{wg1}$  (e.g.

I12) and  $\bar{X}_{wg2}$  (e.g. from I12 on a sheet not shown) represent the alternative system's and the current system's respective values for the mean for planning horizon  $J$ , then the difference  $Z_{wg}$  (e.g. J12) is determined by the equation:

$$\begin{aligned} Z_{wg} &= \bar{X}_{wg1} - \bar{X}_{wg2} && \text{Equation 3.2} \\ &= 6.625 - 7.088 \\ &= -0.463 \end{aligned}$$

This process is executed for all 2 000 trials of the  $G$  different performance criteria.

Part 2.1.1 to 2.G.W shows the statistical analysis of the difference between the alternative system's and the current system's means for the planning horizon  $J$  for different values of  $w$  for the  $G$  different performance criteria. Figure 3.8 shows the statistical analysis for  $w = 50$  of performance criterion 1 only.

	K	L
11	Statistical analysis of A-C	
12	k=	50
13	Sample mean=	2.22
14	Sample variance=	97.04
15	Confidence interval LB=	-0.58
16	Confidence interval UB=	5.02

Figure 3.8 Part 2.1.1 of the First technique

The process is replicated for  $w = 100, 250, 500, 1\ 000$  and  $2\ 000$ . The sample mean for  $w = 50$  for performance criterion  $g$  in cell L13 is computed as:

$$\begin{aligned} \bar{Z}_g &= \frac{\sum_{w=1}^W Z_{wg}}{W} && \text{Equation 2.1} \\ &= \frac{110.928}{50} \\ &= 2.22 \end{aligned}$$

and the sample variance for  $w = 50$  for performance criterion  $g$  in cell L14 by:

$$\begin{aligned} s^2(\bar{Z}_g) &= \frac{\sum_{w=1}^W [Z_{wg} - \bar{Z}_g]^2}{(W - 1)} && \text{Equation 2.2} \\ &= \frac{4755.1509}{49} \\ &= 97.04 \end{aligned}$$

It is now possible to form the confidence interval lower bound for  $w = 50$  for performance criterion  $g$  in cell L15:

$$LB = \bar{Z}_g - t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \tag{Equation 2.3}$$

$$= 2.22 - 2.0096 \sqrt{\frac{97.04}{50}}$$

$$= -0.58$$

as well as the confidence interval upper bound for  $w = 50$  for performance criterion  $g$  in cell L16:

$$UB = \bar{Z}_g + t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \tag{Equation 2.4}$$

$$= 2.22 + 2.0096 \sqrt{\frac{97.04}{50}}$$

$$= 5.02$$

Part 3.1 to 3.G shows the statistical analysis of the difference between the alternative system's and the current system's means for the planning horizon  $J$  for the value of  $w$  determined with the method discussed in paragraph 2.7.3 on page 26 for the specific performance criterion  $g$ . Figure 3.9 below shows the statistical analysis for performance criterion 1 only.

	D	E
12	Performance criterion 1	
13	Sample mean=	2.22
14	Sample variance=	97.04
15	Confidence interval LB=	-0.58
16	Confidence interval UB=	5.02
17	Alternative better than current?	NO
18	Alternative points=	0.00
19	Current points=	8.00

Figure 3.9 Part 3.1 of the First technique

The process is replicated for all the performance criteria. Thus, depending on the sample size used, given in cell B26, the sample mean, variance, confidence interval lower bound and upper bound are chosen from those given in cells K12: L41. From these it is determined whether it can be assumed that the alternative system is better than the current system or not. For maximisation, if the confidence interval lower bound (cell E15) is equal to or smaller than zero, it can be assumed with confidence equal to  $CL_g\%$ (cell B13) that the current system's expectations are better than the alternative system's for specific performance criterion  $g$ . However, if the

confidence interval lower bound (cell E15) is greater than zero, it can be assumed with confidence equal to  $CL_g\%$  that the alternative system's expectations are better than those of the current system for specific performance criterion  $g$ . For minimisation, if the confidence interval upper bound (cell E16) is equal to or greater than zero, it can be assumed with confidence equal to  $CL_g\%$  (cell B13) that the current system's expectations are better than those of the alternative system for specific performance criterion  $g$ . However, if the confidence interval upper bound (cell E16) is smaller than zero, it can be assumed with confidence equal to  $CL_g\%$  (cell B13) that the alternative system's expectations are better than those of the current system for specific performance criterion  $g$ .

In this case, the performance criteria must be maximised (cell C15) and thus the alternative system is not better than the current system, because confidence interval lower bound E15 is smaller than zero. If the alternative system is better than the current system, the importance of the performance criterion  $v_g$  (cells B15: 19) is assigned to the alternative system in cell E18, or else it is assigned to the current system in cell E19. In this case, the 8 points (cell B15) are assigned to the current system cell E19.

Part 4 shown in Figure 3.10 illustrates the final compromise function calculations.

	A	B	C
28	Output:		
29	Compromise function total		
30	Alternative=	7.00	
31	Current=	20.00	
32	Alternative better than current?		NO

Figure 3.10 Part 4 of the First technique

The compromise function is determined for both the alternative system ( $CF_p$ ) in cell B30 and the current system ( $CF_c$ ) in cell B31 with the following equations:

$$CF_p = \sum_{l=1}^L (H_p \times v_l) \tag{Equation 3.3}$$

$$= 0 \times 8 + 0 \times 3 + 0 \times 4 + 0 \times 5 + 1 \times 7$$

$$= 7$$

$$CF_c = \sum_{g=1}^G (H_c \times v_g) \tag{Equation 3.4}$$

$$= 1 \times 8 + 1 \times 3 + 1 \times 4 + 1 \times 5 + 0 \times 7$$

$$= 20$$

Finally, it is possible to compare  $CF_p$  (cell B30) and  $CF_c$  (B31) to determine which of the current or the alternative system is the best. Cell B32 gives the final verdict on whether the alternative system is better than the current system or not. If  $CF_p$  is larger than  $CF_c$ , it can be assumed that the alternative system is better and if  $CF_p$  is smaller or equal to  $CF_c$ , it can be assumed that the current system is still better than the alternative system. In this case,  $CF_c$  is larger and cell B32 shows the alternative system is not better than the current.

### 3.2 The Second technique

The second developed technique is derived, but differs greatly, from Davis's [3] dominance probability density function approach discussed in paragraph 2.3.4 on page 13. The dominance probability density function approach calculates the probability that a control policy will provide an overall performance value that is  $\delta$  greater than the value generated by another control policy, for all possible combinations of control policies for every value of  $\delta$ . The Second technique also uses ideas from the paired-t confidence interval technique discussed in paragraph 2.4.2 on page 16.

The Second technique tests the hypothesis that a confidence level exists between, but not including, 50 and 100 percent, where the alternative system's performance criterion will provide a performance value that is better than the performance value of the current system's performance criterion. It compares the alternative system with the current system by determining the probability (confidence level) with which one can assume that the alternative system's performance criterion will provide a performance value that is greater than the performance value of the current system's performance criterion and *vice versa*. The confidence levels are then aggregated into a single compromise function that is used to determine the control policy to be implemented currently in the real-world system model. The following mathematical formulation and discussion represent only one evaluation, but the technique will be executed every time the system models complete a new replication of any alternative system. An algorithm describing the Second technique is shown in Figure 3.11.

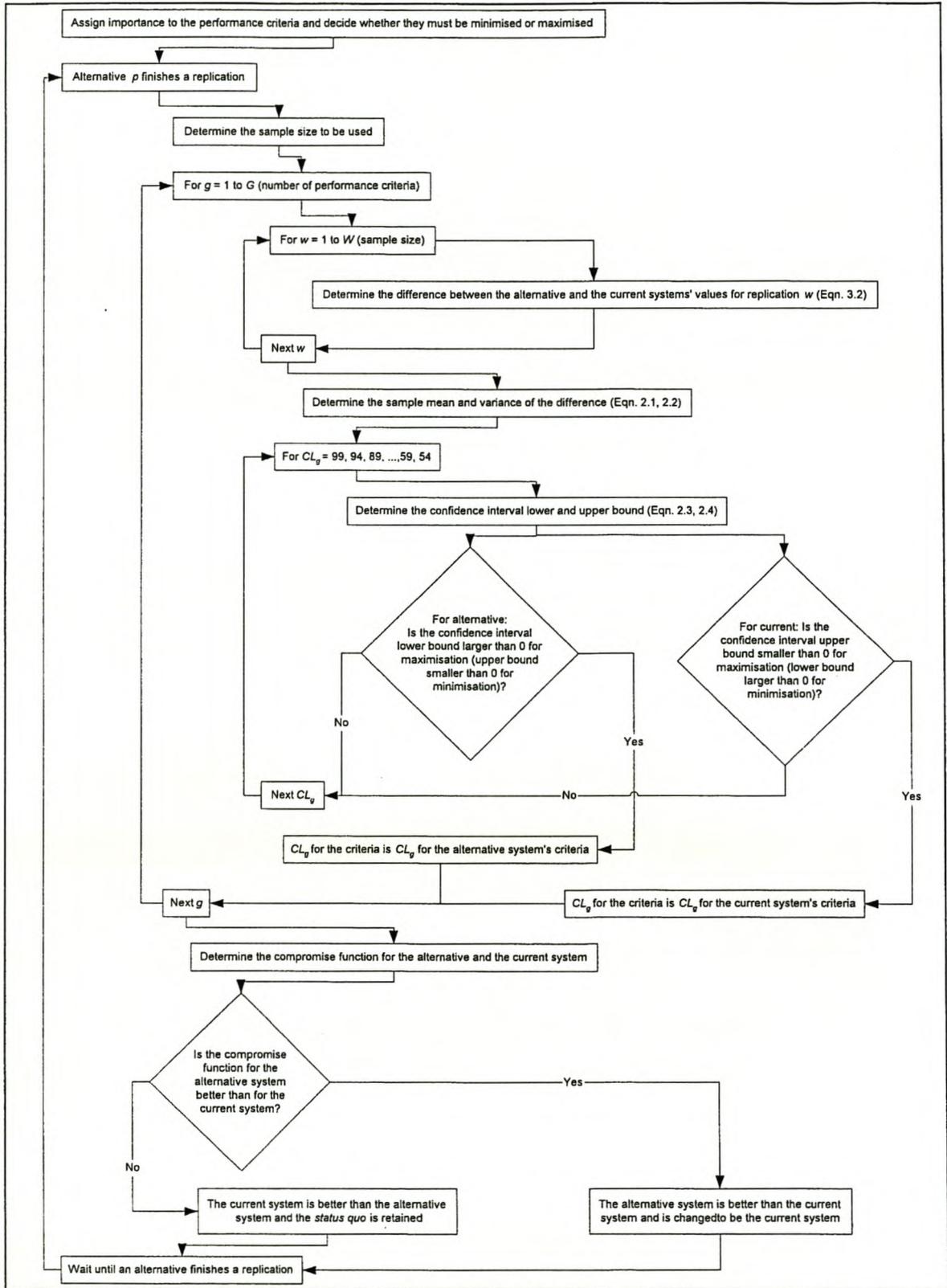


Figure 3.11 Schematic representation of the second developed technique

### 3.2.1 Mathematical formulation of the Second technique

The technique for comparing alternatives, illustrated in Figure 3.11, starts with the assignment of importance values  $v_g$  to the  $G$  different performance criteria in the same way as was discussed in paragraph 3.1.1 on page 32. This gives the operator control over the technique.

Figure 3.11 shows that whenever one of the  $P$  system models has completed a new replication, the alternative system  $p$  needs to be compared with the current system. Once again, the way the technique is implemented makes no distinction between the current system and the  $P-1$  alternative systems, and the incremental increasing of the sample size up to a fixed sample size procedure discussed in paragraph 2.7.3 on page 26 is used.

The next step is the sequential evaluation of the  $G$  performance criteria used to compare the alternative system with the current system. The sample mean and variance are determined in the same way as discussed in paragraph 3.1.1 on page 32.

Now it is possible to determine the highest confidence level ( $CL_g$ ) with which it can be assumed that the alternative system's performance criterion  $g$  is better than the current's performance criterion  $g$  or *vice versa*. This is done by the iteration of the confidence level for comparison. In this case, the confidence level is started at 99%. The confidence interval lower and upper bounds are determined from the sample mean and variance in the same way as described in paragraph 3.1.1 on page 32.

The hypothesis is formulated as follows:

$H_0$ : The alternative system's performance criterion  $g$  is better than that of the current system.

$H_1$ : The alternative system's performance criterion  $g$  is not better than that of the current system.

If the performance criterion needs to be maximised, the alternative system's performance criterion  $g$  will be better than the current system's if the confidence interval lower bound is larger than zero and then there are no grounds to reject the hypothesis. In the same way, the current system's performance criterion  $g$  will be better than the alternative system's if the confidence interval upper bound is smaller than zero, but then the hypothesis is rejected. If it cannot be shown that the confidence interval's lower bound is larger than zero or that the confidence interval's upper bound is smaller than zero, revert to the next confidence level, until a confidence level of 50% is reached.

If the performance criterion needs to be minimised, the alternative system's performance criterion  $g$  will be better than the current system's if the confidence interval upper bound is smaller than zero and then there are no grounds to reject the hypothesis. In the same way, the

current system's performance criterion  $g$  will be better than the alternative system's if the confidence interval lower bound is larger than zero, but then the hypothesis is rejected. If it cannot be shown that the confidence interval's upper bound is smaller than zero or that the confidence interval's lower bound is larger than zero, revert to the next confidence level., until a confidence level of 50% is reached.

This technique determines the iteration's next confidence level by decreasing the current confidence level with 5%. This can be changed to 1% or less for better accuracy, but it will increase the execution time. The ideal would be to replace the iterative system with a calculation that determines the exact confidence level where the lower or upper bound moves above or below zero respectively, but that requires an extremely complex calculation involving the inverse of the t-distribution.

The result of the lower confidence level will be a narrower confidence interval. This new confidence interval is evaluated in the same way as described above and if it cannot be shown that the confidence interval's upper bound is smaller than zero (minimisation) or that the confidence interval's lower bound is larger than zero (maximisation), then the technique will revert to the next confidence level. This will continue until a confidence level of 54% is reached or a confidence level is found where the confidence interval's upper bound is smaller than zero or the confidence interval's lower bound is larger than zero. The iteration stops at 54%, because the next iteration would be for 49%, and at 50% the interval width becomes meaningless.

The result of the iteration is the highest confidence level ( $CL_g$ ) with which it can be assumed that the alternative system's performance criterion  $g$  is better than the current's performance criterion  $g$  or *vice versa*.

Decreasing the confidence level to 54% is open to debate. At confidence levels that low, the confidence interval is so narrow (for a given variance estimator and degrees of freedom) that it would be possible to assume wrongly that the alternative system's performance criterion  $g$  is better than the current's performance criterion  $g$  or *vice versa*. That in itself is worrying, but even worse is that the performance criterion  $g$  is given a weight of 0.5, which is half of what is given to a performance criterion that is assumed to be better at a confidence level of 100%. Two possible solutions exist. The first is to determine a cut-off confidence level, say 75%, where the iteration stops. This will prevent performance criteria that are only marginally better having an influence. The other solution is to convert the highest confidence level with which it can be assumed that the alternative system's performance criterion  $g$  is better than the current's performance criterion  $g$  or *vice versa* (a value between 50% and 100%), to a value to between 0% and 100%. This will

ensure that performance criteria that can only be assumed to be better at a confidence level of 50%, receive a weighting of 0. These suggestions are subjects for further study.

After the  $G$  performance criteria have been compared, the compromise function can be determined from these highest confidence levels. The compromise functions are defined for the alternative system ( $CF_p$ ) and the current system ( $CF_c$ ) with the following equations:

$$CF_p = \sum_{g=1}^G \left( H_r \times v_g \times \frac{CL_g}{100} \right) \quad \text{Equation 3.5}$$

with  $H_p = 0$  if the hypothesis is rejected and  $H_p = 1$  if the hypothesis cannot be rejected, and

$$CF_c = \sum_{g=1}^G \left( H_a \times v_g \times \frac{CL_g}{100} \right) \quad \text{Equation 3.6}$$

with  $H_c = 1$  if the hypothesis is rejected and  $H_c = 0$  if the hypothesis cannot be rejected.

The definition of the compromise functions requires further study as explained in paragraph 3.1.1 on page 32.

The final step before progressing to the next alternative system is to compare  $CF_p$  and  $CF_c$  to determine whether the alternative system's expectations are better and need to be implemented in the real-world system. If  $CF_p$  is larger than  $CF_c$ , it is assumed that the alternative system is better and it is swapped with the current system in the real-world system. If  $CF_p$  is smaller or equal to  $CF_c$ , we assume that the current system is still the best and retain the *status quo*.

### 3.2.2 Schematic implementation with sample calculations for the Second technique

To illustrate the Second technique a complete schematic implementation with sample calculations is shown in this paragraph. Figure 3.12 shows an overview of the second developed technique, with the parts corresponding to the technique coloured in. The darker colour corresponds to information supplied by the user or to results from the replications of alternative systems, while the lighter colour indicates cells that are calculated. Figure 3.13 shows the different parts, with their annotations, that will be discussed as units in the following sections. When the first digit of the annotation is a letter, it refers to information supplied by the user or to results from the replications of alternative systems, while a number refers to calculations. Annotations with the same prefixes are nearly similar, differing only with the digit that is not the same.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V		
11	Second technique							th most recent trial	Total number	Mean for planning horizon J	Alternative-Current	Statistical analysis of A-C				th most recent trial	Total number	Mean for planning horizon J	Alternative-Current	Statistical analysis of A-C				
12	Given input from "Start cell"							1	5459	7.088	-0.463				n=	50	1	5429	3.105	3.535			n=	50
13	Importance (1-10)							2	5438	6.625	5.526			Sample mean	1.96	2	5436	3.424	4.300			Sample mean	1.96	
14	Performance criterion 1=							3	5457	7.288	6.437	Confidence level		Sample variance	68.45	3	5437	2.264	0.254	Confidence level		Sample variance	68.45	
15	Performance criterion 2=							4	5456	9.341	13.771	99	Confidence interval LB=	-0.39	4	5436	1.145	6.162	99	Confidence interval LB=	-0.39			
16	Performance criterion 3=							5	5436	6.125	-4.677	99	Confidence interval UB=	4.31	5	5435	2.433	2.028	99	Confidence interval UB=	4.31			
17	Performance criterion 4=							6	5454	6.708	10.975	94	Confidence interval LB=	0.00	6	5434	1.252	3.355	94	Confidence interval LB=	0.00			
18	Performance criterion 5=							7	5433	7.234	-10.217	94	Confidence interval UB=	3.92	7	5433	2.240	-11.551	94	Confidence interval UB=	3.92			
19								8	5437	1.065	-6.122	89	Confidence interval LB=	0.44	8	5432	2.619	-10.491	89	Confidence interval LB=	0.44			
20	Calculated							9	5431	6.777	-3.644	89	Confidence interval UB=	3.48	9	5431	1.464	1.530	89	Confidence interval UB=	3.48			
21								10	5420	2.667	-9.795	84	Confidence interval LB=	0.97	10	5430	3.684	-3.012	84	Confidence interval LB=	0.97			
22								11	5429	4.833	-15.691	84	Confidence interval UB=	2.95	11	5429	3.312	8.659	84	Confidence interval UB=	2.95			
23								12	5428	4.162	-10.688		Confidence interval LB=	1.34	12	5428	3.056	12.393		Confidence interval LB=	1.34			
24	Sample size used=							13	5437	2.243	3.984		Confidence interval UB=	2.58	13	5427	3.266	-1.706		Confidence interval UB=	2.58			
25								14	5435	2.599	-2.629	64	Confidence interval LB=	1.66	14	5426	3.053	4.644	64	Confidence interval LB=	1.66			
26	Output:							15	5425	6.924	4.948	64	Confidence interval UB=	2.26	15	5425	2.955	8.357	64	Confidence interval UB=	2.26			
27	Compromise function total							16	5424	5.038	-9.463	59	Confidence interval LB=	1.81	16	5424	2.519	-2.472	59	Confidence interval LB=	1.81			
28	Alternative=							17	5423	1.068	0.491	59	Confidence interval UB=	2.11	17	5423	3.439	5.968	59	Confidence interval UB=	2.11			
29	Current=							18	5422	2.036	-10.483	54	Confidence interval LB=	1.89	18	5422	3.469	4.051	54	Confidence interval LB=	1.89			
30	Alternative better than current? YES							19	5421	0.844	-15.374	54	Confidence interval UB=	2.03	19	5421	3.620	5.945	54	Confidence interval UB=	2.03			
31	Confidence level							20	5420	4.573	2.743		n=	100	20	5420	3.676	25.960		n=	100			
32	For smallest LB > 0 =							21	5419	3.822	-1.337		Sample mean	3.44	21	5419	-1.659	-11.374		Sample mean	1.48			
33	For largest UB < 0 =							22	5418	4.428	-9.699	Confidence level		Sample variance	64.96	22	5418	3.054	3.687	Confidence level		Sample variance	64.96	
34	Points							23	5417	5.822	6.174	99	Confidence interval LB=	1.62	23	5417	3.910	8.266	99	Confidence interval LB=	1.62			
35								24	5416	7.893	-7.327	99	Confidence interval UB=	1.27	24	5416	3.800	-3.361	99	Confidence interval UB=	1.27			
36	Performance criterion 2							25	5415	6.178	10.424	94	Confidence interval LB=	1.91	25	5415	3.229	4.667	94	Confidence interval LB=	1.91			
37	Sample mean							26	5414	8.780	4.137	94	Confidence interval UB=	4.98	26	5414	3.465	-7.014	94	Confidence interval UB=	4.98			
38	Confidence level							27	5413	5.156	-8.137	89	Confidence interval LB=	2.26	27	5413	4.000	6.147	89	Confidence interval LB=	2.26			
39	99							28	5412	4.700	-8.876	89	Confidence interval UB=	4.63	28	5412	4.765	10.059	89	Confidence interval UB=	4.63			
40	Confidence interval LB=							29	5411	6.538	7.341	84	Confidence interval LB=	2.67	29	5411	2.895	-2.042	84	Confidence interval LB=	2.67			
41	94							30	5410	6.911	0.443	84	Confidence interval UB=	4.22	30	5410	3.943	8.050	84	Confidence interval UB=	4.22			
42	Confidence interval UB=							31	5409	5.709	19.014		Confidence interval LB=	2.96	31	5409	3.736	-14.778		Confidence interval LB=	2.96			
43	89							32	5408	5.405	6.726		Confidence interval UB=	3.93	32	5408	3.720	-6.832		Confidence interval UB=	3.93			
44	Confidence interval LB=							33	5407	4.128	-3.980	64	Confidence interval LB=	3.21	33	5407	3.673	-0.412	64	Confidence interval LB=	3.21			
45	84							34	5406	6.677	15.716	64	Confidence interval UB=	3.68	34	5406	2.814	2.009	64	Confidence interval UB=	3.68			
46	Confidence interval UB=							35	5405	7.992	5.713	59	Confidence interval LB=	3.33	35	5405	1.959	1.503	59	Confidence interval LB=	3.33			
47	94							36	5404	7.893	13.637	59	Confidence interval UB=	3.56	36	5404	4.039	13.840	59	Confidence interval UB=	3.56			
48	Confidence interval LB=							37	5403	4.376	-22.433	54	Confidence interval LB=	3.39	37	5403	3.961	3.396	54	Confidence interval LB=	3.39			
49	64							38	5402	3.938	-7.038	54	Confidence interval UB=	3.50	38	5402	4.391	-1.031	54	Confidence interval UB=	3.50			
50	Confidence interval UB=							39	5401	5.129	20.936		n=	250	39	5401	2.671	4.052		n=	250			
51	94							40	5400	5.325	9.700		Sample mean	3.08	40	5400	2.165	-2.773		Sample mean	2.93			
52	Confidence interval LB=							41	5399	8.394	15.677	Confidence level		Sample variance	79.69	41	5399	4.574	13.505	Confidence level		Sample variance	79.69	
53	54							42	5398	8.273	10.786	99	Confidence interval LB=	1.97	42	5398	1.660	8.189	99	Confidence interval LB=	1.97			
54	Confidence interval UB=							43	5397	1.933	2.405	99	Confidence interval UB=	4.19	43	5397	4.802	12.280	99	Confidence interval UB=	4.19			
55	Confidence level							44	5396	3.968	5.562	94	Confidence interval LB=	2.15	44	5396	2.765	4.614	94	Confidence interval LB=	2.15			
56	For smallest LB > 0 =							45	5395	4.807	5.019	94	Confidence interval UB=	4.01	45	5395	2.788	5.950	94	Confidence interval UB=	4.01			
57	For largest UB < 0 =							46	5394	1.183	0.323	89	Confidence interval LB=	2.35	46	5394	1.669	19.183	89	Confidence interval LB=	2.35			

Additional performance criteria

Additional performance criteria

Additional trials

Additional sample sizes

Additional trials

Additional sample sizes

Figure 3.12 The Second technique implemented

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	
11		Second technique							th most recent trial	Test number	Value for Student's $t$ -test	Attended classes	Student's $t$ -test	th most recent trial	Test number	Value for Student's $t$ -test	Attended classes	Student's $t$ -test	th most recent trial	Test number	Value for Student's $t$ -test	Attended classes	Student's $t$ -test
12	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
13	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	
14	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
15	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	
16	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	
17	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	
18	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	
19	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	
20	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	
21	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	
22	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	
23	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	
24	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	
25	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	
26	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	
27	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	
28	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	
29	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	
30	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	
31	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	
32	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	
33	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	
34	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	
35	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	
36	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	
37	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	
38	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	
39	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	
40	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	
41	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	
42	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	
43	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	
44	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	
45	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	
46	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	
47	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	
48	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	
49	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	
50	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	
51	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	
52	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	
53	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	
54	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	43	
55	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	
56	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	
57	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	46	

Figure 3.13 The different parts of the Second technique

	A	B	C
13	Importance [1-10]		
14	Performance criterion 1=	8	MAX
15	Performance criterion 2=	3	MAX
16	Performance criterion 3=	4	MIN
17	Performance criterion 4=	5	MAX
18	Performance criterion 5=	7	MIN

Figure 3.14 Part A of the Second technique

Part A, shown in Figure 3.14, comprises of the information supplied by the user. This includes the importance of the different performance criteria ( $v_j$ ) and whether the performance criteria should be maximised or minimised. Cells B14: B18 gives the relevant importance of the performance criteria (on a scale of 1 to 10 with 1 the least important) and cells C14: C18 indicates whether the performance criteria should be minimised or maximised.

	A	B
20	Calculated:	
21		
22		
23		
24	Sample size used=	50

Figure 3.15 Part B of the Second technique

Part B, shown in Figure 3.15, comprises of the fixed sample size discussed in paragraph 2.7.3 on page 26. In this case, the sample size was set to 50 to make the sample calculations easier. Cell B24 gives the sample size to be used in the comparison.

Part C.1 to C.G gives the trial number and mean for planning horizon  $J$  for the  $G$  different performance criteria. Figure 3.16 shows the first 34 entries of performance criterion 1 only.

	I	J
11	Trial number	Mean for planning horizon J
12	5439	7.088
13	5438	6.835
14	5437	7.868
15	5436	9.341
16	5435	8.175
17	5434	6.708
18	5433	2.224
19	5432	1.085
20	5431	5.777
21	5430	2.667
22	5429	4.933
23	5428	4.162
24	5427	2.243
25	5426	2.558
26	5425	8.924
27	5424	5.028
28	5423	1.058
29	5422	2.036
30	5421	0.844
31	5420	4.573
32	5419	3.572
33	5418	4.423
34	5417	5.622
35	5416	7.883
36	5415	6.176
37	5414	8.750
38	5413	5.155
39	5412	4.700
40	5411	6.538
41	5410	6.911
42	5409	5.709
43	5408	5.405
44	5407	-1.123
45	5406	6.677
46	5405	7.392

Figure 3.16 Part C.1 of the Second technique

The trial numbers are given in cells I12: I2012 and the means for planning horizon  $J$  in cells J12: J2012.

Part 1.1 to 1.G shows the calculation of the difference between the alternative system's mean for planning horizon  $J$  and the current system's mean for planning horizon  $J$  for the  $G$  different performance criteria. Figure 3.17 shows the first 34 entries of performance criterion 1 only.

	K
11	Alternative-Current
12	-0.463
13	5.526
14	8.437
15	13.771
16	-4.677
17	10.975
18	-10.217
19	-6.122
20	-3.644
21	-9.756
22	-15.691
23	-10.868
24	3.984
25	-2.629
26	4.948
27	-9.463
28	0.491
29	-10.483
30	-15.374
31	2.743
32	-1.337
33	-9.698
34	6.174
35	-7.327
36	10.424
37	4.137
38	-8.137
39	-8.876
40	7.341
41	0.443
42	19.014
43	6.726
44	-3.890
45	15.718
46	5.713

Figure 3.17 Part 1.1 of the Second technique

For every performance criterion  $g$  the difference between the alternative system's and the current system's values for a specific replication  $w$  is then computed for the last  $W$  replications. If  $\bar{X}_{wK1}$  (e.g. J12) and  $\bar{X}_{wK2}$  (e.g. from J12 on a sheet not shown) represent the alternative system's and the current system's respective values for the mean for planning horizon  $J$ , then the difference  $Z_{wK}$  (e.g. K12) is determined by the equation:

$$Z_{wK} = \bar{X}_{wK1} - \bar{X}_{wK2} \tag{Equation 3.2}$$

$$= 6.625 - 7.088$$

$$= -0.463$$

This process is executed for all 2 000 trials for all the  $G$  different performance criteria.

Part 2.1.1 to 2.G.W shows the statistical analysis of the difference between the alternative system's and the current system's means for the planning horizon  $J$  for different values of  $w$  for the  $G$  different performance criteria. Figure 3.18 shows the statistical analysis for  $w = 50$  of performance criterion 1 only.

	L	M	N
11	Statistical analysis of A-C		
12		w=	50
13		Sample mean=	1.96
14	Confidence level	Sample variance=	68.45
15	99	Confidence interval LB=	-0.39
16	99	Confidence interval UB=	4.31
17	94	Confidence interval LB=	0.00
18	94	Confidence interval UB=	3.92
19	89	Confidence interval LB=	0.44
20	89	Confidence interval UB=	3.48
21	84	Confidence interval LB=	0.97
22	84	Confidence interval UB=	2.95
23	...	Confidence interval LB=	1.34
24	...	Confidence interval UB=	2.58
25	64	Confidence interval LB=	1.66
26	64	Confidence interval UB=	2.26
27	59	Confidence interval LB=	1.81
28	59	Confidence interval UB=	2.11
29	54	Confidence interval LB=	1.89
30	54	Confidence interval UB=	2.03

Figure 3.18 Part 2.1.1 of the Second technique

The process is replicated for  $w = 100, 250, 500, 1\ 000$  and  $2\ 000$ . The sample mean for  $w = 50$  for performance criterion  $g$  in cell N13 is computed as:

$$\bar{Z}_g = \frac{\sum_{w=1}^W Z_{wg}}{W}$$

Equation 2.1

$$= \frac{98.132}{50}$$

$$= 1.96$$

and the sample variance for  $w = 50$  for performance criterion  $g$  in cell N14 by:

$$s^2(\bar{Z}_g) = \frac{\sum_{w=1}^W [Z_{wg} - \bar{Z}_g]^2}{(W-1)} \quad \text{Equation 2.2}$$

$$= \frac{3354.05}{49}$$

$$= 68.45$$

It is now possible to form the confidence interval lower and upper bound for confidence levels corresponding to 99, 94, 89, ..., 59 and 54 percent. First, the alpha ( $\alpha_c$ ) is computed. For the confidence level of 99% in cell L15 it is calculated according to the following equation:

$$\alpha_c = 1 - \left( \frac{CL_g}{100} \right) \quad \text{Equation 3.1}$$

$$= 1 - \left( \frac{99}{100} \right)$$

$$= 0.01$$

It is now possible to form the confidence interval lower bound for  $w = 50$  for performance criterion  $g$  in cell N15:

$$LB = \bar{Z}_g - t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.3}$$

$$= 1.96 - 2.008 \sqrt{\frac{68.45}{50}}$$

$$= -0.39$$

as well as the confidence interval upper bound for  $w = 50$  for performance criterion  $g$  in cell N16:

$$UB = \bar{Z}_g + t_{n-1, 1-\alpha/2} \sqrt{\frac{s^2(\bar{Z}_g)}{W}} \quad \text{Equation 2.4}$$

$$= 1.96 + 2.008 \sqrt{\frac{68.45}{50}}$$

$$= 4.31$$

This process is repeated for confidence levels corresponding to 99, 94, 89, ..., 59 and 54 percent in cells N17: N30.

Part 3.1 to 3.G shows the statistical analysis of the difference between the alternative system's and the current system's means for the planning horizon  $J$  for the value of  $w$  as determined with the method discussed in paragraph 2.7.3 on page 26 for the specific performance criterion  $g$ . Figure 3.19 shows the statistical analysis for performance criterion 1 only.

	D	E	F
12	Performance criterion 1		
13		Sample mean=	2.22
14	Confidence level	Sample variance=	97.04
15	99	Confidence interval LB=	-0.39
16	99	Confidence interval UB=	4.31
17	94	Confidence interval LB=	0.00
18	94	Confidence interval UB=	3.92
19	89	Confidence interval LB=	0.44
20	89	Confidence interval UB=	3.48
21	84	Confidence interval LB=	0.97
22	84	Confidence interval UB=	2.95
23	...	Confidence interval LB=	1.34
24	...	Confidence interval UB=	2.58
25	64	Confidence interval LB=	1.66
26	64	Confidence interval UB=	2.26
27	59	Confidence interval LB=	1.81
28	59	Confidence interval UB=	2.11
29	54	Confidence interval LB=	1.89
30	54	Confidence interval UB=	2.03
31	Confidence level	For smallest LB > 0 =	0.89
32		For largest UB < 0 =	0.00
33	Points	Alternative=	7.12
34		Current=	0.00

Figure 3.19 Part 3.1 of the Second technique

The process is replicated for all the performance criteria. Thus, depending on the sample size used, given in cell B24, the sample mean, variance, confidence interval lower bound and upper bound are chosen from those given in cells M12: N125. Now it can be determined what the highest confidence level ( $CL_g\%$ ) is with which it can be assumed that the alternative system's performance criterion  $g$  is better than the current system's or *vice versa*.

If the performance criterion needs to be maximised, the alternative system's performance criterion  $g$  will be better than the current system's if the confidence interval lower bound is larger than zero and then there are no grounds to reject the null hypothesis. In the same way, the current system's performance criterion  $g$  will be better than the alternative system's if the confidence interval upper bound is smaller than or equal to zero, but then the hypothesis is rejected. If the performance criterion needs to be minimised, the alternative system's performance criterion  $g$  will be better than the current system's if the confidence interval upper

bound is smaller than zero and then there are no grounds to reject the hypothesis. In the same way, the current system's performance criterion  $g$  will be better than the alternative system's if the confidence interval lower bound is greater than or equal to zero, but then the hypothesis is rejected. The result is the highest confidence level ( $CL_g$ ) with which it can be assumed that the alternative system's performance criterion  $g$  is better than the current system's performance criterion  $g$  or *vice versa*.

In this case, the performance criterion should be maximised (cell C14 shown in Figure 3.14 on page 52), so that the highest confidence level for which the confidence interval lower bound is larger than zero (cell F19 equals 0.44) is equal to 89% (cell D19). This is shown in cell F31. If the alternative system is better than the current system, the importance of the performance criterion  $v_g$  (cells B14: B18) is multiplied with the confidence level in cells F31: F32 and shown in cell F33, or else it is multiplied with the confidence level in cells F31: F32 and shown in cell F34. In this case, the 8 points (cells B14:C14, shown in Figure 3.14 on page 52) are multiplied with the confidence level of 0.89 in cell F31 and shown as 7.12 in cell F33.

Part 4, shown in Figure 3.20, illustrates the final compromise function calculations.

	A	B
26	Output:	
27	Compromise function total	
28	Alternative=	18.97
29	Current=	0.00
30	Alternative better than current?	YES

Figure 3.20 Part 4 of the Second technique

The compromise function is determined for both the alternative system ( $CF_p$ ) in cell B28 and the current system ( $CF_c$ ) in cell B29 with the following equations:

$$CF_p = \sum_{g=1}^i \left( H_r \times v_g \times \frac{CL_g}{100} \right) \tag{Equation 3.5}$$

$$\begin{aligned}
 &= 1 \times 8 \times \frac{89}{100} + 1 \times 3 \times \frac{79}{100} + 1 \times 4 \times \frac{54}{100} + 1 \times 5 \times \frac{54}{100} + 1 \times 7 \times \frac{64}{100} \\
 &= 7.12 + 2.4 + 2.16 + 2.7 + 4.48 \\
 &= 18.97
 \end{aligned}$$

$$CF_c = \sum_{g=1}^i \left( H_a \times v_g \times \frac{CL_g}{100} \right) \tag{Equation 3.6}$$

$$= 0 \times 8 \times \frac{0}{100} + 0 \times 3 \times \frac{0}{100} + 0 \times 4 \times \frac{0}{100} + 0 \times 5 \times \frac{0}{100} + 0 \times 7 \times \frac{0}{100}$$

$$= 0$$

Finally, it is possible to compare  $CF_p$  (cell B28) and  $CF_c$  (cell B29) to determine which of the current or the alternative system is the best. Cell B30 gives the final verdict on whether the alternative system is better than the current system or not, if  $CF_p$  is larger than  $CF_c$ , it can be assumed that the alternative system is better and if  $CF_p$  is smaller or equal to  $CF_c$ , it can be assumed that the current system is still better. In this case,  $CF_p$  is larger and cell B30 shows the alternative system is better than the current.

### 3.3 Summary

This chapter discussed two techniques that were developed to automate the real-time compromise analysis function. It introduced the techniques, followed with a detailed mathematical formulation and finished with a schematic implementation with sample calculations for both of the techniques.

It was shown that for the First technique, the paired-t confidence interval technique is used to compare the alternative system with the current system by building confidence intervals of the expected differences for the respective performance criteria and by testing the hypothesis that the alternative system's expectations of the performance criteria are better than those of the current system. The results of the comparisons made with the paired-t confidence interval technique are then consolidated into a compromise function that is used to determine the control policy to be implemented currently in the real-world system model.

The Second technique tests the hypothesis that a confidence level exists between, but not including, 50 and 100 percent, where the alternative system's performance criterion will provide a performance value that is better than the performance value of the current system's performance criterion. It compares the alternative system with the current system by determining the probability (confidence level) with which one can assume that the alternative system's performance criterion provides a performance value that is greater than the performance value of the current system's performance criterion and *vice versa*. The confidence levels are then aggregated into a single compromise function that is used to determine the control policy to be implemented currently in the real-world system model.

From the discussion of the techniques in the chapter, the differences between the two techniques become apparent. The differences between the two techniques are summarised in Table 3.2.

Table 3.2 The differences between the two techniques

First technique	Second technique
Based on the paired-t confidence interval technique.	Loosely derived from Davis's [3] dominance probability density function approach, but uses ideas from the paired-t confidence interval technique.
Tests the hypothesis that the alternative system's expectations of the performance criteria are better than those of the current system.	Tests the hypothesis that a confidence level exists between, but not including, 50 and 100 percent, where the alternative system's performance criterion will provide a performance value that is better than the performance value of the current system's performance criterion.
The confidence level is fixed and user-assignable.	The confidence level is varied by the technique.
The upper and lower bound are evaluated only once to determine whether it is smaller or larger than zero respectively.	An iterative process is followed to find the confidence level where the upper bound or lower bound is smaller or larger than zero respectively.
The compromise functions are only concerned with whether the expectations of the performance criteria of the current system are better than the expectations of the performance criteria of the alternative system or <i>vice versa</i> .	The compromise functions contain the probability with which it can be assumed that the expectations of the performance criteria of the current system are better than the expectations of the performance criteria of the alternative system or <i>vice versa</i> .

While discussing the techniques, some subjects that require further study were noted. These are summarised in Table 3.3.

Table 3.3 Subjects that require further study, as identified during the development of the techniques

Subject	Paragraph	Page
The effect of the settings of the techniques, i.e. the confidence level of the First technique, on the efficiency of the techniques.	3.1.1	32
	3.2.1	47
The effect of the sample size of the calculations on the efficiency of the techniques.	3.1.1	32
Adapting the techniques to accommodate performance criteria consisting of proportions, percentiles, minimums and maximums.	3.1.1	32
The definition and effect of the compromise functions on the efficiency of the techniques.	3.1.1	32
	3.2.1	47

This chapter was only concerned with the formulation and execution of the techniques. It did not make any claims to the efficiency of the techniques. The techniques need to be implemented to determine their efficiency. In Chapter 4 on page 62 the Emulator that is used to evaluate the techniques is described, while the results of the evaluations are presented in Chapter 5, starting on page 84.

## 4 THE DEVELOPMENT OF THE EMULATOR

The previous chapter looked at the two techniques that were developed. They were only described; it was not possible to say anything about their efficiency. In paragraph 2.6 on page 21 it was shown that an emulator is needed to evaluate the techniques used to select a control policy during the on-line planning and control process. This chapter will discuss the Emulator that will be used to evaluate the two techniques that were developed.

The systems that will be controlled with the on-line planning and control process will be complex systems, e.g. FMSs. However, the system controlled in the Emulator is only a straightforward M/M/1/FIFO/ $\infty$ / $\infty$  server. This is necessary because this is only an introductory project that is exploring ideas in an uncharted research area. Problems that have not been addressed sufficiently, e.g. the effortless initialising of the system models to the current state of the real-world system model, can be bypassed by having a first iteration with a straightforward system. It is hoped that later studies will look at increasingly complex systems.

This chapter only focuses on subjects that contribute to a better understanding of the Emulator. The Arena<sup>®</sup> model logic and its detailed explanation, the Visual Basic<sup>®</sup> code and the description of its subprograms and functions and Visual Basic<sup>®</sup> forms can be found in Appendix B. Information on operating the Emulator can be found in Appendix C. The chapter will introduce the Emulator, look at the Arena<sup>®</sup> models, the Visual Basic<sup>®</sup> programming, and the validation and verification of the Emulator.

### 4.1 Introduction to the Emulator

In essence, the Emulator is a Visual Basic<sup>®</sup> program that uses Arena<sup>®</sup> models. In paragraph 2.6 on page 21 it was said that a language like Java<sup>®</sup> would allow for concurrent operations, but because Arena<sup>®</sup> is only compatible with Visual Basic<sup>®</sup>, the Emulator was programmed in Visual Basic<sup>®</sup>. The development of the Emulator in a language that allows concurrent operations and effortless initialisation is a subject for further study.

Figure 2.2 on page 22 is reproduced as Figure 4.1, with the two main parts highlighted. The first part consists of the models (both the real-world system model and the alternative system models). The second part contains the initialiser and the output analysis and real-time compromise analysis function. Although the models are mainly programmed in Arena<sup>®</sup> and the rest mainly programmed in Visual Basic<sup>®</sup>, the entire Emulator is an intricate combination of Arena<sup>®</sup> models and Visual Basic<sup>®</sup> routines. The operation of the Emulator and additional aspects of the operation of the Emulator are discussed in the next two sections.

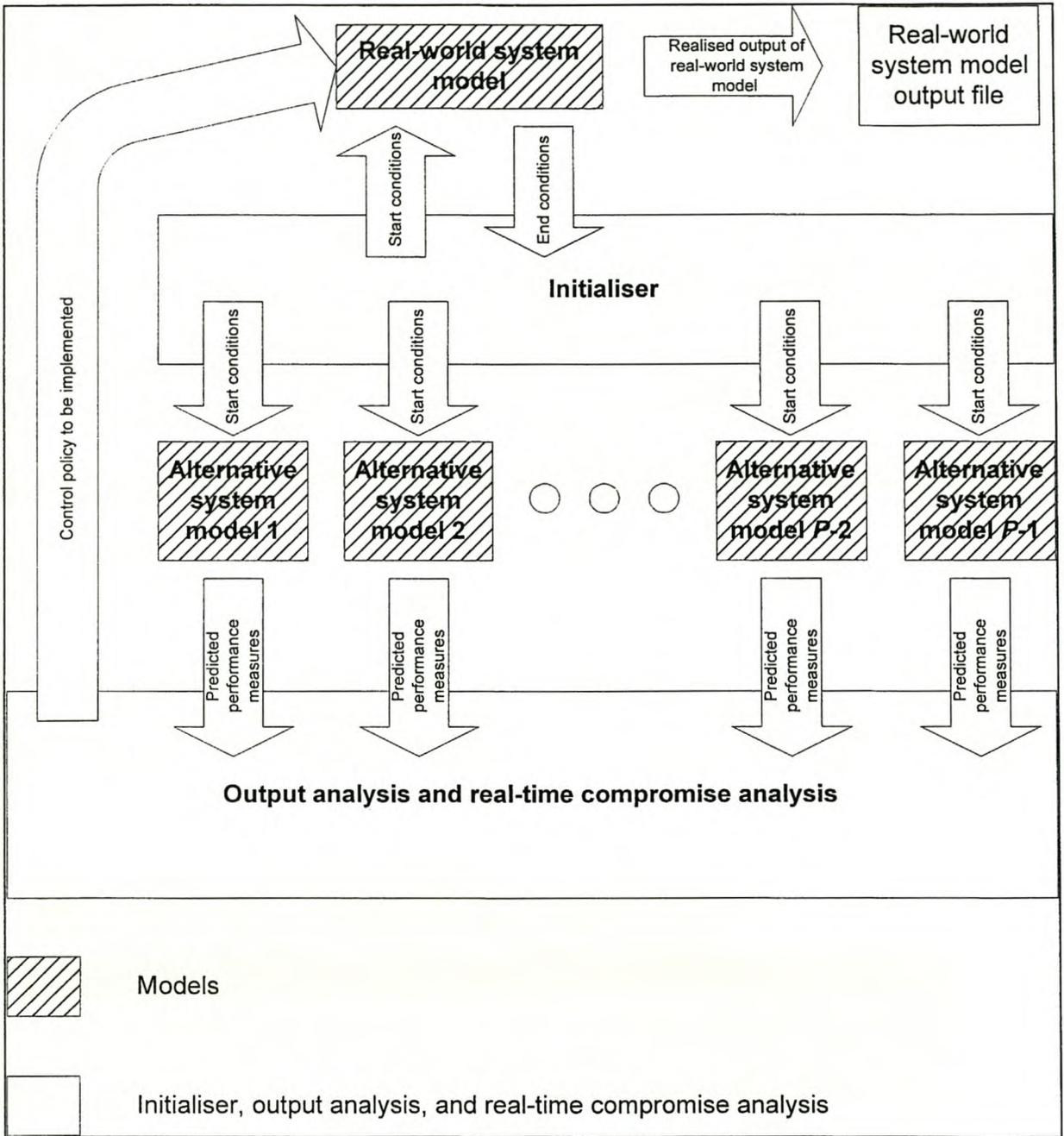


Figure 4.1 Schematic representation of the Emulator with the different parts distinguished

4.1.1 A brief overview of the operation of the Emulator

Before the Emulator is discussed in detail, it is necessary to get a general idea of the way it operates. A schematic representation of its operation is shown in Figure 4.2.

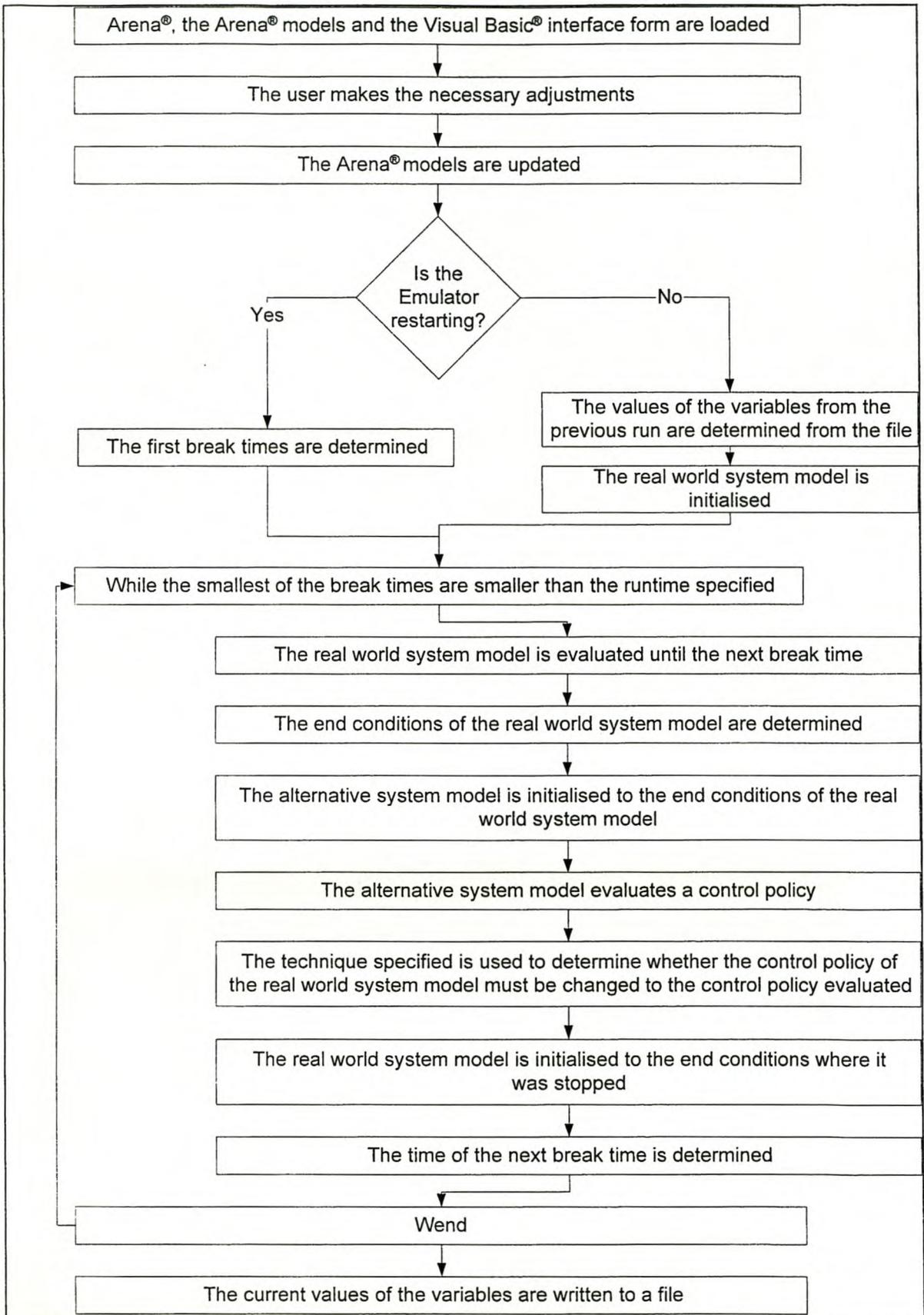


Figure 4.2 Schematic representation of the Emulator operations

When the Emulator is started, Arena<sup>®</sup>, the Arena<sup>®</sup> models and the Visual Basic<sup>®</sup> form are loaded. For more detail on starting the Emulator, see Appendix C. The Visual Basic<sup>®</sup> form, shown in Figure 4.3, enables the user to make the necessary adjustments before a run. These adjustments include adjusting the variables, selecting the technique to be used and changing the filenames that are used to capture the output.

Figure 4.3 The initial Visual Basic<sup>®</sup> form of the Emulator

The user of the Emulator starts its operation by pressing the required command button. The variables and settings chosen are updated in the Arena<sup>®</sup> models. The next part depends on whether the Emulator is restarting or continuing. Restarting refers to the situation where the Emulator is starting the run from time zero, while continuing refers to the situation where the Emulator continues from where it has finished its previous run. If the Emulator is restarting, the first breaktimes are determined. The breaktimes are the times at which the real-world system model must be stopped to evaluate the different control policies with the alternative system model. The spacing of the breaktimes is considered in the next paragraph. If the Emulator is continuing, the values of the variables from the previous run are determined from the output file

and the real-world system model is initialised to the condition at which it has ended the previous run.

The following loop is executed while the required runtime is still larger than the smallest of the breaktimes. The real-world system model is run until the next breaktime. While the real-world system model is running, its output is continuously written to the output files. The output files are used later to analyse statistically the specific technique used to select the control policy to be implemented. When the real-world system model reaches the next breaktime, the end conditions of the real-world system model are determined and the alternative system model is initialised to these conditions. The alternative system model is then used to evaluate a control policy and the output is used by the technique specified to determine whether or not the control policy in the real-world system model must be changed to the control policy evaluated. The control policy of the real-world system is changed if necessary. The real-world system model is then initialised to the condition where it has been stopped and the new breaktime is determined. If the required runtime is still smaller than the smallest of the breaktimes, the loop is repeated.

When the required runtime is reached, the current values of all the variables are written to a file to be used if the next run continues on this one.

#### *4.1.2 Additional aspects of the operation of the Emulator*

The discussion of the operation of the Emulator in the previous subsection assumed instantaneous evaluation of the control policies. Unfortunately, this is impossible because all calculations take a finite length of time and an evaluation is a combination of calculations. Thus, the evaluation of a control policy by the alternative system model takes a certain amount of time and the results of the evaluation can only be implemented once the evaluation has been finished.

The same alternative system model is used to evaluate the same control policy, so the next evaluation of the specific control policy also can be started only once the previous evaluation has finished. It may be possible to have many models where the same control policy are evaluated, but that would require an increase in computing power. In this case, it is assumed that only one model is available to evaluate a specific control policy. Unfortunately, it is difficult to determine the time an evaluation has taken in the Emulator, because it involves using the system clock. Thus, to simplify the process, it is assumed that the time an evaluation takes, is random and is taken from an exponential distribution. This is not the best procedure and ideally a minimum value added to a random value from an exponential distribution should be used. Once an evaluation is finished, the next breaktime for this specific control policy is calculated as the random time the evaluation is expected to take added to the time the previous evaluation started.

The mean of the exponential distribution is one of the settings of the Emulator, giving the user of the Emulator control over the spacing of the breaktimes. The true time of evaluation and the effect of the time of evaluation require further study.

One purpose of the Emulator, as explained in paragraph 2.6 on page 21, is to enable operations that should be done concurrently to be done sequentially. To explain the process, an example case where there are only one performance criterion and two alternative system models is examined. Figure 4.4 illustrates how the operations that should be done concurrently are done one at a time for this example. For the purpose of the explanation, the examination is started at the end of the  $w - 2^{\text{th}}$  trial of alternative system model 1 (point A), but it should be noted that the same is happening whenever the Emulator is running. The Emulator is run, and as time advances, the real-world system model generates a realised performance criterion value, e.g. **Length of service queue**. However, the technique tries to minimise or maximise this performance criterion value by predicting the performance criterion values of the alternative systems and implementing the alternative system's control policy in the real-world system if the alternative system's predicted performance criterion values are better than the predicted performance criterion values of the control policy currently implemented in the real-world system. Figure 4.4 shows that these trials should be done concurrently with the generation of the realised performance criteria values, but because the Emulator cannot do more than one calculation at a time, the advance of time, and thus also the calculation of the realised performance criterion values, is stopped at point B, and trial  $w + 1$  of alternative system model 2 is evaluated. Ideally, the actual time this calculation took would be determined by using the system clock, but in this case the time the calculation took is taken randomly from the exponential distribution specified. This is then added to the time at which the calculation was started to determine the time at which the calculation would have been finished (point C). This is the time at which the next trial of that specific alternative system model can start, because only one system model of each alternative system is available. If the result of trial  $w + 1$  requires that the control policy of the real-world system be changed to the control policy of alternative system 2, it is done when the Emulator reaches point D. The Emulator's advance of time and the generation of the realised performance criterion values, that were stopped at point B, are then run to the start of the next trial (point E). In this case, it is trial  $w$  for alternative system model 1, and the sequence, that starts with the stopping of the advance of time, is repeated.

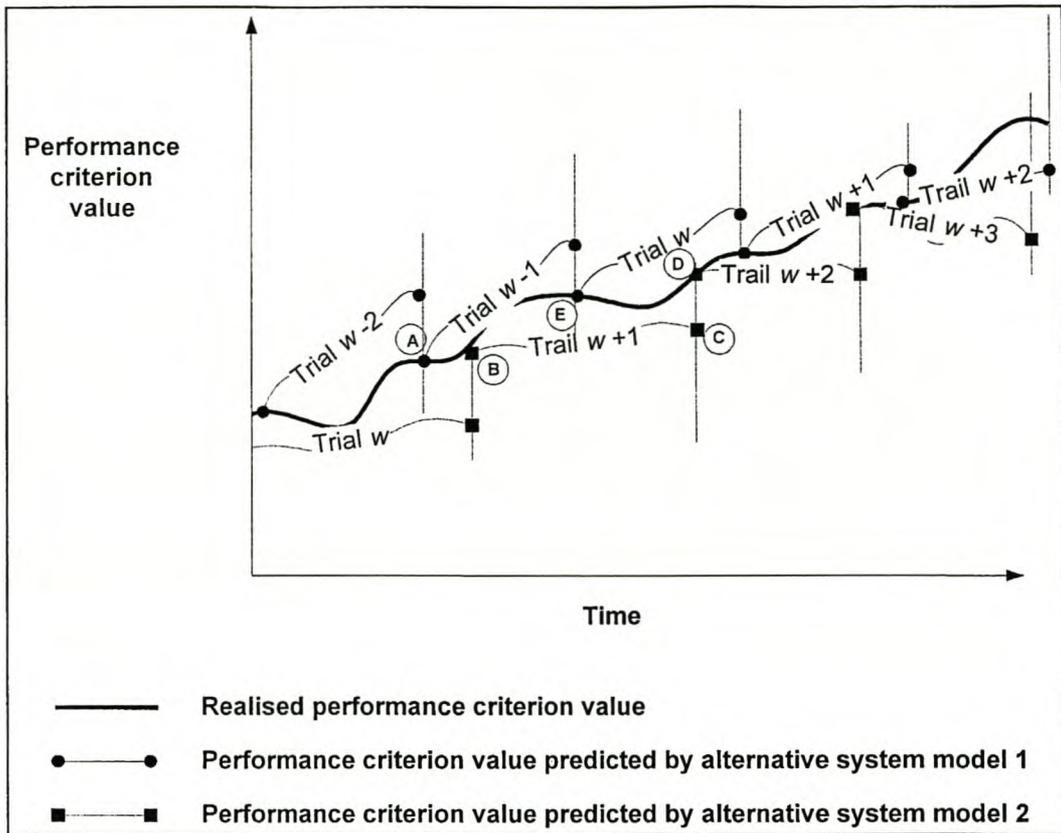


Figure 4.4 The pseudo-concurrent operation of the Emulator illustrated

## 4.2 The Arena<sup>®</sup> models

At the core of the Emulator are the Arena<sup>®</sup> models used for the real-world system model and the alternative system models. This paragraph will look at these Arena<sup>®</sup> models and the factors affecting them.

It is possible to distinguish between the models that are used by the Emulator. These are the real-world system model and the alternative system models. The main difference between the real-world system model and the alternative system models is that the real-world system model is stopped and started continuously to enable the Emulator to work. The alternative system models evaluate an alternative system for a specified period and then stops and starts all over again to evaluate another alternative system. While they are quite similar because they are models of the same system, there are some small differences necessitated by the programming of the Emulator. However, the following discussion of the models and the factors affecting them assumes that the models are identical. The detail of the implementation of the models in Arena<sup>®</sup>, including the small differences between the models, is shown in Appendix B.

#### 4.2.1 Concept model for the Arena® models

The concept model for the Arena® models is shown in Figure 4.5. First, the entities are created with the time between their arrivals taken from an exponential distribution. The Emulator only allows time between arrivals taken from an exponential distribution, and using other distributions is a subject for further study. Then the entities are assigned attributes. These include their process time, their due time, the time they entered the system and the number of the entity. It is then determined whether the server is currently busy or not. If it is busy, the entity is stored in the service queue. If it is not busy, the entity is serviced by the server and disposed of. Once an entity is in the queue, it has to compete with the other entities in the queue as to which is the next entity to be serviced once the server becomes available. The control policy determines which entity will be the next to be serviced.

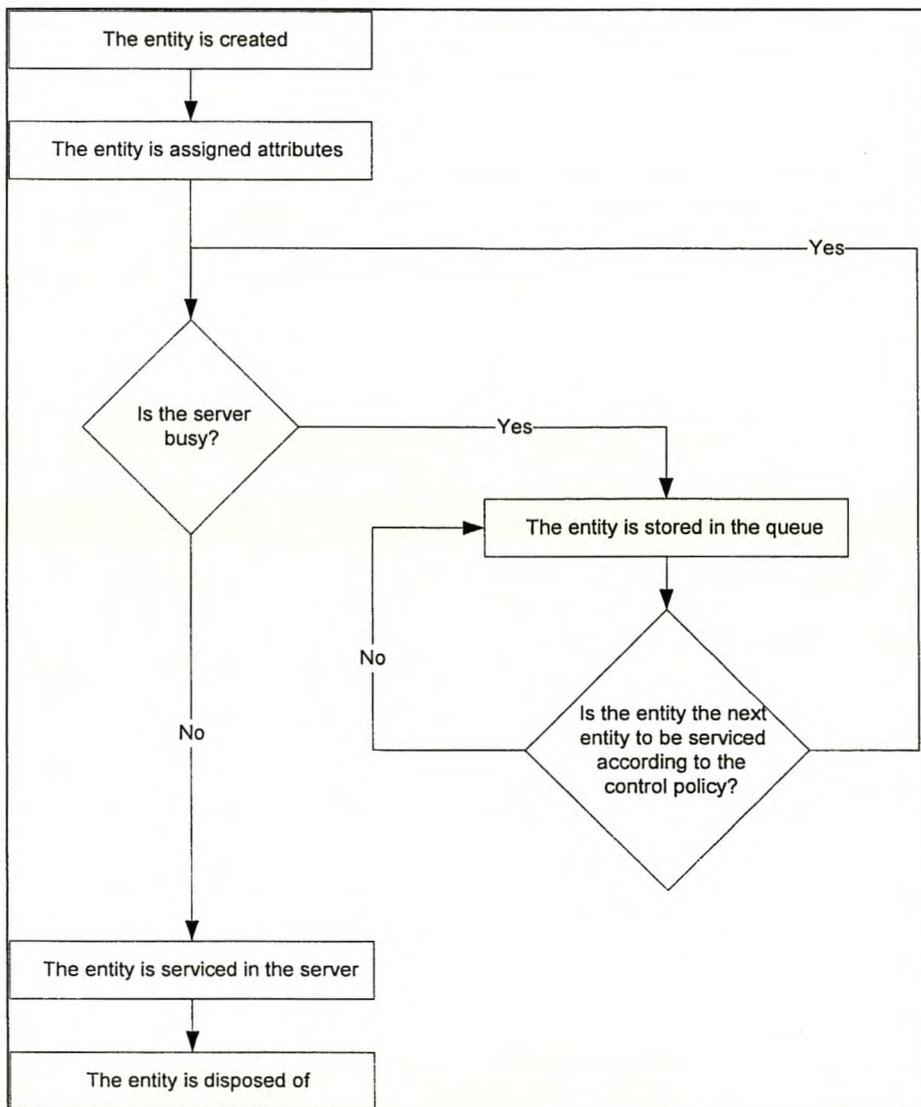


Figure 4.5 The concept model for the Arena® models

#### 4.2.2 *Entities*

To keep the models as generic as possible, the entities as used in this project are not assigned an identity, but possible examples of these entities could be manufacturing jobs that enter a manufacturing system or even people being served at a counter.

#### 4.2.3 *Attributes*

The attributes that are assigned to the entities once they are created are the time to the next arrival, their process time, their due time, the time they entered the system and the number of the entity. Their process times are taken from an exponential distribution. Their due time is the time when the entity should be finished and is computed as the time on arrival plus a random time taken from an exponential distribution. The Emulator only allows the process time and the due time interval to be taken from an exponential distribution, and using other distributions is a subject for further study. The user of the Emulator chooses the characteristics of the exponential distributions from which the time to the next arrival, the process time and the due time are sampled. The time they entered is simply the system time at system entry. The entity number is a unique number used to differentiate between entities and is computed by incrementing a variable every time a new entity is created and then assigning the new value of the variable to the new entity's number attribute.

#### 4.2.4 *Performance criteria*

In paragraph 2.2 on page 5 it was shown that the on-line planning and control process assumes that off-line planning has been performed and that the optimum set of design criteria for the operation of the system has been determined. This is not possible with the Emulator, because there is no actual real-world system, so a set of design criteria was chosen from those used in the literature.

The models use performance criteria to indicate the performance of the system. The following performance criteria are used in the Arena<sup>®</sup> models:

- a) The average time in system (TIS).
- b) The average process productivity (PP).
- c) The average lateness (LN).
- d) The average job productivity (JP).
- e) The average length of service queue (LSQ).

The average time an entity spends in the system is given by the equation:

$$\overline{TIS}_{wp} = \frac{\sum_{j=1}^J TIS_{jwp}}{J} \quad \text{Equation 4.1}$$

The average process productivity is the ratio of the time the server is busy to the total operational time and is given by the equation:

$$\overline{PP}_{wp} = \frac{\sum TimeServerBusy}{EndTime - StartTime} \quad \text{Equation 4.2}$$

The average lateness is given as the difference between the time at which the entity is disposed of and the time the entity should have been disposed of (if the time of disposal is later than the time it is due). It is given by the equation:

$$\overline{LN}_{wp} = \frac{\sum_{j=1}^J (DisposeTime_j - DueTime_j)}{J} \quad \text{If } DisposeTime_j > DueTime_j, \quad \text{Equation 4.3}$$

The average job productivity is calculated as the ratio of the total processing time per entity to the time the entity is in the system and is given by the equation:

$$\overline{JP}_{wp} = \frac{\sum_{j=1}^J \frac{PT_{jwp}}{TIS_{jwp}}}{J} \quad \text{Equation 4.4}$$

The average length of service queue is given by the equation:

$$\overline{LSQ}_{wp} = \frac{\sum_{b=0}^h TimeWith\_b\_entitiesInQueue}{EndTime - StartTime} \quad \text{Equation 4.5}$$

#### 4.2.5 Control policies

The models use different control policies to decide which of the entities in the queue should be serviced next. The different control policies are as follows:

- a) First in first out: The entity that has spent the longest time in the service queue is the next entity to be serviced.
- b) Last in first out: The entity that has spent the shortest time in the service queue is the next to be serviced.
- c) Latest job: The entity of which the due time is the smallest is the next to be serviced.

- d) Longest service time: The entity with the longest processing time is the next entity to be serviced.
- e) Shortest service time: The entity with the shortest service time is the next entity to be serviced.

The control policies are implemented in Arena® in the following way:

- a) First in first out. The server's queue ranking rule is changed to *LowValueFirst* and the expression set to *attEntityNumber*.
- b) Last in first out. The server's queue ranking rule is changed to *HighValueFirst* and the expression set to *attEntityNumber*.
- c) Latest job. The server's queue ranking rule is changed to *LowValueFirst* and the expression set to *attDueTime*.
- d) Longest service time. The server's queue ranking rule is changed to *HighValueFirst* and the expression set to *attProcessTime*.
- e) Shortest service time. The server's queue ranking rule is changed to *LowValueFirst* and the expression set to *attProcessTime*.

#### 4.2.6 Assumptions

The following assumptions allow the models to operate correctly:

- a) There is no time delay for the entity for transportation to and from the server.
- b) The server cannot fail.
- c) There is no limit on the number of entities that may be stored in the service queue.

#### 4.2.7 Initialising the models

One of the major obstacles to the on-line planning and control process is the initialisation of the alternative system models to the current state of the real-world system (and the real-world system model, once the alternative system model has been evaluated). To make it easier to initialise system models, Davis [3] developed a new modelling architecture and made the ease with which system models can be initialised one of its biggest features. Gonzales and Davis [9] showed how to initialise the state of the on-line simulation from the state of a physical operating system using this approach. Unfortunately, this architecture is not yet available in general simulation software.

Therefore, the project was not able to use this approach and the system models had to be initialised by the complex method described here.

The method starts once the real-world system model is stopped. The following are determined:

- a) Whether or not there is an entity in the server. This is possible with the *ResourceNumberBusy* method of the SIMAN<sup>®</sup> object of the real-world system model.
- b) The number of entities in the queue. This is possible with the *QueueNumberOfEntities* method of the SIMAN<sup>®</sup> object of the real-world system model.
- c) All the attributes of all the entities in the service queue. They can be determined with the *QueuedEntityAttribute* method of the SIMAN<sup>®</sup> object of the real-world system model.

These attributes are:

- ⇒ Time between arrivals.
  - ⇒ Entity number.
  - ⇒ Time entered.
  - ⇒ Process time.
  - ⇒ Time processing started (this is still equal to 0 because processing has not yet started).
  - ⇒ Due time.
  - ⇒ Queue time (this is a system generated attribute and does not concern us).
- d) The time the real-world system model was stopped. This is possible with the *RunCurrentTime* method of the SIMAN<sup>®</sup> object of the real-world system model.
  - e) The attributes of the entity being serviced. This is slightly more complex than the entities in the service queue. When the entity starts its service, its attributes are written to an Arena<sup>®</sup> variable. Thus, this variable always contains the attributes of the entity being serviced. These can be accessed with the *VariableArrayValue* method of the SIMAN<sup>®</sup> object of the real-world system model and used to compute the processing time still required by deducting the time since it started processing from the total processing time required. The rest of the attributes listed below are read from the variable.
    - ⇒ Time between arrivals.
    - ⇒ Entity number.
    - ⇒ Time entered.
    - ⇒ Process time.
    - ⇒ Time processing started.

- ⇒ Due time.
  - ⇒ Queue time (this is a system generated attribute and is not a concern).
- f) The time at which the next entity should be created. This is determined by summing the two variables that give the last time between arrivals and the time the delay started. These variables can be accessed with the *VariableArrayValue* method of the SIMAN<sup>®</sup> object of the real-world system model.
- g) The entity number of the next entity to enter. This is determined by summing the number of entities in the system and those that have already left plus one. Those that have already left can be found with the *CounterValue* method of the SIMAN<sup>®</sup> object of the real-world system model.
- h) The seed numbers to be used for the time between arrivals, processing times and due times. The calculation of the seed numbers is discussed in paragraph 4.2.8 on page 75.

To initialise the alternative system model (or the real-world system model, once the alternative system model has been evaluated) the following are done:

- a) The time of first creation is changed to the time the next entity should be created. This is done by finding the *Create* module in the model and changing its *Offset* data.
- b) The warm-up period is changed to the time of initialisation to ensure that the discrete change variables only start calculating at the time of initialisation. This is done by finding the *Simulate* module in the model and changing its *Warm-up* data.
- c) The time at which the entity that was in the server is injected into the system, is changed to the time of initialisation. This is done by finding the *ArrivalFirst* module in the model and changing its *Interval* data.
- d) The time at which the entities that were in the queue are injected into the system is changed to the time of initialisation. This is done by finding the *ArrivalRest* module in the model and changing its *Interval* data.
- e) The number of entities to be injected into the queue is changed to the number of entities that were in the queue. This is done by finding the *ArrivalRest* module in the model and changing its *BatchSize* data.
- f) The attributes of the entities injected into the system are assigned to them when the *Arrivals* module injects them into the model.
- g) The seed numbers are changed to the required seeds.

- h) The variable with the entity number of the new entities to enter the system is changed to reflect the other entities that have entered the system previously. This is accessed with the *VariableArrayValue* method of the SIMAN® object of the model.
- i) To ensure that the sequence of entities is correct, the server is seized until the time the entities arrive. This is done with the *ResourceCapacity* method of the SIMAN® object of the model.

#### 4.2.8 *Simulating the alternative systems under conditions as similar as possible*

The sample size for the comparison of the alternative systems is fixed. To gain more confidence in the results, the variance needs to be reduced. It was shown in paragraph 2.5.2 on page 20 that using common random numbers is an effective way of reducing the variance, if the systems are simple enough to allow it. The Emulator is simple enough, because it is only controlling a M/M/1/FIFO/ $\infty/\infty$  server.

The objective is to simulate the alternative systems under conditions as similar as possible. This necessitates the synchronisation of the random numbers across the alternatives by initialising each of the alternative system models' seed numbers so that they use the same seeds to generate entities similarly spaced with the same process times and due times.

The seed number initialisation works as follows. There are three places in the model where random numbers are used:

- a) The time between arrivals.
- b) The process times.
- c) The due time.

All of these are random samples taken from the exponential distribution and their random numbers are taken from different parts of the random number stream. To start, each of the three random number generators is initialised to a specific starting seed. This is then used to start the random number generation process in the usual way. When the real-world system model is stopped, the last random observation from the exponential distribution used ( $R_{\text{expo}}$ ) is collected for all three cases. If  $\beta$  represents the mean of the distribution, then the random number ( $U$ ), between 0 and 1, used to compute the next random sample from the exponential distribution, is given by:

$$U_i = e^{-\frac{R_{\text{expo}}}{\beta}} \tag{Equation 4.6}$$

Arena<sup>®</sup> uses the multiplicative congruential random number generating method, defined by the following equation:

$$Y_i = (aY_{i-1} + c) \bmod m \quad \text{Equation 4.7}$$

With  $m = 2^{31} - 1$

$$a = 7^5$$

$c = 0$ , and

$$U_i = \frac{Y_i}{m} \quad \text{Equation 4.8}$$

The  $U_i$  needs to be multiplied with  $m$  to get  $Y_i$ . In the initialised model, this value of  $Y_i$  is used as  $Y_{i-1}$  for the seed of the new random number generator. The result is that all of the alternative system models see exactly the same entities as the real-world system model and the only difference affecting the output of the alternative system is the alternative control policies.

#### 4.2.9 *Creating entities with time between arrivals from a specific distribution*

The time between entities entering the system could not be specified with an *Arrive* module, because then it would not have been possible to control the time at which the next entity should be created when the system model is initialised. Therefore, a different method had to be followed. A concept model of this method of creating entities is shown in Figure 4.6. This method creates one entity and duplicates it. The time between duplications is then set to a random observation from the specified distribution, resulting in entities arriving with the time between their arrivals from the same distribution.

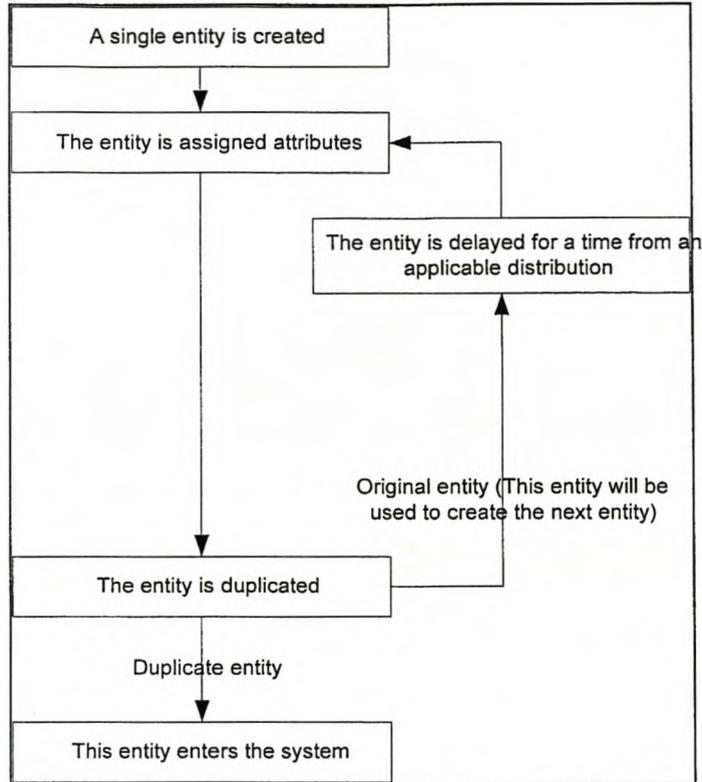


Figure 4.6 The concept model of the model logic of the creation of entities

This concludes the discussion of the Arena<sup>®</sup> part of the Emulator.

### 4.3 The Emulator programmed in Visual Basic<sup>®</sup>

The rest of the Emulator consists mainly of Visual Basic<sup>®</sup> routines. A schematic block diagram and the functions and subprograms used are discussed next. The complete code for the Emulator as well as detailed descriptions of all the subprograms and functions are given in Appendix B.

#### 4.3.1 Schematic block diagram of the Visual Basic<sup>®</sup> code

In Figure 4.7, the main subprograms, their sequence and the decisions that constitute the Emulator are shown. Table 4.1 describes the subprograms that are shown in Figure 4.7.

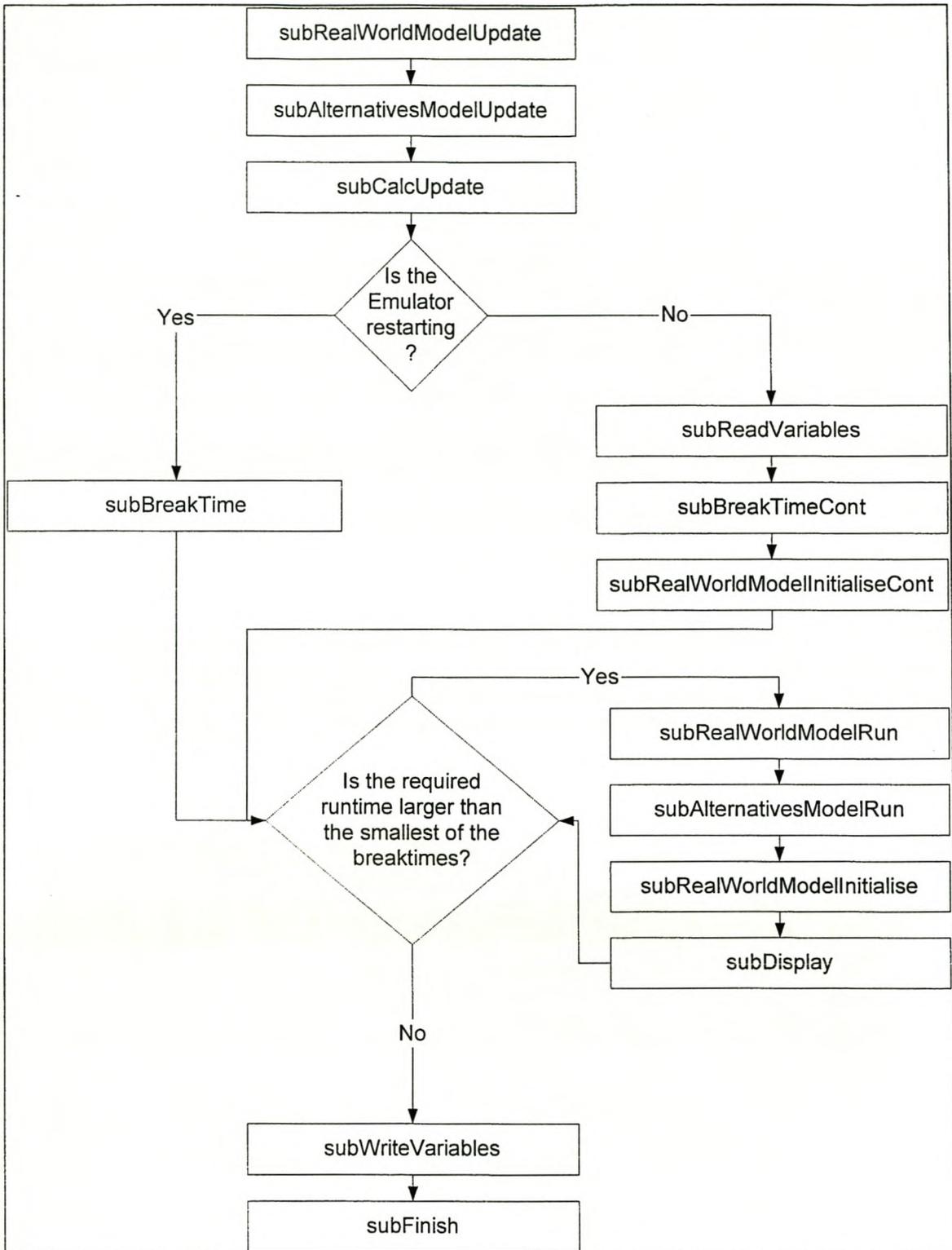


Figure 4.7 Schematic block diagram of the Visual Basic® routines

Table 4.1 The main subprograms of the Emulator

Subprogram	Domain	Usage
<i>subRealWorldModelUpdate</i>	Public	This subprogram updates the real-world system model. It also writes the run specific data to a specified file and opens files for output.
<i>subAlternativesModelUpdate</i>	Public	This subprogram updates the alternative system model.
<i>subCalcUpdate</i>	Public	This subprogram updates the variables and arrays to be used for the calculations.
<i>subBreakTime</i>	Public	This subprogram determines the first breaktimes, enables manual override and disables the buttons and input boxes during operation.
<i>subBreakTimeCont</i>	Public	This subprogram disables the buttons and input boxes during operation for the case where the Emulator is continuing.
<i>subRealWorldModelInitialiseCont</i>	Public	This subprogram initialises the real-world system model to the state it was in before the Emulator stopped.
<i>subRealWorldModelRun</i>	Public	This subprogram runs the real-world system model. It then determines the end state of the real-world system model to which the alternative system model should be initialised.
<i>subAlternativesModelRun</i>	Public	This subprogram decides on the control policy to be evaluated and initialises the alternative system model to the state the real-world system model ended in. It then runs the alternative system model and writes the results of the performance criteria to file.

Subprogram	Domain	Usage
<i>subRealWorldModelInitialise</i>	Public	This subprogram initialises the real-world system model to the state it was in before the alternative system model was evaluated and determines whether the control policy needs to change or not.
<i>subDisplay</i>	Public	This subprogram handles the display of the progress of the Emulator and writes it to an output file.
<i>subFinish</i>	Public	This subprogram closes up everything once the required runtime has been reached.
<i>subReadVariables</i>	Public	This subprogram reads all the variables saved in the previous run from a file.
<i>subWriteVariables</i>	Public	This subprogram writes all the variables needed in the next run to a file.

#### 4.4 Validating and verifying the Emulator

Validation refers to the process of determining whether or not the model is an adequate representation of the real-world system. In this case, the model is the Emulator, and because there is no actual real-world on-line planning and control process, it is assumed that the Emulator is a valid representation of some on-line planning and control process.

Verification refers to the process of determining whether the model operates correctly or not. In this case, the model is the Emulator, and verifying proved to be more difficult than originally envisaged. It was tested step by step as it was developed, correcting syntax errors and logic mistakes, but it proved impossible to validate the completed Emulator completely. Some verification procedures that were possible are described below.

##### 4.4.1 Verifying the initialising of the models

A previous version of the Emulator used a standalone Visual Basic® shell and not Arena®'s Visual Basic® (See Appendix E). Under this version, it was possible to verify the initialising process of the Emulator. The Emulator was run to a specific time where the real-world system model was

stopped. The number of entities in the system, as well as their attributes, were noted. Then the Emulator was allowed to initialise the alternative system model and it was confirmed that the entities and their attributes correspond to those in the real-world system model. The same was done for the re-initialising of the real-world system model. Unfortunately, when Arena's Visual Basic is used, it is not possible to manipulate the Arena models once the Visual Basic program has been stopped. This means that this version's initialising could not be verified, but because it has not been changed from the previous version, it is assumed to be correct.

#### *4.4.2 Verifying the breaking up of the runs*

To verify that it is possible to break-up the runs into a combination of shorter runs, the results of a broken run were compared with those of an unbroken run. It was found that the results immediately after the break were identical to those from the unbroken run, but that they deviated slightly later on. This can be attributed to the breaktimes for the evaluation of the alternative system model not being at the same times, which leads to slightly different results.

## **4.5 Summary**

The chapter showed that it is possible to construct an emulator of Davis [3]'s on-line planning and control process by using Arena models and Visual Basic programming. However, the Emulator is not a complete emulation. Figure 4.8 is a replication of Figure 2.1 from page 5 with the parts that are not used in the Emulator or changed in the Emulator, removed or changed respectively. The block representing the real-world system is changed to represent the real-world system model, because the real-world system is represented by a model. The blocks representing the autovalidation process and the alternative control policy generator are removed, because the technology to construct the autovalidation capability does not currently exist and not much research has been done on the procedures for generating alternative control policies.

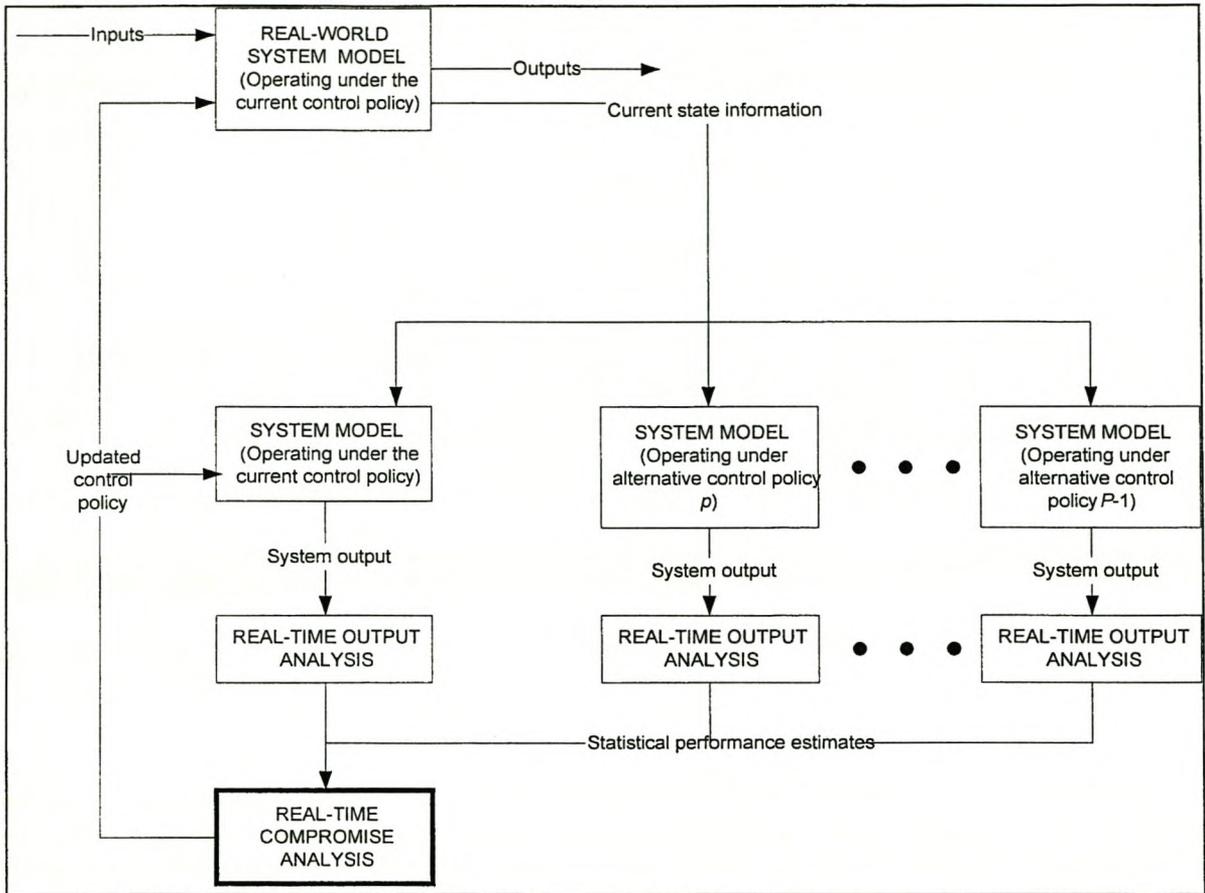


Figure 4.8 The parts of Davis [3]'s on-line planning and control process represented in the Emulator

While discussing the Emulator, some subjects that require further study in other projects were noted. These are summarised in Table 4.2.

Table 4.2 Subjects that require further study, as identified during the development of the Emulator

Subject	Paragraph	Page
The development of the Emulator in a language that allows concurrent operations and effortless initialisation.	4.1	62
The actual time the evaluation of the alternative takes and the effect of this time on the Emulator.	4.1.2	66
Allowing the Emulator to use distributions other than the exponential distribution for the time between arrivals.	4.2.1	69
Allowing the Emulator to use distributions other than the exponential distribution for the process time and the random time added to the time of arrival to compute the due time.	4.2.3	70

The Emulator is used to evaluate the techniques developed in Chapter 3 (page 29 and on). The results of the evaluation are presented in Chapter 5, starting on page 84.

## 5 EVALUATING THE NEW TECHNIQUES

The Emulator described in Chapter 4 (page 61) is used to evaluate the two developed techniques, as discussed in Chapter 3 (page 29). The Emulator generates the data and this data is then analysed statistically to determine whether the techniques are useful or not. The techniques are evaluated for traffic intensities of both 0.7 and 0.9. The evaluation is discussed in this chapter.

The first part of the analysis consists of an off-line non-terminating analysis of the system for both of the traffic intensities. This is used as a base line against which the on-line planning and control process of the Emulator can be evaluated. The on-line planning and control process of the Emulator is also evaluated and the results for the specific traffic intensities compared. The final part is a discussion of the results in general.

This chapter only shows the data that contributes to the evaluation of the techniques. The detailed data sets are given in Appendix F.

### 5.1 Off-line non-terminating analysis for a traffic intensity of 0.7

Before the on-line planning and control process can be evaluated for a traffic intensity of 0.7, the Emulator must first be used to generate data that can be used as a base line for evaluating the two techniques. The Emulator is used to evaluate all five the different control policies on their own in an off-line non-terminating manner.

#### 5.1.1 *Work method*

The runs for the different control policies were made on a single computer. The Emulator was set to "Manual override" which ensures that no control policies are evaluated, and the real-world system model only runs for the required run length with the control policy specified. A run of 1 000 000 simulated minutes takes between five and fifteen minutes on a 600MHz Pentium® III computer, depending on the control policy evaluated.

The output from the Emulator is a text file, containing comma-separated values for each of the five performance criteria. Each of these files needs to be opened in Word® and the commas replaced with spaces. This cannot be done in Excel® or WordPad®, because they cannot handle files this big. Then the files are converted to Arena® output files by using Arena®'s Output analyser's Data file loading function. These files can then be analysed in the usual way with Arena®'s Output analyser.

### 5.1.2 Preliminary experiment

The variables of the Emulator need to be set to certain values before the Emulator is run.

Table 5.1 gives the variable settings for the off-line non-terminating analysis for a traffic intensity of 0.7.

Table 5.1 Variable settings for off-line non-terminating analysis for a traffic intensity of 0.7

Variable	Setting	Explanation
Mean of exponential distribution of time between arrivals (minutes)	10	This should lead to the required traffic intensity of 0.7.
Mean of exponential distribution of process times (minutes)	7	
Mean of exponential distribution of due times (minutes)	20	This value needed to be made up because there is no real-world system where the mean of the exponential distribution of the due times could be determined from. This value seems right, because it is twice as long as the expected time between arrivals. However, this subject requires further study.

### 5.1.3 Method for statistical analysis

The method used is based on the method given by Kelton, *et al* [11] for steady state simulation with batching in a single run, as well as the method described by Law, *et al* in [12] for evaluating non-terminating systems concerned with the long term or steady state behaviour of the system. The batch means approach is used to obtain statistically independent observations.

The truncation points are determined as the time and observation at which the system has reached steady state. The truncation point is determined by inspection from a moving average plot. A typical example of the truncation point determination is shown in Figure 5.1 for the performance criterion **Time in system** for the Longest service time control policy. The warm-up period may seem very long, but because the model is so simple, the runs are very fast and it is thus better to err on the safe side.

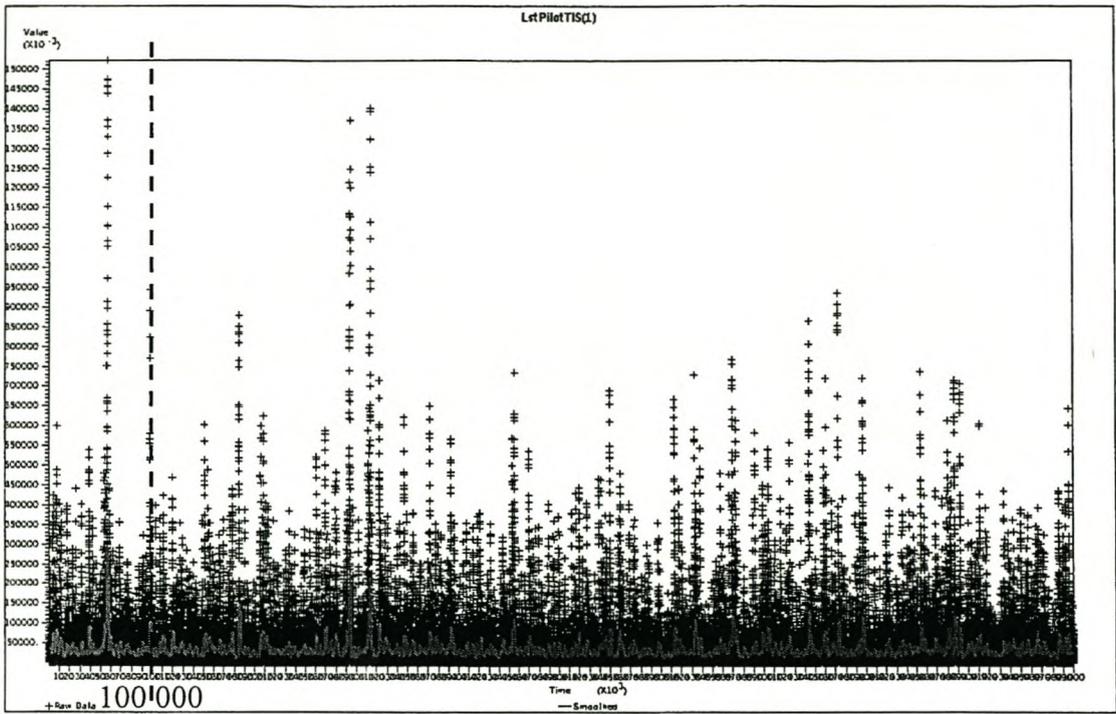


Figure 5.1 Moving average plot of Time in system for Longest service time

A pilot run is made to begin with, and if necessary a production run follows. The discrete change variables need to be batched first to allow the correlogram function in Arena<sup>®</sup>'s Output analyser to work. These batches are called mini-batches.

Arena<sup>®</sup>'s correlogram function is used to determine the lag number where the correlation is approximately zero. An example is given in Figure 5.2 for the performance criterion **Lateness** for the control policy Last in first out. In this case, the correlation is assumed zero at a lag number of 80.

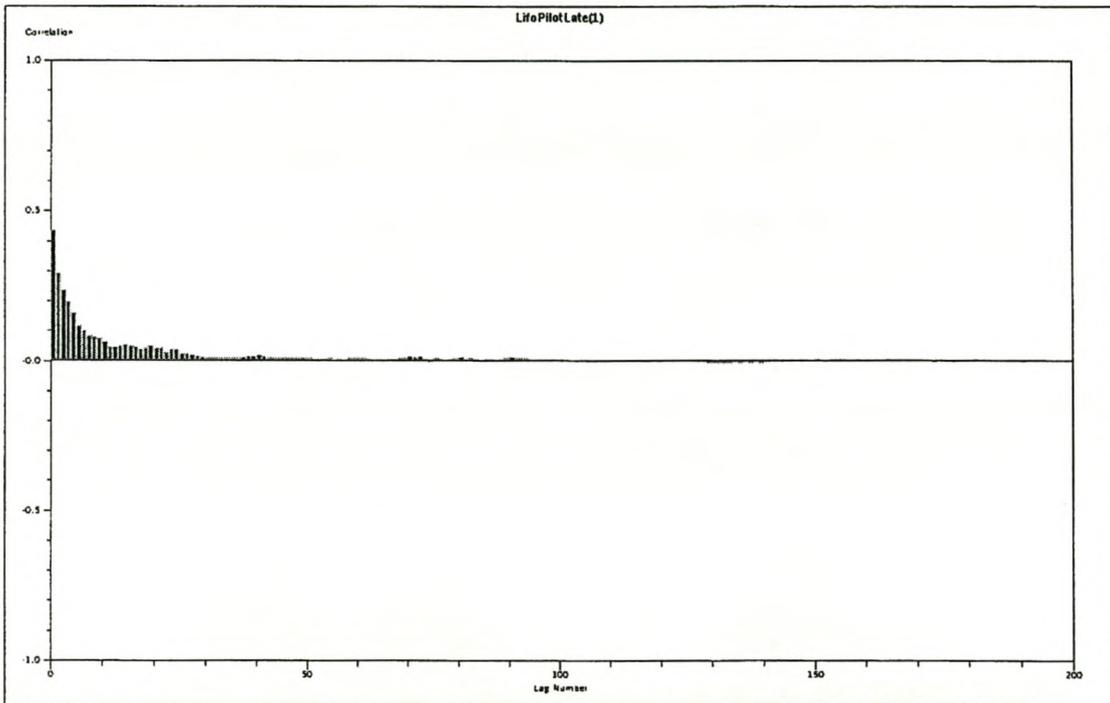


Figure 5.2 Correlogram for the performance criterion **Lateness** for Last in first out

For observation-based batching the batch size used is determined as ten times the lag number where the correlation is approximately zero and for time-based batching the batch size used is determined as ten times the time where the correlation is approximately zero, times the size of the mini-batch. In Figure 5.3 it can be seen that there is no significant correlation in the observation-based batched data when the data is batched in batch sizes of 800 observations.

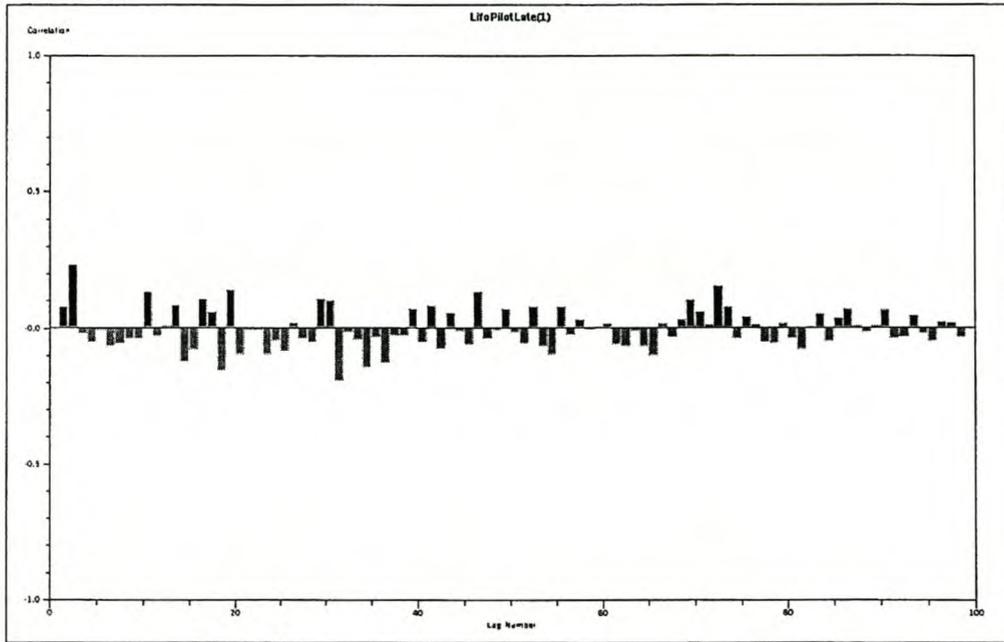


Figure 5.3 Correlogram for the performance criterion **Lateness** for Last in first out batched in batches of size 800

In the case where the pilot run did not result in small enough confidence intervals, the following method can be used to determine the required number of batches that may reduce the half width to the required half width. If  $h$  is the confidence interval half width,  $h^*$  is the desired confidence interval half width,  $n$  is the number of batches and  $n^*$  is the required number of batches,  $n^*$  can be determined by:

$$n^* = n \times \left( \frac{h}{h^*} \right)^2 \quad \text{Equation 5.1}$$

The required run length is determined as  $n^*$  times the time length per batch with the warm-up period added for time-batched observations, and  $n^*$  divided by  $n$  times the length in time units of the pilot run with the warm-up period added for observation-batched observations. The production run will be done for this required run length.

Then the averages of the observations per batch can be determined and the confidence intervals drawn per criterion. Table 5.2 gives the data for the pilot runs. The runtime is chosen as the shortest time that can be emulated with the Emulator, while still expecting reasonable results.

The required half widths are determined as the half widths that can be achieved by the two techniques of the Emulator for the specified runtime.

Table 5.2 Data for pilot runs for a traffic intensity of 0.7

Variable	Setting
Total runtime (simulation minutes)	1 000 000
Truncation point (simulation minutes)	100 000
Required half width	
Time in system (minutes)	1
Process productivity (ratio)	0.01
Lateness (minutes)	1
Job productivity (ratio)	0.01
Length of service queue (entities)	0.1
Size of mini-batch (minutes)	10

5.1.4 Statistical analysis for First in first out pilot run for a traffic intensity of 0.7

The information on the batching of the data is given in Table 5.3. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.3 Batch size information for First in first out for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	75	750
Process productivity	Time (minutes)	23	2 300
Lateness	Observation	80	800
Job productivity	Observation	50	500
Length of service queue	Time (minutes)	70	7 000

The results for the First in first out pilot run are shown in Table 5.4. All tables containing results are shaded. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.4 Results for First in first out pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	22.8	0.803	1
Process productivity (ratio)	0.692	0.006 45	0.01
Lateness (minutes)	12.1	0.627	1
Job productivity (ratio)	0.524	0.007 63	0.01
Length of service queue (entities)	1.57	0.863	0.1

5.1.5 Statistical analysis for Last in first out pilot run for a traffic intensity of 0.7

The information on the batching of the data is given in Table 5.5. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.5 Batch size information for Last in first out for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	50	500
Process productivity	Time (minutes)	30	3 000
Lateness	Observation	80	800
Job productivity	Observation	30	300
Length of service queue	Time (minutes)	100	10 000

The results for the Last in first out pilot run are shown in Table 5.6. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure

that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.6 Results for Last in first out pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	22.7	0.831	1
Process productivity (ratio)	0.692	0.006 19	0.01
Lateness (minutes)	13.6	0.695	1
Job productivity (ratio)	0.608	0.005	0.01
Length of service queue (entities)	1.56	0.080 5	0.1

5.1.6 *Statistical analysis for Latest job pilot run for a traffic intensity of 0.7*

The information on the batching of the data is given in Table 5.7. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.7 Batch size information for Latest job for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	70	700
Process productivity	Time (minutes)	30	3 000
Lateness	Observation	90	900
Job productivity	Observation	40	400
Length of service queue	Time (minutes)	80	8 000

The results for the Latest job pilot run are shown in Table 5.8. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.8 Results for Latest job pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	22.7	0.798	1
Process productivity (ratio)	0.692	0.006 19	0.01
Lateness (minutes)	11.2	0.646	1
Job productivity (ratio)	0.54	0.007 35	0.01
Length of service queue (entities)	1.57	0.080 6	0.1

5.1.7 Statistical analysis for Longest service time pilot run for a traffic intensity of 0.7

The information on the batching of the data is given in Table 5.9. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.9 Batch size information for Longest service time for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	100	1 000
Process productivity	Time (minutes)	25	2 500
Lateness	Observation	110	1 100
Job productivity	Observation	30	300
Length of service queue	Time (minutes)	110	11 000

The results for the Longest service time pilot run are shown in Table 5.1. Unlike the cases of the previous control policies, the pilot run for the Longest service time did not reach the required half width. To determine the required run length, the following are shown for every performance criterion: the average, the number of batches ( $n$ ), the achieved half width ( $b$ ), the required half width ( $b^*$ ) and the required number of batches ( $n^*$ ). This enables the determination of the required run length for every performance criterion. The largest of these, in this case **Length of**

**service queue** (about 4 000 000), is used as the length for the production run. The full results are shown in Appendix F.

Table 5.10 Results for Longest service time pilot run for a traffic intensity of 0.7

Performance criterion	Average	$n$	$b$	$b^*$	$n^*$	Required run length (minutes)
Time in system (minutes)	35.8	89	1.95	1	338.42	3 902 472
Process productivity (ratio)	0.692	359	0.006 42	0.01	n.a	n.a
Lateness (minutes)	25	81	1.92	1	298.6	3 786 420
Job productivity (ratio)	0.558	298	0.006 24	0.01	n.a	n.a
Length of service queue (entities)	2.86	81	0.21	0.1	357.21	4 029 310

5.1.8 *Statistical analysis for Longest service time production run for a traffic intensity of 0.7*

The specific data for the Longest service time production run is shown in Table 5.11.

Table 5.11 Specific data for Longest service time production run for a traffic intensity of 0.7

Variable	Setting
Total runtime (simulation minutes)	4 000 000
Truncation point (simulation minutes)	100 000

The results for the Longest service time production run are shown in Table 5.12. For every performance criterion, the average, half width for the production run, required half width and half width for the pilot run are shown. The latter attempts to ensure that the half width for the production run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.12 Results for Longest service time production run for a traffic intensity of 0.7

Performance criterion	Average	Half width for production run	Required half width	Half width for pilot run
Time in system (minutes)	37.3	0.978	1	1.95
Process productivity (ratio)	0.698	0.002 99	0.01	0.006 42
Lateness (minutes)	26.4	0.964	1	1.92
Job productivity (ratio)	0.552	0.002 86	0.01	0.006 24
Length of service queue (entities)	3.03	0.103	0.1	0.21

5.1.9 Statistical analysis for Shortest service time pilot run for a traffic intensity of 0.7

The information on the batching of the data is given in Table 5.13. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.13 Batch size information for Shortest service time

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	50	500
Process productivity	Time (minutes)	30	3 000
Lateness	Observation	50	500
Job productivity	Observation	50	500
Length of service queue	Time (minutes)	50	5 000

The results for the Shortest service time pilot run are shown in Table 5.14. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.14 Results for Shortest service time pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	15.9	0.341	1
Process productivity (ratio)	0.692	0.006 19	0.01
Lateness (minutes)	7.39	0.279	1
Job productivity (ratio)	0.596	0.005 26	0.01
Length of service queue (entities)	0.887	0.032 9	0.1

This concludes the reporting and discussion of the results of the off-line non-terminating analysis for a traffic intensity of 0.7.

## 5.2 On-line planning and control for a traffic intensity of 0.7

Now that the off-line non-terminating analysis for a traffic density of 0.7 has been finished, the Emulator is used to generate data that is used to evaluate the on-line planning and control process for a traffic intensity of 0.7.

### 5.2.1 Work method

The runs for the different techniques were made on a single computer. A run of 1 000 000 simulated minutes takes 160 to 200 hours on a 600 MHz Pentium® III computer, depending on the technique used. The First technique evaluates faster than the Second technique. The run length of 160 to 200 hours may give rise some concerns towards the viability of the techniques. However, it should be noted that the run length of 1 000 000 simulated minutes is only necessary to evaluate the efficiency of the techniques, because the techniques need to be implemented repeatedly before any statistical conclusions about the their efficiency can be made. After the techniques have been proved to be efficient, they can be used in a real-world system where the only time delay in a single evaluation of the technique would be the evaluation of the alternatives and the subsequent comparison between the current control policy and the alternative control policy.

The output from the Emulator is a set of text files for each of the five performance criteria. Each text file of the set is the result of a run and follows upon the previous file. These sets need to be merged and converted as explained in paragraph 5.1.1 on page 84.

### 5.2.2 Preliminary experiment

The variables of the Emulator need to be set to certain values before the Emulator is run. Table 5.15 gives the values of the variable settings for the on-line planning and control process for a traffic intensity of 0.7.

Table 5.15 Emulator variable settings for a traffic intensity of 0.7

Variable	Setting	Explanation
Mean of exponential distribution of time between arrivals (minutes)	10	This should lead to the required traffic intensity of 0.7.
Mean of exponential distribution of process times (minutes)	7	
Mean of exponential distribution of due times (minutes)	20	This value is kept consistent with the value chosen for the off-line non-terminating analysis in Table 5.1 on page 85.
Evaluation period (minutes)	1 000	This value needed to be made up because there is no real-world system where the required evaluation period could be determined. This value seems right, because it is about a week, which is a good planning horizon. However, this subject requires further study.
Mean of exponential distribution of simulation delay (minutes)	10	This value needed to be made up because there is no real-world system where the time of the simulation period could be determined. This value seems right, because ten minutes is comparable with the time a simulation run may take. However,

Variable	Setting	Explanation
		this subject requires further study.
The starting control policy	Alternative 1	This does not have an effect on the results. See paragraph 2.7.3 on page 26. However, this subject requires further study.
Maximum sample size	500	This value was chosen because it is ten times larger than 50 (transient analysis) and still significantly smaller than 2 000 (steady state). See paragraph 2.7.3 on page 26. However, this subject requires further study.
Confidence level for First technique (%)	95	This is the usual setting for confidence levels. However, this subject requires further study.
Importance of the performance criteria	5	Having all the importances the same enables the comparison of apples with apples by setting all to 5. However, this subject requires further study.
Whether performance criterion 1 should be minimised or maximised	Minimised	These were chosen specifically to propagate switching. However, this subject requires further study.
Whether performance criterion 2 should be minimised or maximised	Minimised	
Whether performance criterion 3 should be minimised or maximised	Maximised	
Whether performance criterion 4 should be minimised or maximised	Minimised	
Whether performance criterion 5 should be minimised or maximised	Maximised	
Whether performance criterion 5 should be minimised or maximised	Maximised	

### 5.2.3 Method for statistical analysis

The method used is based on the method given by Kelton, *et al* [11] for steady state simulation with batching in a single run, as well as the method described by Law, *et al* in [12] for evaluating non-terminating systems concerned with the longterm or steady state behaviour of the system. In this case, however, the model never reaches steady state, and thus it is unnecessary to seek the truncation point; it is set to the time when every alternative system has been evaluated 500 times.

To obtain statistically independent observations, the batch means approach is used. It should be noted that the batch means approach is only valid for steady state operations. However, in this case the system never reaches steady state and no methods exist to examine non-steady state systems, leaving no alternative. Another reason why the batch means method is used is that this enables the Emulator to run for the entire run length without having to re-initialise the seed numbers.

The discrete change variables need to be batched first to allow the correlogram function in Arena<sup>®</sup>'s Output analyser to work and also to discard the extra unnecessary observations generated by starting and stopping the real-world system model. These batches are called mini-batches.

Arena<sup>®</sup>'s correlogram function is used again to determine the lag number where the correlation is approximately zero. An example is given in Figure 5.4 for the performance criterion **Job productivity** for the First technique. In this case, the correlation is assumed zero at lag 35.

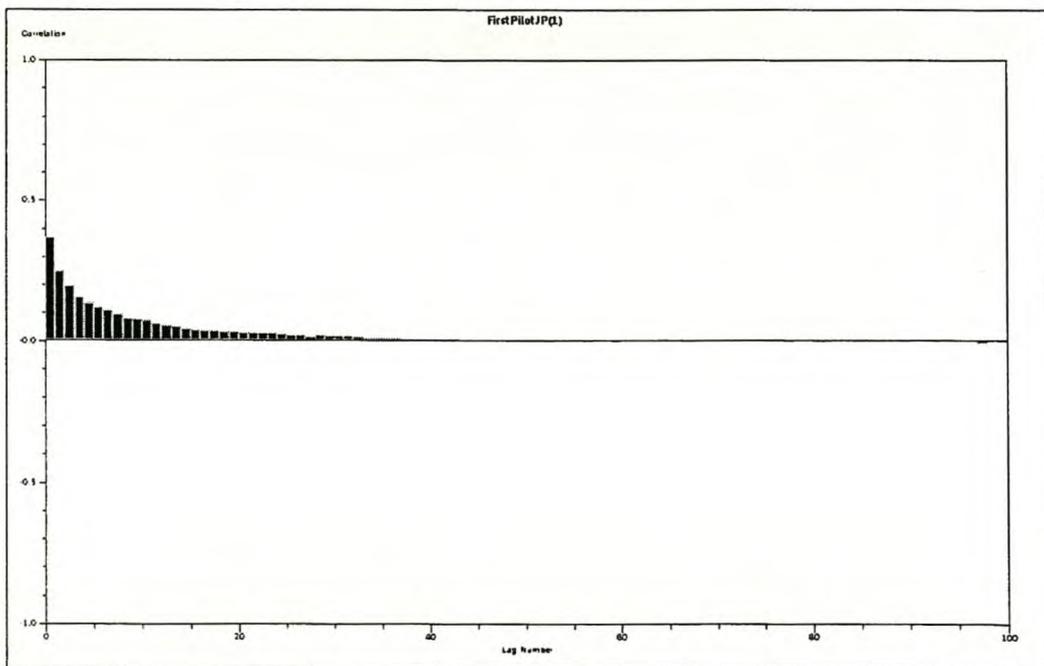


Figure 5.4 Correlogram for the performance criterion **Job productivity** for the First technique

For observation-based batching the batch size used is determined as ten times the lag number where the correlation is approximately zero and for time-based batching the batch size used is determined as ten times the time where the correlation is approximately zero times the size of the mini-batch. In Figure 5.5 it can be seen that there is no significant correlation in the observation-based batched data when the data is batched in batch sizes of 350 observations.

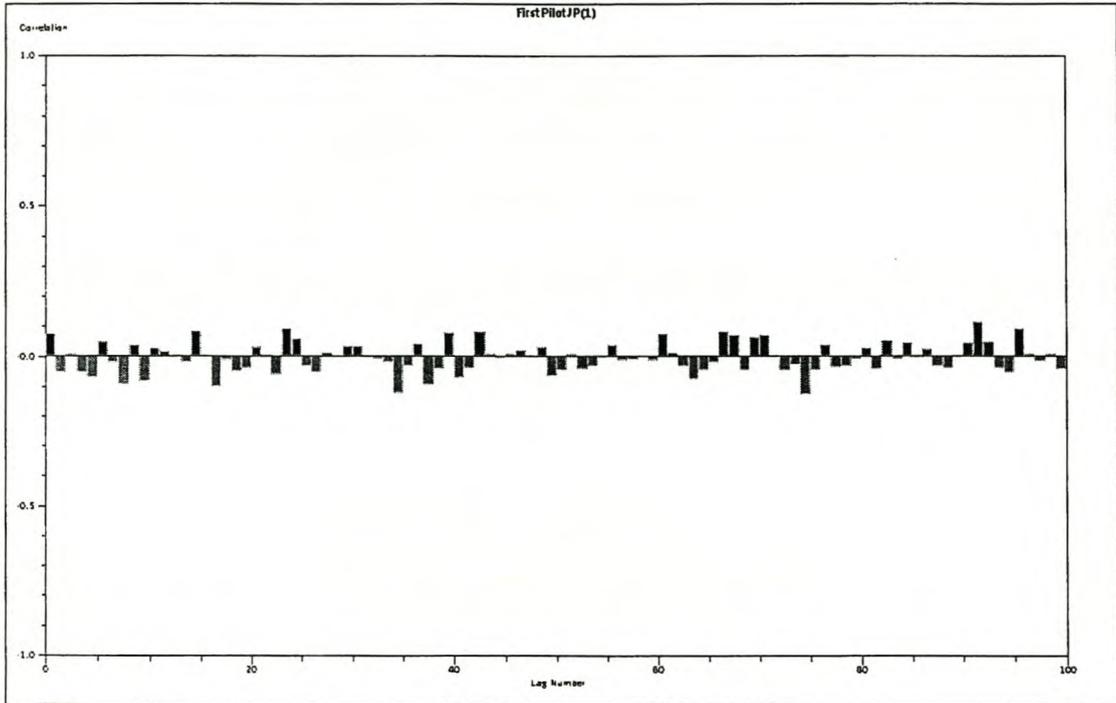


Figure 5.5 Correlogram for the performance criterion **Job productivity** for the First technique with batch size 350

The pilot run is done for the required run length. Then the averages of the observations per batch can be determined and the confidence intervals drawn per criterion.

The use of the different control policies is determined as the time the real-world system model used that specific control policy divided by the total runtime of the Emulator. The average number of trials to the next switch is determined as the number of switches the Emulator made, divided by the number of trials. The program shown in Appendix D is used to determine the percentage time using a specific control policy, and to merge the files.

5.2.4 *Statistical analysis for First technique pilot run for a traffic intensity of 0.7*

The specific data for the First technique pilot run is shown in Table 5.16.

Table 5.16 Specific data for First technique pilot run for a traffic intensity of 0.7

Variable	Setting
Total runtime (simulation minutes)	1 009 982
Truncation point (simulation minutes)	5 028
Size of mini-batch (minutes)	10

The information on the batching of the data is given in Table 5.17. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.17 Batch size information for First technique pilot run for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	75	750
Process productivity	Time (minutes)	25	2 500
Lateness	Observation	75	750
Job productivity	Observation	35	350
Length of service queue	Time (minutes)	75	750

The results for the first technique pilot run are shown in Table 5.18. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.18 Results for First technique pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	22.9	0.849	1
Process productivity (ratio)	0.693	0.005 94	0.01
Lateness (minutes)	13.1	0.766	1
Job productivity (ratio)	0.564	0.005 61	0.01
Length of service queue (entities)	1.58	0.088 7	0.1

5.2.5 Control policy switching information for First technique pilot run for a traffic intensity of 0.7

The information on the switching of control policies is shown in Table 5.19.

Table 5.19 Control policy switching information for First technique pilot run for a traffic intensity of 0.7

Control policy	Usage (%)	Average number of trials to next switch
First in first out	55.23	32.06
Last in first out	5.15	
Latest job	5.83	
Longest service time	33.28	
Shortest service time	0.5	

This concludes the reporting on the results and the analysis of the First technique. The results and analysis of the Second technique will be discussed subsequently.

### 5.2.6 Statistical analysis for Second technique pilot run for a traffic intensity of 0.7

The specific data for the Second technique pilot run is shown in Table 5.20.

Table 5.20 Specific data for Second technique pilot run for a traffic intensity of 0.7

Variable	Setting
Total runtime (simulation minutes)	1 009 970
Truncation point (simulation minutes)	5 028
Size of mini-batch (minutes)	10

The information on the batching of the data is given in Table 5.21 For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.21 Batch size information for Second technique pilot run for a traffic intensity of 0.7

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	90	900
Process productivity	Time (minutes)	25	2 500
Lateness	Observation	100	1 000
Job productivity	Observation	40	400
Length of service queue	Time (minutes)	75	7 500

The results for the Second technique pilot run are shown in Table 5.22. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.22 Results for Second technique pilot run for a traffic intensity of 0.7

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	22.9	0.846	1
Process productivity (ratio)	0.693	0.005 94	0.01
Lateness (minutes)	13.1	0.716	1
Job productivity (ratio)	0.564	0.005 77	0.01
Length of service queue (entities)	1.58	0.088 7	0.1

5.2.7 Control policy switching information for Second technique pilot run for a traffic intensity of 0.7

The information on the switching of control policies is shown in Table 5.23.

Table 5.23 Control policy switching information for Second technique pilot run for a traffic intensity of 0.7

Control policy	Usage (%)	Average number of trials to next switch
First in first out	19.14	6.29
Last in first out	22.25	
Latest job	20.85	
Longest service time	18.73	
Shortest service time	19.03	

This concludes the reporting on the results and the analysis of the Second technique.

5.3 Combining the results for a traffic intensity of 0.7

The combined results from the off-line non-terminating analysis and the on-line planning and control for a traffic intensity of 0.7 are shown in Table 5.24. At first glance these results may be rather disconcerting, but before becoming despondent, another traffic intensity should be evaluated to gain a better view of the situation. In the next section, the analysis is repeated for a

traffic intensity of 0.9. The combined results for both the traffic intensities considered are discussed in paragraph 5.7 on page 121.

Table 5.24 Combined results for a traffic intensity of 0.7

	Min/Max	Off-line analysis					On-line analysis	
		First in first out	Last in first out	Latest job	Longest service time	Shortest service time	First technique	Second technique
Time in system (minutes)	Min	22.8	22.7	22.7	37.3	15.9	22.9	22.9
Process productivity (ratio)	Min	0.692	0.692	0.692	0.698	0.692	0.693	0.693
Lateness (minutes)	Max	12.1	13.6	11.2	26.4	7.39	13.1	13.1
Job productivity (ratio)	Min	0.524	0.608	0.54	0.552	0.596	0.564	0.564
Length of service queue (entities)	Max	1.57	1.56	1.57	3.03	0.887	1.58	1.58

#### 5.4 Off-line non-terminating analysis for a traffic intensity of 0.9

The next traffic intensity to be evaluated is 0.9. The traffic density was increased to increase the average queue length, thus forcing the control policies to have more effect. Before the on-line planning and control process can be evaluated for a traffic intensity of 0.9, the Emulator must first be used to generate data that can be used as a base line for evaluating the two techniques. The Emulator is used to evaluate all five the different control policies on their own in an off-line non-terminating manner for a traffic intensity of 0.9.

##### 5.4.1 Work method

The work method is the same as discussed in paragraph 5.1.1 on page 84.

##### 5.4.2 Preliminary experiment

Table 5.25 gives the variable settings for the off-line non-terminating analysis for a traffic intensity of 0.9.

Table 5.25 Variable settings for off-line non-terminating analysis for a traffic intensity of 0.9

Variable	Setting	Explanation
Mean of exponential distribution of time between arrivals (minutes)	10	This should lead to the required traffic intensity of 0.9.
Mean of exponential distribution of process times (minutes)	9	
Mean of exponential distribution of due times (minutes)	20	This value is kept consistent with the value chosen for the off-line non-terminating analysis of a traffic intensity of 0.7 in Table 5.1 on page 85.

##### 5.4.3 Method for statistical analysis

The method for statistical analysis is the same as discussed in paragraph 5.1.3 on page 85.

The truncation point is again determined by inspection. An example is shown in Figure 5.6 for the performance criterion **Length of service queue** for the control policy First in first out.

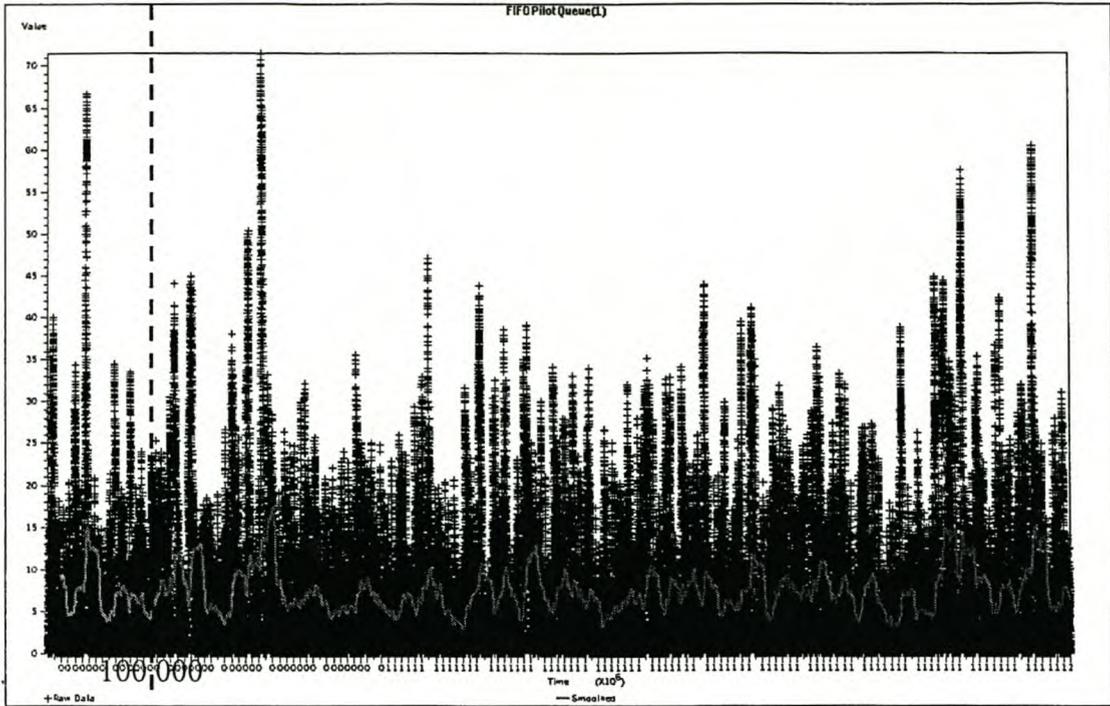


Figure 5.6 Moving average plot of Length of service queue for First in first out

The data for the pilot runs is shown in Table 5.26. The runtime is longer than the runtime used for the traffic intensity of 0.7, because the system is more congested, resulting in more variation that needs to be accommodated. The required half widths are once again determined as the half widths that can be achieved by the two techniques of the Emulator for the specified runtime. The half widths are the same as those for a traffic intensity of 0.7 (Table 5.2 on page 89), except for **Lateness**, where the half width is increased to 10. This ensures that the required total runtime stays within a viable limit.

Table 5.26 Data for pilot runs for a traffic intensity of 0.9

Variable	Setting
Total runtime (simulation minutes)	1 500 000
Truncation point (simulation minutes)	100 000
Required half width	
Time in system (minutes)	10
Process productivity (ratio)	0.01
Lateness (minutes)	10
Job productivity (ratio)	0.01

Variable	Setting
Length of service queue (entities)	1
Size of mini-batch (minutes)	10

#### 5.4.4 Statistical analysis for First in first out pilot run for a traffic intensity of 0.9

The information on the batching of the data is given in Table 5.27. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.27 Batch size information for First in first out for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	420	4 200
Process productivity	Time (minutes)	125	12 500
Lateness	Observation	410	4 100
Job productivity	Observation	350	3 500
Length of service queue	Time (minutes)	410	41 000

The results for the First in first out pilot run are shown in Table 5.28. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.28 Results for First in first out pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	83.1	5.84	10
Process productivity (ratio)	0.893	0.007 07	0.01
Lateness (minutes)	66.7	6.2	10
Job productivity (ratio)	0.268	0.011 2	0.01
Length of service queue (entities)	7.38	0.666	1

5.4.5 Statistical analysis for Last in first out pilot run for a traffic intensity of 0.9

The information on the batching of the data is given in Table 5.29. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.29 Batch size information for Last in first out for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	160	1 600
Process productivity	Time (minutes)	130	13 000
Lateness	Observation	150	1 500
Job productivity	Observation	80	800
Length of service queue	Time (minutes)	420	42 000

The results for the Last in first out pilot run are shown in Table 5.30. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.30 Results for Last in first out pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	82.8	5.67	10
Process productivity (ratio)	0.893	0.006 82	0.01
Lateness (minutes)	70.9	6.35	10
Job productivity (ratio)	0.472	0.004 85	0.01
Length of service queue (entities)	7.35	0.646	1

5.4.6 Statistical analysis for Latest job pilot run for a traffic intensity of 0.9

The information on the batching of the data is given in Table 5.31. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.31 Batch size information for Latest job for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	410	4 100
Process productivity	Time (minutes)	130	13 000
Lateness	Observation	410	4 100
Job productivity	Observation	360	3 600
Length of service queue	Time (minutes)	400	40 000

The results for the Latest job pilot run are shown in Table 5.32. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.32 Results for Latest job pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	82.9	6.39	10
Process productivity (ratio)	0.893	0.006 82	0.01
Lateness (minutes)	66	6.22	10
Job productivity (ratio)	0.28	0.010 7	0.01
Length of service queue (entities)	7.41	0.707	1

5.4.7 Statistical analysis for Longest service time pilot run for a traffic intensity of 0.9

The information on the batching of the data is given in Table 5.33. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.33 Batch size information for Longest service time for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	370	3 700
Process productivity	Time (minutes)	180	18 000
Lateness	Observation	380	3 800
Job productivity	Observation	80	8 000
Length of service queue	Time (minutes)	440	44 000

The results for the Longest service time pilot run are shown in Table 5.34. Unlike the case of the other control policies, the pilot run for the Longest service time did not reach the required half width. To determine the required run length, the following are shown for every performance criterion: the average, the number of batches ( $n$ ), the achieved half width ( $h$ ), the required half width ( $h^*$ ) and the required number of batches ( $n^*$ ). This enables the determination of the

required run length for every performance criterion. The largest of these, in this case **Lateness** (about 20 000 000), is used as the length for the production run. The full results are shown in Appendix F.

Table 5.34 Results for Longest service time pilot run for a traffic intensity of 0.9

Performance criterion	Average	n	h	$b^*$	$n^*$	Required run length (minutes)
Time in system (minutes)	237	37	24.4	10	221	9 059 429
Process productivity (ratio)	0.893	77	0.005 69	0.01	n.a.	n.a.
Lateness (minutes)	222	36	27.8	10	467	19 558 338
Job productivity (ratio)	0.38	17	0.008 08	0.01	n.a.	n.a.
Length of service queue (entities)	22.7	31	2.67	1	221	9 824 000

#### 5.4.8 Statistical analysis for Longest service time production run for a traffic intensity of 0.9

The specific data for the Longest service time production run is shown in Table 5.35. The required runtime is 20 000 000, but the files generated are too large to be analysed, so the longest runtime for which the files could be analysed, was used.

Table 5.35 Specific data for Longest service time production run for a traffic intensity of 0.9

Variable	Setting
Total runtime (simulation minutes)	10 000 000
Truncation point (simulation minutes)	100 000

The results for the Longest service time production run are shown in Table 5.36. For every performance criterion, the average, half width for the production run, required half width and the half width for the pilot run are shown. It can be seen that the production run half width is lower than the pilot run half width, but the required half width is still not achieved. Unfortunately, longer runs were impossible because the output files became too large, so these results were used as is. The full results are shown in Appendix F.

Table 5.36 Results for Longest service time production run for a traffic intensity of 0.9

Performance criterion	Average	Half width for production run	Required half width	Half width for pilot run
Time in system (minutes)	260	15.1	10	24.4
Process productivity (ratio)	0.898	0.002 28	0.01	0.005 69
Lateness (minutes)	245	15.6	10	27.8
Job productivity (ratio)	0.375	0.002 46	0.01	0.008 08
Length of service queue (entities)	25	1.66	1	2.67

#### 5.4.9 Statistical analysis for Shortest service time pilot run for a traffic intensity of 0.9

The information on the batching of the data is given in Table 5.37. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.37 Batch size information for Shortest service time for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	160	1 600
Process productivity	Time (minutes)	170	17 000
Lateness	Observation	170	1 700
Job productivity	Observation	120	1 200
Length of service queue	Time (minutes)	400	40 000

The results for the Shortest service time pilot run are shown in Table 5.38. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.38 Results for Shortest service time pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	36.1	1.43	10
Process productivity (ratio)	0.893	0.006 44	0.01
Lateness (minutes)	25	1.43	10
Job productivity (ratio)	0.444	0.005 55	0.01
Length of service queue (entities)	2.71	0.167	1

This concludes the reporting and discussion of the results of the off-line non-terminating analysis for a traffic intensity of 0.9.

## 5.5 On-line planning and control for a traffic intensity of 0.9

The Emulator generates the data for the two techniques for a traffic intensity of 0.9.

### 5.5.1 Work method

The work method is the same as discussed in paragraph 5.2.1 on page 96.

### 5.5.2 Preliminary experiment

Table 5.39 gives the settings of the variables that have been changed from Table 5.15, page 97.

Table 5.39 Emulator variable settings for a traffic intensity of 0.9

Variable	Setting	Explanation
Mean of exponential distribution of process times (minutes)	9	This should lead to the required traffic intensity 0.9.

### 5.5.3 Method for statistical analysis

The method for statistical analysis is the same as discussed in paragraph 5.2.3 on page 99.

5.5.4 Statistical analysis for First technique pilot run for a traffic intensity of 0.9

The specific data for the First technique pilot run is shown in Table 5.40. The runtime is longer than for the traffic intensity of 0.7, because the longer expected queue lengths will result in more variation. This increased variation is accommodated with the longer runtime.

Table 5.40 Specific data for First technique pilot run for a traffic intensity of 0.9

Variable	Setting
Total runtime (simulation minutes)	1 509 982
Truncation point (simulation minutes)	5 028
Size of mini-batch (minutes)	10

The information on the batching of the data is given in Table 5.41. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.41 Batch size information for First technique pilot run for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	400	4 000
Process productivity	Time (minutes)	180	18 000
Lateness	Observation	410	4 100
Job productivity	Observation	140	1 400
Length of service queue	Time (minutes)	400	4 000

The results for the First technique pilot run are shown in Table 5.42. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.42 Results for First technique pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	84	6.22	10
Process productivity (ratio)	0.894	0.006 53	0.01
Lateness (minutes)	70.3	5	10
Job productivity (ratio)	0.388	0.006 9	0.01
Length of service queue (entities)	7.43	0.551	1

5.5.5 Control policy switching information for First technique pilot run for a traffic intensity of 0.9

The information on the switching of control policies is shown in Table 5.43.

Table 5.43 Control policy switching information for First technique pilot run for a traffic intensity of 0.9

Control policy	Usage (%)	Average number of trials to next switch
First in first out	67.65	27.28
Last in first out	5.88	
Latest job	7.87	
Longest service time	17.72	
Shortest service time	0.87	

This concludes the reporting on the results and the analysis of the First technique. The results and analysis of the Second technique will be discussed subsequently.

5.5.6 Statistical analysis for Second technique pilot run for a traffic intensity of 0.9

The specific data for the Second technique pilot run is shown in Table 5.44.

Table 5.44 Specific data for Second technique pilot run for a traffic intensity of 0.9

Variable	Setting
Total runtime (simulation minutes)	1 529 972
Truncation point (simulation minutes)	5 028
Size of mini-batch (minutes)	10

The information on the batching of the data is given in Table 5.45. For every performance criterion it is indicated whether it is time- or observation-based batching, what the lag number is where the correlation is approximately zero and what batch size is used.

Table 5.45 Batch size information for Second technique pilot run for a traffic intensity of 0.9

Performance criterion	Time- or observation-based batching	Lag number where correlation is approximately zero	Batch size used
Time in system	Observation	420	4 200
Process productivity	Time (minutes)	170	17 000
Lateness	Observation	420	4 200
Job productivity	Observation	110	1 100
Length of service queue	Time (minutes)	400	40 000

The results for the Second technique pilot run are shown in Table 5.46. For every performance criterion, the average, half width and required half width are shown. The latter attempts to ensure that the half width for the pilot run is smaller than the required half width. The full results are shown in Appendix F.

Table 5.46 Results for Second technique pilot run for a traffic intensity of 0.9

Performance criterion	Average	Half width for pilot run	Required half width
Time in system (minutes)	83.5	5.4	10
Process productivity (ratio)	0.893	0.005 76	0.01
Lateness (minutes)	70	5.32	10
Job productivity (ratio)	0.389	0.006 77	0.01
Length of service queue (entities)	7.42	0.518	1

5.5.7 Control policy switching information for Second technique pilot run for a traffic intensity of 0.9

The information on the switching of control policies is shown in Table 5.47.

Table 5.47 Control policy switching information for Second technique pilot run for a traffic intensity of 0.9

Control policy	Usage (%)	Average number of trials to next switch
First in first out	20.51	10.38
Last in first out	19.15	
Latest job	19.47	
Longest service time	18.91	
Shortest service time	21.91	

This concludes the reporting on the results and the analysis of the Second technique.

5.6 Combining the results for a traffic intensity of 0.9

The combined results from the off-line non-terminating analysis and the on-line planning and control for a traffic intensity of 0.9 are shown in Table 5.48. The combined results for both the traffic intensities considered are discussed in paragraph 5.7 on page 121.

Table 5.48 Combined results for a traffic intensity of 0.9

	Min/Max	Off-line analysis					On-line analysis	
		First in first out	Last in first out	Latest job	Longest service time	Shortest service time	First technique	Second technique
Time in system (minutes)	Min	83.1	82.8	82.9	260	36.1	84	83.5
Process productivity (ratio)	Min	0.893	0.893	0.893	0.898	0.893	0.894	0.893
Lateness (minutes)	Max	66.7	70.9	66	245	25	70.3	70
Job productivity (ratio)	Min	0.268	0.472	0.28	0.375	0.444	0.388	0.389
Length of service queue (entities)	Max	7.38	7.35	7.41	25	2.71	7.43	7.42

## 5.7 Discussion of the results of the evaluation of the techniques

Testing the techniques for the two different traffic intensities of 0.7 and 0.9 respectively enables the evaluation of the efficiency of the techniques at those two intensities. The first consideration is the explanation of the results of the evaluation of the techniques. Then the results of this comparison are used to predict the efficiency of the techniques at other traffic intensities. The second consideration is how the two techniques compare with regard to each other. Finally, some additional considerations are given.

### 5.7.1 *Explaining the results of the evaluation of the techniques*

To simplify the explanation of the results evaluation of the techniques, Table 5.24 on page 106 and Table 5.48 on page 120, are combined in Table 5.49 on page 124. Table 5.49 shows the combined results of the evaluation for both the traffic intensities. For both traffic intensities, the following are given for every performance criterion:

- a) Whether the performance criterion needs to be minimised or maximised.
- b) The expected value of the performance criterion under the First and Second technique.
- c) The expected value of the performance criterion under the five individual control policies.

The results can be explained as follows. The performance criteria, and whether they should be minimised or maximised, were chosen specifically to encourage switching between the different control policies. In this uncomplicated model, it was only possible by making contradictory choices on whether the performance criteria should be minimised or maximised. Consequently, the results of the two techniques converge to the results of the two random control policies (First in first out and Last in first out). However, the most important observation is that the techniques resulted in performance criteria that were very similar to the best results achieved by the individual control policies.

The next step is to predict the effect of the techniques at other intensities that were not evaluated. Unfortunately, only the information available can be used for prediction. It is expected that at different traffic intensities, different control policies may be better than the techniques themselves, while the techniques will only give marginally worse results. In addition, in the introduction of the project it was noted that the on-line planning and control process intends to respond to changing conditions. Thus, it can be assumed that the techniques will excel in those conditions of the on-line planning and control process where the input distribution is not

constant, as it will give slightly worse results than an individual control policy, but never poor results as would be the case for an individual control policy not suited to the traffic intensity. The techniques may also be advantageous in cases where it is not possible to determine beforehand which of the individual control policies to use because it is impossible to predict the input distribution that will occur. It is expected that the techniques will give good (but unfortunately, not necessarily the best) results for any input distribution, while an individual control policy that may give the best results for one input distribution, may prove disastrous for another input distribution.

The preceding discussion is mere guess work and the techniques should be evaluated in increasingly complex systems with random mixed intensities before any conclusive results can be claimed. The evaluation of the techniques in increasingly complex systems with random mixed intensities is a subject for further study. These evaluations will necessarily lead to the evaluation of the techniques in increasingly complex systems, culminating in the evaluation of the techniques in a real-world FMS. This is also a subject for further study. Therefore, while it is not possible to make complete conclusions about the techniques before more tests are done, it has been shown that they warrant further examination.

### *5.7.2 Comparing the techniques with each other*

A method of comparison of the techniques would be to look at the control policy switching information. Table 5.19 on page 102, Table 5.23 on page 104, Table 5.43 on page 117 and Table 5.47 on page 119 are combined in Table 5.50 on page 125. Thus, Table 5.50 gives the combined usage and switching information for both traffic intensities and both techniques. From Table 5.50 it can be seen that the First technique, on average, evaluates more trials than the Second technique before switching, while the Second technique has a more even usage of control policies than the First technique. Therefore, for the two traffic intensities evaluated, it can be said that the First technique should be used in situations where the switching of control policies is unwanted and should be minimised, while the Second technique should be used in situations where the different control policies use different resources and the difference between loads on the resources should be minimised. The reason for this difference between the two techniques is not known and is a subject for further study. Once again, the behaviour of the techniques in increasingly complex systems is unclear, and is a subject for further study.

### *5.7.3 Additional considerations*

A final consideration relates to the fact that the Emulator does not choose one of the control policies that have been shown to be obviously better i.e. Longest service time, and does not use it all the time. The reason for this is that the individual control policy is shown to be better on the long run by steady state analysis. The Emulator, however, only evaluates the next 1 000 jobs (a time period of about a week) and decides what is the best control policy to use in the system for this short time horizon.

Table 5.49 Combined results for both traffic intensities

		Min / Max	First technique	Second technique	First in first out	Last in first out	Latest job	Longest service time	Shortest service time
Traffic intensity of 0.7	Time in system (minutes)	Min	22.9	22.9	22.8	22.7	22.7	37.3	15.9
	Process productivity (ratio)	Min	0.693	0.693	0.692	0.692	0.692	0.698	0.692
	Lateness (minutes)	Max	13.1	13.1	12.1	13.6	11.2	26.4	7.39
	Job productivity (ratio)	Min	0.564	0.564	0.524	0.608	0.54	0.552	0.596
	Length of service queue (entities)	Max	1.58	1.58	1.57	1.56	1.57	3.03	0.887
Traffic intensity of 0.9	Time in system (minutes)	Min	84	83.5	83.1	82.8	82.9	260	36.1
	Process productivity (ratio)	Min	0.894	0.893	0.893	0.893	0.893	0.898	0.893
	Lateness (minutes)	Max	70.3	70	66.7	70.9	66	245	25
	Job productivity (ratio)	Min	0.388	0.389	0.268	0.472	0.28	0.375	0.444
	Length of service queue (entities)	Max	7.43	7.42	7.38	7.35	7.41	25	2.71

Table 5.50 Combined usage and switching information for both traffic intensities and both techniques

	Traffic intensity of 0.7				Traffic intensity of 0.9			
	First technique		Second technique		First technique		Second technique	
Control policy	Usage (%)	Average number of trials to next switch	Usage (%)	Average number of trials to next switch	Usage (%)	Average number of trials to next switch	Usage (%)	Average number of trials to next switch
First in first out	55.23	32.06	19.14	6.29	67.65	27.28	20.51	10.38
Last in first out	5.15		22.25		5.88		19.15	
Latest job	5.83		20.85		7.87		19.47	
Longest service time	33.28		18.73		17.72		18.91	
Shortest service time	0.5		19.03		0.87		21.91	

## 5.8 Summary

In this chapter, the Emulator described in Chapter 4 (starting on page 61) was used to evaluate the two techniques discussed in Chapter 3 (page 29) for two different traffic intensities. It was shown that the advantage of the use of the techniques would come when it is not possible to determine beforehand which of the individual control policies to use and in systems where the input distribution is not constant. In addition, the First technique should be used in situations where the switching of control policies is unwanted and should be minimised, while the Second technique should be used in situations where the different control policies use different resources and the load on the resources should be equal.

While discussing the analysis of the techniques, some subjects that require further study in other projects were noted. These subjects should preferably be studied in increasingly complex systems to enable the techniques to show their true worth. These are summarised in Table 5.51.

Table 5.51 Subjects that require further study in increasingly complex systems, as identified during the evaluation of the techniques

Subject	Paragraph	Page
The effect of the mean of the exponential distribution of the due times on the efficiency of the techniques.	5.1.2	85
The effect of the evaluation period on the efficiency of the techniques.	5.2.2	97
The effect of the mean of the exponential distribution of the simulation delay on the efficiency of the techniques.	5.2.2	97
The effect of the starting control policy on the efficiency of the techniques.	5.2.2	97
The effect of the confidence level for the First technique on the efficiency of the techniques.	5.2.2	97
The effect of the importance of the performance criteria on the efficiency of the techniques.	5.2.2	97
The effect of whether the performance criteria should be minimised or maximised on the efficiency of the techniques.	5.2.2	97
The evaluation of the techniques at random mixed traffic intensities.	5.7.1	121
The evaluation of the techniques in increasingly complex systems, culminating in the evaluation of the techniques in a real-world FMS.	5.7.1	121
The reasons for the differences between the operational characteristics of the two techniques.	5.7.2	122
Whether or not the average number of trials before switching and control policy usage differ for different traffic intensities.	5.7.2	122

## 6 CONCLUSIONS

The project described in the preceding chapters enables the formulation of a conclusion for each of the three main chapters. They are, from Chapters 3, 4 and 5 respectively, as follows:

- a) It is possible to automate the real-time compromise analysis function.
- b) An emulator can be developed to evaluate the techniques that automate the real-time compromise analysis function.
- c) The techniques developed in this project to automate the real-time compromise analysis function warrant further study.

Each of these three conclusions is now discussed in more detail.

### 6.1 It is possible to automate the real-time compromise analysis function

In Chapter 3, starting on page 29, it was shown that it is possible to automate the real-time compromise analysis function. This is beneficial, because it eliminates the need for a human observer to evaluate the results of the real-time output analysis function and to make a decision from it. The automation can be done with both of the techniques that were developed, but unfortunately, the techniques are not necessarily efficient. While it would be ideal to evaluate the techniques mathematically to quantify their efficiency, it is not possible because the assumptions in Table 3.1 on page 30 are not necessarily correct. Consequently, the statistical basis of the techniques is suspect and the only way of determining their efficiency is by evaluating them.

### 6.2 An emulator can be developed to evaluate the techniques that automate the real-time compromise analysis function

Chapter 4, starting on page 62, showed that it is possible to develop an emulator of the on-line planning and control process that can be used to evaluate the techniques that automate the real-time compromise analysis function.

In this case, some modifications were needed to make it possible. Firstly, a straightforward M/M/1/FIFO/ $\infty$ / $\infty$  system is used to allow the initialising of the alternative system models to the end state of the real-world system model, because it is extremely problematic to determine the end conditions and to initialise models in Arena<sup>®</sup>. Secondly, it is not a complete emulation and parts for which the technology does not currently exist (i.e. the autovalidation process and the alternative control policy generator) have been excluded. Thirdly, it was necessary to do operations sequentially that should actually be done concurrently, because the language used for the Emulator (Visual Basic<sup>®</sup>) does not allow for operations to be done concurrently as a language

like Java<sup>®</sup> would. Finally, it must be noted that there is not a genuine real-world system that is being controlled and that an Arena<sup>®</sup> model is used to represent the real-world system.

The greatest advantage of the Emulator is that it can run significantly faster than real-time, enabling the generation of enough data to make significant statistical comparisons to evaluate the techniques that automate the real-time compromise analysis function.

### **6.3 The techniques developed in this project to automate the real-time compromise analysis function warrant further study**

In Chapter 5, starting on page 84, it was shown that the results of the analysis of the techniques for real-time compromise analysis warrant further study. This was done by showing that at the traffic intensities evaluated, the results of the techniques were very similar to those of the individual control policies. From these results, it was speculated that at different traffic intensities, different control policies would be better than the techniques themselves, while the techniques will only give slightly worse results. In addition, because the on-line planning and control process attempts to respond to changing conditions, it can be assumed that the techniques will excel in those conditions where the input distribution is changing continuously. It is also speculated that the techniques may be advantageous in cases where it is not possible to determine beforehand which of the individual control policies to use because it is impossible to predict the input distribution that will occur. It is expected that the techniques will give good (but unfortunately, not necessarily the best) results for any input distribution, while an individual control policy that may give the best results for one input distribution, may prove inadequate for another input distribution. Therefore, while it is not possible to make complete conclusions about the techniques before more tests are done, it has been shown that they warrant further examination.

## 7 RECOMMENDATIONS

The project described in the preceding chapters leads to the following recommendations for further study. It is possible to make a recommendation from each of the three main chapters. Thus the three main recommendations, for Chapters 3, 4 and 5 respectively, are:

- a) The developed techniques need to be studied further.
- b) The developed Emulator needs to be improved.
- c) The developed techniques need to be evaluated further.

Each of these three recommendations is now discussed in more detail.

### 7.1 The developed techniques need to be studied further

It cannot be claimed that the techniques that were developed are perfect or that they are the only possible techniques that will work. In fact, they are merely techniques that may work. For this reason the techniques need to be studied further, refined and improved. The possibility that entirely new techniques will evolve from this further study cannot be excluded. Table 3.3 on page 61 gives a summary of the subjects that require further study. These are now discussed in more detail.

The effect of the settings of the techniques (i.e. the confidence level of the First technique) on the efficiency of the techniques needs to be studied further. This will require the evaluation of many different combinations of settings for many different systems of increasingly complex nature.

The effect of the sample size used in the calculations on the efficiency of the techniques also needs to be studied further. This will require the evaluation of different sample sizes as well as other methods of deciding on the sample size, i.e. Antonacci's method, for many different combinations of settings for many different systems of increasingly complex nature. The different sample sizes and size selection methods must be evaluated with different settings of the techniques.

The techniques must be adapted to accommodate performance criteria that represent proportions, percentiles, minimums and maximums. This will ensure that the techniques are functional in real systems where not all the performance criteria represent means.

The effect of the compromise functions on the efficiency of the techniques also needs to be studied further. This will require the evaluation of different compromise functions for many different combinations of settings, sample sizes and methods of deciding on sample sizes for lots

of different systems of increasingly complex nature. The different compromise functions must be evaluated with the perceived optimal settings of the techniques and sample sizes or method of sample size determination, but it must also be investigated whether or not other compromise functions require other settings or sample sizes.

## 7.2 The developed Emulator needs to be improved

It would be foolhardy to assume that the Emulator is complete. It is easy to identify improvements that need to be made. Some of these improvements are summarised in Table 4.2 on page 83 and are discussed here.

The first need is critical, and that is to develop the Emulator in a programming language and simulation package that allows concurrent operations and effortless initialisation. The concurrent operations will allow for a more realistic emulation of the on-line planning and control process because the operations that are actually being done concurrently do not need to be done sequentially. A programming language-simulation package combination that allows effortless initialisation would allow the Emulator to be much faster, because even though a  $M/M/1/FIFO/\infty/\infty$  system was used for the project to make the initialisation as effortless as possible, the initialisation is currently taking the largest part of the time. However, more importantly, it would allow the Emulator to be less dependent on the model. At the moment the models form an intricate part of the Emulator, because the initialisation is a complex set of calculations that is dependent on specialised programming as well as on small differences between the model initialised and the models whose end conditions it is initialised to. In the ideal case, the Emulator would provide the framework that will be able to take any system and emulate the on-line planning and control process with that specific system. Davis [3] has developed a new modelling architecture that makes it easier to initialise system models.

If a language that allows concurrent operations is used and the Emulator is not doing concurrent operations sequentially any more, the actual time the evaluation of an alternative takes will be incorporated into the Emulator. The effect of this time on the Emulator can then be determined. This is especially important in the case of the complex models, where the evaluation of an alternative may take a long time. If the evaluation of the alternatives takes too long, additional models may be needed to evaluate the same alternative system concurrently. An ideal situation would be if the different alternatives are evaluated on different computers connected with a network. However, this will bring about its own set of networking problems.

The current Emulator is also very inflexible with regard to input. The Emulator must be changed to allow the use of distributions other than the exponential distribution for the time between

arrivals, the process time and the random time added to the time of arrival to compute the due time. This will greatly increase the number of possible scenarios that can be evaluated with the Emulator. Ideally, the Emulator must be allowed to generate inputs from distributions that are continually changing, as would be expected in the real-world.

### 7.3 The developed techniques need to be evaluated further

The techniques need to be proved to be efficient in more complex systems (This would be a lot easier with the Emulator recommended in the previous paragraph.). Unfortunately, due to a time constraint on the project, the techniques could not be evaluated for more complex systems and it could only be speculated on the efficiency of the techniques in more complex systems. It is recommended that the following situations, identified in Table 5.51 on page 127, be evaluated further in more complex systems.

Firstly, for the traffic intensities evaluated (0.7 and 0.9), the following additional situations need to be evaluated for more complex systems:

- a) Different means of the exponential distribution of the due times.
- b) Different evaluation periods.
- c) Different means of the exponential distribution of the simulation delay on the efficiency of the techniques.
- d) Different starting control policies.
- e) Different control policies.
- f) Different importances of the performance criteria.
- g) Different combinations of minimising or maximising the performance criteria.

The techniques also need to be evaluated for the same system, in the same way as was done in this project, but for traffic intensities other than 0.7 and 0.9. The additional aspects mentioned above will also need to be evaluated before any conclusive remarks can be made on the efficiency of the techniques in this specific model.

The final test would be the implementation of the on-line planning and control process with the techniques included in a real-world FMS. Only if the techniques can be proved to be efficient in a real-world FMS, can they be assumed to be useful. Gonzalez and Davis [9] address the complex task of initialising an on-line simulation to a current system state collected from an operational system, which will be very helpful in the implementation of the techniques in a real-world FMS.

It is also recommended that the reasons for the differences between the operational characteristics of the two techniques be inspected, as well as whether or not the average number of trials before switching, as well as control policy usage differs for different traffic intensities. This would give valuable information on the techniques. This could be used to improve the techniques and help to decide which technique is more suitable for given conditions.

## 8 SUMMARY

The initial objective of this project was to develop techniques to select a control policy during on-line planning and control. This is aimed at automating the real-time compromise analysis function to eliminate the need for a human observer. Chapter 3 showed that this initial objective was fulfilled with the development of two techniques. To determine whether or not these techniques were efficient, an emulator was developed, which is described in Chapter 4. In Chapter 5, it was shown how this Emulator was used to show that no conclusive decision on the efficiency of the techniques can be made, but that they definitely deserve further study. Chapter 6 gave the conclusions of the project while Chapter 7 made some recommendations that followed from the project.

The output from the project is dual. Firstly, the project was a valuable learning experience, because it was instrumental in the understanding of the on-line planning and control process. In addition, it also cleared up some aspects of off-line simulation analysis, as well as simulation analysis with Arena<sup>®</sup> and Visual Basic<sup>®</sup> programming. Secondly, it is hoped that this project will form a valuable building block that will facilitate making on-line planning and control a viable alternative for controlling complex systems, and that it will enable these systems to respond better to the changing conditions that are currently becoming the norm.

## 9 REFERENCES

- [1] BANKS, J., ed., *Handbook of Simulation*, John Wiley & Sons, New York, 1998
- [2] DAVIS, W. J., "Looking into the future of simulation", IIE Solutions, vol. 30, no. 5, pp. 24-30, 1998
- [3] DAVIS, W. J., "On-line simulation: need and evolving research requirements", *Handbook of Simulation*, ed. Banks, J., John Wiley & Sons, New York, 1998
- [4] DAVIS, W. J., "Simulation: Technologies in the new millennium", Winter Simulation Conference Proceedings 1999, vol. 1, pp. 141-147, 1999
- [5] DAVIS, W. J., H. WANG and C. A. HSIEH, "Experimental studies in real-time, Monte Carlo simulation", IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no. 4, pp. 802-814, 1991
- [6] DAVIS, W. J., X. CHEN, A. BROOK and F. A. AWAD, "Implementing on-line simulation upon the World-Wide Web", Winter Simulation Conference Proceedings 1998, pp. 87-95, 1998
- [7] DRAKE, G. R. and J. S. SMITH, "Simulation system for real-time planning, scheduling, and control", Winter Simulation Conference Proceedings 1996, pp. 1 083-1 090, 1996
- [8] ELMARAGHY, H. A., I. B. ABDALLAH and W. H. ELMARAGHY, "On-line simulation and control in manufacturing systems", Annals of the CIRP, vol. 47, pp. 401-404, 1998
- [9] GONZALEZ, F. G. and W. J. DAVIS, "Initialising on-line simulations from the state of a distributed system", Winter Simulation Conference Proceedings 1998; vol. 1, pp. 507-513, 1998
- [10] HOLTER, T., X. YAO, L. C. RABELO, A. JONES and Y. YIH, *Integration of neural networks and genetic algorithms for an intelligent manufacturing controller*, Computers and Industrial Engineering, vol. 29, no. 1-4, pp. 211-215, 1995
- [11] KELTON, W. D., R. P. SADOWSKI and D. A. SADOWSKI, *Simulation with Arena*, MacGraw-Hill, Boston, 1998
- [12] LAW, A. M. and W. D. KELTON, *Simulation modeling and analysis, second edition*, MacGraw-Hill, New York, 1991
- [13] RODGERS, P. and R. J. GORDON, "Dynamic finite scheduling through on-line simulation: prioritising research and application issues", International Journal of Industrial Engineering - Applications and Practice, vol. 1, pp. 223-232, 1994
- [14] SADOWSKI, D. A., "Overview of ActiveX<sup>TM</sup> Automation, VBA, and Arena<sup>®</sup>", <http://www.sm.com/overview/whitepapers/ActiveX.htm>, 2000
- [15] WHITT, W., "Comparing probability measures on a set with an intransitive preference relation", Management Science, vol. 25, no. 6, pp. 505-511, 1979

## 10 BIBLIOGRAPHY

- BANKS, J., ed., *Handbook of Simulation*, John Wiley & Sons, New York, 1998
- DAVIS, W. J., "Looking into the future of simulation", IIE Solutions, vol. 30, no. 5, pp. 24-30, 1998
- DAVIS, W. J., "On-line simulation: need and evolving research requirements", *Handbook of Simulation*, ed. Banks, J., John Wiley & Sons, New York, 1998
- DAVIS, W. J., "Simulation: Technologies in the new millennium", Winter Simulation Conference Proceedings 1999, vol. 1, pp. 141-147, 1999
- DAVIS, W. J., H. WANG and C. A. HSIEH, "Experimental studies in real-time, Monte Carlo simulation", IEEE Transactions on Systems, Man and Cybernetics, vol. 21, no. 4, pp. 802-814, 1991
- DAVIS, W. J., X. CHEN, A. BROOK and F. A. AWAD, "Implementing on-line simulation upon the World-Wide Web", Winter Simulation Conference Proceedings 1998, pp. 87-95, 1998
- DRAKE, G. R. and J. S. SMITH, "Simulation system for real-time planning, scheduling, and control", Winter Simulation Conference Proceedings 1996, pp. 1 083-1 090, 1996
- ELMARAGHY, H. A., I. B. ABDALLAH and W. H. ELMARAGHY, "On-line simulation and control in manufacturing systems", Annals of the CIRP, vol. 47, pp. 401-404, 1998
- FISHBURN, P.C., "Stochastic dominance without transitive preferences", Management Science, vol. 24, no. 12, pp. 1 268-1 277, 1978
- GONZALEZ, F. G. and W. J. DAVIS, "Initialising on-line simulations from the state of a distributed system", Winter Simulation Conference Proceedings 1998; vol. 1, pp. 507-513, 1998
- HOLTER, T., X. YAO, L. C. RABELO, A. JONES and Y. YIH, *Integration of neural networks and genetic algorithms for an intelligent manufacturing controller*, Computers and Industrial Engineering, vol. 29, no. 1-4, pp. 211-215, 1995
- KELTON, W. D., R. P. SADOWSKI and D. A. SADOWSKI, *Simulation with Arena*®, MacGraw-Hill, Boston, 1998
- LAW, A. M. and W. D. KELTON, *Simulation modeling and analysis, second edition*, MacGraw-Hill, New York, 1991
- PAULSON, E., "On the comparison of several experimental categories with a control", Annals of Mathematical Statistics, vol. 23, pp. 239-246, 1952
- RODGERS, P. and R. J. GORDON, "Dynamic finite scheduling through on-line simulation: prioritising research and application issues", International Journal of Industrial Engineering - Applications and Practice, vol. 1, pp. 223-232, 1994
- SADOWSKI, D. A., "Overview of ActiveX™ Automation, VBA, and Arena", <http://www.sm.com/overview/whitepapers/ActiveX.htm>, 2000
- WHITT, W., "Comparing probability measures on a set with an intransitive preference relation", Management Science, vol. 25, no. 6, pp. 505-511, 1979

YAMAMOTO, M. and S. Y. NOF, , "*Scheduling/rescheduling in the manufacturing operating system environment*", International Journal of Production Research, vol. 23, no. 4, pp. 705-722, 1985

**APPENDIX A:  
IMPLEMENTATION OF  
AND SAMPLE  
CALCULATIONS FOR  
ANTONACCI'S METHOD**

## **A IMPLEMENTATION OF AND SAMPLE CALCULATIONS FOR ANTONACCI'S METHOD**

Antonacci's method was implemented in Excel<sup>®</sup> to gain a better understanding of it. Figure A. 1 shows Antonacci's method implemented in Excel<sup>®</sup>, with the parts corresponding to Antonacci's method coloured in. The darker colour corresponds to information that is given, while the lighter colour indicates cells that are calculated. Sample calculations are shown to prove that it was implemented correctly. Figure A. 2 shows the different parts, with their annotations, that will be discussed together. When the first digit of the annotation is a letter, it refers to given information, while a number refers to calculations. Annotations with the same prefixes are analogous, differing only for digits that are not the same.

	A	B	C	D	E	F	G	H	I	J	L=	L	M	N	O	P	Q	R	S
1						Performance criterion 1							Performance criterion 2						
2		Antonacci's method		Given input from "Start off"		n	Sample mean	Confidence interval lower bound	Confidence interval upper bound	Confidence interval entirely inside previous sample size's confidence interval?				Sample mean	Confidence interval lower bound	Confidence interval upper bound	Confidence interval entirely inside previous sample size's confidence interval?		
3			Confidence level (%) =	95	50	5.062	4.375	5.760						3.418	3.219	3.617			
4			Calculated:	100	4.463	3.918	5.007	NO						3.534	3.306	3.672	NO		
5			Alpha =	0.05	250	4.489	4.114	4.863	YES					3.588	3.505	3.672	NO		
6			Output:	50	600	4.446	4.182	4.706	YES					3.604	3.547	3.662	YES		
7			Sample size for alternative	50	4.500	4.327	4.689	YES						3.621	3.582	3.660	YES		
8			Sample size for performance criteria	50	4.566	4.433	4.697	NO						3.604	3.578	3.631	NO		
9																			
10																			
11			First technique					...in most recent trial					...in most recent trial						
12		Given input from "Start off"		Performance criterion 1		trial	Trial number	Mean for planning horizon	Alternative-Current	Statistical analysis of A-C			trial	Trial number	Mean for planning horizon	Alternative-Current	Statistical analysis of A-C		
13		Confidence level (%) =	95	Sample mean =	2.22	1	5439	2.069	-0.463	w =	50	1	5438	3.309	3.535				
14		Importance (1-10)		Sample variance =	97.04	2	5438	8.836	5.536	Sample mean =	2.22	2	5439	3.424	4.390	Sample mean =	1.96		
15		Performance criterion 1 =	6	MA Confidence interval LB =	-0.58	3	5437	7.869	8.437	Sample variance =	97.04	3	5437	2.394	0.254	Sample variance =	68.45		
16		Performance criterion 2 =	3	MA Confidence interval UB =	5.02	4	5436	9.941	13.771	Confidence interval LB =	-0.58	4	5436	3.146	6.162	Confidence interval LB =	-0.39		
17		Performance criterion 3 =	4	MA Alternative better than	NO	5	5435	8.175	-4.677	Confidence interval UB =	5.02	5	5435	2.439	2.028	Confidence interval UB =	4.31		
18		Performance criterion 4 =	5	MA Alternative points =	0.00	6	5434	6.708	10.975	w =	100	6	5434	4.252	-3.365	w =	100		
19		Performance criterion 5 =	7	MA Current points =	8.00	7	5433	1.224	-10.217	Sample mean =	3.44	7	5433	2.347	-11.551	Sample mean =	1.48		
20		Calculated:		Performance criterion 2		8	5432	1.085	-6.122	Sample variance =	84.96	8	5432	2.473	-10.491	Sample variance =	64.62		
21		Alpha =	0.05	Sample mean =	1.96	9	5431	5.777	-9.544	Confidence interval LB =	1.62	9	5431	1.484	1.530	Confidence interval LB =	-0.11		
22		Sample size		Sample variance =	68.45	10	5430	2.687	-9.756	Confidence interval UB =	5.27	10	5430	2.664	-3.012	Confidence interval UB =	3.08		
23		Sample size of alternative =	50	Confidence interval UB =	4.31	11	5429	4.333	-15.691	w =	250	11	5429	3.312	-8.699	w =	250		
24		Current sample size of current =	50	Alternative better than	NO	12	5428	4.182	-10.868	Sample mean =	3.08	12	5428	3.056	12.393	Sample mean =	2.93		
25		Sample size used =	50	Alternative points =	0.00	13	5427	2.243	3.984	Sample variance =	79.69	13	5427	3.226	-1.706	Sample variance =	54.84		
26						14	5426	2.659	-2.629	Confidence interval LB =	1.97	14	5426	3.053	4.644	Confidence interval LB =	2.01		
						15	5425	8.924	4.948	Confidence interval UB =	4.19	15	5425	2.956	-8.357	Confidence interval UB =	3.85		

Figure A. 1 Antonacci's method implemented in Excel®

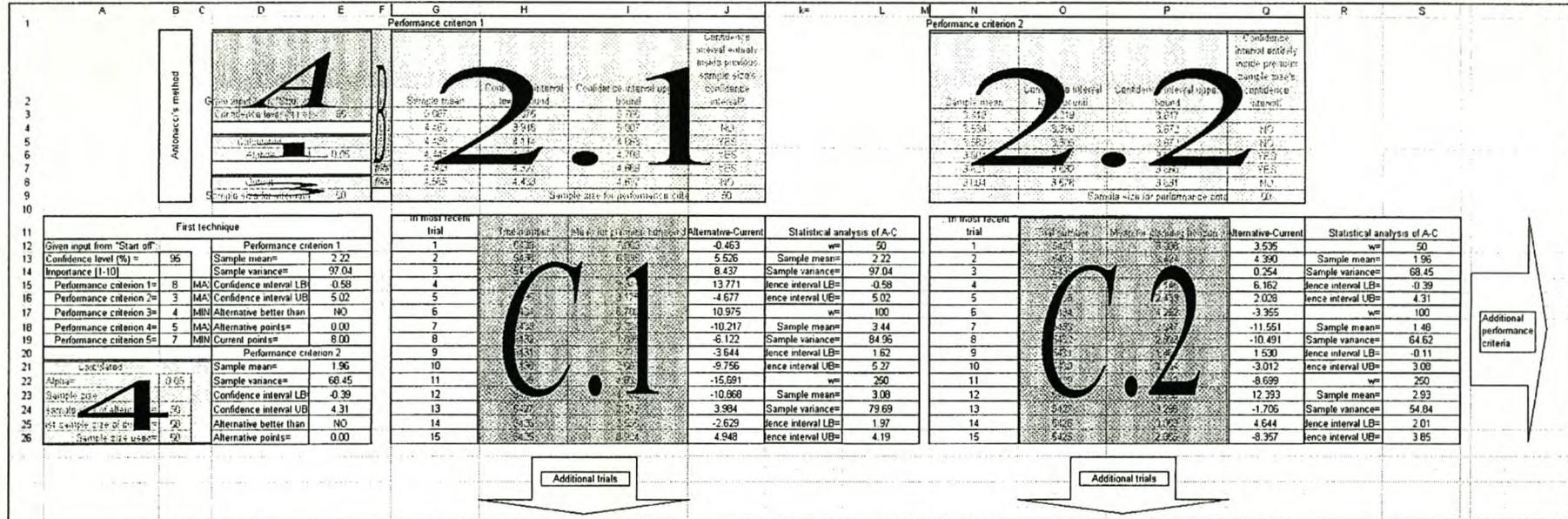


Figure A. 2 Parts of Excel® spreadsheet corresponding to Antonacci's method discussed together

Part A, shown in Figure A. 3, comprises of the given information on the confidence level for the comparison. Cell E3 gives the confidence level.

	D	E
2	Given input from "Start off":	
3	Confidence level (%) =	95

Figure A. 3 Part A of Antonacci's method

Part B, shown in Figure A. 4, comprises of the information on the different values of  $w$  that Antonacci proposes. Thus cells F2:F8 give  $w = 50, 100, 250, 500, 1\ 000, 2\ 000$ .

	F
2	$w$
3	50
4	100
5	250
6	500
7	1000
8	2000

Figure A. 4 Part B of Antonacci's method

Part C.G gives the trial number and mean for planning horizon  $J$  for the  $G$  different performance criteria. Figure A. 5 only shows Part C.1 with only the first 14 entries of performance criterion  $g$ . The trial number is given in cells H12:H2012 and the mean for planning horizon  $J$  in cells I12:I2012.

	H	I
11	Trial number	Mean for planning horizon J
12	5439	7.088
13	5438	6.835
14	5437	7.868
15	5436	9.341
16	5435	8.175
17	5434	6.708
18	5433	2.224
19	5432	1.085
20	5431	5.777
21	5430	2.667
22	5429	4.933
23	5428	4.162
24	5427	2.243
25	5426	2.558
26	5425	8.924

Figure A. 5 Part C.1 of Antonacci's method

Figure A. 6 shows Part 1, corresponding to cells D5:E6, that is concerned with the calculation of alpha ( $\alpha_a$ ) from the confidence level given in cell E3 by the following equation (all equation numbers refer to equations in the main document):

$$\begin{aligned} \alpha_a &= 1 - \left( \frac{CL_a}{100} \right) \\ &= 1 - \left( \frac{95}{100} \right) \\ &= 0.05 \end{aligned} \tag{Equation 2.6}$$

	D	E
5	Calculated:	
6	Alpha=	0.05

Figure A. 6 Part 1 of Antonacci’s method

Figure A. 7 shows Part 2, corresponding to cells G1:J9, that is concerned with the determination of the sample size for the specific performance criterion. In this case, it is performance criterion 1 from cell G1. It starts in cells G3:G8 where the sample mean for the  $w$  (given in cells F3:F8) most recent means for planning horizon  $J$  given in cells I12:I2012. Thus, for cell G3 the sample mean for the last 50 (from cell F3) is given by the equation:

$$\begin{aligned} \bar{\bar{X}}_{sp} &= \frac{\sum_{w=1}^W \bar{X}_{w/p}}{W} \\ &= \frac{254.260}{50} \\ &= 5.067 \end{aligned} \tag{Equation 2.7}$$

In cells H3:H8, the confidence interval lower bound is calculated for the  $w$  (given in cells F3:F8) most recent means for planning horizon  $J$ , given in cells I12:I2012. First, the unbiased estimator of the sample variance is given for the  $w$  (given in cells F3:F8) most recent means for planning horizon  $J$ , given in cells I12:I2012, by:

$$\begin{aligned} s^2(\bar{X}_{w/p}) &= \frac{\sum_{w=1}^w (\bar{X}_{w/p} - \bar{\bar{X}}_{sp})^2}{W - 1} \\ &= \frac{296.1054}{49} \\ &= 6.0429 \end{aligned} \tag{Equation 2.8}$$

Hence, for cell H3 the confidence interval lower bound for the last 50 (from cell F3) is given by the equation:

$$\begin{aligned}
 LB &= \bar{X}_{gp} - t_{n-1, 1-\alpha/2} \sqrt{\frac{s(\bar{X}_{wgp})^2}{W}} && \text{Equation 2.9} \\
 &= 5.067 - 2.0095 * \sqrt{\frac{6.0429}{50}} \\
 &= 4.375
 \end{aligned}$$

Furthermore, for cell I3 the confidence interval upper bound for the last 50 (from cell F3) is given by the equation:

$$\begin{aligned}
 UB &= \bar{X}_{gp} + t_{n-1, 1-\alpha/2} \sqrt{\frac{s(\bar{X}_{wgp})^2}{W}} && \text{Equation 2.10} \\
 &= 5.067 + 2.0095 * \sqrt{\frac{6.0429}{50}} \\
 &= 5.760
 \end{aligned}$$

The cells J4:J8 determine if the confidence interval for  $w$ , given in cells F4:F8, falls entirely inside the previous confidence interval. For cell J4 it is not true, because the confidence interval lower bound for  $w = 100$  in cell H4 is smaller than the confidence interval upper bound for  $w = 50$  given in cell H3.

Cell J9 determines the sample size for the specific performance criterion. This is done by using the value of  $w$  (cells F3:F8) that is one increment smaller than the first value of  $w$  for which the next larger confidence interval does not fit entirely inside the previous confidence interval. Thus, in this case J9 indicates that the smallest  $w$  for which the confidence interval does not fall entirely inside confidence level is 100 (cell J4) and the sample size is 50.

	G	H	I	J
1	Performance criterion 1			
2	Sample mean	Confidence interval lower bound	Confidence interval upper bound	Confidence interval entirely inside previous sample size's confidence interval?
3	5.067	4.375	5.760	
4	4.463	3.918	5.007	NO
5	4.489	4.114	4.863	YES
6	4.445	4.182	4.708	YES
7	4.508	4.327	4.689	YES
8	4.565	4.433	4.697	NO
9			Sample size for performance criterion:	50

Figure A. 7 Part 2.1 of Antonacci's method

Figure A. 8 shows Part 3, corresponding to cells D8:E9, that is concerned with the calculation of the sample size for the specific alternative system. The sample sizes for the different performance criteria e.g. the sample size of 50 for performance criterion 1 given in cell J9, are compared and the smallest one chosen. In this case, it is 50, as shown in cell E9.

	D	E
8	Output:	
9	Sample size for alternative	50

Figure A. 8 Part 3 of Antonacci's method

Figure A. 9 shows Part 4, corresponding to cells A21:B26, that is concerned with the calculation of the sample size for the comparison. Cell B24 gives the sample size for the alternative system as given by cell E9, and cell B25 gives the sample size for the current system as given by cell E9 of the current system's worksheet. In this case, both are equal to 50. Cell B26, which gives the sample size to be used in the comparison, is determined as the smallest of cells B24:B25 and is equal to 50.

	A	B
21	Calculated:	
22	Alpha=	0.05
23	Sample size	
24	Smallest sample size of alternative=	2000
25	Smallest sample size of current=	50
26	Sample size used=	50

Figure A. 9 Part 4 of Antonacci's method

**APPENDIX B:**  
**THE EMULATOR**  
**IMPLEMENTED IN**  
**ARENA<sup>®</sup> AND VISUAL**  
**BASIC<sup>®</sup>**

## **B THE EMULATOR IMPLEMENTED IN ARENA<sup>®</sup> AND VISUAL BASIC<sup>®</sup>**

This appendix describes how the Emulator used in this project was implemented in Arena<sup>®</sup> and Visual Basic<sup>®</sup>. Firstly, it gives the Arena<sup>®</sup> model logic and describes it. This is followed by the Visual Basic<sup>®</sup> code and a summary of the subprograms and functions used by the Emulator. Finally, the Visual Basic<sup>®</sup> interface form is shown.

### **B.1 Arena<sup>®</sup> model logic**

The model logic for the real-world system model (*RealworldModel.doe*) is shown in Figure B. 1 and the model logic for the alternative system model (*AlternativesModel.doe*) in Figure B. 2.

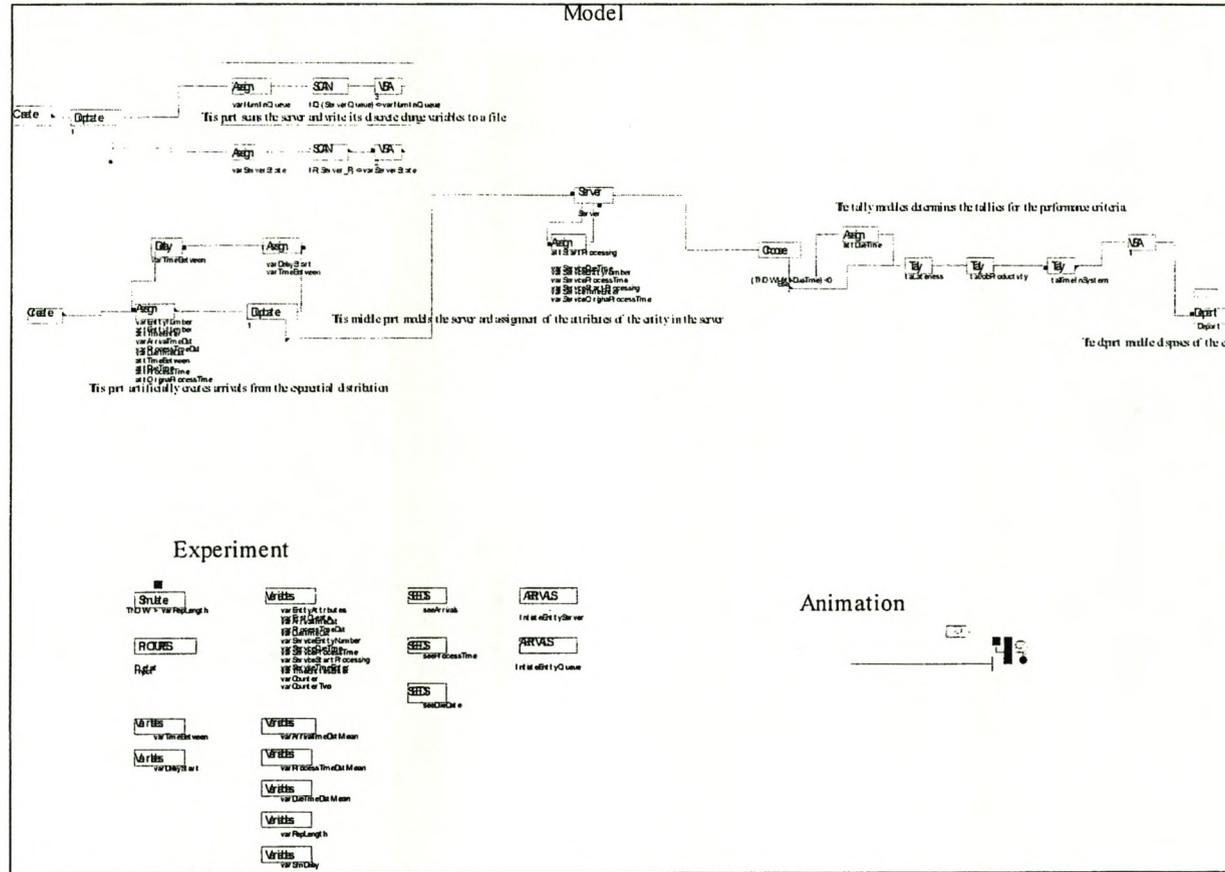


Figure B. 1 The overall model logic for the real-world system model



The different parts of the model logic are now discussed. The parts where the real-world system model and alternative system models are identical are discussed only once, but parts that differ are discussed separately.

### B.1.1 The model logic of the creation of the entities

The implementation of the model logic of the creation of entities for the real-world system model or the alternative system model is shown in Figure B. 3. It consists of a *Create* module, two *Assign* modules, a *Duplicate* module and a *Delay* module. The *Create* module creates one entity at time *varFirstCreate*. It then advances to the first *Assign* module where the following variables are assigned new values:

- a) *varEntityNumber*: This variable is computed by incrementing the variable *varEntityNumber* by one every time an entity passes through the *Assign* module.
- b) *varArrivalTimeDist*: This variable is taken from the exponential distribution, with a mean given by the variable *varArrivalTimeDistMean*.
- c) *varProcessTimeDist*: This variable is taken from the exponential distribution, with a mean of *varProcessTimeDistMean*.
- d) *varDueTimeDist*: This variable is taken from the exponential distribution, with a mean of the *varDueTimeDistMean*.

The entity is also assigned the following attributes:

- a) *attEntityNumber*: The attribute *attEntityNumber* is a unique number used to differentiate between entities and is assigned the new value of variable *varEntityNumber*.
- b) *attTimeEnter*: The attribute *attTimeEnter* is assigned the value of the system variable *TNOW* when the entity passes through the *Assign* module.
- c) *attTimeBetween*: The attribute *attTimeBetween* gives the delay before the next duplicate entity is entered into the system and is assigned the new value of variable *varArrivalTimeDist*.
- d) *attDueTime*: The attribute *attDueTime* is the time when the entity's processing should be finished and is computed as the time on arrival plus the new value of variable *varDueTimeDist*.
- e) *attProcessTime*: The attribute *attProcessTime* is the processing time of this entity and is assigned the new value of variable *varProcessTimeDist*.

- f) *AttOriginalProcessTime*: The attribute *attOriginalProcessTime* stores the original process time of the entity.

The entity is then duplicated once in the *Duplicate* module. The one duplicate is allowed to go to the server, while the other goes to the next *Assign* block. Here the entity is assigned the variables:

- a) *varDelayStart*: This is the time at which the current delay is started (the value of the system variable *TNOW*), and it is used to compute the time at which the next entity should be created in the initialised model.
- b) *varTimeBetween*: This is assigned as the value of the attribute *attTimeBetween* of the entity currently being delayed.

The entity is then delayed for the time given by *varTimeBetween* in the *Delay* module. After the delay, the entity is allowed to move to the first *Arrive* module and the creation process starts all over again.

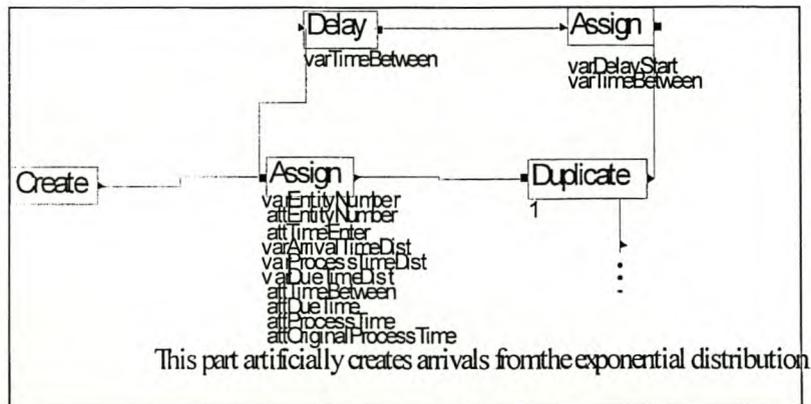


Figure B. 3 The model logic of the creation of entities for the real-world system or the alternative system model

### B.1.2 The model logic of the server

The model logic of the server for the real-world system or alternative system model is shown in Figure B. 4. If the server is currently busy, the entity is placed in the server's queue with the name *ServerQueue*. The entities are then taken from the queue in the sequence as specified by the control policy. Before the entity is serviced, external logic is accessed and the following attribute is assigned:

- a) *attStartProcessing*: The attribute *attStartProcessing* is assigned as the value of the system variable *TNOW* when the entity begins to be serviced.

The variables that store the values of the respective attributes of the entity currently being serviced, are also assigned:

- ⇒ *varServiceDueTime*.
- ⇒ *varServiceEntityNumber*.
- ⇒ *varServiceProcessTime*.
- ⇒ *varServiceStartProcessing*.
- ⇒ *varServiceTimeEnter*.
- ⇒ *varServiceOriginalProcessTime*.

At the server, the entity is delayed as long as its attribute *attProcessTime* requires. After the delay, the entity is allowed to advance to the *Choose* module.

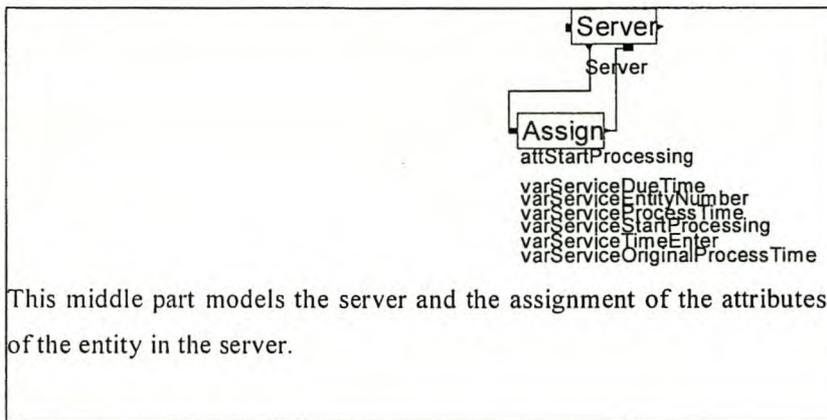


Figure B. 4 The model logic of the server for the real-world system or alternative system model

### B.1.3 The model logic of the tallies and the disposal of the entities

The model logic of the tallies and the disposal of the entities of the real-world system model is shown in Figure B. 5 and in Figure B. 6 for the alternative system model. Firstly, the entity arrives at the *Choose* module. Here it is determined if the entity's attribute *attDueTime* is larger than the current simulation time. If it is, the entity is forwarded to the *Assign* module where the entity's *attDueTime* is assigned the value of the system variable *TNOW*, so that the next tally only incorporates late entities. If the entity's attribute *attDueTime* is smaller than the value of the system variable *TNOW*, it is sent straight to the first tally module. The first tally, *talLateness*, is computed as the difference between the time the entity is disposed of and its attribute *attDueTime*. Only the entities that were early are assigned a new attribute *attDueTime*, so it only incorporates entities that are late. The next tally, *talJobProductivity*, is computed as the entity's processing time divided by the time it spent in the system. The final tally, *talTimeInSystem*, is computed as the difference between the time when the entity was created and the time when it was disposed of.

The real-world system model also has a *VBA* block (code shown in paragraph B.2) between the last *Tally* block and the *Depart* block. It is used to write the output of the real-world system

continuously to a file for analysis later on. It writes the last values of the three tallies to a file whenever an entity passes through.

The entity is disposed of in the *Depart* module and a counter, *cntDepart*, counts the departing entities.

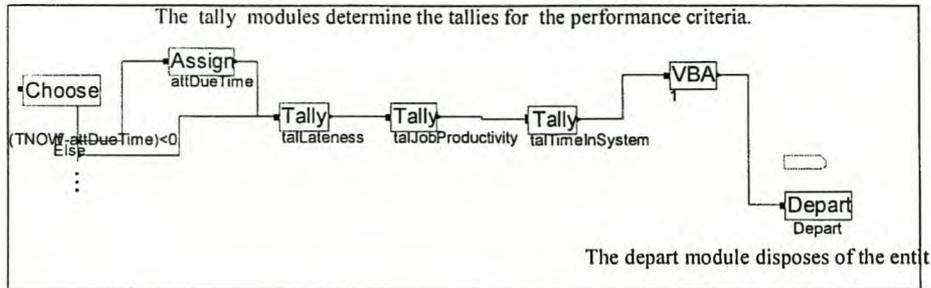


Figure B. 5 The model logic of the tallies and the disposal of the entities of the real-world system model

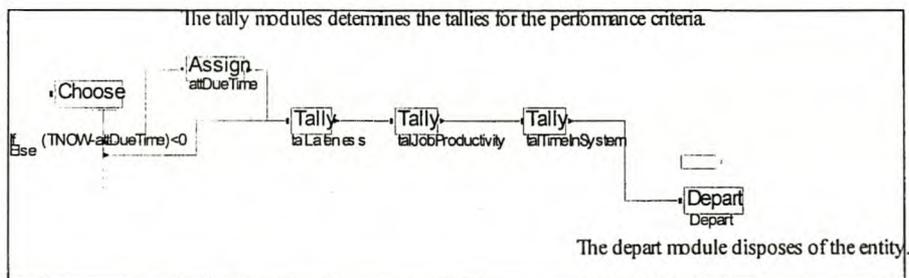


Figure B. 6 The model logic of the tallies and the disposal of the entities of the alternative system model

#### B.1.4 The model experiment parameters

The model experiment parameters for the real-world system model are shown in Figure B. 7 and those for the alternative system model in Figure B. 8. The analysis part comprises of a *Simulate* block, a *Pictures* block, six *Variables* blocks, three *Seeds* blocks and two *Arrivals* blocks. The *Simulate* block indicates that only one replication is done. The terminating condition is when the number of disposed entities, as counted by counter *cntDepart*, is equal to the variable *varNumberJobs* for the alternative system model, as well as to the run length for the real-world system model. The *Pictures* block eases verification by colouring the inserted entities pink. The *Variable* block assigns initial conditions to all the variables. The reason for the many *Variables* blocks is to enable the initialising of individual variables. The same is true for the multiple *Seeds* and *Arrival* blocks. The *Seeds* blocks assign the initial values of the random number streams that are used to generate values for the arrival times, process times and due times (*seeArrivals*, *seeProcessTime* and *seeDueDate* respectively). The *Arrival* blocks inject the initialising entities into the system at the specified time.

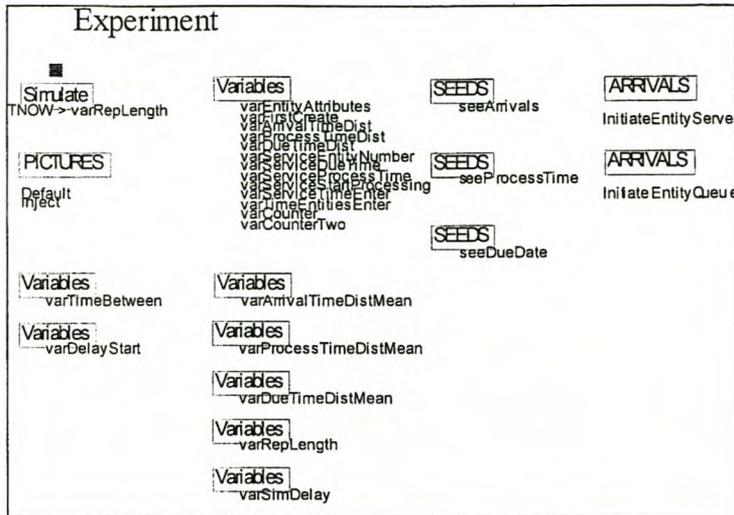


Figure B. 7 The definition of the experimental parameters for the real-world system model

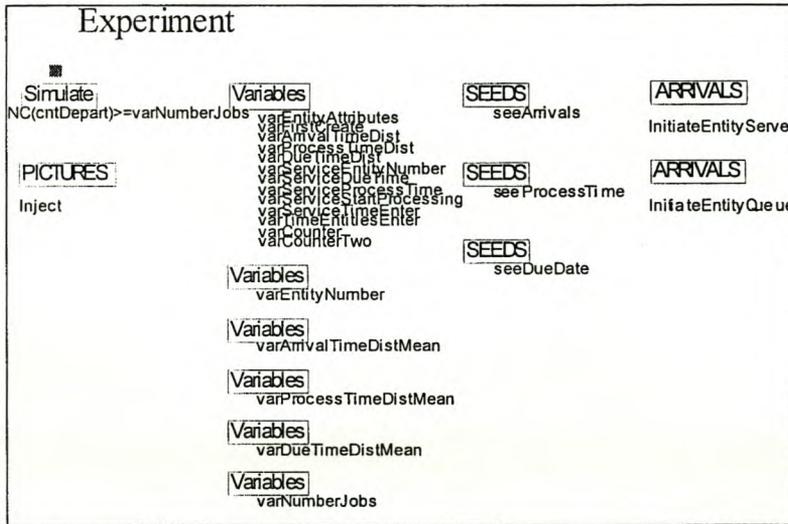


Figure B. 8 The definition of the experimental parameters for the alternative system model

*B.1.5 The model animation*

The model animation for both the real-world system model and the alternative system model is shown in Figure B. 9. The animation only shows the server, the entity in the server and the first few entities in the queue.

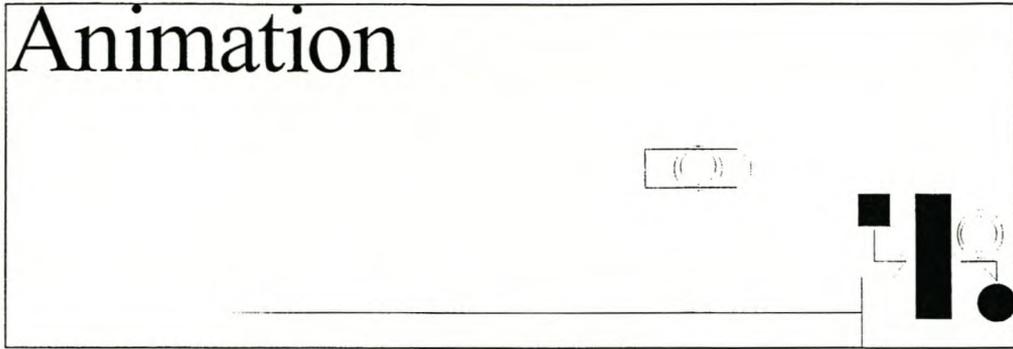


Figure B. 9 The model animation

*B.1.6 The determination of the real-world system's output*

The writing of the discrete change variables to file for the real-world system model is shown in Figure B. 10. It consists of two sets, each containing an *Assign* block, a *Scan* block and a *VBA* block. The set concerned with the discrete change variable of the number of entities in the queue assigns the current number of entities in the queue to a variable, *varNumInQueue*. As soon as the number of entities in the queue is different from the variable, the *Scan* block picks it up and the *VBA* block writes the value of the system variable *TNOW* and the new number in the queue to a file (code shown in paragraph B.2). The state of the server is recorded in the same way, but with the other set.

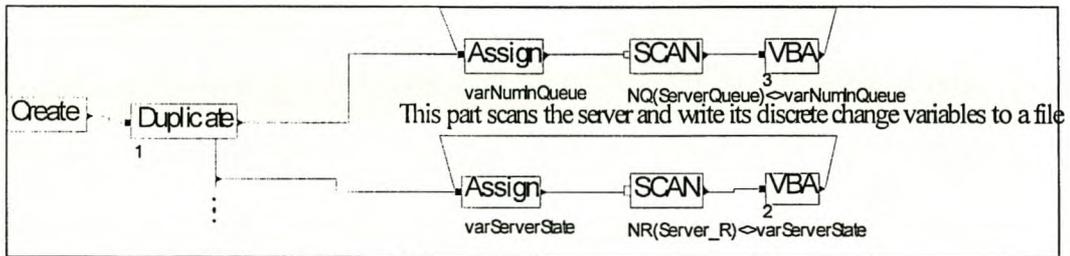


Figure B. 10 Code for the writing of discrete change variables to file

## B.2 Visual Basic® code

The Visual Basic® code used in the Emulator is given here.

### B.2.1 *The Visual Basic® code for ThisDocument of the real-world system model*

```
Option Explicit
Option Base 1
'


---


Private Sub ModelLogic_DocumentOpen()
'This is the subprogram that is executed when the Real-world
'model is opened (that is the emulator is started)

'It does the initial initialising and settings and
'it gets the previous settings of the emulator and display them
'in the initial form

'Dim the variables
Dim k As Integer
Dim g_varFileName As String

'The variables that accesses the Real-world model are set
Set g_ModelRW = Arena.Models(1)
Set g_SIMANRW = g_ModelRW.SIMAN

'Now the alternatives Arena model is opened
Arena.Application.Models.Open ("AlternativesModel.doe")

'And the variables that accesses this Alternatives model are set
Set g_ModelAL = Arena.Models(2)
Set g_SIMANAL = g_ModelAL.SIMAN

'Arena has been initialised, so to ensure that it is only
'closed when run through make
g_varRunThrough = False

'The previous settings of the emulator is found with this subprogram
Call subGetPrevious

'The text that need to be determined are displayed
Call subEndTime

'We also need to load the names of the control policies into
'the array of their names. In our case as follows
g_varPolicyNames(1) = "Alternative 1"
g_varPolicyNames(2) = "Alternative 2"
g_varPolicyNames(3) = "Alternative 3"
g_varPolicyNames(4) = "Alternative 4"
g_varPolicyNames(5) = "Alternative 5"

>Show the form
frmInitial.Show

End Sub
'


---


Private Sub ModelLogic_DocumentSave()
'This closes all the files once the simulation is finished
Close
End Sub
```

---

```

Private Sub VBA_Block_1_Fire()
'this writes the individual values of the tallies to a file as the
'entities passes through VBA block 1

'Dim the variables
Dim g_SIMANRW As Arena.SIMAN

'Set the variables
Set g_SIMANRW = Arena.Models(1).SIMAN

'Write to files
Write #1, g_SIMANRW.RunCurrentTime, g_SIMANRW.TallyLastObservation _
(fncTallyIDRW("talTimeInSystem"))
Write #3, g_SIMANRW.RunCurrentTime, g_SIMANRW.TallyLastObservation _
(fncTallyIDRW("talLateness"))
Write #4, g_SIMANRW.RunCurrentTime, g_SIMANRW.TallyLastObservation _
(fncTallyIDRW("talJobProductivity"))

End Sub

```

---

```

Private Sub VBA_Block_2_Fire()

'Dim the variables
Dim g_SIMANRW As Arena.SIMAN

'Set the variables
Set g_SIMANRW = Arena.Models(1).SIMAN

'Write to files
Write #2, g_SIMANRW.RunCurrentTime, g_SIMANRW.ResourceNumberBusy(1)

End Sub

```

---

```

Private Sub VBA_Block_3_Fire()
'this writes the individual values of the discrete change variables
'to a file as the entities passes through the VBA block

'Dim the variables
Dim g_SIMANRW As Arena.SIMAN

'Set the variables
Set g_SIMANRW = Arena.Models(1).SIMAN

'Write to files
Write #5, g_SIMANRW.RunCurrentTime, g_SIMANRW.QueueNumberOfEntities(1)

End Sub

```

### *B.2.2 The Visual Basic<sup>®</sup> code for form (frmInitial)*

Option Explicit

```

'Dim the global variables used throughout the program
Dim g_varSimDelay As Double
Dim g_varAttributes() As Double
Dim g_varServerBusy As Double
Dim g_varDelayStart As Double
Dim g_varTimeBetween As Double
Dim g_varArrivalTimeDistLast As Double
Dim g_varArrivalTimeDistMean As Double

```

```

Dim g_varProcessTimeDistLast As Double
Dim g_varProcessTimeDistMean As Double
Dim g_varDueTimeDistLast As Double
Dim g_varDueTimeDistMean As Double
Dim g_varEntitiesCount As Double
Dim g_varEntityAttributesID As Double
Dim g_varDueSeed As Long
Dim g_varProcessSeed As Long
Dim g_varArrivalSeed As Long
Dim g_varNextCreation As Double
Dim g_varQueueID As Double
Dim g_varResourceID As Double
Dim g_varRunTime As Double
Dim g_varStartTime As Double
Dim g_varServerReq As Double
Dim g_varServiceProcessTime As Double
Dim g_varTNow As Double
Dim g_varServiceStartProcessing As Double
Dim g_varNumEntitiesQueue As Integer
Dim g_varNumEntitiesServer As Integer
Dim g_varNumAttributes As Integer
Dim g_varTrialTotal As Double
Dim g_varFileName As String
Dim g_varCurrent As Integer

```

---

```

Private Sub cmdEdit_Click()
'This is the subprogram that allows you to edit
'RealWorldModel.doe

```

```

'Close the model Alternatives.doe
g_ModelAL.Close

```

```

'End the program
End

```

```

End Sub

```

---

```

Private Sub cmdEnd_Click()
'This is the subprogram that allows you to stop
'the emulator in the middle

```

```

'Now that we are finished with the model, we can end it
ActiveModel.End

```

```

'we can also close the file with the information on changing the
'control policy
Close #7

```

```

'Now close Arena
Arena.Application.Quit

```

```

'To ensure Arena is not closed again
g_varRunThrough = True

```

```

'Take away the initial screen
Unload frmInitial

```

```

End Sub

```

---

```

Private Sub cmdStart_Click()

```

'This is the main part of the emulator.  
'It is started when the command button on the initial form is  
'pressed  
'It calls different subprograms that forms the emulator

'This is the subprogram that updates the Real-world model  
'It also writes the run specific data to a file specified in  
'txtOut6 (#7) and opens files for Output (#1-#5)  
Call subRealWorldModelUpdate

'This is the subprogram that updates the Alternatives model  
Call subAlternativesModelUpdate

'This is a subprogram that updates the variables and arrays to be  
'used for the calculations  
'And writes detail to file #7  
Call subCalcUpdate

'The following part differs depending on whether the emulator is  
'restarted or continuing  
Select Case frmInitial.OptContinue

Case False  
    'For restarting

        'This subprogram determines the first breaktimes  
        'and enables manual override  
        'and disables the buttons and boxes during operation  
        Call subBreakTime

Case True  
    'For continuing

        'This is the subprogram that reads from the file with all the  
        'variables saved in the previous run  
        Call subReadVariables

        'This subprogram determines the first breaktimes  
        'and enables manual override  
        'and disables the buttons and boxes during operation  
        'especially for continuing  
        Call subBreakTimeCont

        'This is the subprogram that initialise the Real-world model  
        'to the state it was before the emulator stopped and is used to  
        'update the Real-world model when the emulator is continued  
        Call subRealWorldModelInitialiseCont

End Select

'The loop is executed as long as the required runtime is  
'larger than the smallest of the next breaktimes  
While g\_varRunTime > fncSmallest(g\_varBreaktime())

    'This is the subprogram that runs the real-world model  
    'It then gets the state to which we want to initialise the  
    'Alternatives model  
    Call subRealWorldModelRun

    'This is the subprogram that decides on the control policy to be evaluated  
    'initialises the Alternatives model to the state the real-world model ended at

'And then runs it and writes the results of the performance criteria to file  
Call subAlternativesModelRun

'This is the subprogram that initialise the Real-world model  
'to the state it was before the Alternative model was evaluated  
'And determines whether the control policy need to change  
Call subRealWorldModelInitialise

'This is the subprogram that handles the display of the progress  
'of the emulator  
'As well as write it to the output file #7  
Call subDisplay

Wend

'This is the subprogram that writes to the file all the  
'variables needed in the next run  
Call subWriteVariables

'This subprogram just closes up everything once the  
'required runtime has been reached  
Call subFinish

End Sub

---

Private Sub cmdStop\_Click()

'This is the subprogram that closes the emulator  
'without using it

'first it reads the changes to file  
Call subSaveNext

'then close Arena  
Arena.Application.Quit

End Sub

---

Private Sub OptMonitorNo\_Click()

'This subprogram makes the textboxes in the Monitor frame  
'invisible once the option button is changed

frmInitial.txt11.visible = False  
frmInitial.txt12.visible = False  
frmInitial.txt13.visible = False  
frmInitial.txt14.visible = False  
frmInitial.txt15.visible = False  
frmInitial.txt16.visible = False  
frmInitial.txt17.visible = False  
frmInitial.txt18.visible = False  
End Sub

---

Private Sub OptMonitorYes\_Click()

'This subprogram makes the textboxes in the Monitor frame  
'visible once the option button is changed

frmInitial.txt11.visible = True  
frmInitial.txt12.visible = True  
frmInitial.txt13.visible = True  
frmInitial.txt14.visible = True

```

frmInitial.txt15.visible = True
frmInitial.txt16.visible = True
frmInitial.txt17.visible = True
frmInitial.txt18.visible = True
End Sub
'
-----
Private Sub OptRestart_Change()
'This changes the entime when there is a change from continue/restart

Call subEndTime

End Sub
'
-----
Private Sub OptTech1_Click()

'The the confidence level for First technique must be enabled again
frmInitial.txtCL.Enabled = True

End Sub
'
-----
Private Sub optTech2_Click()

'Then the confidence level for First technique must be cleared
frmInitial.txtCL.Enabled = False

End Sub
'
-----
Public Sub subRealWorldModelUpdate()
'This is the subprogram that updates the Real-world model
'It also writes the run specific data to a file specified in
'txtOut6 (#7) and opens files for Output (#1-#5)

'Set the variables that accesses this model
Set g_ModelRW = Arena.Models(1)
Set g_SIMANRW = g_ModelRW.SIMAN
Set g_ModelAL = Arena.Models(2)
Set g_SIMANAL = g_ModelAL.SIMAN

'Open the file to which the changes in control policy is written
'Every time the initial values are written to the file
Open frmInitial.txtOut6 For Output As #7

'First the variables that must be changed in the real-world model
'are updated
'Starting with the Mean for exponential distribution of time
'between arrivals
fncFindModRW("varArrive").Data("Value") = frmInitial.txtArive.Text

'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Mean for exponential distribution of time between " _
        & " arrivals", frmInitial.txtArive.Text
End If

'now the Mean for exponential distribution of process times
fncFindModRW("varProcess").Data("Value") = frmInitial.txtProcess.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Mean for exponential distribution of process times ", _
        frmInitial.txtProcess.Text

```

End If

```
'and Mean for exponential distribution of due times
fncFindModRW("varDue").Data("Value") = frmInitial.txtDue.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Mean for exponential distribution of due times ", _
        frmInitial.txtDue.Text
End If
```

```
'To get the Emulator runtime the model of the real-world
fncFindModRW("varRep").Data("Value") = frmInitial.txtEnd.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Emulator runtime ", frmInitial.txtRuntime.Text
End If
```

```
'It is also set equal to a variable that is used to control the
'while wend loop
g_varRunTime = frmInitial.txtEnd.Text
```

```
'Also write to variable that the starttime is
g_varStartTime = frmInitial.txtEnd.Text - frmInitial.txtRuntime.Text
```

```
'Now for the starting control policy for the real-world system model
If optFIFO.value = True Then
    fncFindModRW("Server").Data("QRule") = "LVF"
    fncFindModRW("Server").Data("RankingExp") = "attEntityNumber"
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Starting control policy ", "Alternative 1"
    End If
End If
```

```
If optLIFO.value = True Then
    fncFindModRW("Server").Data("QRule") = "HVF"
    fncFindModRW("Server").Data("RankingExp") = "attEntityNumber"
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Starting control policy ", "Alternative 2"
    End If
End If
```

```
If optLJ.value = True Then
    fncFindModRW("Server").Data("QRule") = "LVF"
    fncFindModRW("Server").Data("RankingExp") = "attDueTime"
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Starting control policy ", "Alternative 3"
    End If
End If
```

```
If optLST.value = True Then
    fncFindModRW("Server").Data("QRule") = "HVF"
    fncFindModRW("Server").Data("RankingExp") = "attProcessTime"
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Starting control policy ", "Alternative 4"
    End If
End If
```

```
If optSST.value = True Then
    fncFindModRW("Server").Data("QRule") = "LVF"
    fncFindModRW("Server").Data("RankingExp") = "attProcessTime"
```

```

'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Starting control policy ", "Alternative 5"
End If
End If

'these initialisations insures that that the model start with the correct
'information

'This suppresses the message at the end of the
'run asking whether the user wants to see results.
g_ModelRW.QuietMode = True

'The warmup period initialised to ensure that the discrete change
'variables start calculating at the beginning
fncFindModRW("Simulate").Data("Warmup") = 0

'First initialise the time of first creation to 0 again
'Set the time of first creation 0
fncFindModRW("Create").Data("Offset") = "0"

'The Time at which the entities enter the system is changed
'to a high value to ensure that they only arrive when needed
'Set the time the entities should be injected into the system
fncFindModRW("ArrivalRest").Data("Interval") = 1000000000
fncFindModRW("ArrivalFirst").Data("Interval") = 1000000000

'The variables used to determine the time of next creation must be
'initialised
fncFindModRW("varTimeBetween").Data("Value") = 0
fncFindModRW("varDelayStart").Data("Value") = 0

'The seed numbers must be initialised to the values specified in
'originally
fncFindModRW("seeArrivals").Data("Seed") = 14561
fncFindModRW("seeProcessTime").Data("Seed") = 25971
fncFindModRW("seeDueDate").Data("Seed") = 31131

'The Arena model is used to store this values
'first Mean for exponential distribution of simulation delay
fncFindModRW("varSimDelay").Data("Value") = frmInitial.txtDelay.Text

'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Mean for exponential distribution of simulation delay ", _
        frmInitial.txtDelay.Text
End If

'This is the subprogram that save the current settings of the emulator
'so that it can be recalled next time
Call subSaveNext

'#7 is file control policy
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Output files:"
    Write #7, frmInitial.txtOut1.Text
    Write #7, frmInitial.txtOut2.Text
    Write #7, frmInitial.txtOut3.Text
    Write #7, frmInitial.txtout4.Text

```

```

    Write #7, frmInitial.txtOut5.Text
    Write #7, frmInitial.txtOut6.Text
End If

```

```

'This opens all the output files so that the VBA blocks
'can write to them
Open frmInitial.txtOut1.Text For Output As #1
Open frmInitial.txtOut2.Text For Output As #2
Open frmInitial.txtOut3.Text For Output As #3
Open frmInitial.txtOut4.Text For Output As #4
Open frmInitial.txtOut5.Text For Output As #5

```

```

'This is just to get the correct initial values in the files
'But it must only be done if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    'The file with the "Number in queue" needs to start with 0 at time 0
    Write #5, 0, 0
    'and the "Process productivity" at 1 at time 0
    Write #2, 0, 1
End If

```

```
End Sub
```

---

```
Public Sub subAlternativesModelUpdate()
```

```
'This is the subprogram that updates the Alternatives model
```

```
'The updating starts with the Mean for exponential distribution of time
'between arrivals
```

```
fncFindModAL("varArrive").Data("Value") = frmInitial.txtArive.Text
```

```
'now the Mean for exponential distribution of process times
```

```
fncFindModAL("varProcess").Data("Value") = frmInitial.txtProcess. _
    Text
```

```
'and Mean for exponential distribution of due times
```

```
fncFindModAL("varDue").Data("Value") = frmInitial.txtDue.Text
```

```
'To get the Emulator runtime the model of the real-world
```

```
fncFindModAL("varJobs").Data("Value") = frmInitial.txtPeriod.Text
```

```
'The output must only be written to file if the emulator is restarting
```

```
If frmInitial.OptRestart.value = True Then
```

```
    Write #7, "Evaluation period for alternatives (next T jobs) ", _
        frmInitial.txtPeriod.Text
```

```
End If
```

```
'This suppresses the message at the end of the
```

```
'run asking whether the user wants to see results.
```

```
g_ModelAL.QuietMode = True
```

```
'The warmup period initialised to ensure that the discrete change
```

```
'variables start calculating at the beginning
```

```
fncFindModAL("Simulate").Data("Warmup") = 0
```

```
'First initialise the time of first creation to 0 again
```

```
'Set the time of first creation 0
```

```
fncFindModAL("Create").Data("Offset") = "0"
```

```
'The Time at which the entities enter the system is changed
```

```
'to a high value to ensure that they only arrive when needed
```

```
'Set the time the entities should be injected into the system
```

```
fncFindModAL("ArrivalRest").Data("Interval") = 1000000000
fncFindModAL("ArrivalFirst").Data("Interval") = 1000000000
```

```
'The seed numbers must be initialised to the values specified in
'originally
```

```
fncFindModAL("seeArrivals").Data("Seed") = 14561
fncFindModAL("seeProcessTime").Data("Seed") = 25971
fncFindModAL("seeDueDate").Data("Seed") = 31131
```

```
'Mean for exponential distribution of simulation delay
```

```
g_varSimDelay = frmInitial.txtDelay.Text
fncFindModRW("varSimDelay").Data("Value") = g_varSimDelay
End Sub
```

---

```
Public Sub subCalcUpdate()
```

```
' This is a subprogram that updates the variables and arrays to be
'used for the calculations
'And writes detail to file #7
```

```
'Dim the variables
```

```
Dim value As Double
Dim k As Integer
Dim t As Integer
```

```
'The variable with the maximum sample size is initialised
```

```
g_varSampleSize = frmInitial.txtMaxSample.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Maximum sample size used ", _
        frmInitial.txtMaxSample.Text
End If
```

```
'The variable with the technique used is initialised
```

```
If frmInitial.OptTech1.value = True Then
    g_varTechnique = 1
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "First technique"
    End If
    g_varConfidenceLevel = frmInitial.txtCL.Text
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Confidence level ", frmInitial.txtCL.Text
    End If
```

```
Else
```

```
    g_varTechnique = 2
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Second technique"
    End If
End If
```

```
'The variable with the starting control policy
```

```
If optFIFO.value = True Then
    g_varCurrent = 1
End If
If optLIFO.value = True Then
    g_varCurrent = 2
End If
If optLJ.value = True Then
```

```

    g_varCurrent = 3
End If
If optLST.value = True Then
    g_varCurrent = 4
End If
If optSST.value = True Then
    g_varCurrent = 5
End If

'Here the data used to compute TINV is read into the array
'Open the file
Open "TINVdata.txt" For Input As #15

'Read the values that may be used into an array
For k = 1 To 6
    For t = 100 To 50 Step -1
        Input #15, value
        g_arrTINVdata(k, t) = value
    Next t
Next k
Close #15

'Now the importance of the 5 performance measures
g_arrImportance(1) = frmInitial.txtImp1.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average time in System ", frmInitial.txtImp1.Text
End If

g_arrImportance(2) = frmInitial.txtImp2.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average process productivity ", frmInitial.txtImp2.Text
End If

g_arrImportance(3) = frmInitial.txtImp3.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average lateness ", frmInitial.txtImp3.Text
End If

g_arrImportance(4) = frmInitial.txtImp4.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average job productivity ", frmInitial.txtImp4.Text
End If

g_arrImportance(5) = frmInitial.txtImp5.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average length of service queue ", frmInitial.txtImp5. _
    Text
End If

'As well as whether they should be minimised or maximised
g_arrMinMax(1) = frmInitial.txtMM1.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average time in System ", frmInitial.txtMM1.Text
End If

```

```

g_arrMinMax(2) = frmInitial.txtMM2.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average process productivity ", frmInitial.txtMM2.Text
End If

g_arrMinMax(3) = frmInitial.txtMM3.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average lateness ", frmInitial.txtMM3.Text
End If

g_arrMinMax(4) = frmInitial.txtMM4.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average job productivity ", frmInitial.txtMM4.Text
End If

g_arrMinMax(5) = frmInitial.txtMM5.Text
'The output must only be written to file if the emulator is restarting
If frmInitial.OptRestart.value = True Then
    Write #7, "Average length of service queue ", frmInitial.txtMM5.Text
End If

End Sub
'

```

---

```

Public Sub subBreakTime()
'This subprogram determines the first breaktimes
'and enables manual override
'and disables the buttons and boxes during operation

'The variables are dimensioned
Dim p As Double

'The times at which the first model runs are made is now
'determined randomly from the exponential distribution
'and the trial numbers set equal to 0 and the next policy to be
'implemented is cleared
For p = 1 To 5
    g_varBreaktime(p) = -g_varSimDelay * Log(Rnd())
    g_varTrialNumber(p) = 0
    g_varPolicyChange(p) = ""
Next p

'This is necessary to ensures manual override by making one breaktime
'incrementally smaller than the run length and the others larger
If frmInitial.optManYes = True Then
    g_varBreaktime(1) = g_varRunTime - 0.0000001
    For p = 2 To 5
        g_varBreaktime(p) = g_varRunTime + 1
    Next p
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Manual override ", "Yes"
    End If
Else
    'The output must only be written to file if the emulator is restarting
    If frmInitial.OptRestart.value = True Then
        Write #7, "Manual override ", "No"
    End If
End Sub

```

End If  
End If

'To ensure that variables are not changed and the command button  
'not presses during operation we  
'deactivate the options of the initial form  
frmInitial.cmdStart.Enabled = False  
frmInitial.fraControlPolicy.Enabled = False  
frmInitial.fraManual.Enabled = False  
frmInitial.fraSwitch.Enabled = False  
frmInitial.fraPerformance.Enabled = False  
frmInitial.fratech.Enabled = False  
frmInitial.fraVariables.Enabled = False  
frmInitial.fraMonitor.Enabled = False  
frmInitial.fraRestart.Enabled = False  
frmInitial.cmdEnd.Enabled = True  
frmInitial.cmdEdit.Enabled = False  
frmInitial.cmdStop.Enabled = False

'The real-world model is started over  
g\_ModelRW.StartOver

End Sub

---

Public Sub subBreakTimeCont()

'This subprogram disables the buttons and boxes during operation  
'especially for continuing

'To ensure that variables are not changed and the command button  
'not presses during operation we  
'deactivate the options of the initial form  
frmInitial.cmdStart.Enabled = False  
frmInitial.fraControlPolicy.Enabled = False  
frmInitial.fraManual.Enabled = False  
frmInitial.fraSwitch.Enabled = False  
frmInitial.fraPerformance.Enabled = False  
frmInitial.fratech.Enabled = False  
frmInitial.fraVariables.Enabled = False  
frmInitial.fraMonitor.Enabled = False  
frmInitial.fraRestart.Enabled = False  
frmInitial.cmdEnd.Enabled = True  
frmInitial.cmdEdit.Enabled = False  
frmInitial.cmdStop.Enabled = False

'The real-world model is started over  
g\_ModelRW.StartOver

End Sub

---

Public Sub subRealWorldModelInitialiseCont()

'This is the subprogram that initialise the Real-world model  
'to the state it was before the emulator stopped and is used to  
'update the Real-world model when the emulator is continued

'Dim variables  
Dim k As Long  
Dim i As Long

'Now that the alternatives model had been stopped, start  
'the real-world model up again

`g_ModelRW.StartOver`

'Now here is where it must be decided whether the control policy  
'must change

'It is first changed before it is determined whether it must be  
'changed, because it must be changed when the alternative is  
'evaluated the next time

```
Select Case g_varPolicyChange(fncPolicyNum(g_varPolicyEval))
  Case "Alternative 1"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attEntityNumber"
    g_varCurrent = 1
  Case "Alternative 2"
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attEntityNumber"
    g_varCurrent = 2
  Case "Alternative 3"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = "attDueTime"
    g_varCurrent = 3
  Case "Alternative 4"
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attProcessTime"
    g_varCurrent = 4
  Case "Alternative 5"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attProcessTime"
    g_varCurrent = 5
End Select
```

'Now it can be determined whether it must change next time  
If `fncSmallest(g_varTrialNumber()) > 50` Then  
  `subTechnique g_varCurrent, fncPolicyNum(g_varPolicyEval)`  
End If

'The Time of first creation is changed to ensure that no entities  
'enter before the model is initialised  
'Set the time of first creation equal to the time the next entity  
'should be created  
`fncFindModRW("Create").Data("Offset") = g_varNextCreation`

'The Time at which the entities enter the system is changed  
'to ensure that the entities enter at the initialised time  
`fncFindModRW("ArrivalFirst").Data("Interval") = g_varTNow`

'The number of entities in the sever is set  
`fncFindModRW("ArrivalFirst").Data("BatchSize") = g_varNumEntitiesServer`

'Set the time the entities should be injected into the system  
`fncFindModRW("ArrivalRest").Data("Interval") = g_varTNow`

'Set the number of entities to be entered equal to the number of  
'Entities in the queue  
`fncFindModRW("ArrivalRest").Data("BatchSize") = _`  
  `g_varNumEntitiesQueue`

```

'The variables used to determine the time of next creation must be
'initialised
fncFindModRW("varTimeBetween").Data("Value") = _
    g_varTimeBetween
fncFindModRW("varDelayStart").Data("Value") = _
    g_varDelayStart

'The seed numbers must be changed to the values computed
fncFindModRW("seeArrivals").Data("Seed") = g_varArrivalSeed
fncFindModRW("seeProcessTime").Data("Seed") = g_varProcessSeed
fncFindModRW("seeDueDate").Data("Seed") = g_varDueSeed

'This junk is only so that Arena see the change in module
'Arrivals
g_ModelRW.End
g_ModelRW.StartOver

'Change the variable entity number, so that the entities coming
'in after the injection have the correct numbers
g_SIMANRW.VariableArrayValue(fncIdenNumRW("varEntityNumber")) _
    = g_varEntitiesCount

'Import the variable with the entity data to Arena
'Get the variable data
For i = 1 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
    For k = 1 To (g_varNumAttributes)
        'Get the variable iD
        g_varEntityAttributesID = g_SIMANRW.SymbolNumber _
            ("varEntityAttributes", i, k)
        g_SIMANRW.VariableArrayValue(g_varEntityAttributesID) _
            = g_varAttributes(i, k)
    Next k
Next i

'To ensure that the sequence of entities is correct the server
'is seized until the time the entities arrive
g_SIMANRW.ResourceCapacity(1) = 0

'then the model is stopped at the time the previous one stopped
g_ModelRW.BreakOnTime (g_varTNow)
g_ModelRW.FastForward

'And the resource is unseized again
g_SIMANRW.ResourceCapacity(1) = 1

'The entities is now entered into the system by the model itself
'The real-world system model is run to the next break point
g_ModelRW.QuietMode = True

End Sub
'


---


Public Sub subRealWorldModelRun()
'This is the subprogram that runs the real-world model
'It then gets the state to which we want to initialise the
'Alternatives model

'Dimension the variables
Dim i As Long

'Break the model at the smallest of the breaktimes

```

```

g_ModelRW.BreakOnTime (fncSmallest(g_varBreaktime()))

'Determine with which alternative it corresponds
g_varPolicyEval = fncWhichAlter()
g_ModelRW.FastForward

'Increment the total amount of trials
g_varTrialTotal = g_varTrialTotal + 1

'Now the running is finished and we want to get the state to which we
'want to initialise the Alternatives model

'Begin to get the necessary data
'Get the queueID
'As there is only one queue in the model its ID is 1
g_varQueueID = 1

'Get the resourceID
'As there is only one resource its id is 1
g_varResourceID = 1

'Determine whether there is an entity in the server
If g_SIMANRW.ResourceNumberBusy(g_varResourceID) = 1 Then
    g_varNumEntitiesServer = 1
Else
    g_varNumEntitiesServer = 0
End If

'Get number of entities in Queue
g_varNumEntitiesQueue = g_SIMANRW.QueueNumberOfEntities _
    (g_varQueueID)

'Get the number attributes that entities has
g_varNumAttributes = g_SIMANRW.AttributesMaximum

'Redim the array size to the required size of entities
ReDim g_varAttributes(1 To (g_varNumEntitiesQueue + 1), _
    1 To (g_varNumAttributes)) As Double

'Get all the entities' attributes
'Starting at i = 2, because i = 1 is the entity in the server
For i = 2 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
    g_varAttributes(i, 1) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW _
            ("attTimeBetween"))
    g_varAttributes(i, 2) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW _
            ("attEntityNumber"))
    g_varAttributes(i, 3) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW("attTimeEnter"))
    g_varAttributes(i, 4) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW _
            ("attProcessTime"))
    g_varAttributes(i, 5) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW _
            ("attStartProcessing"))
    g_varAttributes(i, 6) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW("attDueTime"))
    g_varAttributes(i, 7) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW("QueueTime"))

```

```

    g_varAttributes(i, 8) = g_SIMANRW.QueuedEntityAttribute _
        (g_varQueueID, (i - 1), fncIIdenNumRW("attOriginalProcessTime"))
Next i

'Get the time now
g_varTNow = g_SIMANRW.RunCurrentTime

If g_varNumEntitiesServer = 1 Then
    'Get the attributes of the entity in server
    'Get the specified service time
    g_varServiceProcessTime = g_SIMANRW.VariableArrayValue _
        (fncIIdenNumRW("varServiceProcessTime"))

    'Get the time the processing started
    g_varServiceStartProcessing = g_SIMANRW. _
        VariableArrayValue(fncIIdenNumRW _
            ("varServiceStartProcessing"))

    'Get the time the server has been busy on this entity
    g_varServerBusy = (g_varTNow - g_varServiceStartProcessing)

    'Get the time the entity still needs to be serviced
    g_varServerReq = Abs(g_varServiceProcessTime - g_varServerBusy)

    'For the entity in the server
    g_varAttributes(1, 1) = 0
    g_varAttributes(1, 2) = g_SIMANRW. _
        VariableArrayValue(fncIIdenNumRW _
            ("varServiceEntityNumber"))
    g_varAttributes(1, 3) = g_SIMANRW. _
        VariableArrayValue(fncIIdenNumRW("varServiceTimeEnter"))
    g_varAttributes(1, 4) = g_varServerReq
    g_varAttributes(1, 5) = g_varServiceStartProcessing
    g_varAttributes(1, 6) = g_SIMANRW. _
        VariableArrayValue(fncIIdenNumRW("varServiceDueTime"))
    g_varAttributes(1, 7) = 0
    g_varAttributes(1, 8) = g_SIMANRW. _
        VariableArrayValue(fncIIdenNumRW("varServiceOriginalProcessTime"))
End If

'Get the time of next creation
g_varTimeBetween = g_SIMANRW. _
    VariableArrayValue(fncIIdenNumRW("varTimeBetween"))
g_varDelayStart = g_SIMANRW. _
    VariableArrayValue(fncIIdenNumRW("varDelayStart"))
g_varNextCreation = (g_varDelayStart + g_varTimeBetween)

'Get the number of entities that has been created
g_varEntitiesCount = g_SIMANRW.CounterValue(1) + _
    g_varNumEntitiesQueue + g_varNumEntitiesServer

'The seed numbers only change if an entity was created in the
'interval
If g_SIMANRW.VariableArrayValue(fncIIdenNumRW _
    ("varArrivalTimeDist")) <> 0 _
Then
    'Find the next seed number to be used for Arrivals
    'Find the last exponential random variable generated for
    'ARRIVALS
    'Get the last exponential variable

```

```

g_varArrivalTimeDistLast = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW("varArrivalTimeDist"))
'Get the Arrival Time distribution mean
g_varArrivalTimeDistMean = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW _
    ("varArrivalTimeDistMean"))
'Determines it's number from the uniform distribution
'And then multiply it with  $m (= 2^{31}-1)$  to get Z
g_varArrivalSeed = CLng((Exp(-g_varArrivalTimeDistLast / _
  g_varArrivalTimeDistMean) * (2 ^ 31 - 1)))
'This is the seed to use in the next replication

'Find the next seed number to be used for Process times
'Find the last exponential random variable generated for
'PROCESS TIMES
'Get the last exponential variable
g_varProcessTimeDistLast = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW("varProcessTimeDist"))
'Get the Process Time distribution mean
g_varProcessTimeDistMean = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW _
    ("varProcessTimeDistMean"))
'Determines it's number from the uniform distribution
'And then multiply it with  $m (= 2^{31}-1)$  to get Z
g_varProcessSeed = CLng((Exp(-g_varProcessTimeDistLast / _
  g_varProcessTimeDistMean) * (2 ^ 31 - 1)))
'This is the seed to use in the next replication

'Find the next seed number to be used for Due Dates
'Find the last exponential random variable generated for
'DUE DATES
'Get the last exponential variable
g_varDueTimeDistLast = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW("varDueTimeDist"))
'Get the Due Date distribution mean
g_varDueTimeDistMean = g_SIMANRW. _
  VariableArrayValue(fncIIdenNumRW("varDueTimeDistMean"))
'Determines it's number from the uniform distribution
'And then multiply it with  $m (= 2^{31}-1)$  to get Z
g_varDueSeed = CLng((Exp(-g_varDueTimeDistLast / _
  g_varDueTimeDistMean) * (2 ^ 31 - 1)))
'This is the seed to use in the next replication
End If

'Shut down the model
g_ModelRW.End
End Sub
'

```

---

```

Public Sub subAlternativesModelRun()
'This is the subprogram that decides on the control policy to be evaluated
'initialises the Alternatives model to the state the real-world model ended at
'And then runs it and writes the results of the performance criteria to file

'Dimension the variables
Dim i As Long
Dim k As Long
Dim varMean1 As Double
Dim varMean2 As Double
Dim varMean3 As Double
Dim varMean4 As Double

```

Dim varMean5 As Double

'Now that the real-world model had been ended the alternatives  
'model can be start up  
g\_ModelAL.StartOver

'This is where the control policy to be evaluated must be chosen

```
Select Case g_varPolicyEval
  Case "Alternative 1"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attEntityNumber"
  Case "Alternative 2"
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attEntityNumber"
  Case "Alternative 3"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attDueTime"
  Case "Alternative 4"
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attProcessTime"
  Case "Alternative 5"
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
      "attProcessTime"
End Select
```

'The Time of first creation is changed to ensure that no  
'entities enter before the model is initialised  
'Set the time of first creation equal to the time the next  
'entity should be created  
fncFindModAL("Create").Data("Offset") = g\_varNextCreation

'The warmup period is changed to ensure that the discrete  
'change variables only start calculating at the time of  
'initialisation  
fncFindModAL("Simulate").Data("Warmup") = g\_varTNow

'The Time at which the entities enter the system is changed  
'to ensure that the entities enter at the initialised time  
fncFindModAL("ArrivalFirst").Data("Interval") = g\_varTNow

'The number of entities in the sever is set  
fncFindModAL("ArrivalFirst").Data("BatchSize") = g\_varNumEntitiesServer

'Set the time the entities should be injected into the system  
fncFindModAL("ArrivalRest").Data("Interval") = g\_varTNow  
'Set the number of entities to be entered equal to the number of  
'Entities in the queue  
fncFindModAL("ArrivalRest").Data("BatchSize") = \_  
g\_varNumEntitiesQueue

'The seed numbers must be changed to the values computed  
fncFindModAL("seeArrivals").Data("Seed") = g\_varArrivalSeed  
fncFindModAL("seeProcessTime").Data("Seed") = g\_varProcessSeed  
fncFindModAL("seeDueDate").Data("Seed") = g\_varDueSeed

```

'This junk is only so that Arena see the change in module
'Arrivals
g_ModelAL.End
g_ModelAL.StartOver

'Change the variable entity number, so that the entities coming
'in after the injection have the correct numbers
g_SIMANAL.VariableArrayValue(fncIdenNumAL("varEntityNumber")) _
    = g_varEntitiesCount

'Import the variable with the entity data to Arena
'Get the variable data
For i = 1 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
    For k = 1 To (g_varNumAttributes)
        'Get the variable iD
        g_varEntityAttributesID = g_SIMANAL.SymbolNumber _
            ("varEntityAttributes", i, k)
        g_SIMANAL.VariableArrayValue(g_varEntityAttributesID) _
            = g_varAttributes(i, k)
    Next k
Next i

'To ensure that the sequence of entities is correct the server
'is seized Until the time the entities arrive
g_SIMANAL.ResourceCapacity(1) = 0

'then the model is stopped at the time the previous one stopped
g_ModelAL.BreakOnTime (g_varTNow)
g_ModelAL.FastForward

'And the resource is unseized again
g_SIMANAL.ResourceCapacity(1) = 1

'The entities is now entered into the system by the model itself
'The alternatives model is run to the end
g_ModelAL.QuietMode = True
g_ModelAL.FastForward

'Now that the alternative is finished evaluated, its
'results and the replication number must be written to the
'Excel file
'First increment the specific alternative's trial number
g_varTrialNumber(fncPolicyNum(g_varPolicyEval)) = _
    g_varTrialNumber(fncPolicyNum(g_varPolicyEval)) + 1

'Determine the first attributes value: Average time in system
varMean1 = g_SIMANAL.TallyAverage(fncTallyIDAL("talTimeInSystem"))

'Determine the second attributes value: Average process
'productivity
varMean2 = g_SIMANAL.DStatAverage(3)

'Determine the third attributes value: Average lateness
varMean3 = g_SIMANAL.TallyAverage(fncTallyIDAL("talLateness"))

'Determine the fourth attributes value: Average Job productivity
varMean4 = g_SIMANAL.TallyAverage(fncTallyIDAL("talJobProductivity"))

'Determine the fifth attributes value: Average queue length
varMean5 = g_SIMANAL.DStatAverage(2)

```

```
subAddTrial fncPolicyNum(g_varPolicyEval), varMean1, varMean2, varMean3, _
    varMean4, varMean5, g_varTrialNumber(fncPolicyNum(g_varPolicyEval))
```

```
'As we are finished with the Alternatives model we can
'end it
g_ModelAL.End
```

```
End Sub
```

---

```
Public Sub subRealWorldModelInitialise()
```

```
'This is the subprogram that initialise the Real-world model
'to the state it was before the Alternative model was evaluated
'And determines whether the control policy need to change
```

```
'Dim variables
Dim k As Long
Dim i As Long
```

```
'Now that the alternatives model had been stopped, start
'the real-world model up again
g_ModelRW.StartOver
```

```
'Now here is where it must be decided whether the control policy
'must change
```

```
'It is first changed before it is determined whether it must be
'changed, because it must be changed when the alternative is
'evaluated the next time
```

```
Select Case g_varPolicyChange(fncPolicyNum(g_varPolicyEval))
```

```
Case "Alternative 1"
```

```
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
        "attEntityNumber"
```

```
    g_varCurrent = 1
```

```
Case "Alternative 2"
```

```
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
        "attEntityNumber"
```

```
    g_varCurrent = 2
```

```
Case "Alternative 3"
```

```
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = "attDueTime"
```

```
    g_varCurrent = 3
```

```
Case "Alternative 4"
```

```
    fncFindModAL("Server").Data("QRule") = "HVF"
    fncFindModAL("Server").Data("RankingExp") = _
        "attProcessTime"
```

```
    g_varCurrent = 4
```

```
Case "Alternative 5"
```

```
    fncFindModAL("Server").Data("QRule") = "LVF"
    fncFindModAL("Server").Data("RankingExp") = _
        "attProcessTime"
```

```
    g_varCurrent = 5
```

```
End Select
```

```
'Now it can be determined whether it must change next time
```

```
If fncSmallest(g_varTrialNumber()) > 50 Then
```

```
    subTechnique g_varCurrent, fncPolicyNum(g_varPolicyEval)
```

```
End If
```

```

'The Time of first creation is changed to ensure that no entities
'enter before the model is initialised
'Set the time of first creation equal to the time the next entity
'should be created
fncFindModRW("Create").Data("Offset") = g_varNextCreation

'The Time at which the entities enter the system is changed
'to ensure that the entities enter at the initialised time
fncFindModRW("ArrivalFirst").Data("Interval") = g_varTNow

'The number of entities in the sever is set
fncFindModRW("ArrivalFirst").Data("BatchSize") = g_varNumEntitiesServer

'Set the time the entities should be injected into the system
fncFindModRW("ArrivalRest").Data("Interval") = g_varTNow

'Set the number of entities to be entered equal to the number of
'Entities in the queue
fncFindModRW("ArrivalRest").Data("BatchSize") = _
    g_varNumEntitiesQueue

'The variables used to determine the time of next creation must be
'initialised
fncFindModRW("varTimeBetween").Data("Value") = _
    g_varTimeBetween
fncFindModRW("varDelayStart").Data("Value") = _
    g_varDelayStart

'The seed numbers must be changed to the values computed
fncFindModRW("seeArrivals").Data("Seed") = g_varArrivalSeed
fncFindModRW("seeProcessTime").Data("Seed") = g_varProcessSeed
fncFindModRW("seeDueDate").Data("Seed") = g_varDueSeed

'This junk is only so that Arena see the change in module
'Arrivals
g_ModelRW.End
g_ModelRW.StartOver

'Change the variable entity number, so that the entities coming
'in after the injection have the correct numbers
g_SIMANRW.VariableArrayValue(fncIdenNumRW("varEntityNumber")) _
    = g_varEntitiesCount

'Import the variable with the entity data to Arena
'Get the variable data
For i = 1 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
    For k = 1 To (g_varNumAttributes)
        'Get the variable iD
        g_varEntityAttributesID = g_SIMANRW.SymbolNumber _
            ("varEntityAttributes", i, k)
        g_SIMANRW.VariableArrayValue(g_varEntityAttributesID) _
            = g_varAttributes(i, k)
    Next k
Next i

'To ensure that the sequence of entities is correct the server
'is seized until the time the entities arrive
g_SIMANRW.ResourceCapacity(1) = 0

'then the model is stopped at the time the previous one stopped

```

```

g_ModelRW.BreakOnTime (g_varTNow)
g_ModelRW.FastForward

'And the resource is unseized again
g_SIMANRW.ResourceCapacity(1) = 1

'The entities is now entered into the system by the model itself
'The real-world system model is run to the next break point
g_ModelRW.QuietMode = True
End Sub
'

```

---

```

Public Sub subDisplay()
'This is the subprogram that handles the display of the progress
'of the emulator
'As well as write it to the output file #7

'Dim variables
Dim i As Long

'This is where I want to display the current situation
'First the total trial number
frmInitial.txt11.Text = g_varTrialTotal

'Then the breaktime
frmInitial.txt12.Text = Format(g_varBreaktime(fncPolicyNum(g_varPolicyEval)) _
, "##.####")

'Then the control policy currently used
Select Case g_varCurrent
Case 1
frmInitial.txt13.Text = "Alternative 1"
Case 2
frmInitial.txt13.Text = "Alternative 2"
Case 3
frmInitial.txt13.Text = "Alternative 3"
Case 4
frmInitial.txt13.Text = "Alternative 4"
Case 5
frmInitial.txt13.Text = "Alternative 5"
End Select

'Then the policy evaluated
frmInitial.txt14.Text = g_varPolicyEval

'Then the trial number of the specific control policy
frmInitial.txt15.Text = g_varTrialNumber(fncPolicyNum(g_varPolicyEval))

'Then whether all the trial numbers are larger than 50
frmInitial.txt16.Text = fncSmallest(g_varTrialNumber()) > 50

'Then whether the control policy must change
If g_varAlternativesTotal > g_varCurrentTotal Then
frmInitial.txt17.Text = "Yes"
Else
frmInitial.txt17.Text = "No"
End If

'This just writes all the above to output file #7
Write #7, g_varTrialTotal, g_varBreaktime(fncPolicyNum(g_varPolicyEval)), _
g_varCurrent, _

```

```
g_varPolicyEval, g_varTrialNumber(fncPolicyNum(g_varPolicyEval)), _
fncSmallest(g_varTrialNumber()) > 50, _
g_varAlternativesTotal > g_varCurrentTotal
```

```
'The new breakpoint for this specific alternative must
'first be determined
'First it is determine the number of the alternative and then
'its specific breaktime is changed
g_varBreakeTime(fncPolicyNum(g_varPolicyEval)) = _
g_varTNow - g_varSimDelay * Log(Rnd())
```

```
'Now just display the time the control policy must change
Select Case g_varAlternativesTotal > g_varCurrentTotal
    Case True
        frmInitial.txt18.Text = Format(g_varBreakeTime(fncPolicyNum _
            (g_varPolicyEval)), "##.####")
    Case False
        frmInitial.txt18.Text = "No switch required"
End Select
```

```
End Sub
```

---

```
Public Sub subFinish()
```

```
'This subprogram just closes up everything once the
'required runtime has been reached
```

```
'Now that we are finished with the real-world model, we can end it
g_ModelRW.End
```

```
'we can also close the file with the information on changing the
'control policy
Close #7
```

```
'Close Arena models
g_ModelAL.Save
g_ModelRW.Save
```

```
'Now close Arena
Arena.Application.Quit
```

```
'To ensure Arena and Excel is not closed again
g_varRunThrough = True
```

```
'Take away the initial screen and monitor screen
Unload frmInitial
```

```
End Sub
```

---

```
Private Sub txtRuntime_Change()
```

```
'This changes the end time every time the runtime is changed
```

```
Call subEndTime
```

```
End Sub
```

---

```
Public Sub subReadVariables()
```

```
'This is the subprogram that reads from the file with all the
'variables saved in the previous run
```

```
'Dim variables
```

```

Dim k As Long
Dim i As Long
Dim j As Long
Dim varDummyString As String
Dim varDummyDouble As Double
Dim varDummyInteger As Integer
Dim varDummyLong As Long
Dim varDummyBoolean As Boolean

'First open the file
Open "Continue.txt" For Input As #23

'Input the position it is at
Input #23, varDummyDouble

'Input the number of the emulation
Input #23, varDummyDouble

'Input the variable g_varPolicyEval
Input #23, varDummyString
g_varPolicyEval = varDummyString

'Input the variable g_varCurrent
Input #23, varDummyInteger
g_varCurrent = varDummyInteger

'Input the array g_varPolicyChange
For k = 1 To 5
    Input #23, varDummyString
    g_varPolicyChange(k) = varDummyString
Next k

'Input the array g_varTrialNumber
For k = 1 To 5
    Input #23, varDummyDouble
    g_varTrialNumber(k) = varDummyDouble
Next k

'Input the variable g_varNextCreation
Input #23, varDummyDouble
g_varNextCreation = varDummyDouble

'Input the variable g_varTNow
Input #23, varDummyDouble
g_varTNow = varDummyDouble

'Input the variable g_varNumEntitiesServer
Input #23, varDummyInteger
g_varNumEntitiesServer = varDummyInteger

'Input the variable g_varNumEntitiesQueue
Input #23, varDummyInteger
g_varNumEntitiesQueue = varDummyInteger

'Input the variable g_varTimeBetween
Input #23, varDummyDouble
g_varTimeBetween = varDummyDouble

'Input the variable g_varDelayStart
Input #23, varDummyDouble

```

```

g_varDelayStart = varDummyDouble

'Input the variable g_varArrivalSeed
Input #23, varDummyLong
g_varArrivalSeed = varDummyLong

'Input the variable g_varProcessSeed
Input #23, varDummyLong
g_varProcessSeed = varDummyLong

'Input the variable g_varDueSeed
Input #23, varDummyLong
g_varDueSeed = varDummyLong

'Input the variable g_varEntitiesCount
Input #23, varDummyDouble
g_varEntitiesCount = varDummyDouble

'Input the variable g_varNumAttributes
Input #23, varDummyInteger
g_varNumAttributes = varDummyInteger

'Redim the array size to the required size of entities
ReDim g_varAttributes(1 To (g_varNumEntitiesQueue + 1), _
    1 To (g_varNumAttributes)) As Double

'Input the array g_varAttributes
For i = 1 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
    For k = 1 To (g_varNumAttributes)
        Input #23, g_varAttributes(i, k)
    Next k
Next i

'Input the variable g_varSimDelay
Input #23, varDummyDouble
g_varSimDelay = varDummyDouble

'Input the variable g_varServerBusy
Input #23, varDummyDouble
g_varServerBusy = varDummyDouble

'Input the variable g_varArrivalTimeDistLast
Input #23, varDummyDouble
g_varArrivalTimeDistLast = varDummyDouble

'Input the variable g_varArrivalTimeDistMean
Input #23, varDummyDouble
g_varArrivalTimeDistMean = varDummyDouble

'Input the variable g_varProcessTimeDistLast
Input #23, varDummyDouble
g_varProcessTimeDistLast = varDummyDouble

'Input the variable g_varProcessTimeDistMean
Input #23, varDummyDouble
g_varProcessTimeDistMean = varDummyDouble

'Input the variable g_varDueTimeDistLast
Input #23, varDummyDouble
g_varDueTimeDistLast = varDummyDouble

```

```

'Input the variable g_varDueTimeDistMean
Input #23, varDummyDouble
g_varDueTimeDistMean = varDummyDouble

'Input the variable g_varEntityAttributesID
Input #23, varDummyDouble
g_varEntityAttributesID = varDummyDouble

'Input the variable g_varQueueID
Input #23, varDummyDouble
g_varQueueID = varDummyDouble

'Input the variable g_varResourceID
Input #23, varDummyDouble
g_varResourceID = varDummyDouble

'Input the variable g_varStartTime
Input #23, varDummyDouble
g_varStartTime = varDummyDouble

'Input the variable g_varServerReq
Input #23, varDummyDouble
g_varServerReq = varDummyDouble

'Input the variable g_varServiceProcessTime
Input #23, varDummyDouble
g_varServiceProcessTime = varDummyDouble

'Input the variable g_varServiceStartProcessing
Input #23, varDummyDouble
g_varServiceStartProcessing = varDummyDouble

'Input the variable g_varTrialTotal
Input #23, varDummyDouble
g_varTrialTotal = varDummyDouble

'Input the variable g_varFileName
Input #23, varDummyString
g_varFileName = varDummyString

'Input the array g_arrTINVdata(1 To 6, 1 To 100)
For k = 1 To 6
  For i = 1 To 100
    Input #23, varDummyDouble
    g_arrTINVdata(k, i) = varDummyDouble
  Next i
Next k

'Input the array g_arrData(1 To 5, 1 To 6, 1 To 2000)
For k = 1 To 5
  For i = 1 To 6
    For j = 1 To 2000
      Input #23, varDummyDouble
      g_arrData(k, i, j) = varDummyDouble
    Next j
  Next i
Next k

'Input the array g_arrCalculations(1 To 5, 1 To 2000)

```

```

For k = 1 To 5
  For i = 1 To 2000
    Input #23, varDummyDouble
    g_arrCalculations(k, i) = varDummyDouble
  Next i
Next k

```

```

'Input the array g_varBreaktime
For k = 1 To 5
  Input #23, varDummyDouble
  g_varBreaktime(k) = varDummyDouble
Next k

```

```

'Input the array g_arrImportance(1 To 5)
For k = 1 To 5
  Input #23, varDummyDouble
  g_arrImportance(k) = varDummyDouble
Next k

```

```

'Input the variable g_varConfidenceLevel
Input #23, varDummyDouble
g_varConfidenceLevel = varDummyDouble

```

```

'Input the variable g_varSampleSizeUsed
Input #23, varDummyDouble
g_varSampleSizeUsed = varDummyDouble

```

```

'Input the variable g_varSampleSize
Input #23, varDummyDouble
g_varSampleSize = varDummyDouble

```

```

'Input the variable g_varAlternativesTotal
Input #23, varDummyDouble
g_varAlternativesTotal = varDummyDouble

```

```

'Input the variable g_varCurrentTotal
Input #23, varDummyDouble
g_varCurrentTotal = varDummyDouble

```

```

'Input the array g_varPolicyNames
For k = 1 To 5
  Input #23, varDummyString
  g_varPolicyNames(k) = varDummyString
Next k

```

```

'Input the array g_arrMinMax(1 To 5)
For k = 1 To 5
  Input #23, varDummyString
  g_arrMinMax(k) = varDummyString
Next k

```

```

'Input the variable g_varRunThrough
Input #23, varDummyBoolean
g_varRunThrough = varDummyBoolean

```

```

'Input the variable g_varTechnique
Input #23, varDummyInteger
g_varTechnique = varDummyInteger

```

```

'Close the file again

```

Close #23

End Sub

---

```
Public Sub subWriteVariables()
'This is the subprogram that writes to the file all the
'variables needed in the next run

'Dim variables
Dim k As Long
Dim i As Long
Dim j As Long

'First open the file
Open "Continue.txt" For Output As #24

'Write the position it is at
Write #24, g_varTNow

'Write the number of the emulation
Write #24, CDb1(frmInitial.txtNumber.Text)

'Write the variable g_varPolicyEval
Write #24, g_varPolicyEval

'Write the variable g_varCurrent
Write #24, g_varCurrent

'Write the array g_varPolicyChange
For k = 1 To 5
    Write #24, g_varPolicyChange(k)
Next k

'Write the array g_varTrialNumber
For k = 1 To 5
    Write #24, g_varTrialNumber(k)
Next k

'Write the variable g_varNextCreation
Write #24, g_varNextCreation

'Write the variable g_varTNow
Write #24, g_varTNow

'Write the variable g_varNumEntitiesServer
Write #24, g_varNumEntitiesServer

'Write the variable g_varNumEntitiesQueue
Write #24, g_varNumEntitiesQueue

'Write the variable g_varTimeBetween
Write #24, g_varTimeBetween

'Write the variable g_varDelayStart
Write #24, g_varDelayStart

'Write the variable g_varArrivalSeed
Write #24, g_varArrivalSeed

'Write the variable g_varProcessSeed
```

```

Write #24, g_varProcessSeed

'Write the variable g_varDueSeed
Write #24, g_varDueSeed

'Write the variable g_varEntitiesCount
Write #24, g_varEntitiesCount

'Write the variable g_varNumAttributes
Write #24, g_varNumAttributes

'Write the array g_varAttributes
For i = 1 To (g_varNumEntitiesQueue + g_varNumEntitiesServer)
  For k = 1 To (g_varNumAttributes)
    Write #24, g_varAttributes(i, k)
  Next k
Next i

'Write the variable g_varSimDelay
Write #24, g_varSimDelay

'Write the variable g_varServerBusy
Write #24, g_varServerBusy

'Write the variable g_varArrivalTimeDistLast
Write #24, g_varArrivalTimeDistLast

'Write the variable g_varArrivalTimeDistMean
Write #24, g_varArrivalTimeDistMean

'Write the variable g_varProcessTimeDistLast
Write #24, g_varProcessTimeDistLast

'Write the variable g_varProcessTimeDistMean
Write #24, g_varProcessTimeDistMean

'Write the variable g_varDueTimeDistLast
Write #24, g_varDueTimeDistLast

'Write the variable g_varDueTimeDistMean
Write #24, g_varDueTimeDistMean

'Write the variable g_varEntityAttributesID
Write #24, g_varEntityAttributesID

'Write the variable g_varQueueID
Write #24, g_varQueueID

'Write the variable g_varResourceID
Write #24, g_varResourceID

'Write the variable g_varStartTime
Write #24, g_varStartTime

'Write the variable g_varServerReq
Write #24, g_varServerReq

'Write the variable g_varServiceProcessTime
Write #24, g_varServiceProcessTime

```

```
'Write the variable g_varServiceStartProcessing  
Write #24, g_varServiceStartProcessing
```

```
'Write the variable g_varTrialTotal  
Write #24, g_varTrialTotal
```

```
'Write the variable g_varFileName  
Write #24, g_varFileName
```

```
'Write the array g_arrTINVdata(1 To 6, 1 To 100)  
For k = 1 To 6  
  For i = 1 To 100  
    Write #24, g_arrTINVdata(k, i)  
  Next i  
Next k
```

```
'Write the array g_arrData(1 To 5, 1 To 6, 1 To 2000)  
For k = 1 To 5  
  For i = 1 To 6  
    For j = 1 To 2000  
      Write #24, g_arrData(k, i, j)  
    Next j  
  Next i  
Next k
```

```
'Write the array g_arrCalculations(1 To 5, 1 To 2000)  
For k = 1 To 5  
  For i = 1 To 2000  
    Write #24, g_arrCalculations(k, i)  
  Next i  
Next k
```

```
'Write the array g_varBreaktime  
For k = 1 To 5  
  Write #24, g_varBreaktime(k)  
Next k
```

```
'Write the array g_arrImportance(1 To 5)  
For k = 1 To 5  
  Write #24, g_arrImportance(k)  
Next k
```

```
'Write the variable g_varConfidenceLevel  
Write #24, g_varConfidenceLevel
```

```
'Write the variable g_varSampleSizeUsed  
Write #24, g_varSampleSizeUsed
```

```
'Write the variable g_varSampleSize  
Write #24, g_varSampleSize
```

```
'Write the variable g_varAlternativesTotal  
Write #24, g_varAlternativesTotal
```

```
'Write the variable g_varCurrentTotal  
Write #24, g_varCurrentTotal
```

```
'Write the array g_varPolicyNames  
For k = 1 To 5  
  Write #24, g_varPolicyNames(k)
```

Next k

'Write the array g\_arrMinMax(1 To 5)

For k = 1 To 5

    Write #24, g\_arrMinMax(k)

Next k

'Write the variable g\_varRunThrough

Write #24, g\_varRunThrough

'Write the variable g\_varTechnique

Write #24, g\_varTechnique

'Close the file again

Close #24

End Sub

### B.2.3 *The Visual Basic® code for Module 2*

Option Explicit

```
'The global variables are dimensioned
Public g_varBreaktime(5) As Double
Public g_varPolicyNames(5) As String
Public g_varRunThrough As Boolean
Public g_ModelRW As Arena.Model
Public g_ModelAL As Arena.Model
Public g_SIMANRW As Arena.SIMAN
Public g_SIMANAL As Arena.SIMAN
Public g_arrData(1 To 5, 1 To 6, 1 To 2000) As Double
Public g_arrCalculations(1 To 5, 1 To 2000) As Double
Public g_varTrialNumber(5) As Double
Public g_arrMinMax(1 To 5) As String
Public g_arrImportance(1 To 5) As Double
Public g_arrTINVdata(1 To 6, 1 To 100) As Double
Public g_varPolicyEval As String
Public g_varPolicyChange(5) As String
Public g_varTechnique As Integer
Public g_varConfidenceLevel As Double
Public g_varSampleSizeUsed As Double
Public g_varSampleSize As Double
Public g_varAlternativesTotal As Double
Public g_varCurrentTotal As Double
'


---


Public Function fncIdenNumRW(varArenaVariable As String) As Double
'This function finds the SIMAN identification number for given variables
'in the Real-world Arena model

fncIdenNumRW = g_SIMANRW.SymbolNumber(varArenaVariable)

End Function
'


---


Public Function fncIdenNumAL(varArenaVariable As String) As Double
'This function finds the SIMAN identification number for given variables
'in the Alternatives Arena model

fncIdenNumAL = g_SIMANAL.SymbolNumber(varArenaVariable)

End Function
'


---


Public Function fncFindModRW(varArenaMod As String) As Module
'This function finds modules in the Real-world Arena model, given its name

'Dim the variables
Dim p As Integer

'Find the module
p = g_ModelRW.Modules.Find(smFindTag, varArenaMod)

If p > 0 Then
    Set fncFindModRW = g_ModelRW.Modules(p)
Else
    'If the module was not found, display a message and exit
    MsgBox "Did not find module "
End If
```

```

End Function
'


---


Public Function fncFindModAL(varArenaMod As String) As Module
'This function finds modules in the Alternatives Arena model, given its name

'Dim the variables
Dim p As Integer

'Find the module
p = g_ModelAL.Modules.Find(smFindTag, varArenaMod)

If p > 0 Then
    Set fncFindModAL = g_ModelAL.Modules(p)
Else
    'If the module was not found, display a message and exit
    MsgBox "Did not find module "
End If

End Function
'


---


Public Function fncSmallest(varArray() As Double) As Double
'This function determines the smallest value of 5 variables
'in an array

'Dim variables
Dim i As Integer
Dim temp As Double
Dim varArrayTemp(5) As Double

'First read the values into a temporary array, so that the
'sequence do not get messed up
For i = 1 To 5
    varArrayTemp(i) = varArray(i)
Next i

'Now sort them
For i = 1 To 4
    If varArrayTemp(i) < varArrayTemp(i + 1) Then
        temp = varArrayTemp(i)
        varArrayTemp(i) = varArrayTemp(i + 1)
        varArrayTemp(i + 1) = temp
    End If
Next i

fncSmallest = varArrayTemp(5)

End Function
'


---


Public Function fncWhichAlter() As String
'This function determines which alternative control policy is
'evaluated for a specific Breaktime(p)

'Dim the variables
Dim i As Integer

'Find the control policy
For i = 1 To 5
    If g_varBreaktime(i) = fncSmallest(g_varBreaktime()) Then
        fncWhichAlter = g_varPolicyNames(i)
    End If

```

Next i

End Function

---

Public Function fncPolicyNum(name As String) As Double  
 'This function determines the number corresponding to the  
 'alternatives name

If name = "Alternative 1" Then fncPolicyNum = 1  
 If name = "Alternative 2" Then fncPolicyNum = 2  
 If name = "Alternative 3" Then fncPolicyNum = 3  
 If name = "Alternative 4" Then fncPolicyNum = 4  
 If name = "Alternative 5" Then fncPolicyNum = 5

End Function

---

Public Function fncTallyIDAL(name As String) As Double  
 'this is a function that seeks the SIMAN ID of a tally  
 'in the Real-world model

'Set the variables  
 Set g\_ModelAL = Arena.Models(2)  
 Set g\_SIMANAL = g\_ModelAL.SIMAN

fncTallyIDAL = g\_SIMANAL.SymbolNumber(name)

End Function

---

Public Function fncTallyIDRW(name As String) As Double  
 'this is a function that seeks the ID of a tally  
 'in the Alternatives model

'Set the variables  
 Set g\_SIMANRW = Arena.Models(1).SIMAN

fncTallyIDRW = g\_SIMANRW.SymbolNumber(name)

End Function

---

Public Sub subGetPrevious()  
 'The previous values for the values are displayed on the  
 'Initial form here

'Dim variables  
 Dim g\_varFileName As String

'Open the file  
 Open "Config.txt" For Input As #1

'First for frame Variables  
 Input #1, g\_varFileName  
 frmInitial.txtArive.Text = g\_varFileName  
 Input #1, g\_varFileName  
 frmInitial.txtProcess.Text = g\_varFileName  
 Input #1, g\_varFileName  
 frmInitial.txtDue.Text = g\_varFileName  
 Input #1, g\_varFileName  
 frmInitial.txtRuntime.Text = g\_varFileName  
 Input #1, g\_varFileName  
 frmInitial.txtPeriod.Text = g\_varFileName

```
Input #1, g_varFileName
frmInitial.txtDelay.Text = g_varFileName

'Then frame Control policy usage
Input #1, g_varFileName
frmInitial.txtOut6.Text = g_varFileName

'Then frame Technique used
Input #1, g_varFileName
frmInitial.txtMaxSample.Text = g_varFileName
Input #1, g_varFileName
frmInitial.OptTech1.value = g_varFileName
Input #1, g_varFileName
frmInitial.txtCL.Text = g_varFileName
Input #1, g_varFileName
frmInitial.OptTech2.value = g_varFileName

'Then frame Starting control policy
Input #1, g_varFileName
frmInitial.optFIFO.value = g_varFileName
Input #1, g_varFileName
frmInitial.optLIFO.value = g_varFileName
Input #1, g_varFileName
frmInitial.optLJ.value = g_varFileName
Input #1, g_varFileName
frmInitial.optLST.value = g_varFileName
Input #1, g_varFileName
frmInitial.optSST.value = g_varFileName

'Then frame Manual override
Input #1, g_varFileName
frmInitial.optManYes.value = g_varFileName
Input #1, g_varFileName
frmInitial.optManNo.value = g_varFileName

'Then frame progress monitor
Input #1, g_varFileName
frmInitial.OptMonitorYes.value = g_varFileName
Input #1, g_varFileName
frmInitial.OptMonitorNo.value = g_varFileName

'Then frame Performance measures
Input #1, g_varFileName
frmInitial.txtImp1.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtImp2.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtImp3.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtImp4.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtImp5.Text = g_varFileName

Input #1, g_varFileName
frmInitial.txtMM1.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtMM2.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtMM3.Text = g_varFileName
Input #1, g_varFileName
```

```
frmInitial.txtMM4.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtMM5.Text = g_varFileName
```

```
Input #1, g_varFileName
frmInitial.txtOut1.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtOut2.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtOut3.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtout4.Text = g_varFileName
Input #1, g_varFileName
frmInitial.txtOut5.Text = g_varFileName
```

```
'Then frame restart
Input #1, g_varFileName
frmInitial.OptContinue.value = g_varFileName
Input #1, g_varFileName
frmInitial.OptRestart.value = g_varFileName
```

```
'Close the file
Close #1
```

```
End Sub
```

---

```
Public Sub subSaveNext()
```

```
'Here we want to write the current settings to a file so that we can
'remember the settings when we use the emulator again
```

```
'Open the file
Open "Config.txt" For Output As #11
```

```
'First for frame Variables
Write #11, frmInitial.txtArive.Text
Write #11, frmInitial.txtProcess.Text
Write #11, frmInitial.txtDue.Text
Write #11, frmInitial.txtRuntime.Text
Write #11, frmInitial.txtPeriod.Text
Write #11, frmInitial.txtDelay.Text
```

```
'Then frame Control policy usage
Write #11, frmInitial.txtOut6.Text
```

```
'Then frame Technique used
Write #11, frmInitial.txtMaxSample.Text
Write #11, frmInitial.OptTech1.value
Write #11, frmInitial.txtCL.Text
Write #11, frmInitial.OptTech2.value
```

```
'Then frame Starting control policy
Write #11, frmInitial.optFIFO.value
Write #11, frmInitial.optLIFO.value
Write #11, frmInitial.optLJ.value
Write #11, frmInitial.optLST.value
Write #11, frmInitial.optSST.value
```

```
'Then frame Manual override
Write #11, frmInitial.optManYes.value
Write #11, frmInitial.optManNo.value
```

```
'Then frame progress monitor
Write #11, frmInitial.OptMonitorYes.value
Write #11, frmInitial.OptMonitorNo.value
```

```
'Then frame Performance measures
Write #11, frmInitial.txtImp1.Text
Write #11, frmInitial.txtImp2.Text
Write #11, frmInitial.txtImp3.Text
Write #11, frmInitial.txtImp4.Text
Write #11, frmInitial.txtImp5.Text
Write #11, frmInitial.txtMM1.Text
Write #11, frmInitial.txtMM2.Text
Write #11, frmInitial.txtMM3.Text
Write #11, frmInitial.txtMM4.Text
Write #11, frmInitial.txtMM5.Text
Write #11, frmInitial.txtOut1.Text
Write #11, frmInitial.txtOut2.Text
Write #11, frmInitial.txtOut3.Text
Write #11, frmInitial.txtOut4.Text
Write #11, frmInitial.txtOut5.Text
```

```
'Then frame restart monitor
Write #11, frmInitial.OptContinue.value
Write #11, frmInitial.OptRestart.value
```

```
'Close the file
Close #11
```

```
End Sub
'
```

---

```
Public Sub subAddTrial(IndAlternative As Integer, varMean1 As Double, _
    varMean2 As Double, varMean3 As Double, varMean4 As Double, _
    varMean5 As Double, varTrialNumber As Double)
```

```
'This is a subprogram that adds a trial to the top of the array with the data
```

```
'Dim variables
Dim k As Integer
Dim p As Integer
```

```
'First move all the previous values down the stack
For k = 2000 To 2 Step -1
    For p = 1 To 6
        g_arrData(IndAlternative, p, k) = g_arrData(IndAlternative, p, k - 1)
    Next p
Next k
```

```
'Now enter the new data at the top
g_arrData(IndAlternative, 1, 1) = varMean1
g_arrData(IndAlternative, 2, 1) = varMean2
g_arrData(IndAlternative, 3, 1) = varMean3
g_arrData(IndAlternative, 4, 1) = varMean4
g_arrData(IndAlternative, 5, 1) = varMean5
g_arrData(IndAlternative, 6, 1) = varTrialNumber
```

```
End Sub
'
```

---

```
Public Sub subDifference(IndCurrent As Integer, IndAlternative As Integer)
'This is a subprogram that writes the difference between the array values of the
'Alternative and the current to a new array used for the calculations
```

```

'Dim variables
Dim k As Integer
Dim p As Integer

'determine the difference for 5 criteria and 2000 values each
For k = 1 To 5
    For p = 1 To 2000
        g_arrCalculations(k, p) = g_arrData(IndAlternative, k, p) _
            - g_arrData(IndCurrent, k, p)
    Next p
Next k

End Sub
'
Public Function fncAverage(IndCriteria As Integer)
'This is a function that determines the average of the
'sample size used

'Dim variables
Dim varSum As Double
Dim k As Integer

'Initialise variables
varSum = 0

'determine sum
For k = 1 To g_varSampleSizeUsed
    varSum = varSum + g_arrCalculations(IndCriteria, k)
Next k

'determine average
fncAverage = varSum / g_varSampleSizeUsed

End Function
'
Public Function fncSampleSize()
'This is a function that determines the sample size used in the calculations

Select Case g_varSampleSize
    'It first selects depending on the maximum sample size _
    specified (g_varSampleSize)
    Case 0
        fncSampleSize = 0
    'It first selects depending on the maximum sample size specified _
    (g_varSampleSize)
    Case 50
        Select Case fncSmallest(g_varTrialNumber())
            'And then on the smallest current trial
            'number (fncSmallest(g_varTrialNumber()))
            Case 0 To 49
                fncSampleSize = 0
            'And then on the smallest current trial
            'number (fncSmallest(g_varTrialNumber()))
            Case Is >= 50
                fncSampleSize = 50
        End Select
    'It first selects depending on the maximum sample size specified _
    (g_varSampleSize)
    Case 100

```

```

Select Case fncSmallest(g_varTrialNumber())
  'And then on the smallest current trial
  'number (fncSmallest(g_varTrialNumber()))
  Case 0 To 49
    fncSampleSize = 0
  'And then on the smallest current trial
  'number (fncSmallest(g_varTrialNumber()))
  Case 50 To 99
    fncSampleSize = 50
  'And then on the smallest current trial
  'number (fncSmallest(g_varTrialNumber()))
  Case Is >= 100
    fncSampleSize = 100
End Select
'It first selects depending on the maximum sample size specified _
(g_varSampleSize)
Case 250
  Select Case fncSmallest(g_varTrialNumber())
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 0 To 49
      fncSampleSize = 0
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 50 To 99
      fncSampleSize = 50
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 100 To 249
      fncSampleSize = 100
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case Is >= 250
      fncSampleSize = 250
  End Select
'It first selects depending on the maximum sample size specified _
(g_varSampleSize)
Case 500
  Select Case fncSmallest(g_varTrialNumber())
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 0 To 49
      fncSampleSize = 0
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 50 To 99
      fncSampleSize = 50
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 100 To 249
      fncSampleSize = 100
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case 250 To 499
      fncSampleSize = 250
    'And then on the smallest current trial
    'number (fncSmallest(g_varTrialNumber()))
    Case Is >= 500
      fncSampleSize = 500
  End Select

```

'It first selects depending on the maximum sample size specified \_  
(g\_varSampleSize)

Case 1000

```
Select Case fncSmallest(g_varTrialNumber())
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 0 To 49
    fncSampleSize = 0
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 50 To 99
    fncSampleSize = 50
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 100 To 249
    fncSampleSize = 100
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 250 To 499
    fncSampleSize = 250
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 500 To 999
    fncSampleSize = 500
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case Is >= 1000
    fncSampleSize = 1000
```

End Select

'It first selects depending on the maximum sample size specified \_  
(g\_varSampleSize)

Case 2000

```
Select Case fncSmallest(g_varTrialNumber())
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 0 To 49
    fncSampleSize = 0
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 50 To 99
    fncSampleSize = 50
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 100 To 249
    fncSampleSize = 100
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 250 To 499
    fncSampleSize = 250
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 500 To 999
    fncSampleSize = 500
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case 1000 To 1999
    fncSampleSize = 1000
'And then on the smallest current trial
'number (fncSmallest(g_varTrialNumber()))
Case Is >= 2000
```

```

        fncSampleSize = 2000
    End Select
End Select

End Function
'
Public Function fncVariance(IndCriteria As Integer)
'This is a function that determines the variance of a sample

'Dim variables
Dim varSum As Double
Dim k As Integer

'Initialise variables
varSum = 0

'First sum the squares
For k = 1 To g_varSampleSizeUsed
    varSum = varSum + (g_arrCalculations(IndCriteria, k)) _
        * (g_arrCalculations(IndCriteria, k))
Next k

'Then determine the variance
fncVariance = varSum / (g_varSampleSizeUsed - 1)
End Function
'
Public Function fncTINV()
'This is a function that determines the value for t given
'the confidence level (%) and sample size (g_varSampleSizeUsed)

'Now get the t value for different sample sizes and
'confidence levels
Select Case g_varSampleSizeUsed
    Case 50
        fncTINV = g_arrTINVdata(1, g_varConfidenceLevel)
    Case 100
        fncTINV = g_arrTINVdata(2, g_varConfidenceLevel)
    Case 250
        fncTINV = g_arrTINVdata(3, g_varConfidenceLevel)
    Case 500
        fncTINV = g_arrTINVdata(4, g_varConfidenceLevel)
    Case 1000
        fncTINV = g_arrTINVdata(5, g_varConfidenceLevel)
    Case 2000
        fncTINV = g_arrTINVdata(6, g_varConfidenceLevel)
End Select

End Function
'
Public Function fncLowerBound(IndCriteria As Integer)
'This is a function that determines the lower bound on a confidence interval

fncLowerBound = fncAverage(IndCriteria) - fncTINV() * _
    Sqr(fncVariance(IndCriteria) / g_varSampleSizeUsed)

End Function
'
Public Function fncUpperBound(IndCriteria As Integer)
'This is a function that determines the upper bound on a confidence interval

```

```
fncUpperBound = fncAverage(IndCriteria) + fncTINV() * _
  Sqr(fncVariance(IndCriteria) / g_varSampleSizeUsed)
```

```
End Function
```

---

```
Public Function fncLargestGreater(IndCriteria As Integer)
```

```
'This is a function that finds the largest confidence level
'for which the lower bound is larger than 0
```

```
'Dim variables
```

```
Dim flag As Boolean
```

```
'Initialise variables
```

```
flag = True
```

```
g_varConfidenceLevel = 99
```

```
'Find largest confidence level
```

```
While g_varConfidenceLevel >= 50 And flag = True
```

```
  If fncLowerBound(IndCriteria) > 0 Then
```

```
    fncLargestGreater = g_varConfidenceLevel
```

```
    flag = False
```

```
  End If
```

```
  g_varConfidenceLevel = g_varConfidenceLevel - 1
```

```
Wend
```

```
'If nothing was found set equal to zero
```

```
If g_varConfidenceLevel = 49 Then
```

```
  fncLargestGreater = 0
```

```
End If
```

```
End Function
```

---

```
Public Function fncLargestSmaller(IndCriteria As Integer)
```

```
'This is a function that finds the largest confidence level
```

```
'for which the lower bound is smaller than 0
```

```
'Dim variables
```

```
Dim flag As Boolean
```

```
'Initialise variables
```

```
flag = True
```

```
g_varConfidenceLevel = 99
```

```
'Find largest confidence level
```

```
While g_varConfidenceLevel >= 50 And flag = True
```

```
  If fncUpperBound(IndCriteria) > 0 Then
```

```
    fncLargestSmaller = g_varConfidenceLevel
```

```
    flag = False
```

```
  End If
```

```
  g_varConfidenceLevel = g_varConfidenceLevel - 1
```

```
Wend
```

```
'If nothing was found set equal to zero
```

```
If g_varConfidenceLevel = 49 Then
```

```
  fncLargestSmaller = 0
```

```
End If
```

```
End Function
```

---

```
Public Sub subTechnique(IndCurrent As Integer, IndAlternative As Integer)
```

'This is the subprogram that determines whether a alternative must switch  
'with the current

'Dim variables  
Dim k As Integer

'Initialise variables  
g\_varAlternativesTotal = 0  
g\_varCurrentTotal = 0

'First ready the array for use  
subDifference IndCurrent, IndAlternative

'Now determine the sample size  
g\_varSampleSizeUsed = fncSampleSize

'Now decide what technique to use  
Select Case g\_varTechnique

'For technique 1

Case 1

'For the 5 performance criteria

For k = 1 To 5

'For maximisation

If g\_arrMinMax(k) = "MAX" Then

If fncLowerBound(k) > 0 Then

g\_varAlternativesTotal = g\_varAlternativesTotal \_  
+ g\_arrImportance(k)

Else

g\_varCurrentTotal = g\_varCurrentTotal + g\_arrImportance(k)

End If

End If

'For minimisation

If g\_arrMinMax(k) = "MIN" Then

If fncUpperBound(k) < 0 Then

g\_varAlternativesTotal = g\_varAlternativesTotal \_  
+ g\_arrImportance(k)

Else

g\_varCurrentTotal = g\_varCurrentTotal + g\_arrImportance(k)

End If

End If

Next k

'For technique 2

Case 2

'For the 5 performance criteria

For k = 1 To 5

'For maximisation

If g\_arrMinMax(k) = "MAX" Then

g\_varAlternativesTotal = g\_varAlternativesTotal \_  
+ fncLargestGreater(k) \* g\_arrImportance(k)

g\_varCurrentTotal = g\_varCurrentTotal \_  
+ fncLargestSmaller(k) \* g\_arrImportance(k)

End If

'For minimisation

If g\_arrMinMax(k) = "MIN" Then

g\_varAlternativesTotal = g\_varAlternativesTotal \_  
+ fncLargestSmaller(k) \* g\_arrImportance(k)

g\_varCurrentTotal = g\_varCurrentTotal \_  
+ fncLargestGreater(k) \* g\_arrImportance(k)

End If

Next k

End Select

```
'Determine whether it must switch
If (g_varAlternativesTotal > g_varCurrentTotal) Then
    g_varPolicyChange(fncPolicyNum(g_varPolicyEval)) = _
        g_varPolicyEval
    Else
        g_varPolicyChange(fncPolicyNum(g_varPolicyEval)) = ""
End If
```

End Sub

---

Public Sub subEndTime()

```
'This is an subprogram that determines the end time and run number to be shown on
'the form
'This is done at startup, when option buttons restart/continue are changed and
'when the runtime are changed
```

```
'Dim the variables
Dim varPreviousTime As Double
Dim varPreviousNum As Double
```

```
'Depending on whether the program are restarted or continued, it is determined
Select Case frmInitial.OptContinue
```

```
    Case True
        'The continue file need to be opened
        Open "Continue.txt" For Input As #19
        Input #19, varPreviousTime, varPreviousNum
        Close #19
```

```
    'If the emulator need to continue, the number is incremented
    frmInitial.txtNumber.Text = varPreviousNum + 1
    'and the runtime is added to the time already done
    frmInitial.txtEnd.Text = CDbl(frmInitial.txtRuntime.Text) + varPreviousTime
    'The filenames need to get their number beforehand
    frmInitial.txtOut1 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtOut1, 2)
    frmInitial.txtOut2 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtOut2, 2)
    frmInitial.txtOut3 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtOut3, 2)
    frmInitial.txtout4 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtout4, 2)
    frmInitial.txtOut5 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtOut5, 2)
    frmInitial.txtOut6 = CStr(varPreviousNum + 1) & Mid(frmInitial.txtOut6, 2)
```

```
    Case False
```

```
    'If the emulator need to restart, the number is changed to 1
    frmInitial.txtNumber.Text = 1
    'and the end to the runtime
    frmInitial.txtEnd.Text = frmInitial.txtRuntime.Text
    'the filename's prefix need to be changed back to 1
    frmInitial.txtOut1 = "1" & Mid(frmInitial.txtOut1, 2)
    frmInitial.txtOut2 = "1" & Mid(frmInitial.txtOut2, 2)
    frmInitial.txtOut3 = "1" & Mid(frmInitial.txtOut3, 2)
    frmInitial.txtout4 = "1" & Mid(frmInitial.txtout4, 2)
    frmInitial.txtOut5 = "1" & Mid(frmInitial.txtOut5, 2)
    frmInitial.txtOut6 = "1" & Mid(frmInitial.txtOut6, 2)
```

End Select

End Sub

### B.3 Summary of the subprograms and functions used in the Emulator

The functions and subprograms used in the Emulator, which were not discussed in the main document, are discussed in Table B. 1 to Figure B. 4. Table B. 1 gives the subprograms in the *ThisDocument* part of the real-world system model. Figure B. 2 gives the secondary subprograms (the main subprograms are shown in the main document) of the Visual Basic® form *frmInitial*. Figure B. 3 gives the functions of the Visual Basic® module, *Module 2* and Figure B. 4 the subprograms from the same Visual Basic® module.

Table B. 1 The subprograms in the *ThisDocument* part of the real-world system model

Subprogram	Domain	Usage
<i>ModelLogic_DocumentOpen</i>	Private	This subprogram is executed when the real-world system model is opened and the Emulator is started. It does the initialising and it gets the previous settings of the Emulator and displays them on the initial Visual Basic® form.
<i>ModelLogic_DocumentSave</i>	Private	This subprogram closes all the files once the simulation is finished.
<i>VBA_Block_1_Fire</i>	Private	This subprogram writes the individual values of the tallies to a file as the entities pass through <i>VBA</i> block 1.
<i>VBA_Block_2_Fire</i>	Private	This subprogram writes the individual values of the discrete change variables to a file as the entities pass through <i>VBA</i> block 2.
<i>VBA_Block_3_Fire</i>	Private	This subprogram writes the individual values of the discrete change variables to a file as the entities pass through <i>VBA</i> block 3.

Table B. 2 The secondary subprograms of the Visual Basic® form *fmbInitial*

Subprogram	Domain	Usage
<i>cmdEdit_Click</i>	Private	This subprogram allows the editing of the real-world system model.
<i>cmdEnd_Click</i>	Private	This subprogram allows the stopping of the Emulator during a run.
<i>cmdStart_Click</i>	Private	This subprogram starts the Emulator and calls the different subprograms that constitute the Emulator.
<i>cmdStop_Click</i>	Private	This subprogram closes the Emulator.
<i>OptMonitorNo_Click</i>	Private	This subprogram makes the textboxes in the monitor frame invisible once the invisible option button is activated.
<i>OptMonitorYes_Click</i>	Private	This subprogram makes the textboxes in the monitor frame visible once the visible option button is activated.
<i>OptRestart_Change</i>	Private	This subprogram changes the end time when it is required from the restart option button.
<i>OptTech1_Click</i>	Private	This subprogram enables the confidence level input text box for the First technique.
<i>optTech2_Click</i>	Private	This subprogram disables the confidence level input text box for the First technique.
<i>txtRuntime_Change</i>	Private	This subprogram changes the end time when the runtime is changed.

Table B. 3 The functions of the Visual Basic® module, *Module 2*

Function and type	Domain	Parameters and type	Usage
<i>fncIdenNumRW</i> As Double	Public	<i>varArenaVariable</i> As String	This function finds the SIMAN® identification number for a given variable in the real-world system model.
<i>fncIdenNumAL</i> As Double	Public	<i>varArenaVariable</i> As String	This function finds the SIMAN® identification number for a given variable in the alternative system model.
<i>fncFindModRW</i> As Module	Public	<i>varArenaMod</i> As String	This function finds modules in the real-world system model.
<i>fncFindModAL</i> As Module	Public	<i>varArenaMod</i> As String	This function finds modules in the alternative system model.
<i>fncSmallest</i> As Double	Public	<i>varArray()</i> As Double	This function determines the smallest value of five variables in an array.
<i>fncWhichAlter</i> As String	Public		This function determines which alternative control policy is evaluated at a specific breaktime.
<i>fncPolicyNum</i> As Double	Public	<i>name</i> As String	This function determines the number corresponding to the alternative system's name.

Function and type	Domain	Parameters and type	Usage
<i>fncTallyIDAL</i> As Double	Public	<i>name</i> As String	This function seeks the SIMAN® ID of a tally in the real-world system model.
<i>fncTallyIDRW</i> As Double	Public	<i>name</i> As String	This function seeks the SIMAN® ID of a tally in the alternative system model.
<i>fncAverage</i>	Public	<i>IndCriteria</i> As Integer	This function determines the average of a sample.
<i>fncSampleSize</i>	Public		This function determines the sample size used in the calculations.
<i>fncVariance</i>	Public	<i>IndCriteria</i> As Integer	This function determines the variance of a sample.
<i>fncTINV</i>	Public		This function determines the value from the t-distribution, corresponding to the given confidence level (%) and sample size.
<i>fncLowerBound</i>	Public	<i>IndCriteria</i> As Integer	This function determines the lower bound on a confidence interval.
<i>fncUpperBound</i>	Public	<i>IndCriteria</i> As Integer	This function determines the upper bound on a confidence interval.
<i>fncLargestGreater</i>	Public	<i>IndCriteria</i> As Integer	This function finds the largest confidence level for which the lower bound is larger than 0.

Function and type	Domain	Parameters and type	Usage
<i>fncLargestSmaller</i>	Public	<i>IndCriteria</i> As Integer	This function finds the largest confidence level for which the lower bound is smaller than 0.

Table B. 4 The subprograms in the Visual Basic® module, *Module 2*

Subprogram	Domain	Parameters and type	Usage
<i>subGetPrevious</i>	Public		This subprogram gets the previous values of the settings and displays it on the initial Visual Basic® form.
<i>subSaveNext</i>	Public		This subprogram writes the current settings to a file so that they are available when the Emulator is used again.
<i>subAddTrial</i>	Public	<i>IndAlternative</i> As Integer <i>varMean1</i> As Double <i>varMean2</i> As Double <i>varMean3</i> As Double <i>varMean4</i> As Double <i>varMean5</i> As Double <i>varTrialNumber</i> As Double	This subprogram adds a trial to the top of the array with the trial data.
<i>subDifference</i>	Public	<i>IndCurrent</i> As Integer <i>IndAlternative</i> As Integer	This subprogram calculates the difference between the array values of the alternative and the current system and writes them to a new array used for the calculations.
<i>subTechnique</i>	Public	<i>IndCurrent</i> As Integer <i>IndAlternative</i> As Integer	This subprogram determines whether the alternative control policy must be switched with the current control policy or not.

Subprogram	Domain	Parameters and type	Usage
<i>subEndTime</i>	Public		This subprogram determines the end time and run number to be shown on the form. This is done at start-up, when the restart/continue option buttons are changed and when the runtime is changed.

### B.4 Visual Basic® form

The Visual Basic® form, *fmbinitial*, used in the Emulator, is shown in Figure B. 11.

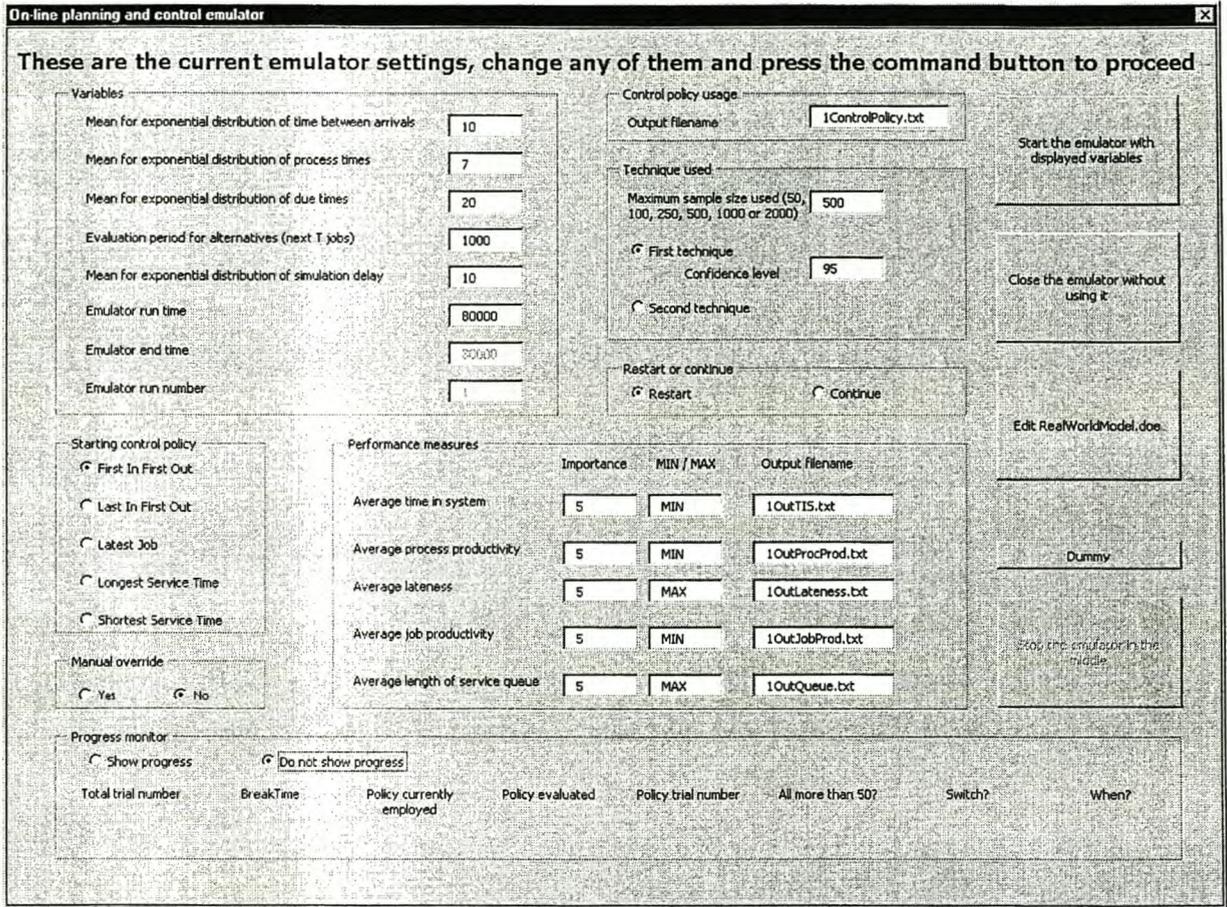


Figure B. 11 *Fmbinitial* of the Emulator

# **APPENDIX C: OPERATING THE EMULATOR**

## C OPERATING THE EMULATOR

Operating the Emulator used in this project (shown in Appendix B on page B-1) requires the following:

- a) A workstation with Windows NT<sup>®</sup> as operating system.
- b) Arena<sup>®</sup> version 3 installed on the workstation.
- c) A folder on the workstation containing the following: two Arena<sup>®</sup> models, *AlternativesModel.doe* and *ReakworldModel.doe*, and three text files, *TINVdata.txt*, *Continue.txt* and *Config.txt*.

The Arena<sup>®</sup> model, *ReakworldModel.doe*, contains the Visual Basic<sup>®</sup> code for the Emulator in addition to the model, while the other model, *AlternativesModel.doe*, is used by the Emulator to evaluate different alternatives.

The first text file, *TINVdata.txt*, is used as a look-up table to determine the inverse of the t-distribution. The inverse of the t-distribution is a very complex function, usually determined by iteration. The Emulator uses discrete values for the confidence level, so it was decided to use a look-up table rather than trying to program it.

The *Continue.txt* file is used to collect the end conditions of all the variables at the end of a run, and, if the Emulator is not restarting, to initialise the variables at the start of a new run to their settings of the previous run. This enables the Emulator to continue a run where it ended the previous one.

The final text file, *Config.txt*, is used to remember the settings of the Emulator from the previous run on the initial form, so that it is not necessary to enter it repeatedly.

The Emulator is started by opening the Arena<sup>®</sup> model, *ReakworldModel.doe*, with Arena<sup>®</sup> version 3. Sometimes one or both of the models may become damaged during operation, and then it needs to be replaced with a copy of that specific file before the next run can be executed. This does not influence the results of the run.

The results of a run are placed in the same folder as the Arena<sup>®</sup> models as text files with names as specified at the start of the Emulator on the initial form. When it is indicated in the initial form that the Emulator is continuing on the previous run, the number before the text file name is changed to the run number. This ensures that the output files do not overwrite each other and that the files can be strung together in the correct sequence when the data is analysed.

**APPENDIX D:**  
**VISUAL BASIC<sup>®</sup>**  
**PROGRAM “FILES”**

## D VISUAL BASIC<sup>®</sup> PROGRAM “FILES”

The Visual Basic<sup>®</sup> program “Files” is used to add the output files together so that they can be analysed as a single data file and to determine what percentage of time the Emulator spends under the different control policies. Firstly, the Visual Basic<sup>®</sup> code that comprises the program is shown. Then the Visual Basic<sup>®</sup> form, that is part of the program, is shown and in the final part the verification of the program is discussed.

### D.1 Visual Basic<sup>®</sup> code

```
Private Sub cmdAdd_Click()
'This subprogram adds the file to the end of the new file

'The new file and file to be added are opened
Open txtNewName.Text For Append As #1
Open txtAddName.Text For Input As #2

'The file is added row for row
While Not EOF(2)
  If OptSeven = True Then
    'For files with seven coloms
    Input #2, var1, var2, var3, var4, var5, var6, var7
    Write #1, var1, var2, var3, var4, var5, var6, var7
  Else
    'For files with two coloms
    Input #2, var1, var2
    Write #1, var1, var2
  End If
Wend

'Close the files
Close #1
Close #2

'Detail to prevent misuse
cmdAdd.Enabled = False
cmdAddFile.Enabled = True
txtAddName.Text = ""

End Sub
'
```

---

```
Private Sub cmdAddFile_Click()
'This subproram displays the filename to be added

'First determine whether a \ is needed
If Right(dirlist.Path, 1) <> "\" Then
  DummyName = ""
Else
  DummyName = ""
End If

'If a file is highlighted display the full name
If filList.FileName <> "" Then
  txtAddName.Text = dirlist.Path & DummyName & filList.FileName
  cmdAdd.Enabled = True
  cmdAddFile.Enabled = False
End If
End Sub
'
```

```

Else
    'Error message
    MsgBox "Please choose the name of the file to be added", , "Error"
End If

End Sub
'


---


Private Sub cmdCalcFile_Click()
'This subproram displays the filename to be calculated

'First determine whether a \ is needed
If Right(dirlist.Path, 1) <> "\" Then
    DummyName = "\"
Else
    DummyName = ""
End If

'If a file is highlighted display the full name
If filList.FileName <> "" Then
    txtCalcName.Text = dirlist.Path & DummyName & filList.FileName
    cmdCalculate.Enabled = True
Else
    'Error message
    MsgBox "Please choose the name of the file to be calculated", , "Error"
End If

End Sub
'


---


Private Sub cmdCalculate_Click()
'This subprogram determines what percentage of time each control policy
'was used as well as the average number of breaks between switches

'Open the file
Open txtCalcName.Text For Input As #3

'Initialise the variables
varTotalcount = 0
varCount1 = 0
varCount2 = 0
varCount3 = 0
varCount4 = 0
varCount5 = 0
varCounter = 1
varNumberCounter = 0
varAverage = 0

'Input the first set of variables
Input #3, varPrevious1, varPrevious2, varPrevious3, varPrevious4, varPrevious5 _
, varPrevious6, varPrevious7

'Do the calculation for every row
While Not EOF(3)
    Input #3, varCurrent1, varCurrent2, varCurrent3, varCurrent4, varCurrent5 _
, varCurrent6, varCurrent7
    'Depending on the current control policy, count the time spend on it
    Select Case varCurrent3
        Case 1
            varCount1 = varCount1 + varCurrent2 - varPrevious2
        Case 2
            varCount2 = varCount2 + varCurrent2 - varPrevious2
    End Select
    varCounter = varCounter + 1
    varPrevious1 = varPrevious2
    varPrevious2 = varPrevious3
    varPrevious3 = varPrevious4
    varPrevious4 = varPrevious5
    varPrevious5 = varPrevious6
    varPrevious6 = varPrevious7
    varPrevious7 = varCurrent1
    varPrevious8 = varCurrent2
    varPrevious9 = varCurrent3
    varPrevious10 = varCurrent4
    varPrevious11 = varCurrent5
    varPrevious12 = varCurrent6
    varPrevious13 = varCurrent7
End While

'Calculate the average number of breaks between switches
varAverage = varNumberCounter / varTotalcount

'Close the file
Close #3
End Sub

```

```

Case 3
varCount3 = varCount3 + varCurrent2 - varPrevious2
Case 4
varCount4 = varCount4 + varCurrent2 - varPrevious2
Case 5
varCount5 = varCount5 + varCurrent2 - varPrevious2
End Select

If varCurrent3 = varPrevious3 Then
    'Count the number of breaks that use the same control policy
    varCounter = varCounter + 1
Else
    'Count the number of switches
    varNumberCounter = varNumberCounter + 1
    'Determine the average
    varAverage = (varAverage * (varNumberCounter - 1) + varCounter) / _
        varNumberCounter
    'Initialise the counter
    varCounter = 1
End If

'Initialise the "Previous" variables
varPrevious1 = varCurrent1
varPrevious2 = varCurrent2
varPrevious3 = varCurrent3
varPrevious4 = varCurrent4
varPrevious5 = varCurrent5
varPrevious6 = varCurrent6
varPrevious7 = varCurrent7

Wend

'Close the file
Close #3

'Determine the total time
varTotalcount = varCount1 + varCount2 + varCount3 + varCount4 + varCount5

'Display the different percentage usages
txtUse(1) = Format(varCount1 / varTotalcount * 100, "##.##")
txtUse(2) = Format(varCount2 / varTotalcount * 100, "##.##")
txtUse(3) = Format(varCount3 / varTotalcount * 100, "##.##")
txtUse(4) = Format(varCount4 / varTotalcount * 100, "##.##")
txtUse(5) = Format(varCount5 / varTotalcount * 100, "##.##")

'Do the final calculation of average
varNumberCounter = varNumberCounter + 1
varAverage = (varAverage * (varNumberCounter - 1) + varCounter) / _
    varNumberCounter

'Display the average number of breaks between switches
txtSwitch = Format(varAverage, "##.##")

End Sub
'


---


Private Sub cmdNewFile_Click()
'This subprogram displays the name of the file to which the other files
' will be added

'Determine whether the \ is needed

```

```
If Right(dirlist.Path, 1) <> "\" Then
    DummyName = "\"
    Else
    DummyName = ""
End If

'If a name is specified display full path
If txtNewFile.Text <> "" Then
    txtNewName.Text = dirlist.Path & DummyName & txtNewFile.Text
    'The command button is disabled
    cmdNewFile.Enabled = False
    'The file is cleaned
    Open txtNewName.Text For Output As #1
    Close #1
Else
    'Error message
    MsgBox "Please fill in a new file name and remember the .txt", , "Error"
End If

End Sub
'
```

---

```
Private Sub dirList_change()
'This subprogram changes the directory
    fillList.Path = dirlist.Path
End Sub
'
```

---

```
Private Sub drvList_Change()
'This subprogram changes the drive
    dirlist.Path = drvList.Drive
End Sub
```

## D.2 Visual Basic® form

The Visual Basic® form used by the program is shown in Figure D. 1.

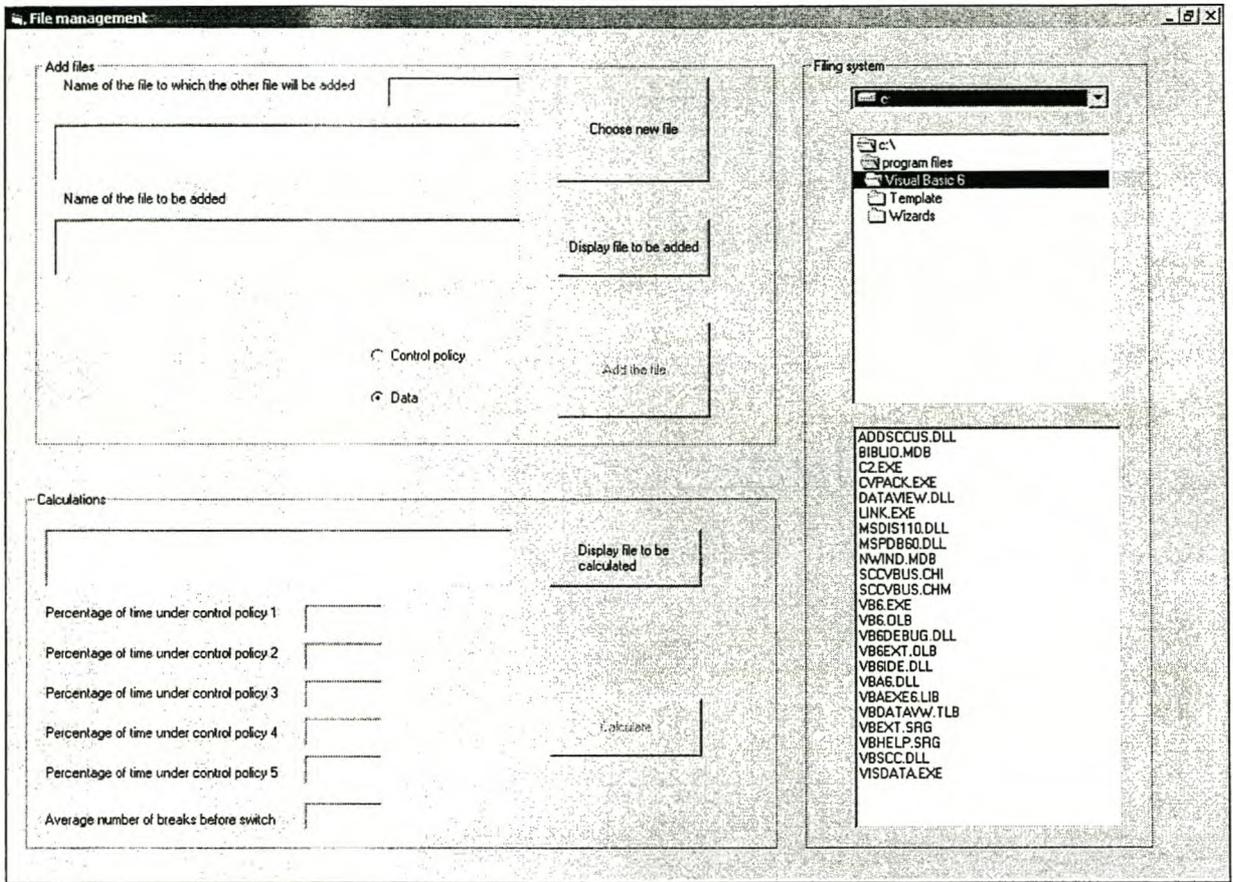


Figure D. 1 The Visual Basic® form used by Visual Basic® program "Files"

### D.3 Verification

To verify the Visual Basic™ program “Files”, the data files shown in Figure D. 2, Figure D. 3 and Figure D. 4 are combined by the program to the data file shown in Figure D. 4. This can be checked by inspection to ensure that the program operates correctly.

#### D.3.1 Adding files



```

ControlPolicyOne - Notepad
File Edit Format Help
2595,5028.2081095331,1,"Alternative 1",521,#TRUE#,#FALSE#
2596,5031.02237675163,1,"Alternative 4",530,#TRUE#,#FALSE#
2597,5031.55011164325,1,"Alternative 1",522,#TRUE#,#FALSE#
2598,5038.64718613179,1,"Alternative 4",531,#TRUE#,#FALSE#
2599,5041.54675367617,1,"Alternative 5",513,#TRUE#,#FALSE#
2600,5042.76018989414,1,"Alternative 5",514,#TRUE#,#FALSE#
2601,5043.87771256851,1,"Alternative 3",501,#TRUE#,#FALSE#
2602,5045.86427504705,1,"Alternative 2",534,#TRUE#,#TRUE#
2603,5047.63196137917,1,"Alternative 1",523,#TRUE#,#FALSE#
    
```

Figure D. 2 Data for *ControlPolicyOne*



```
ControlPolicyTwo - Notepad
File Edit Format Help
2604,5050.95457721613,1,"Alternative 3",502,#TRUE#,#FALSE#
2605,5051.29293488565,2,"Alternative 2",535,#TRUE#,#FALSE#
2606,5051.47623547306,2,"Alternative 2",536,#TRUE#,#FALSE#
2607,5052.66610661464,2,"Alternative 5",515,#TRUE#,#TRUE#
2608,5053.34754006818,2,"Alternative 4",532,#TRUE#,#TRUE#
2609,5054.0564241557,5,"Alternative 5",516,#TRUE#,#FALSE#
2610,5055.05707685464,5,"Alternative 5",517,#TRUE#,#FALSE#
2611,5055.88519554769,5,"Alternative 2",537,#TRUE#,#FALSE#
```

Figure D. 3 Data for *ControlPolicyTwo*



```
ControlPolicyThree - Notepad
File Edit Format Help
2612,5056.89636180744,5,"Alternative 1",524,#TRUE#,#TRUE#
2613,5057.70379802732,5,"Alternative 5",518,#TRUE#,#FALSE#
2614,5060.12996667991,5,"Alternative 3",503,#TRUE#,#TRUE#
2615,5060.7387853573,5,"Alternative 5",519,#TRUE#,#FALSE#
2616,5061.69078739092,5,"Alternative 5",520,#TRUE#,#FALSE#
```

Figure D. 4 Data for *ControlPolicyThree*

```

ControlPolicyCombined - Notepad
File Edit Format Help
2595, 5028.2081095331, 1, "Alternative 1", 521, #TRUE#, #FALSE#
2596, 5031.02237675163, 1, "Alternative 4", 530, #TRUE#, #FALSE#
2597, 5031.55011164325, 1, "Alternative 1", 522, #TRUE#, #FALSE#
2598, 5038.64718613179, 1, "Alternative 4", 531, #TRUE#, #FALSE#
2599, 5041.54675367617, 1, "Alternative 5", 513, #TRUE#, #FALSE#
2600, 5042.76018989414, 1, "Alternative 5", 514, #TRUE#, #FALSE#
2601, 5043.87771256851, 1, "Alternative 3", 501, #TRUE#, #FALSE#
2602, 5045.86427504705, 1, "Alternative 2", 534, #TRUE#, #TRUE#
2603, 5047.63196137917, 1, "Alternative 1", 523, #TRUE#, #FALSE#
2604, 5050.95457721613, 1, "Alternative 3", 502, #TRUE#, #FALSE#
2605, 5051.29293488565, 2, "Alternative 2", 535, #TRUE#, #FALSE#
2606, 5051.47623547306, 2, "Alternative 2", 536, #TRUE#, #FALSE#
2607, 5052.66610661464, 2, "Alternative 5", 515, #TRUE#, #TRUE#
2608, 5053.34754006818, 2, "Alternative 4", 532, #TRUE#, #TRUE#
2609, 5054.0564241557, 5, "Alternative 5", 516, #TRUE#, #FALSE#
2610, 5055.05707685464, 5, "Alternative 5", 517, #TRUE#, #FALSE#
2611, 5055.88519554769, 5, "Alternative 2", 537, #TRUE#, #FALSE#
2612, 5056.89636180744, 5, "Alternative 1", 524, #TRUE#, #TRUE#
2613, 5057.70379802732, 5, "Alternative 5", 518, #TRUE#, #FALSE#
2614, 5060.12996667991, 5, "Alternative 3", 503, #TRUE#, #TRUE#
2615, 5060.7387853573, 5, "Alternative 5", 519, #TRUE#, #FALSE#
2616, 5061.69078739092, 5, "Alternative 5", 520, #TRUE#, #FALSE#
    
```

Figure D. 5 Data for *ControlPolicyCombined*

### D.3.2 Calculations on files

The calculations for the data of text file *ControlPolicyCombined* shown in Figure D. 5 can be calculated easily, as shown in Table D. 1 and Table D. 2.

Table D. 1 Calculated values for percentage of time spend under control policy

	Time spend under control policy (minutes)	Percentage of total time spend under control policy (%)
Control policy 1	22.74	67.93
Control policy 2	2.39	7.14
Control policy 3	0	0
Control policy 4	0	0
Control policy 5	8.34	24.91
Total	33.48	

Table D. 2 Calculated values for average number of breaks between switches

	Number of breaks	Average number of breaks between switches
Control policy 1	10	
Control policy 2	4	
Control policy 5	8	
Total	22	7.33

These are the same as the results from the program shown in Figure D. 6.

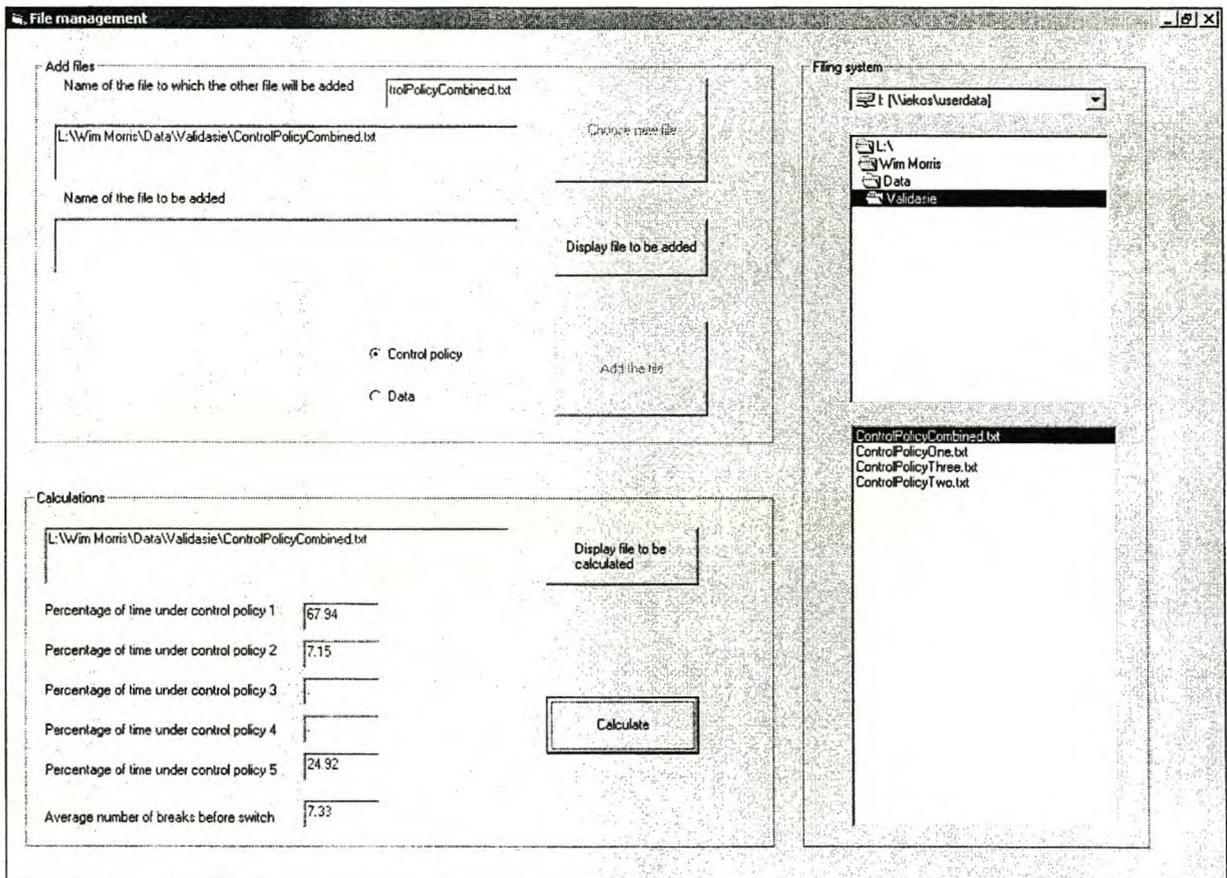


Figure D. 6 Results of calculations on *ControlPolicyCombined*

**APPENDIX E:**  
**THE EVOLUTION OF**  
**THE EMULATOR**

## **E THE EVOLUTION OF THE EMULATOR**

The development of the Emulator did not proceed effortlessly. The main programming setback was that it proved a lot more difficult than was originally thought to determine the end conditions of the real-world system models and then to initialise the alternative system models to that state. However, when at last the initialising was accomplished and the runs were started to collect the data, the Emulator proved to be very slow and prone to crashes. This necessitated three different iterations of the Emulator that will now be described.

### **E.1 Visual Basic<sup>®</sup>, Arena<sup>®</sup> and Microsoft<sup>®</sup> Excel<sup>®</sup>**

The initial version of the Emulator was a standalone Visual Basic<sup>®</sup> shell that interacted with Arena<sup>®</sup> to simulate the models and to execute the developed techniques with Microsoft<sup>®</sup> Excel<sup>®</sup>. It had three major problems. The first was that it was much too slow to be viable, the second was that it crashed before enough data could be gathered to make statistical comparisons of the techniques and the third was that the Excel<sup>®</sup> spreadsheets used to implement the techniques were very large.

Both the Visual Basic<sup>®</sup> code and the Excel<sup>®</sup> spreadsheets executed fast, but the main delay was when switches between the different applications occurred and there was a waiting period before the next application started.

It was not possible to determine why this version crashed when it was run for a long time, but it was noted that different operating systems resulted in different periods before it crashed, giving rise to speculation that the application integration was to blame.

The Excel<sup>®</sup> spreadsheets used to implement the techniques were both larger than 10 MB. This created a problem with both storage and virtual memory.

### **E.2 Visual Basic<sup>®</sup> and Arena<sup>®</sup>**

The next version attempted to eliminate one of the interactions with the applications. To achieve this, the techniques that were implemented in Excel<sup>®</sup> were programmed directly into Visual Basic<sup>®</sup>. This resulted in much better usage of memory and more elegant programming, but it is harder to understand the way the techniques operate from the code than from the Excel<sup>®</sup> spreadsheets. The result was a faster program that crashed less, but it was still not a viable option to evaluate the techniques.

### **E.3 Arena<sup>®</sup>'s Visual Basic<sup>®</sup> and Arena<sup>®</sup>**

For the version used in this project, the Visual Basic<sup>®</sup> code was programmed directly into the Arena<sup>®</sup> model's Visual Basic<sup>®</sup>. This resulted in significantly faster operation, and while it was not possible to eliminate the crashing of the program, the program ran for long periods before it crashed. However, to ensure that no data was lost when the Emulator crashed, the Emulator was changed to run for a certain runtime shorter than the period when it crashed. The Emulator was then restarted, but initialised to the end conditions of the previous run. This enables the output files from the runs following upon each other to be merged together (with the program given in Appendix D) to form continuous output files. This enabled the gathering of data of the required period without the Emulator crashing.

## F OUTPUT RESULTS

The detailed output results of the individual control policies, as well as the results for the two techniques, are shown in this appendix. First the results for a traffic intensity of 0.7 are shown, and then for a traffic intensity of 0.9. For both the traffic intensities, classical confidence interval information is given for all the pilot runs of the individual control policies, for the necessary production runs of the individual control policies, and for the two different techniques.

### F.1 Output results for a traffic density of 0.7

#### F.1.1 Classical C.I. intervals summary for *First in first out* Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
FifoPilotTIS	22.8	4.42	0.803	15.5	41.2	119
FifoPilotProc	0.692	0.064 8	0.006 45	0.517	0.975	391
FifoPilotLate	12.1	3.33	0.627	6.43	21.6	111
FifoPilotJp	0.524	0.051 6	0.007 63	0.33	0.642	178
FifoPilotQueue	1.57	0.493	0.086 3	0.781	3.64	128

*F.1.2 Classical C.I. intervals summary for Last in first out Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LifoPilotTIS	22.7	5.62	0.831	13.5	48.5	178
LifoPilotProc	0.692	0.054 4	0.006 19	0.511	0.827	299
LifoPilotLate	13.6	3.69	0.695	7.57	27.4	111
LifoPilotJp	0.608	0.043 9	0.005	0.43	0.727	298
LifoPilotQueue	1.56	0.382	0.080 5	0.765	3.15	89

*F.1.3 Classical C.I. intervals summary for Latest job Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LjPilotTIS	22.7	4.55	0.798	15.4	43.5	127
LjPilotProc	0.692	0.054 4	0.006 19	0.511	0.827	299
LjPilotLate	11.2	3.25	0.648	6.39	23.9	99
LjPilotJp	0.54	0.055 7	0.007 35	0.352	0.661	223
LjPilotQueue	1.57	0.431	0.080 6	0.84	2.77	112

F.1.4 Classical C.I. intervals summary for Longest service time Pilot run Stellenbosch University <http://scholar.sun.ac.za>

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LstPilotTIS	35.8	9.25	1.95	18.8	65.6	89
LstPilotProc	0.692	0.061 8	0.006 42	0.484	0.858	359
LstPilotLate	25	8.69	1.92	11.8	58.3	81
LstPilotJP	0.558	0.054 7	0.006 24	0.339	0.699	298
LstPilotQueue	2.86	0.951	0.21	1.25	6.7	81

F.1.5 Classical C.I. intervals summary for Longest service time Production run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LstProdTIS	37.3	9.82	0.978	18.8	83.4	389
LstProdProc	0.698	0.060 1	0.002 99	0.484	0.919	1 559
LstProdLate	26.4	9.22	0.964	11.1	63.8	354
LstProdJp	0.552	0.052 4	0.002 86	0.339	0.7	1 298
LstProdQueue	3.03	0.986	0.103	1.2	6.97	354

F.1.6 Classical C.I. intervals summary for Shortest service time Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
SstPilotTIS	15.9	2.31	0.341	11.5	23.7	178
SstPilotProc	0.692	0.054 4	0.006 19	0.511	0.827	299
SstPilotLate	7.39	1.88	0.279	3.97	14.1	178
SstPilotJP	0.596	0.035 6	0.005 26	0.462	0.68	178
SstPilotQueue	0.887	0.223	0.032 9	0.396	2	179

F.1.7 Classical C.I. intervals summary for First technique Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
FirstPilotTIS	22.9	4.95	0.849	15	48	133
FirstPilotProc	0.693	0.060 5	0.005 94	0.485	0.861	401
FirstPilotLate	13.1	4.45	0.766	6.93	36.5	132
FirstPilotJP	0.564	0.048 1	0.005 61	0.377	0.679	285
FirstPilotQueue	1.58	0.517	0.088 7	0.851	4.09	133

*F.1.8 Classical C.I. intervals summary for Second technique Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
SecondPilotTIS	22.9	4.48	0.846	14.9	41.9	110
SecondPilotProc	0.693	0.060 5	0.005 94	0.485	0.861	401
SecondPilotLate	13.1	3.59	0.716	6.98	26.8	99
SecondPilotJP	0.564	0.046 3	0.005 77	0.374	0.672	249
SecondPilotQueue	1.58	0.517	0.088 7	0.79	4.09	133

**F.2 Output results for a traffic density of 0.9**

*F.2.1 Classical C.I. intervals summary for First in first out Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
FIFOPilotTIS	83.1	16.5	5.84	51.2	119	33
FIFOPilotProcProd	0.893	0.037 6	0.007 07	0.779	0.976	111
FIFOPilotLateness	66.7	17.8	6.2	39.6	105	34
FIFOPilotJobProd	0.268	0.034 5	0.011 2	0.196	0.355	39
FIFOPilotQueue	7.38	1.9	0.663	4.39	11.2	34

*F.2.2 Classical C.I. intervals summary for Last in first out Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LIFOPilotTIS	82.8	26.6	5.67	42.9	182	87
LIFOPilotProcProd	0.893	0.035 6	0.006 82	0.81	0.976	107
LIFOPilotLateness	70.9	30.9	6.35	31.9	189	93
LIFOPilotJobProd	0.472	0.032 4	0.004 85	0.398	0.55	174
LIFOPilotQueue	7.35	1.82	0.646	4.34	11.4	33

*F.2.3 Classical C.I. intervals summary for Latest job Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LJPilotTIS	82.9	18.3	6.39	53.8	122	34
LJPilotProcProd	0.893	0.035 6	0.006 82	0.81	0.976	107
LJPilotLateness	66	17.8	6.22	38.3	104	34
LJPilotJobProd	0.28	0.032 6	0.010 7	0.206	0.353	38
LJPilotQueue	7.41	2.03	0.707	4.55	11.9	34

F.2.4 Classical C.I. intervals summary for Longest service time Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LSTPilotTIS	237	73.1	24.4	101	400	37
LSTPilotProcProd	0.893	0.025 1	0.005 69	0.826	0.942	77
LSTPilotLateness	222	82.1	27.8	109	483	36
LSTPilotJobProd	0.38	0.015 7	0.008 08	0.351	0.408	17
LSTPilotQueue	22.7	7.28	2.67	12.9	37.9	31

F.2.5 Classical C.I. intervals summary for Longest service time Production run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
LSTProdTIS	260	125	15.1	83.1	1.55e+003	267
LSTProdProcProd	0.898	0.027 2	0.002 28	0.826	0.991	549
LSTProdLateness	245	128	15.6	85.3	1.55e+003	260
LSTProdJobProd	0.375	0.013 8	0.002 46	0.341	0.412	123
LSTProdQueue	25	8.3	1.66	13.6	76	98

F.2.6 Classical C.I. intervals summary for Shortest service time Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
SSTPilotTIS	36.1	6.73	1.43	24.3	58.3	87
SSTPilotProcProd	0.893	0.029 3	0.006 44	0.819	0.952	82
SSTPilotLateness	25	6.01	1.32	14.5	41.4	82
SSTPilotJobProd	0.444	0.030 2	0.005 55	0.368	0.508	116
SSTPilotQueue	2.71	0.478	0.167	2.01	3.83	34

F.2.7 Classical C.I. intervals summary for First technique Pilot run

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
FirstPiloTIS	84	18.7	6.22	52.7	134	37
FirstPilotProcProd	0.894	0.029 9	0.006 53	0.831	0.949	83
FirstPilotLateness	70.3	14.8	5	49.2	115	36
FirstPilotJobProd	0.388	0.036	0.006 9	0.305	0.474	107
FirstPilotQueue	7.43	5.43	0.551	1.32	36.6	376

*F.2.8 Classical C.I. intervals summary for Second technique Pilot run*

IDENTIFIER	AVERAGE	STANDARD DEVIATION	0.950 C.I. HALF-WIDTH	MINIMUM VALUE	MAXIMUM VALUE	NUMBER OF OBS.
SecondPilotTIS	83.5	16	5.4	55.5	128	36
SecondPilotProcProd	0.893	0.027 4	0.005 76	0.823	0.958	89
SecondPilotLateness	70	15.7	5.32	42.3	114	36
SecondPilotJobProd	0.389	0.040 2	0.006 77	0.286	0.488	138
SecondPilotQueue	7.42	1.58	0.518	4.44	12.4	38