



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

**THE EFFECT OF TIME-ADVANCE MECHANISM
IN MODELING AND SIMULATION**

by

Ahmed Ali Alrowaie

September 2011

Dissertation Supervisor:

Arnold H. Buss

**This dissertation was performed at the MOVES Institute
Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2011	3. REPORT TYPE AND DATES COVERED Dissertation	
4. TITLE AND SUBTITLE: The Effect of Time-Advance Mechanism in Modeling And Simulation			5. FUNDING NUMBERS	
6. AUTHOR(S) Ahmed Ali Alrowaie				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) As the discipline of Modeling and Simulation (M&S) becomes more complex, modelers are faced with mounting challenges to design and analyze simulations that effectively address difficult problems across military, industrial, and societal fields. Understanding the effects of time-advance mechanisms (TAMs) is essential to making advances in the design and use of M&S across a wide variety of domains. We perform a series of empirical studies to characterize and compare the influence of discrete event simulation (DES) and discrete time simulation (DTS) approaches, and describe the effects of changes in time "step" sizes across a number of vital simulation areas including queuing systems, combat systems, and human behavior representations of military significance. Our results illustrate that the choice of TAM can have a significant impact on the behavior of models, the output obtained from simulation tools, and the recommendations that are likely to result. We describe inconsistencies and the emergence of unintended behaviors resulting from the use of different TAM approaches and DTS time "steps." We conclude that the DES approach is more likely to produce trustworthy simulation results for decision-making applications, and that the time step approach carries additional inherent risks that are often invisible to modelers of complex systems.				
14. SUBJECT TERMS Time-advance mechanism, discrete event simulation, discrete time simulation, time-step, simulation time representation, event-driven, time-driven, fixed increment time.			15. NUMBER OF PAGES 318	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**THE EFFECT OF TIME-ADVANCE MECHANISM
IN MODELING AND SIMULATION**

Ahmed Ali Alrowaie
Major, Royal Bahraini Air Force
B.S., Cranfield University, 1998
M.S., Naval Postgraduate School, 2005

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN MODELING, VIRTUAL ENVIRONMENTS,
AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2011**

Author:

Ahmed Ali Alrowaie

Approved by:

Arnold H. Buss
Research Associate Professor
of MOVES
Dissertation Supervisor

Rudolph P. Darken
Professor of Computer Science

Christian J. Darken
Associate Professor of Computer
Science

Paul Sanchez, Ph.D.
Senior Lecturer of Operation
Research

Arijit Das
Research Associate of Computer
Science

Approved by:

Mathias Kölsch, Chair, MOVES Academic Committee

Approved by:

Peter J. Denning, Chair, Department of Computer Science

Approved by:

Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

As the discipline of Modeling and Simulation (M&S) becomes more complex, modelers are faced with mounting challenges to design and analyze simulations that effectively address difficult problems across military, industrial, and societal fields. Understanding the effects of time-advance mechanisms (TAMs) is essential to making advances in the design and use of M&S across a wide variety of domains. We perform a series of empirical studies to characterize and compare the influence of discrete event simulation (DES) and discrete time simulation (DTS) approaches, and describe the effects of changes in time “step” sizes across a number of vital simulation areas including queuing systems, combat systems, and human behavior representations of military significance. Our results illustrate that the choice of TAM can have a significant impact on the behavior of models, the output obtained from simulation tools, and the recommendations that are likely to result. We describe inconsistencies and the emergence of unintended behaviors resulting from the use of different TAM approaches and DTS time “steps.” We conclude that the DES approach is more likely to produce trustworthy simulation results for decision-making applications, and that the time step approach carries additional inherent risks that are often invisible to modelers of complex systems.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational errors, they cannot be considered validated. All additional use of these programs without additional verification is at the risk of the user.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	BACKGROUND AND MOTIVATION	1
1.	Time-Advance Mechanisms in Modeling and Simulation	1
2.	Problems and Challenges of Time-Advance Mechanism	6
B.	RESEARCH OBJECTIVES.....	12
1.	Research Questions.....	12
2.	Research Scope.....	14
C.	GENERAL METHODOLOGY AND RESULTS.....	15
1.	Methodology	15
2.	Research Results	17
D.	SIGNIFICANCE AND CONTRIBUTIONS	20
E.	OVERVIEW OF THE DISSERTATION.....	21
II.	LITERATURE REVIEW	23
A.	RELATED WORK	23
1.	Time Advance Mechanism in Simulation	23
a.	Discrete Time Approach	27
b.	Discrete Event Approach	32
2.	Military Modeling and Simulation	37
3.	Related Comparison Studies.....	39
III.	CASE I: QUEUEING SYSTEMS AND THEIR APPLICATIONS.....	45
A.	INTRODUCTION.....	45
1.	Background	45
B.	LITERATURE REVIEW	47
1.	Queueing Systems: Analytical Algorithms	47
2.	Queueing Simulation Models.....	53
3.	Toll Plaza Systems.....	56
C.	MULTI-SERVER QUEUEING MODELS: M/M/k.....	59
1.	Simple M/M/k Queueing Systems.....	59
a.	Model Description.....	59
b.	Results and Comparative Analysis	63
2.	Multi-server Queueing Model with Balking: M/M/K/C.....	72
a.	Model Description.....	72
b.	Results and Comparative Analysis	75
D.	QUEUEING APPLICATION: TOLL PLAZA SYSTEMS	77
1.	Model Description.....	77
2.	Results and Comparative Analysis.....	81
3.	Summary and Discussions.....	85
IV.	CASE II: COMBAT SYSTEMS SIMULATION	89
A.	INTRODUCTION.....	89
1.	Background and Literature Review.....	90

	a.	<i>Combat Simulation Time Advance Mechanisms.....</i>	<i>91</i>
2.		Combat Simulation Software Packages.....	97
	a.	<i>MANA 4.....</i>	<i>98</i>
	b.	<i>MANA V.....</i>	<i>101</i>
	c.	<i>DAFS and Simkit.....</i>	<i>102</i>
3.		Combat Modeling Elements.....	105
	a.	<i>Entities/Squad units/Platforms.....</i>	<i>105</i>
	b.	<i>Movement.....</i>	<i>107</i>
	c.	<i>Sensors.....</i>	<i>111</i>
	d.	<i>Engagement and Weapons.....</i>	<i>116</i>
	e.	<i>Communications.....</i>	<i>119</i>
4.		Chapter Organization.....	121
B.		SECTION I: AGENT MOVEMENTS.....	122
	1.	Scenario 1: Agent Moving to A Single Waypoint.....	123
		a. <i>Simulation Methods.....</i>	<i>123</i>
		b. <i>Results.....</i>	<i>124</i>
		c. <i>Discussion.....</i>	<i>127</i>
	2.	Scenario 2: Agent Moving to Multiple Waypoints.....	128
		a. <i>Simulation Methods.....</i>	<i>128</i>
		b. <i>Results.....</i>	<i>130</i>
		c. <i>Discussion.....</i>	<i>134</i>
	3.	Scenario 3: Simple Agent Engagements.....	135
		a. <i>Simulation Methods.....</i>	<i>135</i>
		b. <i>Results.....</i>	<i>137</i>
		c. <i>Discussion.....</i>	<i>138</i>
C.		SECTION II: SENSORS AND DETECTION.....	139
	1.	Scenario 4: Changes to Sensor Ranges.....	140
		a. <i>Simulation Methods.....</i>	<i>140</i>
		b. <i>Results.....</i>	<i>142</i>
		c. <i>Discussion.....</i>	<i>143</i>
	2.	Scenario 5: Sensor Type Changes.....	144
		a. <i>Simulation Methods.....</i>	<i>144</i>
		b. <i>Results.....</i>	<i>148</i>
		c. <i>Discussion.....</i>	<i>151</i>
D.		SECTION III: EMERGENT OUTCOMES.....	153
	1.	Scenario 6: Skipping Phenomenon.....	154
		a. <i>Simulation Methods.....</i>	<i>154</i>
		b. <i>Results.....</i>	<i>156</i>
		c. <i>Discussion.....</i>	<i>158</i>
	2.	Scenario 7: Event Scheduling and Ordering Phenomenon.....	159
		a. <i>Simulation Methods.....</i>	<i>159</i>
		b. <i>Results.....</i>	<i>161</i>
		c. <i>Discussion.....</i>	<i>162</i>
E.		SECTION IV: PUTTING IT ALL TOGETHER.....	164
	1.	Littoral Combat Ship (LCS) Battle.....	164

a.	<i>Simulation Methods</i>	164
b.	<i>Results</i>	170
c.	<i>Discussion</i>	175
F.	SUMMARY AND CONCLUSIONS	178
V.	CASE III: HUMAN BEHAVIOR REPRESENTATION	183
A.	INTRODUCTION	183
B.	METHODOLOGY AND LITERATURE REVIEW	185
1.	Methodology	185
2.	Comparison Studies in the Literature	186
C.	SCENARIO DESCRIPTIONS	189
1.	TAM Effects in Pythagoras Simulation Environment	189
a.	<i>Simulation Method</i>	189
b.	<i>Results and Discussions</i>	192
2.	TAM Effects in Peace Support Operation Model (PSOM) Environment	199
a.	<i>Simulation Method</i>	199
b.	<i>Results and Discussions</i>	201
3.	TAM Effects in Cultural Geography (CG) Model Environment	211
a.	<i>Simulation Method</i>	212
b.	<i>Results and Discussions</i>	214
4.	TAM Effects on Agent Behavior in Continuous Systems	222
D.	CONCLUSIONS	236
VI.	CONCLUSION, IMPLICATIONS AND FUTURE WORK	239
A.	CONCLUSION	239
B.	RESEARCH IMPLICATIONS AND FUTURE WORK	244
	APPENDIX A. DEFINITIONS AND TERMINOLOGY	249
	APPENDIX B. QUEUEING SYSTEM COMPUTER CODES	263
A.	M/M/K & M/M/K/C DES PROGRAM IN SIMKIT	263
B.	M/M/K & M/M/K/C DTS PROGRAM IN SIMKIT	266
C.	TOLL PLAZA DES PROGRAM IN SIMKIT	269
	APPENDIX C. CASE II-SCENARIO 8 DESIGN POINTS DATA	275
	LIST OF REFERENCES	277
	INITIAL DISTRIBUTION LIST	291

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	a) Time-stepped execution and b) discrete-event execution of state variables over simulation time	4
Figure 2.	(a) Time-step (time-driven) approach, (b) next-event (event-driven) approach (after Bank et al., 2005).....	5
Figure 3.	Steps in simulation study with modified step 6 (after Law & Kelton, 2000) ..	10
Figure 4.	Illustration of simulation model classifications and study area (best viewed in color).....	12
Figure 5.	Stepwise execution of DTS (after Zeigler, Praehofer, & Kim, 2000)	29
Figure 6.	An illustration of fixed-increment time advance (after Law, & Kelton, 2000)	29
Figure 7.	Preferred illustration of fixed-increment time advance from the simulation view.....	30
Figure 8.	Fixed-increment time advance approach flow chart.....	31
Figure 9.	Illustration of DES (next-event) time-advance approach (after Law & Kelton, 2000)	34
Figure 10.	Next-event time-advance approach flow chart (after Buss, 2011a).....	35
Figure 11.	M/M/k Queueing Systems diagram (from Odoni, 2004).....	48
Figure 12.	Typical concept design for Toll Plaza System (from Corwin, Ganatra, & Rozenblyum, 2005).....	57
Figure 13.	Event Graph for multiple server queue M/M/k (Buss, 2001)	61
Figure 14.	DES model output measures: a) average number in system behavior b) time in system behavior.	64
Figure 15.	DTS model output measures with different time steps: a) average number in system behavior b) time in system behavior.....	65
Figure 16.	Percent error as a function of time-step for DTS models	67
Figure 17.	Simulation execution as a function of time step for DTS models	68
Figure 18.	Plots of percent error (a) and execution time (b) over multiple values of traffic density for $k = 3$	70
Figure 19.	Event Graph for multiple server queue M/M/k/c with balking.....	73
Figure 20.	Output measures for DES, DTS and analytical solution for M/M/k/c queue model with different queue capacity.....	75
Figure 21.	Output measures for DES, DTS and analytical solution for M/M/k/c queue model with different NIS capacity.....	76
Figure 22.	Toll plaza system (Amos, Galstad, & Higgins, 2005)	79
Figure 23.	Event graph model of toll plaza with listeners.....	80
Figure 24.	Toll plaza system wait delay for various service times.	82
Figure 25.	Toll plaza system wait delay for various booth number	84
Figure 26.	Agent tendency movement in MANA 4 (from McIntosh et al., 2007).....	100
Figure 27.	Agent tendency movement in MANA 5 (from McIntosh, 2009)	102
Figure 28.	An example DAFS entity structure (from Havens, 2002)	106
Figure 29.	Basic linear mover component Event Graph	110
Figure 30.	PathMoverManager component Event Graph	111

Figure 31.	Sensor-target interaction for a circular sensor (from Buss & Sanchez, 2005)	113
Figure 32.	A typical Sensor component Event Graph.....	114
Figure 33.	An example of Referee component Event Graph	115
Figure 34.	Cookie-cutter sensor Mediator component Event Graph.....	116
Figure 35.	A Shooter component Event Graph	118
Figure 36.	Illustration of a simple shooter simulation model in DES using the LEGO framework	120
Figure 37.	DAFS input design structure (from Havens, 2002)	121
Figure 38.	Two speeds agent miss-distance to a target location at various time-step sizes in MANA 4	125
Figure 39.	Agent locations and trip time over model runs from DES and DTS models in MANA 5 environment.	126
Figure 40.	Agent cumulative miss distance from target location for multiple time-step sizes in DTS model in MANA 5 environment.....	127
Figure 41.	Illustration of Scenario 2: a single agent and four waypoints in MANA 4 environment	129
Figure 42.	The agent's path traversed during Scenario 2 simulation execution in MANA 4	131
Figure 43.	Illustration of state transition delay caused by the time step method in MANA 4 environment; a) agent speed is not changed after one time step from the first waypoint, b) agent speed changed after two time steps from the first waypoint	132
Figure 44.	Trip duration versus time-step size results from MANA 5 implementation to Scenario 2	133
Figure 45.	Simple agent battle initial settings in MANA 5 GUI display, where 1 is the blue moving agent, 2 the neutral agent and 3 is the enemy (red) agent. Dotted lines show moving paths [best viewed in color].	137
Figure 46.	Illustration of the cumulative casualties between the blue and red agent.....	138
Figure 47.	Simple agent combat in a) Simkit environment and b) MANA 5 environment	141
Figure 48.	Time-step size impact on sensor range change in two agents combat simulation model.....	142
Figure 49.	Comparison of red agent's sensor range change effect on blue kill between Simkit (DES), MANA5, and MANA4 (DTS) results.....	143
Figure 50.	Typical sensor types used in combat scenarios, a) Cookie-cutter and b) two ranges constant-detection-rate Cake sensor	145
Figure 51.	A demonstration of sensor-target interactions for a Cookie-cutter type sensor and a single stationary target, a) without target lateral range and b) with target lateral range	146
Figure 52.	A demonstration of sensor-target interactions for a 2-ranges Cake type sensor and a single stationary target, a) without target lateral range and b) with target lateral range	147
Figure 53.	Two ring "Cake" type sensor Mediator Event Graph.....	148

Figure 54.	Cookie-cutter type sensor results with 4 seconds difference between DES and DTS approaches	149
Figure 55.	A two-ranges “Cake” type sensor results of the difference between DES and DTS approaches	150
Figure 56.	An example of a complex situation in which the DTS approach encounter difficulties to adjust for sensor detection	153
Figure 57.	A simple parallel search operation scenario of one agent and 20 stationary enemy targets in MANA 5 GUI display	155
Figure 58.	A demonstration of skipping phenomena in a simple two heterogeneous agents combat; a) DES model results and b) DTS (MANA 5) model results	156
Figure 59.	Illustration of randomness in the differences between the DTS and DES results for the simple two heterogeneous agents combat.....	157
Figure 60.	Parallel search operation results showing the number of detected enemy targets for various range of time-step sizes [best viewed in color].....	158
Figure 61.	Illustration of initial setup of agent refueling Scenario	160
Figure 62.	Mean casualty percentages of red and blue agents for the refueling Scenario.....	162
Figure 63.	A diagram showing the effect of increasing time-step size on the order of state transition occurrence [not to scale].....	163
Figure 64.	U.S. Navy envision of the LCS mission (from Jacobson 2010).....	165
Figure 65.	a. LCS combat scenario in MANA GUI, b. LCS combat scenario in DAFS GUI [not to scale].....	169
Figure 66.	Illustration of percent differences between MANA results at various time step sizes and DES results for the mean number of kills	171
Figure 67.	Illustration of percent differences between MANA results at various time step sizes and DES results for the mean number of weapons used	172
Figure 68.	MANA and DAFS simulation results of 5 selected design points for; a) the mean number of weapon fired by MH-60R, b) the mean number of weapon fired by the LCS, c) the mean number of enemy boat casualty, and d) the percentage of LCS casualty	174
Figure 69.	a) Simulation execution time as a function of the number of agents in a DES model, b) Simulation execution time as a function of the number of agents in a DTS model (from Matsopoulos, 2007).....	188
Figure 70.	A social network representation in Pythagoras 2.0.0 GUI screen with 100 agents, a) initial network setting before agent communication and b) final network setting after agent communication [best viewed in color].....	191
Figure 71.	a) Illustration of two behavior changes (consent to coalition and security) over time of 100 agents obtained from the social scenario in Pythagoras at $\Delta t = 1.0$ [best viewed in color, but colors do not represent behavior-color]. b) close view to show steady state levels.....	193
Figure 72.	Illustration of oscillations in few agent behavior levels for the 100-agent social scenario at $\Delta t = 3.0$ [best viewed in color, but colors do not represent behavior-color].....	194

Figure 73.	Illustration of oscillations in agent behavior levels for the 100-agent social scenario at $\Delta t = 5.0$ [best viewed in color, but colors do not represent behavior-color].....	195
Figure 74.	Illustration of oscillations in agent behavior levels for the 100-agent social scenario at $\Delta t = 10$ [best viewed in color, but colors do not represent behavior-color].....	196
Figure 75.	a) Illustrations of average differences in the population first belief (security) changes over the simulation time for multiple time-step sizes. b) average differences in the population second belief (consent towards coalition forces) changes over the simulation time for multiple time-step sizes.....	198
Figure 76.	Graphical display of the Iraqi population security level from PSOM [best viewed in color]	200
Figure 77.	Plot and table of the Mean Sunni Consent values towards coalition forces at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days.....	203
Figure 78.	PSOM's Iraqi scenario results for the differences between initial value and the final value at 1 year for the Sunni Consent towards coalition forces, overall security level and the Sunni security level [best viewed in color].....	204
Figure 79.	Plot and table of the mean sunni security values at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days	206
Figure 80.	Plot and table of the Iraq population mean security values at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days.....	207
Figure 81.	Kruskal-Wallis rank sum test for a) sunni consent, b) sunni security, c) Iraq security for different levels of time-step sizes.....	208
Figure 82.	Normal Quantile plots for a) sunni consent to coalitions, b) sunni security, and c) Iraq security with multiple time-step sizes (5,7,14,30,60,90,120 days), [best viewed in color].....	210
Figure 83.	Cultural Geography (CG) model description Chart (TRAC-MRY, 2011)....	212
Figure 84.	A sample illustration of Event Graph methodology used in the CG model (after TRAC-MRY, 2011)	214
Figure 85.	Political belief state changes over time for two arbitrary agents a) agent 1 and b) agent 2.....	216
Figure 86.	A demonstration of CG results of multiple agents' stances towards security over time [best viewed in color].....	217
Figure 87.	Discrete state level of an entity cognitive state over time (Alt, Lieberman & Alrowaei, 2010)	219
Figure 88.	A CG model agent network of communication following an event (from Alt, Lieberman, & Alrowaei, 2010).....	221
Figure 89.	A display of a simple SD model showing the time-step size and integration solvers used in Vensim®	224
Figure 90.	a) Time discretization in DTS and b) State quantization in DES, (from Bolduc, 2002).....	226

Figure 91.	Euler solutions to Stiff equation [best viewed in color], (from Buss & Alrowaei, 2010)	228
Figure 92.	Runge-Kutta Solutions to Stiff Equation [best viewed in color]	229
Figure 93.	Event Graph Quantized Model for 1-dimensional DE (from Buss & Alrowaei, 2010)	230
Figure 94.	Quantized solution for various state quantizations [best viewed in color], (from Buss & Alrowaei, 2010)	231
Figure 95.	Relative Speedup between DTS and DES models with different quantum sizes.....	232
Figure 96.	Sample plots of 1-dimension and 2-dimensions grid of DEs using DEVS methodology [best viewed in color]	234
Figure 97.	Event Graph of 2-dimensional ordinary differential equations using DEVS algorithm	235
Figure 98.	Next-event algorithm used by DES (Buss, 2011a)	251
Figure 99.	Basic Event Graph Construct (Schruben, 1983).....	252
Figure 100.	Listener component listening to Source component (Buss & Sanchez, 2002)	253
Figure 101.	Adapter example: event A in Source component causes event B in Listener component (Buss & Sanchez, 2002).....	254
Figure 102.	Example of a simple queueing service SD model	260
Figure 103.	Illustration of atomic-DEVS Model (from Bolduc, 2002).....	262

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	A time taxonomy in simulation (after Zeigler, Praehofer, & Kim, 2000)	25
Table 2.	Typical military simulation tools that uses DES and DTS approaches	39
Table 3.	Existing study in comparing simulation approaches regarding TAM	44
Table 4.	DES, DTS, and exact solutions for the M/M/2 queue	66
Table 5.	Percent error with different values of k and ρ for the geometric arrival approach (“ExeT” = execution time in minutes)	69
Table 6.	Percent error with different values of k and ρ for the Poisson (Batch) arrival approach and Geometric arrivals at $\Delta t = 0.002$ (“ExeT” = execution time)	71
Table 7.	DES and DTS execution times and accuracy for toll plaza with various service time rates.....	83
Table 8.	DES and DTS execution times and accuracy for toll plaza with various service time rates.....	85
Table 9.	TAM categorization of combat simulation models.	97
Table 10.	Table used in time-step size conversion to grid-based in MANA 4 for agent speed of 25 m/s.....	123
Table 11.	Agent input parameters used in Scenario 3 simulations	136
Table 12.	Two-ranges “Cake” type sensor input parameters.....	147
Table 13.	Input parameters for Scenario 6 combatants.....	154
Table 14.	Agent refueling Scenario input parameters.....	161
Table 15.	Variable factors used in the original study, all distances are in meters (after Jacobson, 2010).....	167

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ABM	Agent-Based Models
ABS	Agent-Based Simulations
ASM	Air-to-Surface Missile
AWARS	Advanced Warfighting Simulation/ Army Warfare Analysis and Requirements System
BTRA-BC	Battlespace Terrain Reasoning and Awareness Battle Command
CASTFOREM	Combined Arms and Support Task Force Evaluation Model
CBS	Corps Battle Simulation
CEM	Concept Evaluation Model
COMBAT XXI	Combined Arms Analysis Tool for the 21st Century
DAFS	Dynamic Allocation of Fires and Sensors
DAGR	Directional Attack Guided Rocket
DES	Discrete Event Simulation
DEVS	Discrete Event System Specification
DIAMOND	Diplomatic and Military Operation in a Non-warfighting Domain
DoD	Department of Defense
DOE	Design of Experiments
DTS	Discrete Time Simulation
EADSIM	Extended Air Defense SIMulation
Eagle	U.S. Air Force Simulation
GUI	Graphical User Interface
HBR	Human Behavior Representation
Hellfire	Heliborne, Laser, Fire and Forget
IW	Irregular Warfare
IWARS	Infantry Warrior System (U.S. Army)
JAS	Joint Analysis System
JICM	Joint Integration Contingency Model
LCS	Littoral Combat Ship

LOGIR	Low-cost Guided Imaging Rocket
M&S	Modeling and Simulations
MAS	Multi-Agent Simulations
MANA	Map Aware Non-Uniform Automata
MOE	Measures of Effectiveness
MOP	Measure of Performance
NOLH	Nearly Orthogonal Latin Hypercube
NLOS-LS	Non Line of Sight Launch System
NPS	Naval Postgraduate School
OneSAF	One Semi-Autonomous Force
PAX	Agent-based model for peace support missions by EADS
Ph	Probability of hit
Pk	Probability of kill
PMESII	Political, military, economic, social, infrastructure and information
PSOM	Peace support Operation Model
Pythagoras	Agent-based model develop to support Project Albert, USMC
SA	Situational Awareness
SD	System Dynamic
SEED	Simulation Experiments and Efficient Designs
SSM	Surface-to-Surface Missile
APKWS	Advanced Precision Kill Weapon System
TACWAR	Tactical Warfare Model
TAM	Time Advance Mechanism
THUNDER	U.S. Air Force Theater Campaign Simulation
TRADOC	Training and Doctrine Command
TRAC-MRY	TRADOC Analysis Center – Monterey
UAV	Unmanned Air Vehicle
VIC	Vector-in-Commander
WARSIM	War Simulation

ACKNOWLEDGMENTS

This dissertation would not be complete without the assistances of many people. First and foremost, I must thank the almighty God “Allah” for making this possible. Without the Lord support, this work would have not been accomplished.

I am sincerely grateful to my dissertation advisor, Professor Arnie Buss, for his supports and supervision throughout the entire process starting from finding a great topic to the final stages of this dissertation. Arnie’s simulation knowledge and experience has been of a great guide and his advice have been an education to improve my knowledge in the modeling and simulation field.

I must also thank all my committee members, Dr. Rudy Darken, Dr. Chris Darken, Dr. Paul Sanchez and Mr. Arijit Das for all their inspiring support and productive discussions towards the right answers.

I also would like to express my thanks to Dr. Ted Lewis and Dr. Tom Lucas for their support during and after my qualifying examinations. Their guidance and ideas have been encouraging my work during my study at NPS. Great appreciations go to Mr. Stephen Lieberman for all his support as well as document editing and discussions during this work.

Above all, my deepest gratitude goes to my beloved wife, Ebtihal, for her enormous sacrifices throughout all the years while I have been at NPS. She has encouraged and supported me in every step especially during her pregnancy difficult times. She is the reason I am pursuing this work. Also, my warmth appreciation go to my parents, Ali and Mariam, who their patience kept me going. All the best wishes go to my lovely triplets Ali, Mohamed (Modi), and Mariam (Meemee). I dedicate this work to my beloved wife and family.

My special thanks go to all faculties at the NPS MOVES Institute for their support as well as making me part of the MOVES family by participating in the agents and combat modeling group every Mondays, the annual MOVES research and education summit, and student researches. Special thanks to the MOVES director CDR Joseph Sullivan for all his collaborative efforts to keep my work on track. Many thanks go to

Curt Blais, Dr. Imre Balogh, LTC Jonathan Alt, Dr. Don Brutzman, Mr. Loren Peitso, Mr. Jimmy Liberato, and TRAC-MRY team. I also would like to extend my thanks to the NPS Operations Research faculties more specifically to Dr. Johannes Royset and Dr. Robert Dell (Chairman of the Operations Research Department) for the excellent intellectual assists throughout my years in the OR department. Also many thanks to Dr. Sam Buttrey, Dr. Matthew Carlyle, Kevin Maher, Dr. Javier Salmeron, Dr. Mike McCauley, Dr. Robert Koyak and Dr. Lyn Whitaker. Thanks to the faculties at the NPS Physics Department in particular my Master thesis advisor Dr. Andres Larraza (Chairman of the Physics Department), as well as to Dr. Daphne Kapolka, Dr. Nancy Heagel, Dr. Bruce Denardo, and Mr. Richard Harkins. I also thank the faculties of the NPS Mechanical and Aerospace Engineering Department, namely my Master thesis advisor Dr. Garth Hobson, as well as to Dr. Millsaps Knox (Chairman of the Mechanical and Aerospace Engineering Department), Dr. Morris Driels, Dr. Fotis Papoulias and Dr. Hebbbar.

I am proud of my country, Kingdom of Bahrain, and I would like to thank all the people who contributed to make this happen. Thanks to H.M. the King of Bahrain, H.R.H. the Crown Prince, H.R.H. the Prime Minister, H.E. the Commander-in-Chief of the BDF, H.E. the Chief of Staff of the BDF, the Commander of the RBAF, all of the Directors at the BDFHQ, and all who supported me directly or indirectly to achieve this work.

Thanks to my Ph.D. fellows at the MOVES institute and the OR department, as well as to Dan Wilkinson, Dr. Ed Rockower, Meredith Thompson, Julia McClenon, Mike Dunhour, and the Delta3D team for being my friends.

Finally yet importantly, I extend my appreciations to the Naval Postgraduate School for allowing me to benefit from working with such professional faculty members and boost my intelligence to a new level. Many thanks go to the team in charge of the International Program Office at NPS starting with COL Gary Roser, Cindy Graham, Kim Andersen, and Debbie Graham.

I. INTRODUCTION

“The goals of DoD’s M&S efforts are to provide tools in the form of models, simulations, and authoritative data that provide timely and credible results. Make capabilities, limitations, and assumptions easily visible.”

– Modeling and Simulation Coordination Office (M&SCO, 2010)

A. BACKGROUND AND MOTIVATION

1. Time-Advance Mechanisms in Modeling and Simulation

Computer simulation is a powerful tool that modelers and researchers use to evaluate and analyze the performance of existing or newly designed systems (Carson, 2004). M&S techniques have been used for decades in situations that are considered too complex to achieve analytical solutions, and in cases where the only alternative to examining the changes in systems is to numerically exercise the engineered models of real-world phenomena. One of the core purposes of simulation is to support decision makers across many disciplines faced with the need to investigate the behavior of dynamic systems. As such, M&S knowledge is considered by many to be one of the most important components the development of modern technology (Law & Kelton, 2000).

The advances in computer technology, particularly in processing efficiency and storage, has been a leading factor in promoting the use of M&S tools to address problems of increasing complexity in recent years. However, the increasing demand for simulations with higher fidelity and resolution, coupled with the necessity to maintain high efficiency and accuracy, produces numerous challenges to the M&S community. Simulation execution time is an important factor, as the complexity of a model is usually measured in time and space required to execute the simulation (Zeigler, Kim, & Praehofer, 2000). In general, the more details added to achieve higher fidelity and resolution, the greater the time to execute the simulation model. In addition, the problems facing government, industry and society are continually growing in size, complexity and uncertainty. The

need for appropriate simulation techniques to resolve such real-time problems in an efficient and accurate manner is apparent.

Decision makers often rely on M&S to provide information to base tactical and strategic decisions (Banks et al., 2005). Because of this increasing reliance on M&S, the amount of trust put in to modeling and simulation results have become increasingly scrutinized. Given that many M&S tools operate on assumptions and foundations that are largely invisible to model developers and M&S users, the trust put in M&S tools has become a major concern of decision makers. A common and important thread throughout the current challenges facing the M&S communities is the selection of an appropriate TAM to model the changes taking place in the real systems over time. The simulation TAM is a method used to keep track of the simulation time and advance the representation of time (simulation clock) variable forward in simulation (Law & Kelton, 2000).

Historically, there are two main TAMs in use within modern simulation systems (Law & Kelton, 2000; Banks, 2005; Deo, 2006). First is the “next-event” or “event-driven” method, implemented by Discrete Event Simulation (DES) models. The second is the “time-step” or “fixed-increment” method, implemented by Discrete Time Simulation (DTS) models. DTS is the most commonly used TAM in military combat simulation and agent-based models (Macal, 2010). Although these two approaches share the same objective of advancing the simulation’s internal clock, they exist as two distinct branches in the M&S community that have evolved their own bodies of literature without much crossover (Paoli & Tisato, 1996).

There are major problems with the representation of time in Modeling and Simulation (M&S) used by the military communities throughout the world. This work attempts to elucidate fundamental truths regarding the impacts of time-advance mechanisms (TAMs) on the accuracy and dependability of modeling and simulation results for use in military decision making contexts.

In this research, we empirically show that the choice of TAM does impact the simulation results and different TAM approaches can lead to different outcomes and

recommendations. We examine simulation tools used by the military community such as MANA, Pythagoras, PSOM, and Vensim in which the fixed time step approach is implemented, and simulation tools including DAFS and Simkit that implement a pure discrete event approach. We also illustrate that the choice of time-step size in DTS models can introduce significant qualitative anomalies. These anomalies are often invisible to modelers using any single time-step size because time effects in DTS models cannot be separated from inherent system properties. DES models tend to produce fewer anomalous behaviors than the DTS models. In DTS models, decreasing the time-step size often but not always produces results that are more accurate, but also tend to be computationally expensive.

Compared to analytical models, simulations have some favorable advantages in terms of modeling the details and capturing the stochastic nature of complex systems without establishing unnatural modeling assumptions. DES is a traditional tool for modeling operational systems wherein the operation of a system is modeled as a chronological sequence of events. The system state variables are updated instantly whenever an event occurs. This type of simulation approach is generally assumed to be suitable when events can be defined and mapped in advance.

Alternately, in DTS there is a fixed time-step size, Δt , that is the uniform increment at which time is advanced throughout the entire simulation. Simulation time typically starts at “0” and is advanced in single increments of size Δt with every state variable updated according to the logic defined by the model. The process repeats until a stopping condition is reached. Time-step approach is currently the preferred mechanism for agent-based models as well as continuous simulations involving differential equations. The system state variables can only change at discrete times in DTS, where as state variables can change at arbitrary times in DES. Figure 1 highlights the flexibility in state variable duration in DTS and DES. Figure 2 illustrates the common set of steps that modelers follow for the DTS approach (a) and the DES approach (b) through in a simulation study. The model components designated in the flowchart shapes with dashed borders show the differences between DTS and DES techniques.

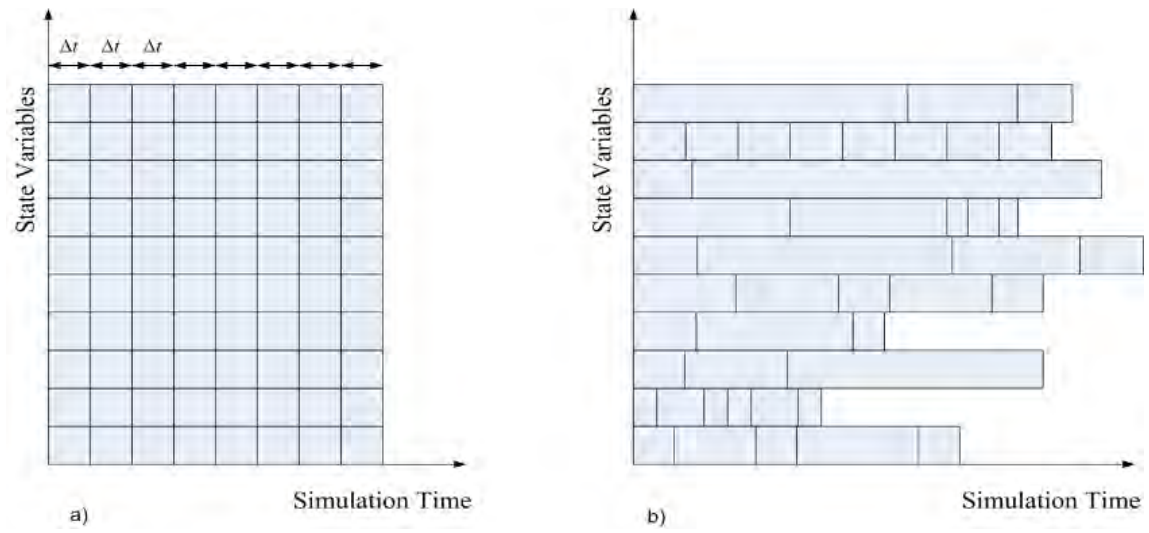


Figure 1. a) Time-stepped execution and b) discrete-event execution of state variables over simulation time

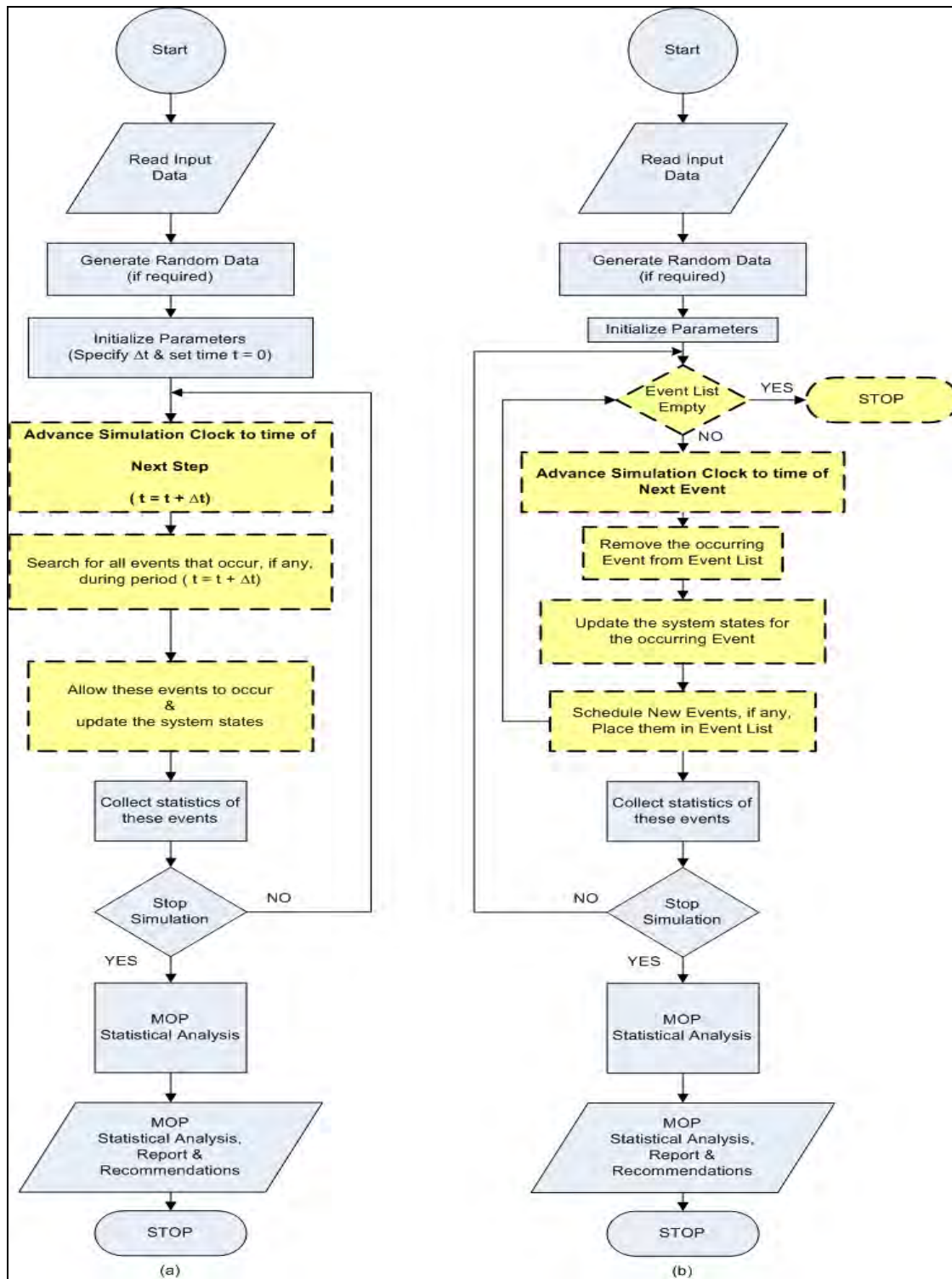


Figure 2. (a) Time-step (time-driven) approach, (b) next-event (event-driven) approach (after Bank et al., 2005)

2. Problems and Challenges of Time-Advance Mechanism

There are several significant problems and challenges germane to time-advance mechanisms (TAM). DES and DTS are heavily used approaches in most of today's simulations, and both are able to produce tools that assist government organizations in strategic development and analysis that gives decision-makers the capability for improvement. However, one approach could be superior to another for a particular class of models, and it is often difficult to determine which approach is superior to simulate a given system unless the two approaches are used side by side in a direct comparison. DES and DTS approaches come with benefits and shortfalls that are often unclear to the modelers. It is typically assumed that whichever simulation approach is used to address a problem, the results of the approaches will converge and lead to analogous decisions. Unfortunately, this is not always a valid claim, as there are cases where DES and DTS can produce significantly different results for comparable models. This can directly affect the decisions of developers, analysts and decisionmakers.

Complicating matters further, modelers are generally predisposed to working within their areas of expertise when simulating real-world problems. Thus, the time-advance method used for a particular model is often nearly invisible to the modeler using special-purpose simulation software and is hardly ever questioned by the decision makers (Tako & Robinson, 2008; Giambiasi & Carmona, 2006). The effects of these biases can reach all stages of the simulation including the analysis, recommendations, and implementations and ultimately forwarded to decision makers for operational actions. Decision makers in turn rely on these analyses to make decisions, and may put a great deal of trust in analyst recommendations from a single approach, while the other approach typically remains unconsidered.

Typically, once an approach is chosen for a range of simulation models, new related models will reuse the simulation many times as a platform for further development. These new models embark on the same approach and are rapidly put into practice to satisfy multiple demands without allowing other alternatives to be explored, creating a snowball effect. Using this strategy in early stages to declare a dominant

approach can be risky without examining what other alternatives have to offer. In many cases, modelers tend to introduce unnecessary assumptions in order to utilize the selected approach to suit various model conditions for closer representation of the real system. However, they tend to ignore a number of limitations related to the chosen approach that could question their means of analysis and put the entire model at jeopardy.

There is a fundamental problem with the way the majority of models, particularly military models and simulations, are currently constructed and interpreted. In particular, problems arise from the assumption that time-step methods can always be used to accurately model the real world. DTS introduces an additional parameter into a model, the size of the time step, that can have a substantial impact on the results of the simulation. In many applications using DTS, the model does not allow changing the size of the time step, and thus introduces the possibility that the results have unknowingly been affected. Moreover, such results may not accurately answer the questions faced in the design or the improvement of a system, and there exist few studies that directly compare the two most widely used simulation techniques, DES and DTS (Tako & Robinson, 2009; Galluscio et al., 1995). This research takes steps toward filling this gap.

As it relates to the problems identified here, this dissertation specifically covers the following issues:

1. The simulation field lacks studies that allow modelers understand the impact of TAM on performance analysis, and the limitations and strengths of each mechanism,
2. There is a great need for efficient, comprehensive and accurate decision support tools to address real-time decision problems specifically related to military domains,
3. It is not generally feasible for modelers of large-scale complex systems to generate multiple simulation approaches for the purpose of identifying a satisfactory approach, and

4. The use and consequences of the DTS approach in simulation models needs to be better understood, especially as DTS is the most common approach in military models.

Many simulation-based models have been developed using either DES or DTS, but few have been developed using both approaches. Furthermore, a substantial number of military-based simulations use DTS instead of DES for its simplicity to provide satisfactory rather than optimal solutions (Kaminski, 1996). This is not sufficient with today's ongoing threats and global crises where large-scale and expensive simulations are involved. The time-step approach has several known deficiencies, such as computational inefficiency with small time-step size, and inaccuracy with large time-step size. Despite these disadvantages, almost all of agent-based simulation software packages used in the defense research are DTS-based. On the other hand, DES modelers face other difficulties such as simulating continuous systems and representing complex human behaviors (Overeinder, 2000; Davidsson, 2000; Sloot, 2003; Tan, 2007).

Few studies in the literature regarding simulation techniques point out the deficiencies of each approach when put in practice. Work on the comparison of the two simulation approaches is limited, consisting mainly of conference presentations and course notes. To our knowledge there has been no methodical investigation of the comparative effects of different size time steps in such simulations, nor have there been any extensive studies of the differences between models constructed in DES and DTS.

As the simulation world is progressively and continuously using these major approaches to model more of the world's problems, the risk of producing inefficient or inaccurate simulation due to modeler's preference or random approach selection becomes extremely high. There is a need for comparison studies of DES to DTS approaches to illustrate which approach provides better solutions to assist government modeling & simulation and refine military and political policies in accurate and efficient ways. Modelers of each approach often lack the insight necessary to fully analyze the impacts of the other approach's capability or fill the gaps that may be introduced through a combined approach. This research aims to address the gap by identifying the significant differences and similarities between the two simulation approaches empirically across a

number of disciplines relevant to security and defense in order to provide future guidelines to select appropriate simulation technique.

The simulation time-advance mechanism is a critical choice for models and requires intelligent handling as it simply can reduce the simulation effectiveness (Kokar & Baclawski, 2000). Since the time-advance mechanism plays an important role in the simulation process, it requires close attention and emphasis. Unfortunately, there has been little said in the literature on the impact of time-advanced mechanism on the simulation results. Furthermore, there is no major study discussing the capabilities and limitations of those mechanisms and how to select an appropriate one for a model specifically when a problem could be modeled with more than a single mechanism.

Every simulation project proceeds through a series of steps to produce a successful project. With the exception of step 6, Figure 3 presents common set of steps in simulation studies; problem definition, problem formulation and plan, model conceptualization, data collection, computer program construction/translation, model verification, model validation, experimental design, production runs and analysis, reports and recommendations, and implementation (Law & Kelton, 2000). Many studies have described each of the above steps in details and provided essential guidelines to understand the behavior of complex systems. However, the step to select an appropriate simulation approach is typically omitted altogether, as are important decisions regarding the choice to program the model in a simulation general-purpose language or use a special-purpose software package (Banks et al., 2005). Often the preference in selecting a simulation approach is correlated to the modeler's area of expertise. There are few, if any, discussions made on providing general guidelines that modelers can follow when selecting simulation approaches, and there are rarely any discussions on the significance of simulation TAM in constructing trustworthy models. This research introduces a new essential step into simulation studies that has been concealed in the "*Computer Program Translation*" step. The new step is called "*Simulation Time-advance Mechanism Selection*." The purpose is to demonstrate that a selection of an appropriate mechanism does make a difference to enhance simulation flexibility, efficiency and accuracy in which it should be highlighted as a major step in simulation.

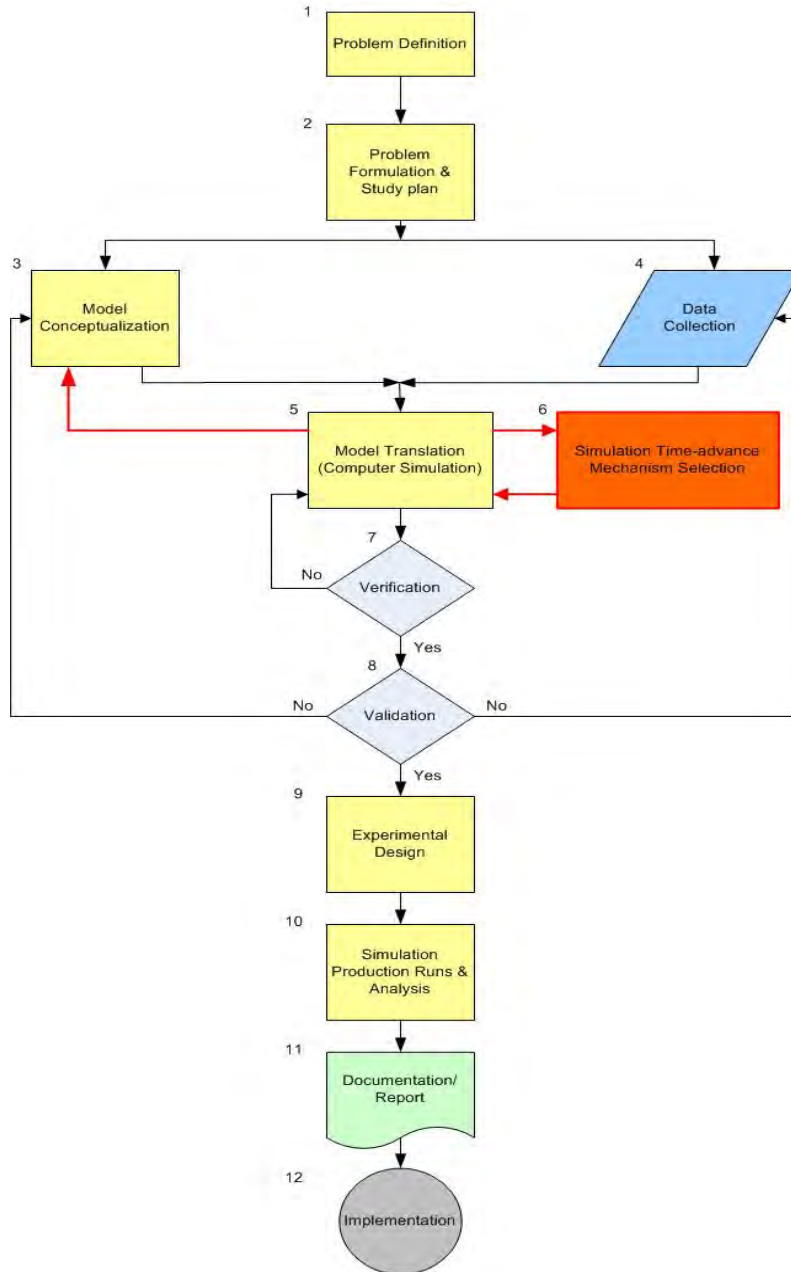


Figure 3. Steps in simulation study with modified step 6 (after Law & Kelton, 2000)

Here we specifically explore the problems associated with the choice of time-advance mechanism in modern military M&S. We do not consider the characterization or identification of optimal time-step sizes. Rather we only consider the impact of time-step size within the context of TAM choice. Likewise, we do not consider measures of

software performance or any measures of performance between modeling software packages, since there already exists an abundance of literature dedicated to these topics. Moreover, we do not attempt to fully exercise and describe new models of behavior, nor do we describe live simulation or training environments, although these areas also present major problems and concerns of their own in military M&S.

There are few studies in the literature, as discussed in Chapter II of this dissertation, that illustrate the significant differences in simulation performances when different TAMs are applied. The management and updating of the simulation's internal clock is extremely important for the efficient programming of dynamic simulations and is required to be utilized in careful manners. Thus, providing investigation that can assist in how to make appropriate selections to provide efficient, accurate, and flexible simulation tools is a significant question which needs to be researched.

Figure 4 presents classification of simulation models and their interactions in the structure of simulation. These classifications are by no means exclusive but they are used to describe the scope of different aspects of system (Cassandras & Lafortune, 2008). Most importantly, these classifications that help us identify the key area of our study in this research. Simulation models can be: 1) Static and Dynamic models, static are those that do not evolve time. In contrast, dynamic models represent systems that evolve over time 2) Linear and nonlinear, linear systems are those that can be described by linear equations 3) Continuous and discrete state, the state variables in continuous state models can take any value, where they are elements of a discrete set otherwise 4) DTS and DES, state variables are time-driven in DTS models and event-driven in DES 5) Deterministic and stochastic, a model is stochastic if there is at least one random variable 6) Discrete and continuous time, where discrete time model is when at least one variable (input, state, output) is defined at discrete point in time only (Cassandras & Lafortune, 2008).

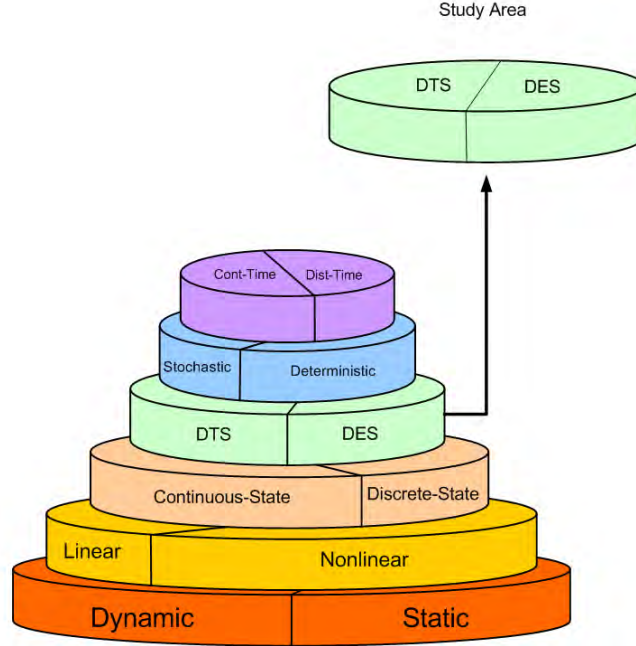


Figure 4. Illustration of simulation model classifications and study area (best viewed in color)

As modelers proceed through the steps of constructing simulation models, the classification choice is well understood except for the time advance mechanism choice. This is because every system is classified by its nature (e.g., static or dynamic) but each can be simulated by the DES approach and the reverse is not always true (Buss, 2011a). We will concentrate on studying DES and DTS approaches for dynamic, stochastic, linear and non-linear, discrete-time and continuous-time systems.

B. RESEARCH OBJECTIVES

1. Research Questions

The goal of this research is to provide modelers and simulation analysts with a comparative analysis between two major TAM simulation approaches, DES and DTS, across four military application domains to reveal key differences and similarities, examine how each approach represents systems used in various real-world applications,

as well as identify the advantages and pitfalls of each approach. Specifically, the study seeks to address the following general questions:

1. Does TAM affect the simulation results of a given system of interest?
2. What are the key limitations and benefits to each TAM approach, DES and DTS?
3. How does each methodology represent systems used in various real-world applications and affect the accuracy of simulation results?
4. How do these results affect the choice of TAM in military application domains, and the recommendations made in military decision making contexts?

We address these four general questions through specific examinations of applicable current military M&S domains as defined below in the next subsection. The null hypothesis is that the choice of time advance mechanism simulation approach (DES or DTS) to advance the simulation clock produces the same simulation system behavior. Our alternative hypothesis is that the time advance mechanism simulation approach does produce differences in the simulation system behavior. Included in this hypothesis is the proposition that changes in the time-step size do produce difference in the simulation system behavior within the DTS simulation approach.

H_0 : the choice of time advance mechanism simulation approach (DES or DTS) to advance the simulation clock produces the same simulation system behavior.

H_A : the time advance mechanism simulation approach does produce differences in the simulation system behavior.

(H_B : changes in the time-step size do produce difference in the simulation system behavior within the DTS simulation approach.)

2. Research Scope

The research here focuses on the impacts of the choice of TAM. We develop a new approach to the study of M&S that reconceptualizes the impacts of TAM choice in the context of decision making in modern military domains. There are few studies in the literature that illustrate the significant differences in simulation outcomes when different TAMs are applied. The management and updating of the simulation's internal clock is extremely important for the efficient programming of dynamic simulations and must be utilized carefully. Thus, determining the factors and process of making appropriate TAM selections to provide efficient, accurate, and flexible simulation tools is a significant question in modeling and simulation research.

The aim of this study is not to give a comprehensive survey of the literature, but rather to discuss whether there are identifiable features of certain systems that make one methodology superior to the other for the systems under consideration. We do not consider the characterization or identification of optimal time-step sizes, although we do consider the impact of time-step size within the context of TAM choice. Likewise, we do not consider measures of software performance or any measures of performance between modeling software packages, as there already exists an abundance of literature dedicated to these topics. Moreover, we do not attempt to fully exercise and describe new models of behavior, nor do we describe live simulation or training environments. We are addressing DTS models that purely implement the fixed increment time advance approach but not any other variations of DTS models. This is because most DoD models implement the pure DTS approach.

The overall objective of this study is to empirically compare DES and DTS time-advance methodologies over a reasonable range of military oriented applications to assist modelers identify an appropriate methodology to improve decision-making. The followings are our objectives:

1. Provide an investigation on how TAM affects the accuracy and dependability of simulation across a range of important military scenarios,

2. Highlight the significant differences in simulation outcomes between DES and DTS to provide guidelines for appropriate approach selection, and
3. Describe the strengths and weaknesses of each TAM implementation when used in complex dynamic system simulation, identifying the most appropriate technique for a given problem.
4. Gain new understanding of the question or issue being modeled when the comparison of two dissimilar models to address the same question or problem is prepared.

Modern military forces have adopted simulation as a useful and cost-efficient means of preparing and training for military operations (Smith, 1998). This comparative analysis between time-advance mechanisms focuses on the type of military constructive simulations that are used for tactical decisions and operational analysis. This is because *“In the future, the military reliance [on simulations] and use of military constructive simulations will increase”* (Hill, Miller, & McIntyre, 2001, p. 787). We do not consider virtual training simulations (e.g., virtual worlds) or similar applications and approaches. This study is centered around a critical challenge in simulation that helps to find a computational model that closely mimics the behavior of the dynamic system, specifically the simulation time-advanced approach. The full scope of this research consists of the analysis and model development of case studies from selected critical domains in modern military simulations. This is because military models have high stakes and the cost of making the wrong decisions is high in terms of both lives and money.

C. GENERAL METHODOLOGY AND RESULTS

1. Methodology

This dissertation uses quantitative and qualitative analysis to explore the impact of TAM on simulation results. Because there is a large number of domains in which this study could be performed, this research only explore scenarios related to military domains. The purpose is to provide DoD’s M&S community assistance to form models

and simulations that provide timely and accurate results, making sure that model capabilities and limitations are easily visible to the modelers. The domains studied here cover a wide range of systems used in several military related applications. These domains are analyzed in separate case studies:

Case I: Traditional Discrete Event Simulation Systems.

Case II: Combat Systems Simulation.

Case III: Human Behavior Representations.

Queuing systems are explored first as one of the most widely used traditional DES systems in simulation. Several widely used queueing systems such as M/M/k and M/M/k/c with bulking and high traffic intensity are considered. The motive behind working with these systems is their utility and applicability to many military situations. We also, investigate the impact of TAM approaches on a toll plaza systems, due to its important in recent research studies and its similarity to many military applications that involve service utilization. Several queueing system scenarios were explored in DES and DTS models in which performance measures were compared. Detailed methodology of this study is provided in Chapter III.

Combat modeling is a major field in military simulations, and agent-based models have been increasingly utilized. The purpose in this case is to generate moderate combat scenarios and utilize them with two currently deployed combat simulation decision support tools. Map-Aware Non-Uniform Automata (MANA) is a candidate for DTS model as it is heavily used in many combat simulation analysis. In using MANA, one can more accurately depict the attributes of the individual agents and MANA gives one the ability to vary these attributes, which allows the simulator to have the ability to observe and quantify the effects of these varying attributes on the battlefield outcomes. Another point in MANA's favor is how easy it is to use. It has a very simple interface that allows the simulator to vary all of the attributes of the agents involved. Dynamic Allocation of Fires and Sensors (DAFS) is the other candidate for DES model. DAFS was used in this case because it is an equivalent low resolution combat simulation tool to MANA. Also, it has the capability of representing individual platforms with sensors and

weapons in a similar manner compared to MANA. Several combat simulation scenarios were explored in DES and DTS models in which performance measures were compared. Detailed methodology of this study is provided in the combat simulation model chapter.

The third case investigates time representation in modern military operations, namely irregular warfare (IW) models involving human behavior representation (HBR). We first study the Peace Support Operation Model (PSOM) simulation environment that uses the DTS approach to measure the population consent towards a political entity actions and overall security over one year in the built in Iraqi scenario. We test the impact of the DTS approach on these measures by varying the time-step size of the duration on which population data is collected. The Pythagoras environment is also examined for a simple multi-agent social network in which agents interact with each other and influence their behaviors to change. Similar sensitivity test to the one in PSOM is carried with various time-step sizes. Finally, output measures such as population stance towards security is observed in the Cultural Geography (CG) model in which the DES approach is implemented. A qualitative comparison analysis is conducted between the DES and DTS approaches to illustrate the pros and cons of these approaches when used to model HBR. Detailed methodology of this study is provided in the human behavior representation chapter.

Our general performance measures for each case study are centered on three key measures 1) simulation efficiency, 2) output accuracy, and 3) system representations (*a qualitative measure*) involving modeling complexity. However, models will be evaluated under several scenarios and specific performance data outcomes will be analyzed for validity and compared against one another. Critical issues that have major impacts on final decisions will be illustrated and recommendations will be made.

2. Research Results

Our experiments show that time advance mechanisms can impact the both the quantitative and qualitative results and consequent decision making resulting from simulation studies across a wide range of military modeling and simulation packages, including MANA, Pythagoras, PSOM, DAFS, Simkit, CG and Vensim. The choice and

implementation of TAM has been shown to have a major effect in queueing systems, combat models, human behavior representation and continuous system models. Both models DTS and DES can provide insight in to the behavior of queueing systems. However, DTS models lose accuracy when compared to analytical solutions and DES models. Small time-step sizes tend to increase the accuracy of a DTS model but at the cost of high execution times particularly at high traffic intensity. DES models tend to provide more stable and efficient results that do not deviate from the analytical solution, but can take longer to design and build.

A set of queueing system scenarios illustrated that TAM has significant effects on the results and recommendations. DES models' estimates showed to be more accurate than the DTS results even under small time-step sizes. The simulation execution time of the DTS models was very different from the DES models and showed to follow a $1/\Delta t$ trend as the time-step sizes were decreased. The DTS approach shows poor performance as the queueing system becomes more complex. We demonstrated this fact by examining queueing systems with large number of servers, high traffic intensity and high queueing capacity limits. In those scenarios, the DTS model was exhausted and showed low performance in comparison with the DES model that maintained high accuracy levels. In a practical queueing application such as toll plaza, the choice of TAM approach had a significant impact on the results and the leading recommendations forwarded to decision makers.

A series of simulated combat scenarios shows that TAM has significant impacts on the outcomes and recommendations. DTS and DES model results did not converge to the same answer for the scenarios studied. DTS models under certain time-step sizes produced inaccurate results and emergent phenomena that would significantly influence the recommendations resulting from the simulation output. Resulting movement, detection, and engagement inaccuracies can all be invisible to the modeler during both development and simulation execution. Thus, these problems could go unnoticed up to and including the stage when recommendations are made to decision makers. These problems and a number of emergent anomalies in the behavior of combat models such as skipping, state transition delays, simultaneous events and event ordering can be caused by

time-step size in DTS models. Conversely, these issues do not arise in the corresponding discrete event simulations. A modeler's TAM approach seems to cause potential issues and the potential for misleading recommendations in combat models.

The choice of TAM also has a substantial impact on human behavior representation (HBR) simulation models such as those that are currently used in irregular warfare and counterinsurgency operations. The DES approach has the capabilities to represent time in which it allows researchers and users to authentically represent both individual and social behaviors, and yields several noteworthy additional benefits including information preservation, event-causal traceability, control for different timescales and assists in evaluating simulation results. Conversely, the DTS approach showed the dependency of the results on time-step sizes and how a vast amount of information may be lost under many circumstances. While the DES approach is shown to be relatively free of the anomalies encountered with time step approaches, this M&S domain is still in its fledgling stages of development for such situations. Thus, it is more difficult to draw general conclusion about HBR relative to more traditional combat simulations. Where we do make considerable progress is towards defining the nature of probable anomalies and steps toward separating the controllable versus invisible factors that influence HBR simulation outcomes.

There are inherent limitations to the time advance mechanisms that are independent of the other aspects of the design of the simulation software tools. While we can posit that the simulation tools are implementing the time advancement theories faithfully, our work here has shown that even the best implementation can lead to grossly erroneous results. DTS approaches generally rely on some extension of, for instance, the Euler integration method which theoretically surmises the simulation model as a set of systems of large differential equations. Thus, the only possible way to solve the system, in this form, is via fixed increments of time, regardless of the accuracy or strength of the software package.

It is impossible to cover all potential applications of M&S in any domain, so it is important to note that there may exist some scenarios in which DTS models would have no anomalous outcomes regardless of time-step size. It is even possible that some

scenarios exist where the DTS and DES results are identical regardless of time-step size. We have, however, shown glaring inconsistencies- both qualitatively and quantitatively- between time advance mechanisms across a large, and we argue representative, sample of military M&S domains. The problems we have encountered and documented are not abstract issues removed from operational reality. Rather, they are real problems with enormous operational significance that influence the outcomes of simulation studies and the resulting recommendations.

D. SIGNIFICANCE AND CONTRIBUTIONS

This dissertation aims to contribute to the growing bodies of important research in areas including generalized time-advance methodologies, discrete event simulation, agent-based simulation, and applied military analytical simulation and decision oriented modeling. The ultimate objective of the study is to provide decision makers with the appropriate tools to help them make better decisions. Providing a quantitative analysis of simulation approaches can lead to advance the state-of-the-art knowledge and help accomplish better decisions. The followings are major contribution of this dissertation:

1. Advance the knowledge of M&S community to understand the risks and benefits of commonly used time advance mechanisms TAM and the corresponding outcomes and decisions.
2. Enhance validation confidence by comparing simulations built independently using different simulation approaches. The process may discover bugs, misinterpretations of model specification, and inherent differences in toolkit implementations.
3. Provide documentation to help modelers identify an appropriate approach for the simulated case to produce accurate and efficient analysis for aid decision makers.
4. Help the military to consider these important aspects in their investment decisions and operational course of action assessment, thus helping them make trusted decisions.

5. Understand the methodological differences and similarities of the two mechanisms in ways that encourage cooperation between the two methods in a natural way (e.g. create hybrid simulation).
6. Create the foundation for future researches to provide guidelines on how modelers select an appropriate simulation approach.
7. Facilitate communication and dialogues between the communities of DTS and DES modelers would have great benefit, particularly in the field of military simulation.

E. OVERVIEW OF THE DISSERTATION

In this dissertation, Chapter II provides a summary of the literature about the comparison between time-driven and event-driven methodologies for a few complex problems in modeling and simulation. Readers and researchers who need to gain the fundamental of the current study can read it separately. The first section in this chapter focuses on qualitative comparisons between the two approaches. The following sections introduce quantitative comparisons including analytical modeling.

Chapter III gives an introductory description of the Queueing System that is developed for this work. Essential queueing scenarios such as M/M/k, and M/M/k/c are developed in DES and DTS then their performance measures are compared with analytical models for validation purposes. Toll Plaza systems are significant queueing applications under concerns of current research is also discussed in this section.

Chapter IV concentrates on the simulation modeling of combat military scenarios to compare agent-based models (DTS) and discrete event simulation (DES). Specific issues regarding combat models such as agents' movement, sensing and detection as well as event ordering are addressed. To support the analysis, and as a critical step of a complete simulation study of this section, computational results for a complex combat scenario involving Littoral Combat Ship (LCS) warfare operation are then discussed.

Chapter V develops a comparison study between DES and DTS approaches in complex simulations involving human behavior representation (HBR). Specifically, the

study investigates the effect of time advance mechanism on the accuracy of human behavior representation in irregular warfare (IW) scenarios. A number of military IW simulation tools such as PSOM, Pythagros and the Cultural Geography model are surveyed. We also explore the differences in representing continuous systems between DES and DTS.

The last chapter presents the summary, conclusions and recommendations for the dissertation by laying out major research contributions and future research directions.

II. LITERATURE REVIEW

“In simulation, time does not simply happen. We need to make it happen.”

(Cellier & Kofman, 2006, p.11)

A. RELATED WORK

In order to support our investigation of the effect of Time Advance Mechanism (TAM) on simulation results as described in the previous section, a range of research areas must be taken into account. Our goal is to accomplish a head-to-head comparison analysis between discrete time simulation (DTS) and discrete event simulation (DES) over a range of military applications between the most commonly used TAMs in simulation. We gain substantial insight from the related work discussed in this chapter. First, the essential theory of TAM in simulation has to be considered. Simulations are dynamic in nature and the method in which each entity in the system change state over time must be described and understood. Second, we study the DTS methodology in detail. Third, we discuss the emergence of the DES methodology, specifically as it relates to the military simulations described here, while we explore the benefits and disadvantages of both approaches as reviewed in the cited literature. Last, we review the existing related comparison studies of simulation techniques in military simulation. It is important to note that there are hardly any examples in the literature that specifically compare the DTS and DES methodologies and their accompanying effects on the simulation outcome accuracy and performance efficiency. The major goal of the current research is to address this gap.

1. Time Advance Mechanism in Simulation

Typically, a modeler goes through the set of steps shown in Figure 3 (in Chapter I) to construct the simulation that addresses the problem of interest. Conceptual modeling is the second step and it is considered one of the most time consuming steps in building a simulation model (Tako & Robinson, 2008). It involves determining the modeling approach and whether simulation is suitable. At this step, modelers should ask whether an

alternative modeling approach would be more suitable to simulate the problem (Law & Kelton, 2000). Once the appropriate approach is selected the conceptual model is then converted to a computer model in a step described as model coding. At this point, the modeler considers how the conceptual model might be implemented in the specific software that is being used for the simulation study. It is during this process when the modeler should review the choice of available time advance mechanisms for suitability in the simulation study.

Simulation is dynamic in nature, depicting changes in the system over time. That is, the description of the state of a system at time instants is necessary to describe the system behavior over those instants of time. Time is an essential element of simulation. We discuss the basis of how time is advanced in a simulation model as a way of representing the flow of the actual time in the real system being modeled.

All dynamic simulations need a system clock (a counter) to mimic real world time advances. Zeigler, Praehofer and Kim (2000) characterize time in simulations as either logical time or physical time. Logical time is described as a time base in an abstract notion, and it is measured by ticks of a clock embedded in a model as the ‘simulation clock’. On the other hand, physical time is also called metric time or wall clock time, is measured by ticks of the physical clocks. Zeigler also classifies time as either local time or global time, depending on the range of entities that are described by the clock. Local time is assumed to be valid only within a component of a system. That is, each individual component used to represent a system in the model has its own clock to process the time. Conversely, global time is valid in the whole system and shared by all components in the system. Therefore, time is classified into two dimensions, one along the logical/physical axis and the other along the local/global axis as shown in Table 1. Simulation time can fall in any one of the four combinations.

		Logical/Physical	
		Logical Time	Physical Time
Local/Global	Global Time	<i>Global, Logical:</i> All components operate on the same abstract time base (e.g. DTS)	<i>Global, Physical:</i> All components operate on the same system clock
	Local Time	<i>Local, Logical:</i> Each component operates on its own abstract time base (e.g. DES)	<i>Local, Physical:</i> Each component operates on its own system clock

Table 1. A time taxonomy in simulation (after Zeigler, Praehofer, & Kim, 2000)

Zeigler, Praehofer and Kim (2000, p. 35) classify M&S by saying that “Traditionally, modeling and simulation considered mainly the first (global, logical) combination.” That is, all components of a modeled system have the same time frame of reference and time is considered as an abstract measure. As we will explain later, this supports the claim that discrete time technique DTS is considered as a common approach in simulation. Notwithstanding, a physical time base must be employed to synchronize between a human perceived time base and the computer logical time base of simulations models involving human-in-the-loop and real-time training. Zeigler also argues that the concept of time as a quantity could introduce difficulties in simulation. These difficulties arise from the fact that while the implementation of time is necessary for simulations, time itself is not considered and not controlled as a model component.

Techniques of TAM were introduced to keep track of the values of model components in simulated time as the simulation progresses. The ‘simulation clock’ (global, logical time) is used in simulation models to present the current value of simulated time (Law & Kelton, 2000). Traditionally in modeling and simulation, two principle approaches have been used for advancing the simulation clock: 1) fixed-increment time and 2) next-event time (Law & Kelton, 2000; Cassandras & Lafortune,

2008; Buss, 2011; Stewart, 2004; Delaney & Vaccari, 1989). These two TAM approaches are the central theme of the literature described here.

Few studies have defined alternative time advancement mechanisms to those suggested above. Márquez introduced TAM as a key design element in model execution (Márquez, 2010) and defined three TAM approaches: 1) time-stepped or fixed-increment, 2) discrete-event, and 3) time parallel. Time parallel involves partitioning simulation time in multiple segments where each segment executes time independently. That is, model components use ‘local, logical’ time. We argue that the time parallel approach can be viewed as a special case of the first or second time-advance approach given that each segment has to advance the simulation time using one of the first two approaches regardless of segment independence.

Galluscio et al. (1995) suggested three approaches to the design of the simulation model: 1) the event-driven approach, 2) the time-driven approach, and 3) the process-driven approach (Galluscio et al., 1995). These three approaches were distinguished by the method used to control how time is advanced in simulation. In the event-driven the simulation clock is advanced by the time of the next scheduled event on the list. In the time-driven approach, the simulation clock is advanced by a fixed-cycle in which a list of activities is examined at each cycle and time is updated at the end of the cycle. In the process-driven approach, the actions related to each model component are incorporated into a process that is scheduled for execution at a later time. In this method, time is advanced by the process that is currently active in the simulation. Simulation time is advanced by processes, from one process to the next. That is, each process is a life cycle of a single entity which consists of a time-sequenced list of events, delays and demands for resources (Banks et al., 2005). In our view, the process time-advance approach is a special case of the event-driven (next-event) approach because simulation time ‘jumps’ to the end of each localized process, and events can be ordered in the form of an event list.

Delaney and Vaccari agreed that there is always a variable present to represent time in a digital computer simulation model (Delaney & Vaccari, 1989). During a simulation run the value of time is modified in some manner to simulate the advancement of physical time. They raise a key point in time advancement procedures: “this may be

accomplished in various ways; what is important is that the state variables be updated in each time advancement.” (Delaney & Vaccari, 1989, p. 418). That is, state variables only change when the simulation clock changes for all TAM approaches.

The modeler must manage the simulation clock when simulating any system. The effectiveness of managing this simulation clock will ultimately depend on the efficiency and accuracy of the simulation run. We now review the two most common TAM approaches, DTS and DES, used in various applications of military M&S.

a. *Discrete Time Approach*

Cassandras and Lafortune (2008) define the root of where DTS approach may emerged from, “Historically, scientists and engineers have concentrated on studying and harnessing natural phenomena which are well-modeled by the law of gravity, classical and non-classical mechanics, physical chemistry, etc..” When modeling physical systems, modelers depend on quantities called “continuous variables,” such as flow rate of fluids, velocity and acceleration of rigid bodies, and electrical current, to provide a full representation of a system behavior over time (Fishwick, 2007). For this reason many mathematical methodologies, such as analytical or numerical methods as well as physics techniques, have been developed to model, control, and analyze real-world systems. The dynamics of a system are modeled using mathematical tools such as difference equations, ordinary and partial differential equations, and other numerical methods. Traditionally, the solutions to these equations contain the procedure of how state transition occurs over time (Zeigler, Praehofer, & Kim, 2000). Real time is continuous, however. As a result, when modeling the system process over time, the state equations could take many different forms. Most of dynamic systems are based on differential equations with time being a variable quantity of the form (Cassandras & Lafortune, 2008):

$$\frac{dx}{dt} = f(x(t), u(t), t) \quad (\text{A.1})$$

$$y(t) = g(x(t), u(t), t) \quad (\text{A.2})$$

where the system modeling process consists of mathematical relationships involving the state variables $x(t)$, input $u(t)$, and output $y(t)$ in functions like f and g over time t .

Typically, numerical solvers provided with a set of initial states, a set of input values, a start time and stop time are utilized towards finding solutions to differential equations. Classical numerical solvers such as implicit and explicit Euler method, multi-order Runge-Kutta, and Adams are few of many solvers implemented to model continuous systems in specific as well as discrete systems (Fishwick, 2007).

Time is a continuous real-value in the real-world, however, “a digital computer has no means of computing numerically any real-valued function of a real-valued argument.” (Cellier & Kofman, 2006). Representing infinitely reducible real time computationally would require an infinite amount of real time. Thus, a complete or 1-to-1 representation of time is not feasible. Instead, the concept of a simulation clock is introduced in which the time axis in the simulation is discretized to keep the time in the finite values needed for digital processing. In this notion time advances from discrete time point to the next value in time. The discretization in time directly introduced what has become known as the discrete-time or time-step approach in simulation. This approach for advancing the simulation clock is sometimes called discrete time simulation DTS (Zeigler, Praehofer, & Kim, 2000; Buss & AlRowaei, 2010), fixed-increment time advance (Law & Kelton, 2000), time-stepped simulation (Wedemann, Barbosa, & Donangelo, 1999), and time-driven approach (Kokar & Baclawski, 2000).

As reviewed in the introduction, the simulation clock in DTS is advanced in increments of exactly Δt time unit for some appropriate choice of Δt specified by the modeler (Law & Kelton, 2000). This increment of time Δt is sometimes referred to as the ‘sampling interval’, another reference to the mathematical lineage and tradition of this TAM. Because a computer simulation cannot process continuous time directly, time is sampled and discretized in to units that can be processed digitally. In this way, continuous time is simulated (as global or local logical time) by the simulation clock. This Δt time units is chosen (e.g., 1 second, 10 minutes, 1 week and so on) to define the simulation clock and the system state variables change from one value, observation, or ‘sampled’ instant to the next value as time progresses. A simple illustration of a system state change in DTS is shown in Figure 5.

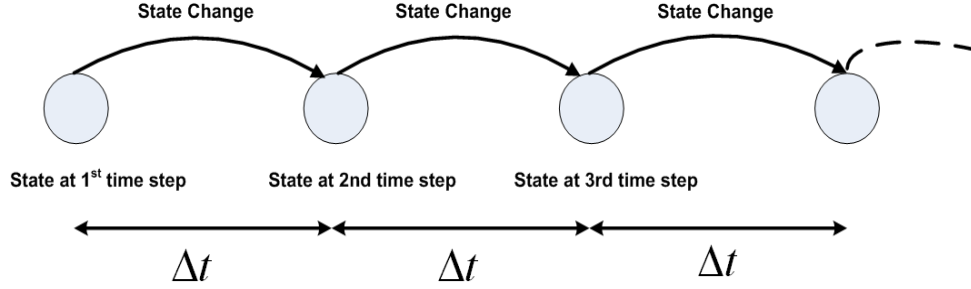


Figure 5. Stepwise execution of DTS (after Zeigler, Praehofer, & Kim, 2000)

After each update of the simulation clock in DTS, a check is made to all system components to determine if any state transition should have occurred during the previous interval of Δt . Any state transition scheduled to occur during time intervals is considered to occur only at the end of the intervals. State transitions occur only at “clock ticks,” and are synchronized by the simulation clock (global, logical) time base for TAM. The global clock is shared by all system components and “The clock alone is responsible for any possible state transition.” (Cassandras & Lafortune, 2008, p. 25). Figure 6 shows Law and Kelton view of fixed-increment time advance with “actual” time of occurrence of i th state transitions. Despite when transitions may occur in the continuous system being modeled (i.e., the natural occurrence), state transitions are only considered at the end on intervals.

We believe Figure 7 provide a better illustration of the DTS world view of the occurrence of the i th state transitions over time. The natural occurrence of state transitions in the real-world view should not be mapped directly to the fixed-increment time advance view.

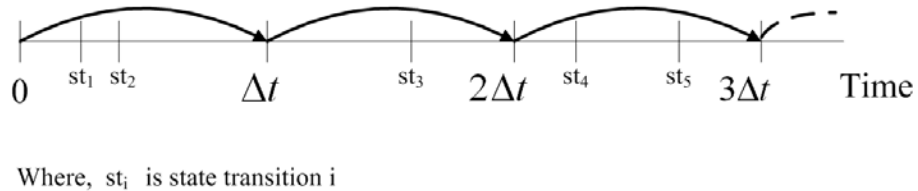


Figure 6. An illustration of fixed-increment time advance (after Law, & Kelton, 2000)

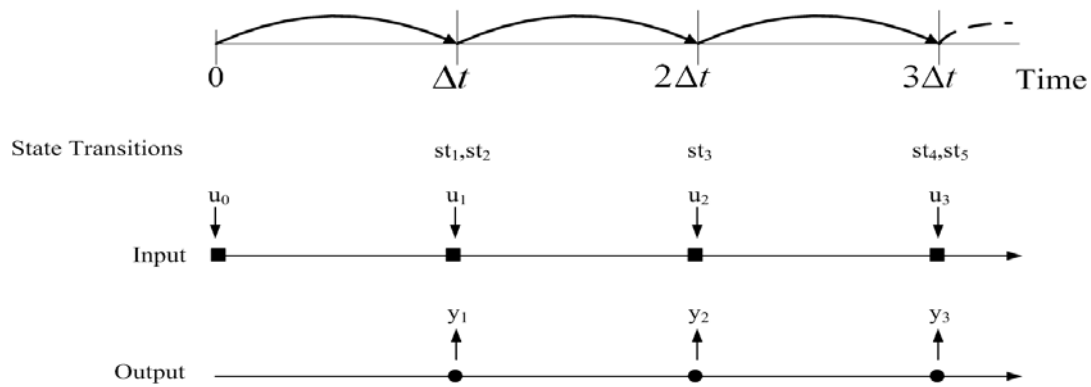


Figure 7. Preferred illustration of fixed-increment time advance from the simulation view

In general, modelers use the following pseudo-code in simulation to compute the state and output trajectories of a model with fixed-increment time advancement (Zeigler, Praehofer, & Kim, 2000):

```

StartSimulation
  //Initialization
  SimulationTime  $\leftarrow$  0.0
  Set simulation end conditions
  Initialize input parameters  $u(t)$ ;
  Initialize state variable  $x(t)$ ;
  Set Time-step size to  $dt$ 

  While (not end condition){
    SimulationTime  $\leftarrow$  SimulationTime +  $dt$ 
    Scan all system Objects
    Change state variables of objects at the end of this
                                time step according to A.1:
     $x(\text{SimulationTime}) = f[x(\text{SimulationTime}), u(\text{SimulationTime})]$ ;

    Update outputs of all objects at the end of this
                                time step according to A.2:
     $y(t) = g(x(\text{SimulationTime}), u(\text{SimulationTime}))$ ;
  }
EndSimulation

```

Figure 8 shows the flow of logical time in DTS models. It is important to recognize that the state variables are only considered at discrete instants of time (i.e., at

each time advance instant.). The process of re-evaluating state variables is repeated at every time advancement even if there is no state variable change.

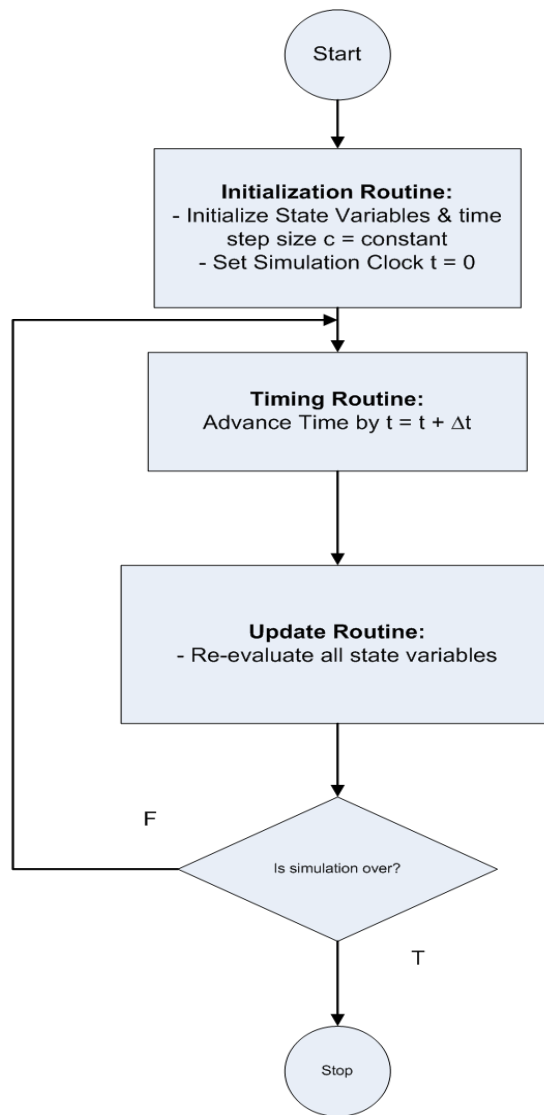


Figure 8. Fixed-increment time advance approach flow chart

The work to identify the benefits and disadvantages of DTS is scarce and not well defined in the M&S literature. The literature emphasizes that most modelers accept the notion that a “small time step will ensure accuracy at the expense of longer simulation time. A large time step will reduce simulation accuracy.” (Al-Hashimi, 1995, p. 46). However, the choice of an appropriate time-step size and the its impact on simulation

accuracy is not well understood. The selection of an appropriate “*small*” time-step size is ill-defined, and it does not necessarily provide an overall satisfactory solution. In fact Ledin stated that for a family of stiff systems, very small time-step sizes can indeed reduce the accuracy of the overall simulation (Ledin, 2001). This is because for some system components (e.g., slow-moving agents) small time steps round off error accumulates over the large number of integration steps. This is not a product of the length of times steps *per se*, but rather an emergent effect of the simultaneous processing of state transitions at discrete intervals.

Model simplicity, low cost and the flexibility to introduce additional components is a major advantage to time-driven approach in DTS. This is because differential equations and many other mathematical equations used to define the system are directly transferred to the model with minor modifications.

Law and Kelton (2000, p. 17) defined a critical disadvantage of fixed-increment time advance in DTS, “the errors introduced by processing events at the end of the interval in which they occur.” Events in this context are state transitions of the dynamic system being modeled. We will quantitatively analyze the effect of this phenomenon on simulation results in our study.

b. Discrete Event Approach

Time in discrete event simulation (DES) is advanced differently than in the time-stepped approach. The time of the natural occurrence of every system state transition is the key to this approach. System state transitions (changes) are represented by a collection of discrete events (Fishman, 2001), and a state transition implies that an event occurs. Said another way, events cause state transitions. The appropriate technique for modeling a particular system depend on the nature of the events in the actual system being modeled. That is, it is dependent upon the natural ordering and timing of events and their time intervals (e.g., between or among events). These intervals can be of varying lengths (i.e., random intervals) or pre-planned lengths (i.e., deterministic intervals).

With this approach, the simulation clock is first initialized to zero, then the times of occurrence of all future events are computed and stored in the future event list (FEL) (Law & Kelton, 2000; Buss, 2011a; Banks *et al.*, 2005). The simulation clock is then advanced to the time of the most recent occurring event on the FEL. The state of the system is updated by processing all state transitions related to current occurring event. The FEL is also updated to the occurrence of a particular event and its associated time is moved to the top of the list. The simulation clock is advanced to the time of the next event on FEL and that event is executed. The procedure is repeated by advancing the simulation time from event to the next event until a pre-specified stopping condition is accomplished. While the execution times of all events on the FEL may be planned in advance, the next event is the only event whose time of occurrence is certain. The execution times of all other events may change as the simulation progresses.

Discrete event simulation can be defined as modeling the system in which state variables are only changed at discrete point in time called events (Banks *et al.*, 2005). The DES simulation clock jumps between events (unequal jumps) and skips inactive periods where nothing is occurring. This is unlike the DTS simulation clock described above where no such skipping exists. The discrete event simulation time-advance approach is also called next-event (Law & Kelton, 2000) and event-driven approach (Kokar & Baclawski, 2000).

According to Zeigler, characterization of time advance in DES falls in the first (global, logical) combination. However, it is possible for such simulation clock to fall in the second (local, logical) combination if each component is driven by the occurrence of its own events. Unlike in DTS where state transitions are synchronized by the simulation clock, DES state transitions are the result of combining asynchronous and concurrent independent event processes (Cassandras & Lafortune, 2008). Since these asynchronously generated discrete events are the only cause for state transitions, then an abstract concept of time (such as that in DTS) is not necessary for driving the simulation. Events control simulation progression, and time cannot be viewed in these models as an independent variable.

Figure 9 shows demonstration of the next-event time-advance approach in DES (Law & Kelton, 2000). Comparing this approach to the fixed-increment time-advance approach clearly shows that simulation clock (represented by curved arrows) advances from the time of an event to the time of the next event.

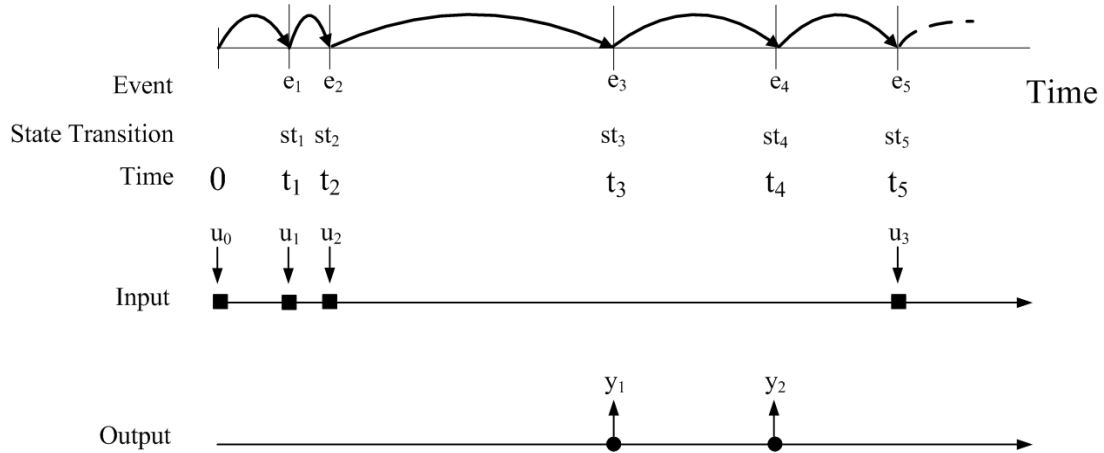


Figure 9. Illustration of DES (next-event) time-advance approach (after Law & Kelton, 2000)

All next-event time advance approaches share essential elements as defined by (Buss, 2011a): input parameters, system state variables, events (defined by state transitions), the future event list (FEL), event-scheduling relations, and a time processing component (timing routine). Buss expresses the interaction between these elements using next-event flow charts as demonstrated in the first figure of Appendix A. Figure 10 provides another view of DES elements interactions categorized by simulation routines.

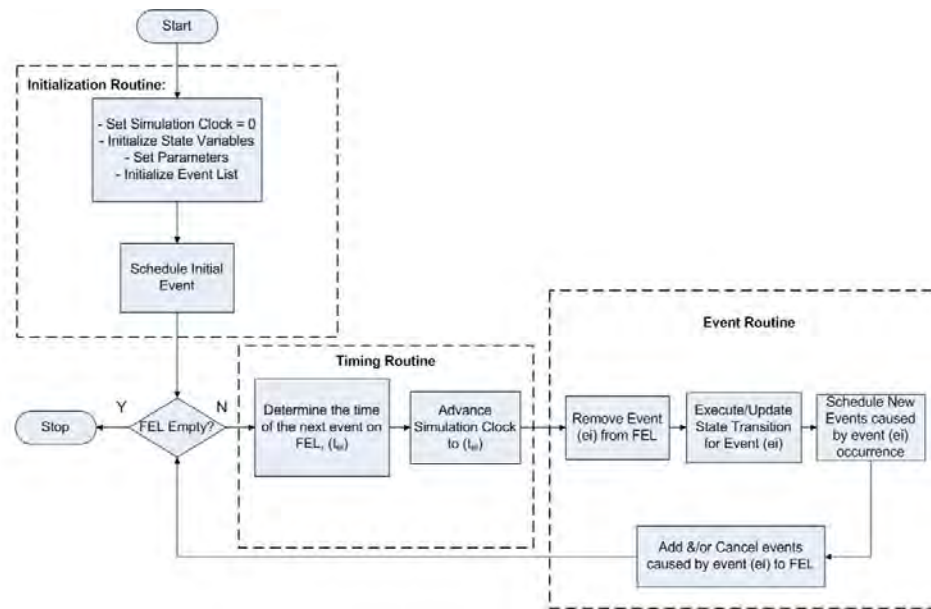


Figure 10. Next-event time-advance approach flow chart (after Buss, 2011a)

When coding the next-event time-advance approach in a general-purpose programming language, DES modelers typically use the ideas expressed in the following pseudo-code as a procedure to advance time and simulates the system behavior:

```

StartSimulation
//Initialization
SimulationTime  $\leftarrow$  0.0
Set simulation end conditions
Initialize parameters
Initialize state variable
Initialize event list
i event index {i=1,...,n}
j system object {j=1,...,m}

While (FEL is not empty){ //not end condition
     $E_i(j) \leftarrow \text{NextEvent}()$ ;
    SimulationTime  $\leftarrow E_i(j).time$ 
    Cancel ( $E_i(j)$ );
    Change state variable of j;
    If(j has interaction with other objects){
        Compute times of new events caused by j interaction
        Schedule() new events caused by j interaction
        Add() new events on FEL
        Cancel() earliest events caused by j if any
    }
    else {
         $E_{i(next)}() \leftarrow \text{NextEvent}()$ ;
    }
}
EndSimulation

```

There is a wide range of simulation software packages used by DES modelers to develop simulation models. These include programming languages such as JAVA, C, C++, as well as dedicated simulation software packages such as Simkit, Arena, AutoMod, Extend, Flexsim, Micro Saint, ProModel, QUEST, SIMUL8, and WITNESS (Banks et al., 2005). We use the Java-based Simkit as a source tool to build our DES models.

DES can be described using Event Graphs (EGs) as discussed in Appendix A. Schruben's (1983) and Schruben and Yücesan (1993) demonstrated the ability of mathematics of graph theory to be comprehensive basis for specifying DES models. They proved that Event Graph methodology is sufficiently powerful to represent any model that can be captured by the DES framework. In fact, they believe that EG methodology can be used as a tool to measure simulation complexity and its implementation to DES models has shown to reduce the overall complexity of the simulation models. Also it can be argued that every model that can represent a system in the real-world can be represented in a pure DES world view (Buss, 2001). In this dissertation, we use Simkit as our main tool to implement DES models.

As with any simulation technique, DES comes with a number of advantages and disadvantages. DES is newer to the M&S field than DTS, whereas differential equations have been around for more than 300 years (Zeigler, Praehofer, & Kim, 2000). Unfortunately, the discussion on the benefits and problems with DES over the few existing studies in the literature is insufficient to guide modelers. Zeigler describes that DES modeling is an attractive approach "*because it is intrinsically tuned to the capabilities and limitations of digital computers*" (*ibid.*, p. 66). He also expresses that DES only considers times and components where events are actually occurring. In another words, DES models concentrate on the times where actions are happening and not to times where the system is not changing states. Depending on how many events are taking place in the simulation model, the DES approach usually has better execution efficiency when compared with DTS (Buss, 2011a). Another benefit of DES is that only state changes related to certain components' behaviors are reported to components but not other state changes (Zeigler, Praehofer, & Kim, 2000).

Delaney and Vaccari (1989) express three problems with DES: 1) there are many “alternative laws of behavior,” ways of expressing state variable changes in the system. Some of these alternatives might be insufficient to represent components at all time. 2) The irregular intervals between events must be defined prior to simulation execution (e.g., as a random or fixed distribution). The randomness involved in these intervals as well as their possible dependence on state variables enhance the complication. 3) Multiple instantaneous events can cause problems with the execution logic of DES architectures. In such cases, a “tie-breaking” mechanism is generally used.

Cassandras and Lafortune (2008) identified a significant disadvantage to the DES approach by asserting that “*Clearly, event-driven systems are more complicated to model and analyze, since there are several asynchronous event-timing mechanisms to be specified as part of our understanding of the system..*” The authors did not provide any details on what measures of complexity is used and the conditions considered in using this approach. However this view is echoed by (Sweetser, 2004; Hill et al., 2001; Buss, 2001).

Buss (2011a) indicates that frequent intractions between objects in a simulated system can exponentially increase the simulation computational complexity of DES models, and in worst case scenarios, simulation execution time reach values equivalent to DTS model values. However, the modeling complexity of DES models remains the same unlike the DTS models.

We now discuss the benefits and limitations of DES over a range of specific classes of simulation models for military applications.

2. Military Modeling and Simulation

The military is the biggest user of models and simulations (Smith, 1998). “The U.S. military alone spends hundreds of millions of dollars acquiring, designing, fielding, and operating simulation systems.” (Hill & McIntyre, 2001). The military use M&S to address problems related to various aspects including training, budgets, acquisitions, analysis, tactical and strategic decisions, logistics, force structure and deployments, and communications. (Davis, 1995). Military simulations fall into three main categories; 1)

live simulation, 2) virtual simulation, and 3) constructive simulation (Smith, 1998). Live simulations are those in which real equipments such as vehicles and rifles are used along with real people. This type is typically practical for combat scenarios where physical testing and exercise are conducted. Virtual simulations are those in which people are immersed in so called ‘virtual environments’. For example, flight simulators where a real individual, such as a pilot, interacts with the virtual environment by taking real-world actions that are applied in the virtual world. Constructive simulations do not involve direct human input and are used to provide analysis for tactical and strategic decisions. These are often called ‘decision-support models.’ This type of simulation is heavily used by the M&S community to address a wide range of military issues (Hill, Miller, & McIntyre, 2001). Typically, constructive simulations are largely influenced by operation research that involves complex mathematical and statistical techniques such as optimization and algorithm applications. Constructive simulations are the main focus in this dissertation.

As the military relies increasingly on M&S for nearly all aspects of operations and planning, the demand to improve simulation efficiency and effectiveness of these tools also increases. It is particularly important to provide full support to the decision-making process in the case of constructive simulations. There is a need to address these issues in the military community, and the work presented here aims to fill some fundamental gaps in the theoretical and applied research necessary to address these problems.

Simulations have been implemented in various military applications. Both DTS and DES approaches have been intensively used across these simulations to facilitate the required representation of system behaviours (Smith, 1998; Hill, Miller, & McIntyre, 2001; M&SCO, 2010). Unfortunately, these time advancement approaches have been applied to different simulation models without an in-depth understanding of their limitations.

DTS approach have been intensively used in a wide variety of military constructive type simulation to serve many aspects. DTS models used for military combat and other applications are shown in Table 2.

On the other hand, DES approach have been recently implanted in a growing number of military constructive simulations to provide support to military M&S community. Some military DES models that are currently in use expressed in Table 2.

DTS Software Tools	DES Software Tools
TACWAR	COMBAT XXI
CEM	OneSAF
JAS	NSS
AWARS	DAFS/ JDAFS
VIC	CG
Eagle	Simkit
WARSIM	JAWAR
CBS	
EADSIM	
THUNDER	
JICM	
MANA	
Pythagoras	
PSOM	
BTRA-B	
DIAMOND	
PAX	
JANUS	

Table 2. Typical military simulation tools that uses DES and DTS approaches

3. Related Comparison Studies

Very little work directly examines TAM comparisons in simulation. Here we review those few studies that have compared different simulation approaches related to our work; however only three specifically address DTS and DES. The rest compare DES with other concepts of time, such as system dynamics.

Galluscio et al. (1995) compared the design and implementation of a parallel computing simulation that relies on the simultaneous execution of code on multiple platforms. They simulated a traffic flow network using both DTS and DES approaches. The overall simulation execution speed was their measure of performance for the study. In both sequential and parallel simulation, at sparse traffic flow, DES performed better than

DTS. However, DES slows down at dense traffic flow. Parallel simulation provided higher computational efficiency than sequential methods when examining both DTS and DES implementations.

Paoli and Tisato (1996) conducted a comparison study between event-driven and time-driven approaches for a model of control theory that sent command signals to multiple embedded agents. They focused on the complementary nature of both approaches to show which one is suitable as a primary approach to model the behavior of a control machine. The study modeled a control machine that schedules actions and dispatches commands to agents using either a DTS or a DES technique. Agents receive commands from a control machine and perform actions in real-time, such as moving to a location, grabbing an item, or dropping an item.

The authors concluded that time-driven approach was found to be superior over DES for several reasons. 1) The time-driven programming paradigms are highly expressive in many real-time application domains, 2) model simplicity makes efficient implementations easier, 3) real-time interactions lead to the use of a virtual clock that synchronize accordingly, and 4) agents are not aware of when and why they perform an action. For these reasons, the authors state that the event-driven was not preferable.

Their paper also highlights the fact that the choice of a suitable TAM approach strongly depends on the interactions of simulation subsystems. If the system has a deterministic temporal behavior, then global plans are better defined in terms of global time. This denotes the use of a DTS approach. Otherwise, if the communication has a non-deterministic temporal behavior, then local plans defined in terms of local time are more appropriate. While their study is informative, it does not address all of the factors needed in a TAM comparison study for military M&S. In fact, the authors state that, “A general comparison of the two approaches falls outside the scope of this paper” (Paoli & Tisato, 1996, p. 853).

Kokar and Baclawski (2000) qualitatively compared DTS and DES when used as methods to simulate dynamic systems. They expressed that quantitative simulation of dynamic systems execute the simulation model typically at a fixed time step to obtain

state variable and output values. They alluded to the fact that most systems could be modeled with DTS and DES approaches separately. These systems are referred to as “fully-driven” in which any approach can be applied.

With the use of two examples, computer processor and software system calling a compiler, the authors shows that DTS is used for general use where DES is used when the events are of particular interest to the analysts. They qualitatively state that several systems may suffer from the limitation of each approach when applied in isolation, and suggested the use of a mixed modeling strategy where possible. They advocate the use DTS for some system components and DES approach for the other components.

Bakeman et al. (2009) modeled observer agreement for simulated time event sequences using two time-based algorithms and three event-based algorithms. Observer coding accuracy considered as the key measure and a parameter called Cohen index varied in the study. Both approaches captured a small number of errors. Time-based approaches tended to overestimate the number of observer decisions, where event-based approaches tended to underestimate those values (Bakeman, Quera, & Gnisci, 2009). They recommended that both approaches be used together, since each provides valuable information.

Qiu and Hu (2010) conducted experiments to compare DTS and DES approaches for pedestrian crowd behavior simulation. They explored movements of agents in a crowd system’s spatial and temporal heterogeneity. Although, the discrete event approach was not purely used, the results showed differences between the two simulation approaches. The DES model achieved better overall performance than the DTS model. They discovered that fewer decisions were made by each agents in the DES model than in the DTS model.

There are a number of studies that contribute to the comparison of DES and System Dynamics (SD) approaches. System Dynamics models consist of a system of stocks and flows where continuous state changes occur over time. These stocks and flows are a set of difference equations (similar to differential equations) but with a fixed time

step (Macal, 2010; Tako & Robinson, 2008). The fixed time aspect of this approach makes it a suitable comparison area for the current research.

Brailsford and Hilton (2001) attempt to compare DES and SD approaches to simulate two cases in the health care system. They discussed that the choice of a simulation approach depends on the purpose of the model and provided few selection criteria. It is important to note that only a single approach was implemented towards each case. That is the SD approach used to model the cardiac surgery, where DES modeled AIDS infecting individuals in a population. Unfortunately, drawing conclusions without employing both approaches to each case, produce unfair comparison and directly influence the outcomes creating biased observations. The study was a qualitative comparison but failed to put the two approaches into a direct comparison.

Tako and Robinson (2008) presented an empirical study on the comparison of model building process in SD and DES using 5 expert modelers for each approach. The study focused on verbal analysis of 7 modeling tasks: problem definition, conceptual modeling, data input, model coding, verification & validation, results & experiment, and implementation. The results show DES modelers follow a more linear path to design completion than the SD modelers. DES modelers also spent less time at the conceptual modeling stages than SD modelers. However, DES modelers spent much more time at the coding stages, relative to SD modelers.

An important observation of their work is that most modelers did not consider using alternative approach before building their simulations. That is, each modeler used the approach they are familiar with rather than the appropriate approach to model the system. Both approaches prove to be fairly equal in data input tasks, but some differences in verification and validation exist. The study attempted to show the differences between DES and SD in a qualitative analysis, but failed to provide recommendations on modeler selections of these approaches.

Lin et al. (2008) conducted a study comparing two time synchronization protocols; DTS (referred as clock-sync) and DES (referred as event-sync) in wireless sensor networks (WSN). The study objective was to identify the appropriate time

protocol that provides accurate registration of observations and precise time synchronization between sensor nodes in a network. Simulation results of both DTS and DES approaches were compared with the analytical solution and showed that; 1) DES provided better accuracy than DTS, 2) with high node mobility level, DTS achieved better accuracy than DES, 3) DTS outperformed DES under high data rates sent from source nodes. The study did not apply the DES approach in the pure form described earlier; it considered local clocks to each node but no event list to control the overall simulation time. This caused the clock of different sink nodes to be not synchronized and produced error.

Özgün and Barlas (2009) also conducted a study comparing DES and SD for a simple M/M/2 queueing system with crowd-dependent arrival rate. The authors show that there are minor differences between DES and SD at steady-state values. They did not show how the choice of time-step size would impact the study, nor did they explain in detail the reasons for the stated similarities.

Peisto (2011) briefly discussed the strength and weaknesses of using the fixed synchronous time step advance approach and the adaptive asynchronous discrete event time advance approach in a study related to hiding communication latency for real-time distributed simulation system of system domain. The work proposes that the asynchronous discrete event approach can provide better benefits in controlling events and effective latency hiding.

Technique	Researchers	Application Area
DES & DTS	Galluscio et al. (1995)	Parallel Computing Simulation
DES & DTS	Paoli & Tisato (1996)	Control Computing Simulation
DES & DTS	Kokar & Baclawski (2000)	Computer Process
DES & SD	Brailsford & Hilton (2001)	Health Care
DES & DTS	Bakeman et al. (2009)	Observer Agreement (CS)
DES & SD	Tako & Robinson (2008)	Simulation Design
DES & SD	Lin et al. (2008)	Wireless Sensor Network
DES & SD	Özgün & Barlas (2009)	Queueing Systems
DES & DTS	Qiu and Hu (2010)	Crowd Behavior
DES & DTS	Peisto (2011)	System of System

Table 3. Existing study in comparing simulation approaches regarding TAM

These findings are summarized above in Table 3. All the studies agreed that selecting the right modeling technique is essential in assuring a better representation of the selected problem in the different areas. We discovered a gap in the amount of work comparing DES with DTS, and we have found no work on comparing the accuracy of DES and DTS results for the study of military applications such as combat systems, agent-based modeling and human representations. This research is intended to fill this gap and focus efforts on investigating the effect of TAM approaches DES and DTS on simulation accuracy and efficiency.

III. CASE I: QUEUEING SYSTEMS AND THEIR APPLICATIONS

“Queueing models will continue to play a major role for most engineering disciplines wherever there exists humans, jobs, or tasks that must wait for a shared resource.”

(Sokolowski & Banks, 2009, p. 89)

A. INTRODUCTION

1. Background

Queueing theory is a mathematical theory about a century old that studies waiting lines. Even today, many real world problems are analyzed and solved using basic queueing theory mechanisms. Queueing systems have increasingly been used to represent many real world phenomena in applications like communications, healthcare systems, traffic flow, manufacturing processes and the scheduling problems of many services (Fazel & Fakheri, 2008). As a result of the widely used queueing systems, queueing representation and queueing behavior have become more and more complex and it has become impossible to obtain analytical results for many systems of interest. Computer simulation is an effective technique for solving queueing problems where analytic solutions are not available. The simulation of queueing models becomes attractive and useful to decision makers due to its ability to provide efficient and accurate for problems that cannot be solved analytically by common queueing theory.

As stated in Chapter I, simulation time-advance mechanisms introduce a number of disparities that can affect the simulation results. Queueing systems have been modeled using both approaches, DES and DTS, as discussed in this chapter’s literature review, and it is important to highlight the impact of these approaches in the results. Although the potentially harmful effects of the size of the time step is well-known in the numerical methods literature, there has been little work in traditional simulation modeling domains, such as queueing systems. This chapter is directed at answering questions such as:

1. Are there any significant differences between measures of performance at steady-state periods when DES and DTS approaches are used to model several types of M/M/k queueing systems?
2. If there are any differences, under what conditions do they occur?
3. Does the choice of the TAM approach affect both the behavior of queueing systems and the decisions to build these systems?
4. What are the advantages and limitations of each approach in modeling queueing systems?

In this chapter we will investigate these differences in a comparative analysis study between DES and DTS for some queueing models. More specifically, we will study the impact of time-step size in DTS on the results as well as on the simulation execution time for some M/M/k queueing system scenarios as well as a simple toll-plaza problem. The results of the discrete time simulation (DTS) will also be compared with analytical solutions and the corresponding discrete event simulation (DES) results. The study presented in this chapter is directed at building a communication bridge between the DES and DST communities. We think it can greatly help analysts in their work to select the most appropriate simulation approach that provides decision makers with efficient and accurate results.

Section B of this chapter provides a summary of the literature about simulation methodologies used to represent queueing systems, as well as a specific queueing application in toll plaza problems. Section C describes a few scenarios in the queueing theory, more specifically the M/M/k models as well as the comparative analysis. Section D discusses the discrete-event and discrete-time simulation models for the toll plaza problem. Finally, section E presents a summary of conclusions drawn from this case study and some discussions. All computations (analytical equations) and simulations (DES and DTS) version of the models were implemented in SimkitTM (Buss, 2011b) a package written in JavaTM. A single laptop with CPU speed 2.00 GHz and RAM 2.00 GHz was used for the study.

B. LITERATURE REVIEW

With the expansions in computer technology and the complexity of queueing problems, simulation-based stochastic system modeling has attracted an increasing amount of interest from both academia and industry. This section reviews some of the main literature related to queueing simulation modeling as well as the toll plaza system, which is a concrete problem in the transportation sector where queueing simulation methods have been applied. The first part in this section describes the analytical methods used to solve the M/M/k queueing problems including a brief on queueing theory. The second part discusses DES and DTS simulation methodologies in solving queueing problems. Finally, the third part discusses an important queueing application in transportation that involves a simple version of the toll plaza problem.

1. Queueing Systems: Analytical Algorithms

Queueing models are typically used to represent and study the behavior of many real-world service facilities that are viewed as queueing systems, such as communications and computer systems, production systems, repair and maintenance systems, healthcare facilities, and transportation systems (Banks et al., 2005). Queueing models can be analyzed mathematically using queueing theory and/or with the use of simulation to provide analysts with a powerful tool to evaluate and predict a number of useful performance measures of queueing systems. Typical performance measures of queueing models are the average number in the queue or the system, the average time spent in the queue or the system, and the server's utilization percentage. Erlang laid the path for modern queueing theory when he considered the problem of determining how many telephone circuits were necessary to provide phone service that would prevent customers from waiting too long for an available circuit (Hillier & Lieberman, 2005). Today, there are tens of books and thousands of papers that study and evaluate queueing systems to provide the most accurate performance measures of these systems. Kendall developed a notation for describing queueing processes. A thorough description of it can be seen in reference (Gross & Harris, 1998). The notation describes a single process as a series of symbols $A/B/X/Y/Z$, where A : is the inter-arrival distribution of the customers, B : is the

service time distribution, X : the number of parallel servers, Y : the maximum number of customers allowed in the system, Z : scheduling discipline/queueing strategy. In many situations, only the first three symbols $A/B/X$ are required because they are the three most important characteristics. The simplest and most commonly used queueing system that has been set as the foundation for a number of queueing applications is the $M/M/k$ queueing system (Willig, 1999). The assumptions in these queueing models are: entities (customers) arrive one by one according to a Poisson process (M for Memoryless or sometimes Markov) and join an infinite capacity queue waiting to be served. Entities are served in the order they have arrived by any one of k servers – each having exponential distribution service time, and depart the system once the entity service is completed. Each server can serve only one entity at a time. There are several queueing systems such as $M/G/k$, $G/M/k$, $G/G/k$ and $M/E_R/k$ that could be studied but the choice of $M/M/k$ models is ideal for the purpose of this study due to the availability of closed-form analytical equations for steady-state measures.

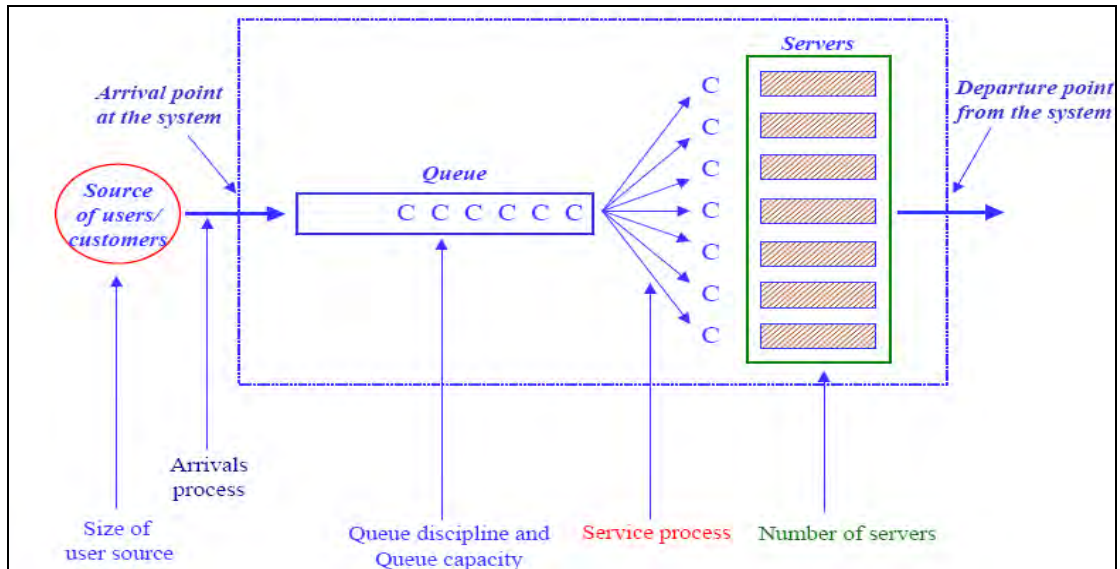


Figure 11. $M/M/k$ Queueing Systems diagram (from Odoni, 2004)

There is a huge body of publications in the operations research that describe the $M/M/k$ queueing systems and provide steady-state closed-form analytical equations for a limited portion of problems; some recommended textbooks for queueing theory

introduction are (Kleinrock , 1975; Ross, 1994; Ross, 2003; Gross & Harris, 1998). For the purpose of this study the M/M/2 queueing systems' steady-state analytical equations are first expressed where the M/M/k multiple server queue models are considered later. The M/M/2 queue model has independently identical distribution of inter-arrival times, which are exponentially distributed and i.i.d. service times with exponential distribution. The system has only two servers ($k = 2$) and uses the FIFO (First-In First-Out) service discipline where the waiting line is of infinite size. This system is a Markovian system and classified as a continuous-time Markov chain CTMC (Ross, 2003). Two measures of performance are of interest in this investigation: the steady-state average number of entities in the system $\bar{L}$ and the steady-state average time spent at the system $\bar{W}$. The assumptions for this simple model are that entities arrive (no batch arrivals) in accordance with the Poisson process with an arrival rate λ . Service times are identical and independent exponentially distributed for each server with service rate μ . To achieve steady-state conditions and queue stability, it is necessary for the server utilization rate $\rho = \frac{\lambda}{2\mu}$ to be less than 1. (Ross, 2003) presented the following equations for the steady-state (long-run) average number of customers in the system $\bar{L}$ in terms of λ and μ , where

λ = the mean number of arrivals per time period (mean arrival rate)

μ = the mean number of services per time period (mean service rate)

k = the number of servers in the system.

ρ = the server utilization rate or factor (ratio of arrivals to service)

The probability that no entities are in the system:

$$P_0 = 1 - \frac{\lambda}{2\mu} \quad (1)$$

The average number of entities in the waiting line:

$$L_q = \frac{\lambda^2}{\mu(2\mu - \lambda)} \quad (2)$$

The average number of entities in the system:

$$\bar{L} = L_q + \frac{\lambda}{2\mu} \quad (3)$$

The average time a entity spends in the waiting line:

$$W_q = \frac{L_q}{\lambda} \quad (4)$$

The average time an entity spends in the system:

$$\bar{W} = W_q + \frac{1}{2\mu} \quad (5)$$

The probability of n entity in the system:

$$P_n = \begin{cases} \left(\frac{\lambda}{\mu}\right)^n P_0, & (0 \leq n \leq k-1) \\ \left(\frac{\lambda}{\mu}\right)^k \frac{1}{k!} \frac{1}{1-\rho} P_0(n) \cdot (1-\rho) \rho^{n-k}, & (n \geq k) \end{cases} \quad (6)$$

Therefore, the average number of entities in the system:

$$\bar{L} = \frac{\frac{\lambda}{\mu}}{1 - \left(\frac{\lambda}{2\mu}\right)^2} \quad (7)$$

In queueing theory, John Little introduced Little's formula, which states that several relationships exist among the operating characteristics when the system is empty. This relationship asserts that the long-term average number of entities in a stable system $\bar{L}$ (known as the offered load), is equal to the long-term average arrival rate, λ , multiplied by the long-term average time an entity spends in the system, $\bar{W}$, or :

$$\bar{L} = \bar{\lambda} \bar{W} \quad (8)$$

$$\therefore \bar{W} = \frac{\bar{L}}{\bar{\lambda}} \quad (9)$$

where, $\bar{\lambda}$ average arrival rate in case of λ_n for each arriving entity.

Willing (1999), Fakheri and Fazel (2007), Lee and Strawderman (2009) later developed analytic equations for the steady-state solutions for the system size of the M/M/k queueing systems. For an M/M/k system, under steady-state conditions, the probability that there are zero arrivals in the system P_0 is given by:

$$P_0 = \frac{1}{\sum_{i=0}^{k-1} \frac{\rho^i}{i!} + \frac{\rho^k k}{k!(k-\rho)}} \quad (10)$$

It was observed that for M/M/k systems, the long-term average number of entities in a stable system is:

$$\bar{L} = L_q + (k - S) \quad (11)$$

where, S is the number of idle servers

and L_q is the average number of entities waiting in the queue, expressed as:

$$L_q = \frac{\rho^{k+1}}{k \cdot k! \left(1 - \frac{\rho}{k}\right)^2} P_0 \quad (12)$$

The average time spent by entities in the system is obtained by applying Little's formula to produce:

$$\bar{W} = \frac{\rho + L_q}{\lambda} \quad (13)$$

The above equations are used in section C to arrive at the exact solutions for our measures of performance, which are used to validate the simulation results.

A special case of the M/M/k queueing systems is the M/M/k/c queueing models with balking at capacity c where the same principles apply. A new entity arriving to the system finding c entities (where $c \geq k$) waiting in the queue, does not enter the system and is considered lost, i.e., it drops from the system and never comes back. This phenomenon is often referred to as balking. Various practical queueing systems, particularly those with balking, have been widely applied to several real-life problems such as situations involving impatient customers. Queueing systems with balking have been studied by many researchers. M/M/1 queue model with customer balking was first proposed by (Haight, 1957). Abou El-Ata and Hariri (1992) produced equations that use probabilities to study the multiple server's queueing system M/M/k/c with balking and reneging. (Zhang et al., 2005) analyzed the M/M/1/c queueing system with balking, reneging and server vacations. They developed equations for the steady-state probabilities, derived the matrix form solution for those probabilities and gave equations for some performance measures. (Hillier & Lieberman, 2005) introduced closed-form formulas for some steady-state performance measures to the M/M/k/c queueing models for the case $\rho = 1$. The textbook of (Banks et al., 2005) presented closed-form algorithms for M/M/k/c queueing models with balking for the case when $\rho \neq 1$ and where the queue has a statistical equilibrium with steady-state characteristics. They defined the effective arrival rate, λ_e , as the mean number of arrivals per time unit for the entity that enters and remains in the system where, $\lambda_e < \lambda$ for queueing systems with balking. The algorithms are probability- dependent and use Little's equations (3) to obtain the steady-state performance measures required for this study. The steady-state parameters for the

M/M/k/c queue model are; $a = \frac{\lambda}{\mu}$, $\rho = \frac{\lambda}{k \cdot \mu}$ where

a = the ratio of mean arrival rate to mean service rate (utilization factor)

c = the allowable maximum number of entities in the system (system capacity)

λ_e = the mean number of arrivals per time unit.

The steady-state probability of finding no entity in the system is:

$$P_0 = \frac{1}{1 + \sum_{n=1}^k \frac{a^n}{n!} + \frac{a^k}{k!} \sum_{n=k+1}^c \rho^{c-k}} \quad (14)$$

The probability that there are c entities in the system (i.e., the system is full) is:

$$P_c = \frac{a^c}{k! k^{c-k}} P_0 \quad (15)$$

The average length of the number of entities waiting in the queue (queue length) is given by

$$L_Q = \frac{P_0 a^k \rho}{k! (1-\rho)^2} [1 - \rho^{c-k} - (c-k) \rho^{c-k} (1-\rho)] \quad (16)$$

The arrival effective rate is given by:

$$\lambda_e = \lambda(1 - P_c) \quad (17)$$

From Little's formula, the average time spent waiting in the queue is:

$$w_Q = \frac{L_Q}{\lambda_e} \quad (18)$$

Therefore, the average time spent in the system is given by:

$$\bar{W} = w_Q + \frac{1}{\mu} \quad (19)$$

Once again, Little's formula is used to produce the average number of entities in the system:

$$\bar{L} = \lambda_e \bar{W} \quad (20)$$

2. Queueing Simulation Models

This section briefly presents and discusses some studies from the literature of each of the two time-advance methods for simulating queueing systems. Simulation is an effective way to solve complicated practical problems when analytical solutions are

impossible to obtain. Although the queueing systems are typically modeled using discrete event simulation (DES) (Lain & Wan, 2007), a number of researchers in the literature have used discrete time simulation (DTS).

Queueing simulation has been studied for a long time using DES. Each event, such as entity arrival, joining the queue, and start and end service occurs at an instant in time and mark a change in system state. In many queueing studies, researchers have generated simulation models that were able to make accurate predictions of quantities, such as the average waiting time in the system and the long-run mean number of entities in the system. Lain and Wan (2007), Law and Kelton (2000), Banks et al. (2005), Fishman (2001) and many others have demonstrated the use of DES technique for a large number of real-world applications, particularly queueing systems, to obtain the above measures of performance. They described the principles of DES method and its role in improving and understanding the behavior of dynamic systems, as well as the capability of solving complex problems such as queueing models. Researchers such as Kiesling & Krieger (2006) and many others have extended the study of queueing systems simulation with DES by introducing techniques, such as parallel DES, to increase simulation efficiency, and accuracy and reduce the development costs.

Despite many studies in the literature that have implemented the DES technique to evaluate the behaviors of queueing systems, there is an increasing number of studies that have implemented the DTS approach when modeling queueing systems. Neuts (1971) conducted a study that dealt with simulating aspects of the transient behavior of queues in discrete time and developed analytical expressions for the single-serve queue models. The study illustrated the numerical and simulation capability of understanding the transient behavior of a substantial class of queueing models, but did not state the time-step size used in the simulation and its impact on the results. (Reed, 1980) presented a computer simulation technique for teaching queueing theory. The study was limited to the single-server M/M/1 queueing model; however, it aimed to illustrate the differences and similarities between analytical solutions and computer simulation of this type of queueing models using discrete time simulation. (Lozano et al., 2004) developed a discrete time simulation to model passengers' traffic within the departure terminal of

Malaga airport using queueing theory. The study suggested that the fixed time-step size is a user choice, but did not provide guide nor recommendations on step size selection. Yi & Shakkottai (2005) developed a hybrid (fluid and packet) simulation called FluNet to simulate computer networks as queueing models using the DTS technique to reduce the simulation time complexity. The study described how, as the system size increases, the time-step size needs to be smaller to produce accurate results which in turn increased the simulation execution time. Kos, Hess and Hess (2006) developed a DTS model to analyze the behavior of seaport systems. The study presented the model as queueing systems type M/M/1 for the cargo terminal and M/D/1 for the loading terminal where the overall aim is to assist decision makers in improving the effectiveness of the seaport system. At each time interval, the simulation generated random variables to account for arrivals and services where the ships were treated as entities and the terminals as servers in the queueing model. Although the study indicates using a data set from previous years, no direct comparison was exhibited between the simulation results and the available data set for the sake of validation.

Literature on the quantitative comparison of the two simulation time-advance techniques—DES and DTS—is scarce. Reed (1980) presented a study that is part of a student assignment to compare the results' analytical solutions, the DTS and DES solutions for the M/M/1 queue models. The analytical approach, as well as the DTS simulation approach, was presented to the students to be used for two main purposes: reinforcing the concepts of queueing theory and illustrating the differences and similarities between these approaches to queueing models. Unfortunately, the findings were not documented in the study and were left to the students to analyze. Wu and Gong (2001) demonstrated the impact of simulation resolution (time-step size) on the simulation results of a simple M/D/1 queue model (D: deterministic distribution) when simulating traffic on computer networks. They developed a DTS simulation tool for computer network traffic engineering for multi-flow queue and tandem queue then compared the results with analytical and DES results for accuracy.

The study concluded that simulation errors of the mean queue length increased at low resolution (i.e., large time-step size) as utilization increases. The authors failed to

discuss the DES model used in this comparison and its limitations to model this large case of complex network simulations. Furthermore, Krull and Horton (2007, 2009) presented Proxel simulation (DTS type) to provide accurate deterministic results for the steady-state performance measures of a specific queueing system. They claimed that Proxel simulation can solve stiff queueing system problems more quickly than DES. Several experiments' results, such as for M/M/1, G/G/1 and M/G/c/k models were validated by running comparison analyses between Proxel simulation (DTS), analytical solutions and DES. Kolker (2008) conducted a comparison analysis between queueing analytical (QA) solutions and DES in the healthcare system. He illustrated that QA is limited to certain applications and is complicated in queueing systems with balking. After conducting a comparison analysis, the study suggested using simulation approaches, DES in particular, to model complex systems, such as healthcare and other simulation approaches, like DTS. Lee and Strawderman (2009) presented a transform-free analytical approximation algorithm for the system size of M/G/c queueing systems and compared the results to simulation results. The study showed that high accuracy was obtained in the case of low traffic intensity, but failed to illustrate which simulation technique is used in the comparison. Kiesling (2005), in part of his PhD dissertation, conducted a study on queueing systems, the M/M/1 queueing model in particular, by comparing parallel DES (state space decomposition) and parallel DTS (temporal decomposition) results. The findings show that time-parallel queueing system simulations produced accurate results, but the simulation execution time tends to grow exponentially. Our work in this chapter can be regarded as a continuation of these efforts to represent an empirical study on quantitative comparisons between DES and DTS models to provide insights for both areas of simulation and improve result accuracy and efficiency for better decisions.

3. Toll Plaza Systems

The analysis in this chapter extends beyond the theoretical work and simulation of simple queueing systems. The study explores an important application of queueing systems in conjunction with traffic simulation. Highway toll plaza and parking entry/exit

plazas represent a unique category of transportation systems that require special analysis in order to understand their operations. These facilities are some of the most effective means for collecting user service fees for highways, tunnels and bridges and play a major part in controlling traffic. Toll plaza systems consist of a number of service booths in which vehicles arrive from freeways at certain rates and must come to a stop in order to be processed. In this type of system, vehicles typically join a single queue for each booth and wait to pay the fees where service rates depend on type of payment. Efficient sizing of toll plazas, minimizing their cost, and vehicle waiting time are critical concerns to many transportation policy makers. Typically, the two most significant measures of performance (MOPs) of toll facilities are the mean length of the queue, and the mean time vehicles spend in the queue or system (Klodzinski & Al-Deek, 2002a). Toll plaza systems have been the focus of many studies in the literature. In fact, multiple-server queueing analysis has been applied in the planning and evaluation of toll plazas (Hall, 1991). Typically, the problem involves the need to optimize traffic flow through toll plazas by determining the minimum number of tollbooths necessary to operate the system with the smallest amount of waiting time.

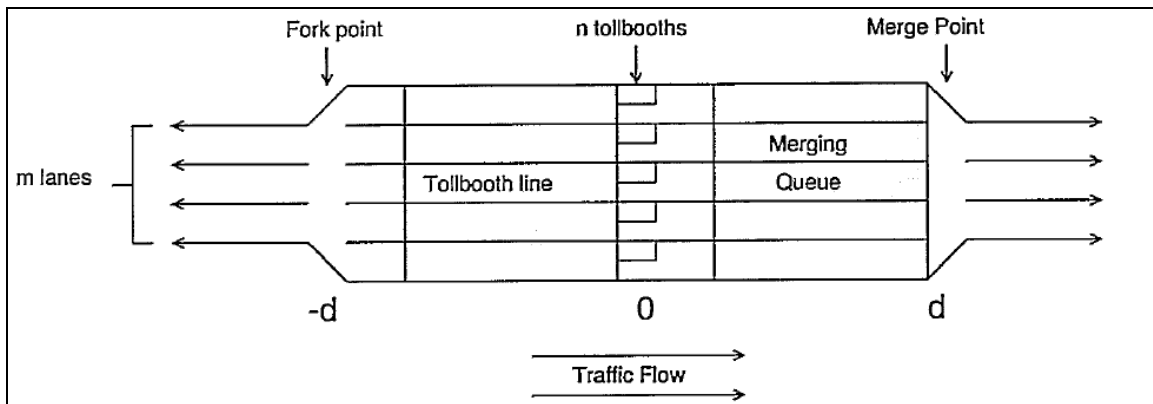


Figure 12. Typical concept design for Toll Plaza System (from Corwin, Ganatra, & Rozenblyum, 2005)

Analytical formulation for this type of model is extremely complex and there is no single set of equations available to estimate the above MOPs. Therefore, traffic simulation models are used instead to enhance planning, design, operation, and management of this type of transportation facility. For example, Mielke (1999) discussed

a discrete event simulation model (VMASC) used in transportation to model traffic and parking plazas. The research expressed the simulation abilities to provide high accuracy on important measures of performance. Similarly, Klodzinski & Al-Deek (2002b) presented a discrete event-based simulation (TPSIM) to model the Dean Toll Plaza on the Orlando-Orange County expressway. Several scenarios were examined and their results showed no significant differences between simulation and actual collected data. Furthermore, some studies used discrete time simulation to model toll plaza transportation systems as an alternative approach to DES. Abdelwahab & Brown (2006) considered the drive-thru system, which is a special case of the toll plaza system, in their study. They built a DTS model to assist decision makers on the best configurations to reduce traffic at a pharmacy.. They briefly described the simulation model without reference to time-step size selection and impact on results. Amos, Galstad & Higgins (2005), Wang (2005), Hoffoss (2005) have generated DTS-based simulations to model toll plazas for various purposes, such as obtaining the optimal number of booths, finding the efficient service/processing times to minimize the mean wait time, predicting more accurate system behavior during high traffic intensity, minimizing the cost of the system (construction and operation costs) and estimating many measures of effectiveness.

Few studies were found directly related to the objective of this dissertation. Galluscio & Douglass (1996) conducted a comparison study between event-driven and time-driven approaches when modeling traffic flow simulation using a parallel simulation method. They showed that parallel simulation helps speed up the simulation execution time and proved to be better than sequential simulation. Their study also illustrated that for many cars in the system, the time-driven approach required longer execution than the event-driven implementation. The research mentions neither the size of time step used in the simulation nor the method of step size selection. Yet another study, by Ceballos & Curtis (2008), developed a discrete time simulation DTS to model toll plaza systems and compared the results with multi-server and multiple single-server queue analytical models. They claimed that discrete event simulation DES models do not incorporate the series of complex traffic interactions and driver behaviors in these types of systems. The comparison between DES and DTS was not conducted in a direct manner; however, the

authors made clear the importance of how the two simulation approaches differ from each other. The purpose of the study in this chapter is to describe and compare the two simulations methods—DES and DTS—currently used to model queueing systems and to explain their differences and similarities.

C. MULTI-SERVER QUEUEING MODELS: M/M/k

Queues are unavoidable aspects of modern life that both humans and non-humans face every day. Multi-server queueing systems are everywhere: standing in lines at grocery stores, at the airport check-in counters and security points, and at toll plaza systems. Many systems exhibit queueing for all kinds of non-human commodities as well. For example, a modern computer system manages queues of computer programs at its central processing, packets wait to be transmitted at each node in a communication network, and items at warehouses wait to be shipped. For these practical reasons, the multi-server queue model M/M/k is the focus of study in this section. On the one hand, the behavior of the system is determined by three performance measures: 1) steady-state average number of entities in the system, 2) steady-state mean time spent in the system and 3) execution time. On the other hand, a comparative analysis is performed using the above performance measures along with a model structural design in both approaches.

1. Simple M/M/k Queueing Systems

Since there are many variations of the M/M/k queue model, this first section begins with the simplest model, M/M/2, to gain insights then develop more complex cases of the model.

a. Model Description

The original queueing system defined above is a continuous-time, discrete-state Markov chain where the system state is the number of people in the system. The M/M/k queueing models used in this chapter follow the basic process assumed by most queueing models. Customers/entities requiring service are generated over time from an infinite population according to a Poisson process. Customer arrivals at the queueing

system occur randomly and independently and are assumed to be at a fixed mean arrival rate λ for all customers. This is equivalent to the assumption that the probability distribution of time between arrivals (inter-arrival times) follows an exponential distribution with mean inter-arrival time $1/\lambda$. Upon arrival, customers join an infinite capacity single queue waiting to get served. The queue discipline in which customers are selected for service is first-in-first-out (FIFO). Once customers join the queue, they are not allowed to leave the system until service is completed. There are k servers in the service facility serving in parallel channels. Each server serves one customer at a time and completes the service with identical independently distributed service times that follow an exponential distribution with mean service time rate μ . The M/M/k model cannot explicitly compute queue length and wait time when the number of customers arriving to be served is greater than the processing capacity, in which case the queue tends to infinity. Therefore, the utilization factor or traffic intensity $\rho = \lambda / (k\mu)$ must be less than one to reach steady-state. The performance measures are not analyzed in transient conditions, where they are greatly affected by initial states, but rather in steady-state conditions. Studying the system in steady-state conditions, where the probability distribution of the state of the system remains unchanged over time, has been the focus of many studies because transient case is analytically very difficult. Assume the system is empty and all servers are idle to run the experiments.

(1) Analytical Model. As illustrated in the first part of section B, analytical closed-form equations exist for a limited number of queueing models. Among these models is the M/M/2 queue model where the system has only two servers ($k = 2$) to serve all customers. The M/M/2 model is specifically used to validate the simulation results for the analysis. Assume that the M/M/2 queue model represents a human-service system such as a bank. Consider that customers arrive according to the Poisson process with mean arrival rate λ and each server service time is approximately exponential with rate μ where $\lambda < (2\mu)$. The system exact steady-state performance measures can be obtained by applying equations (7), (8) and (9).

The mathematical formulation for the M/M/k model becomes more complex as the number of servers, k , exceeds two servers in the queueing system. The

M/M/k queue model analytical closed-form solutions for steady-state performance measures are expressed in Equations (10) through (13) in Section B above.

(2) DES Model. The M/M/k model can easily be simulated with the discrete event approach by using the Event Graph model. (Buss, 2011a) provided a clear description of constructing Event Graphs for such a model and the transformation from the conceptual model (Event Graph) to the implementation in a JAVA-based computer simulation software called SimkitTM (see Appendix B). The reader is advised to refer to (Buss, 2011a) for additional information. The Event Graph model for the M/M/k queue model used in this chapter is shown in Figure 13.

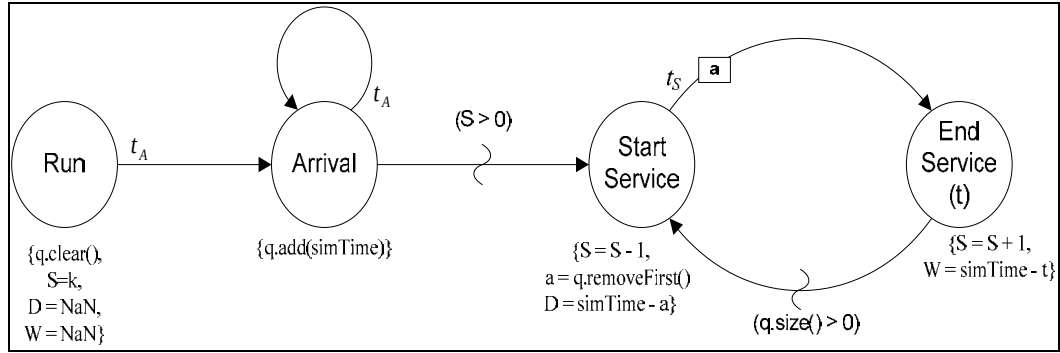


Figure 13. Event Graph for multiple server queue M/M/k (Buss, 2001)

Instead of only keeping track of the number of customers in the queue, a first-in first-out container can be used to keep track of the respective arrival times of the customers. This assists in explicitly tallying the customer delay in queue and the time spent in the system for the multiple server queue model described above. The Run Event schedules an arrival at a time (t_A) with a rate (λ) that follows the Poisson process. Once the Event List executes an arrival, it schedules both another Arrival Event as well as a Start Service Event at different times. A Start Service Event will only occur if there is an available server ($S > 0$). A customer is served by one of the servers for specific service time (t_s) where End Service Event is scheduled. Once End Service Event for a customer is executed, the server is freed and a Start Service Event gets scheduled if there are customers waiting in line ($q.size() > 0$). A global variable, `simTime` is maintained by

the Event List. The value of `simTime`, when an Arrival Event occurs, is the time when the corresponding customer arrived in the system, and the value of `simTime` when the Start Service Event occurs is the time when a customer started service. The delay in queue is simply computed by the difference between the current `simTime` and the `simTime` when that customer entered the system. Similarly, the difference between the `simTime` when End Service Event occurred and when the corresponding Arrival Event occurred is the time in the system. The M/M/k model parameters and state variables are listed below:

Parameters

k is the total number of servers in the system

$\{t_A\}$ is the sequence of (possibly random) times between the arrival of customers in the system and it is called inter-arrival time.

$\{t_s\}$ is the sequence of (possibly random) service times of each successive customer and it is called service times.

State Variables

q is a fifo container holding the arrival times of each respective customer in the queue. Initially it is empty.

S is the total number of available servers (between 0 and k). The initial value is k

D = delay in queue for the last customer (initially undefined)

W = time in the system for the last customer (initially undefined).

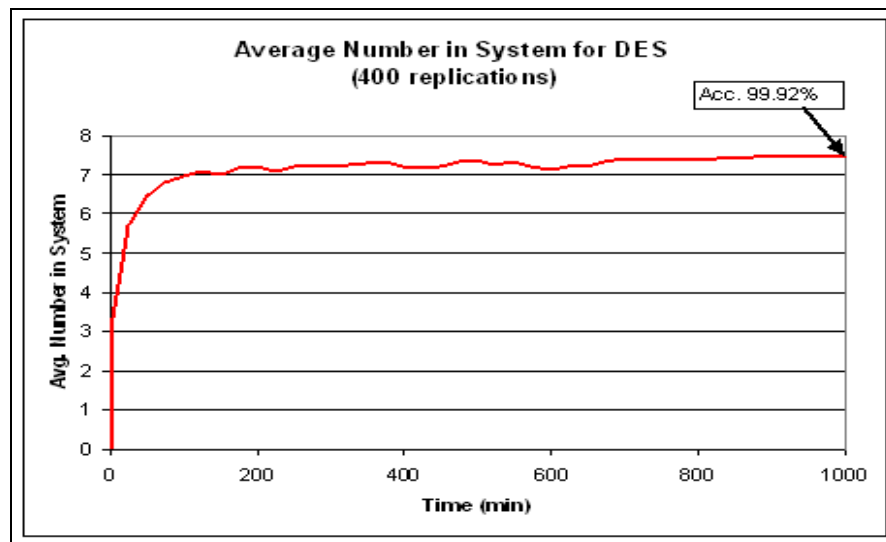
The first measure of performance (the average time spent in the system) is computed by averaging out the time spent by all customers in the system ($\sum W_i / \text{number of customers}$). To estimate the second performance measure, for that is the average number of customers in queue (or in the system), it is necessary to collect a time-average on the number of items in q rather than the value itself.

(3) DTS Model. Although queueing models are traditionally modeled using the DES approach, occasionally a time-step approach has been used as shown in the literature in section B. Two methods were used to create a time-step version of the M/M/k queue. The first was based on arrivals being considered as successive Bernoulli trials at each time step. The second method used assumed arrivals at each time step as a batch of customers. The first approach requires the observation that the exponential distribution used for inter-arrival times and service times in the DES model can be approximated by the appropriate geometric random variables, which in turn can be generated by successive Bernoulli trials. Thus, for example, at each time step an arrival will occur with probability p_A . For a time step of Δt , the inter-arrival distribution is $\Delta t \cdot N$ where $N \sim \text{Geom}(p_A)$. The mean is $\Delta t / p_A$ so a mean inter-arrival time of $1/\lambda$ would require $p_A = \Delta t \cdot \lambda$. The service times are handled in a similar manner where $p_s = \Delta t \cdot \mu$. The second approach that deals with batch arrivals handles the service time in a similar manner, as previously described; the only difference is that the number of arrivals is randomly generated with a Poisson distribution at each time step to account for batches. The purpose of this method is to eliminate any bias in the arrival process, as the Bernoulli method accounts for one and only one arrival at each time step. Bernoulli approach can cause problems with a larger time-step size in which there are fewer arrivals than might actually be observed. Once batch arrivals occur, each customer is added to the queue individually to keep the system operating in the same manner as before (see Appendix B).

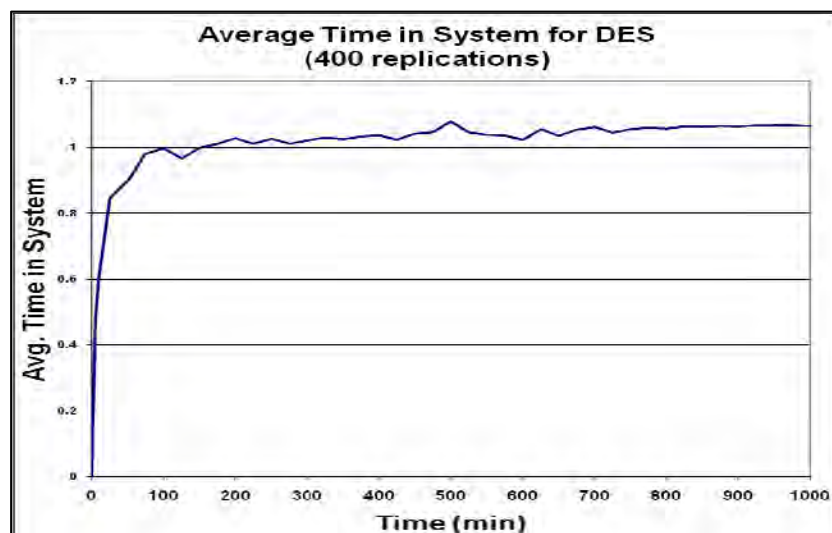
b. Results and Comparative Analysis

The simulation models were executed starting in an empty and idle state. For the baseline system to study, we chose inter-arrival rate $\lambda = 7$, service time rate $\mu = 4$, and number of servers in the system $k = 2$, resulting in a traffic intensity of $\rho = 0.875$. Analytical solutions were calculated producing $\bar{L} = 7.467$ customers and $\bar{W} = 1.067$ minutes. Later, higher intensities will be considered. Although normal analysis in this context dictates having a “warm-up” period, for this study we wanted to examine the

differences in how steady state was achieved, and collected output for all customers starting from the first initial conditions. For 400 independent replications, the DES results were averaged and, as shown in Figure 14, the steady state is approached fairly rapidly. By time 1000, the modeler can be fairly certain that steady-state has been approximately reached. A great level of accuracy of 99.92% in comparison with the exact solution was achieved at the steady-state 1000 minute point for the average number in system. The average time in system, that showed similar behavior, is illustrated in Figure 14-b.



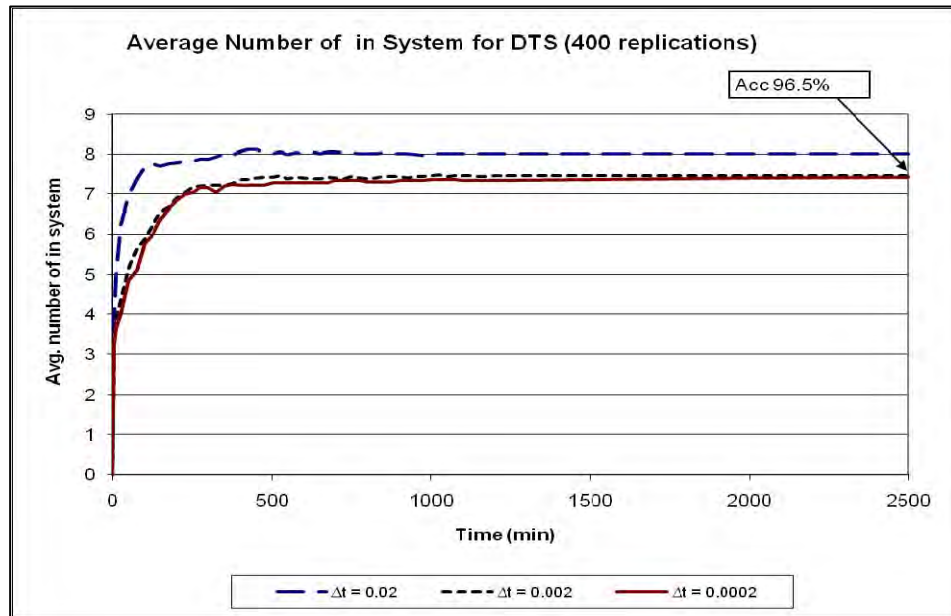
a)



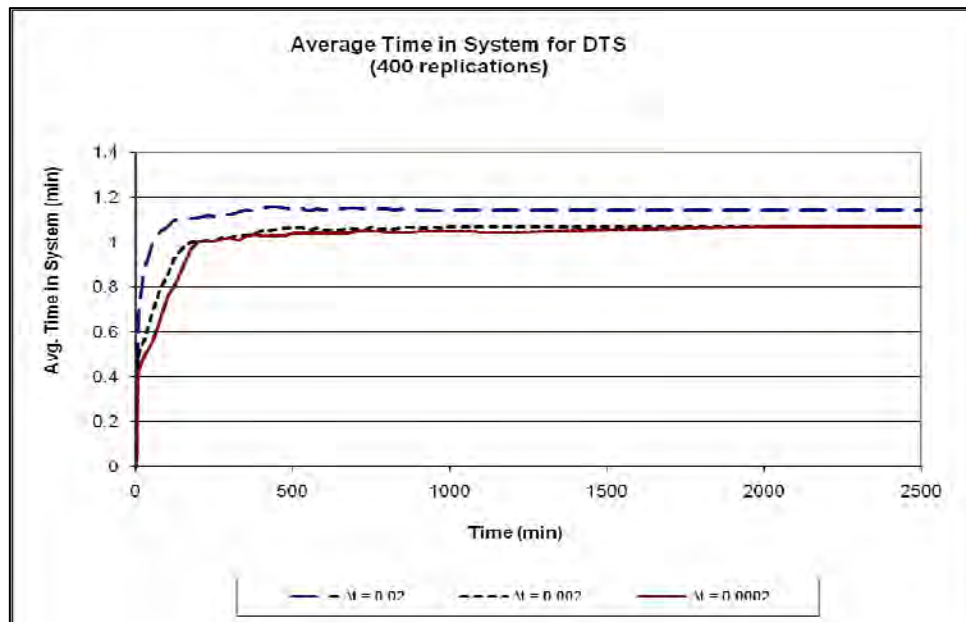
b)

Figure 14. DES model output measures: a) average number in system behavior b) time in system behavior.

The DTS model was then run in a similar manner for three different sizes of time step, 0.02, 0.002, and 0.0002 as shown in Figure 15.



a)



b)

Figure 15. DTS model output measures with different time steps: a) average number in system behavior b) time in system behavior.

Figure 15 results show that even though the qualitative behavior is similar to the DES output for very smaller time steps ($\Delta t = 0.002$ and 0.0002), for the larger time step there is a substantial gap, even when the simulation was run for a substantially longer time. With a larger time-step, steady-state is reached faster than a smaller time-step size but with a larger error. Although at small time-step size $\Delta t = 0.0002$ accuracy level showed 96.5%, it was still less than the results obtained by the DES method.

In order to have more reliable steady-state estimates, the DES and DTS models were then run even longer, for 3000 time units, with 1000 replications, and confidence intervals were computed for two time steps. The results are summarized in Table 4.

Results of 1000 Repl.	Exact	DES		DTS			
				$\Delta t = 0.02$		$\Delta t = 0.002$	
		Mean	95% CI	Mean	95% CI	Mean	95% CI
Avg. # Cust. in System	7.466	7.449	(7.374,7.524)	8.088	(8.037,8.139)	7.47	(7.426,7.514)
Avg. Time in System	1.066	1.065	(1.064,1.067)	1.157	(1.150,1.164)	1.068	(1.062,1.074)

Table 4. DES, DTS, and exact solutions for the M/M/2 queue

As can be seen in Table 4, the DES model produced highly accurate estimates, whereas for the DTS models, the smaller size resulted in somewhat less accurate estimates (although the confidence intervals did include the true means), whereas the larger time step produced estimates that were substantially off, with confidence intervals that did not cover the true means.

The situation deteriorates even more as the time step is increased. Figure 16 shows the percent error for the average number in the system as the time-step size increased. As expected, the error grows considerably as time-step sizes as large as 0.1 are used.

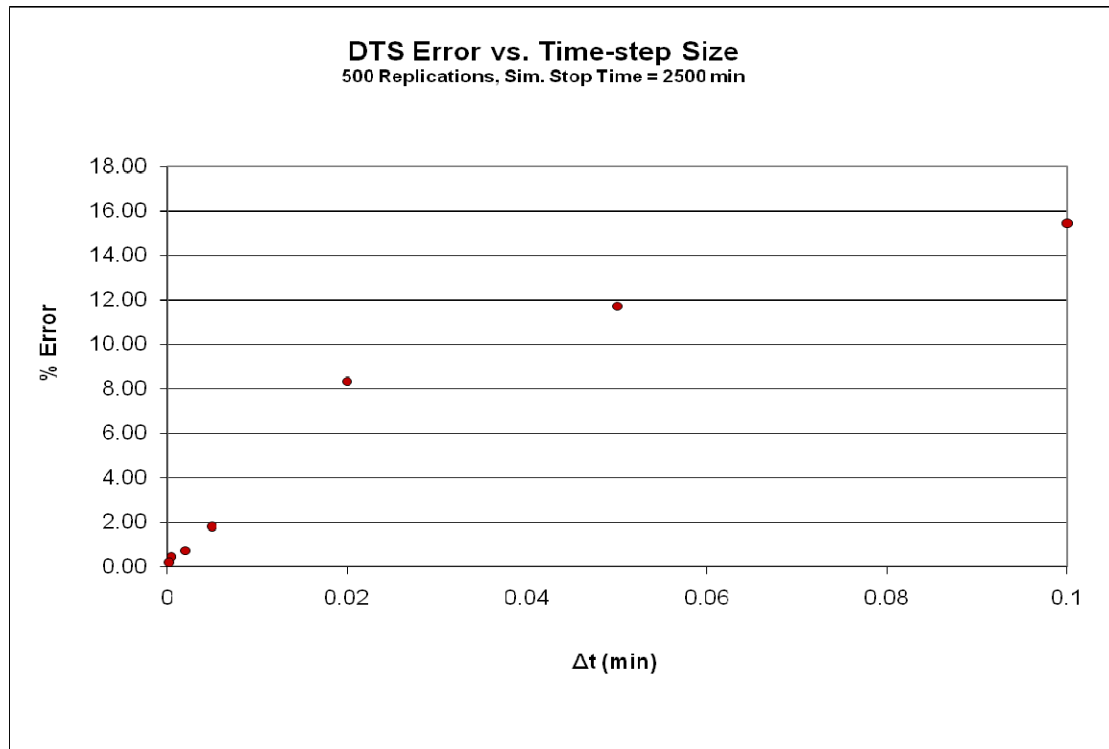


Figure 16. Percent error as a function of time-step for DTS models

Another matter critical to modelers is simulation efficiency. Figure 17 shows the impact of time-step size on the simulation execution time. The simulation execution time dramatically increases in an $1/\Delta t$ pattern as the time-step size decreases to small values. It is observed that the DTS approach increases the problem complexity at smaller time-step sizes, causing modelers to negotiate a trade-off between simulation efficiency and accuracy.

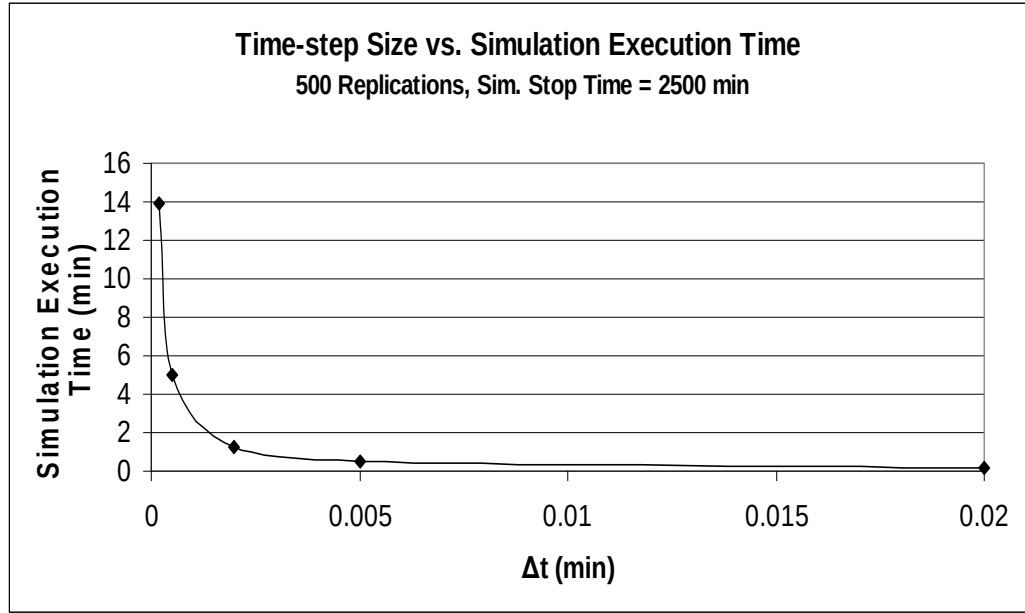
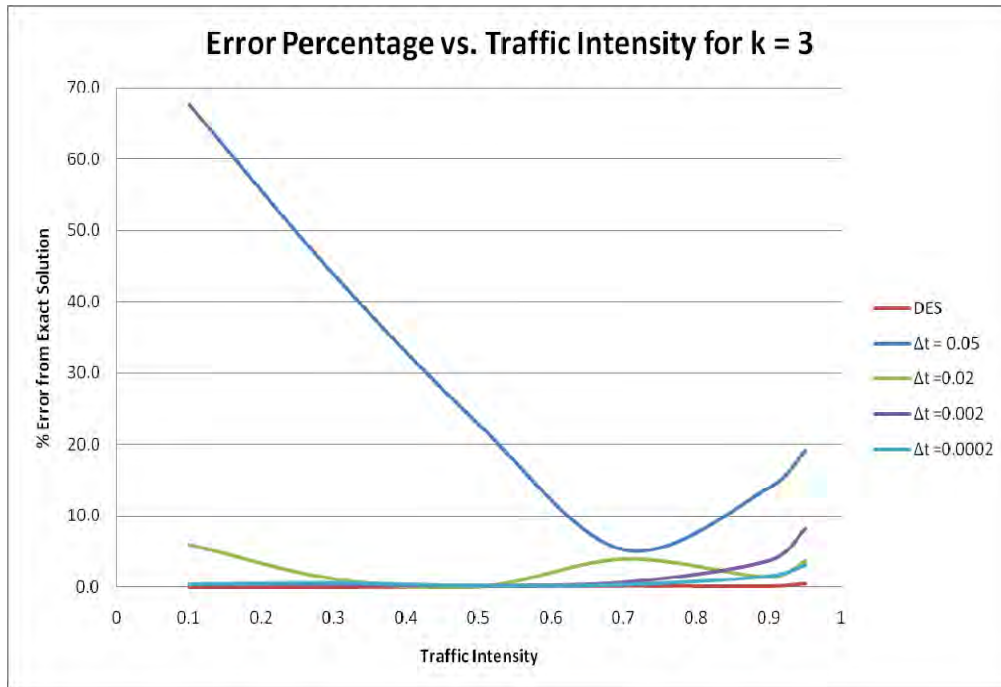


Figure 17. Simulation execution as a function of time step for DTS models

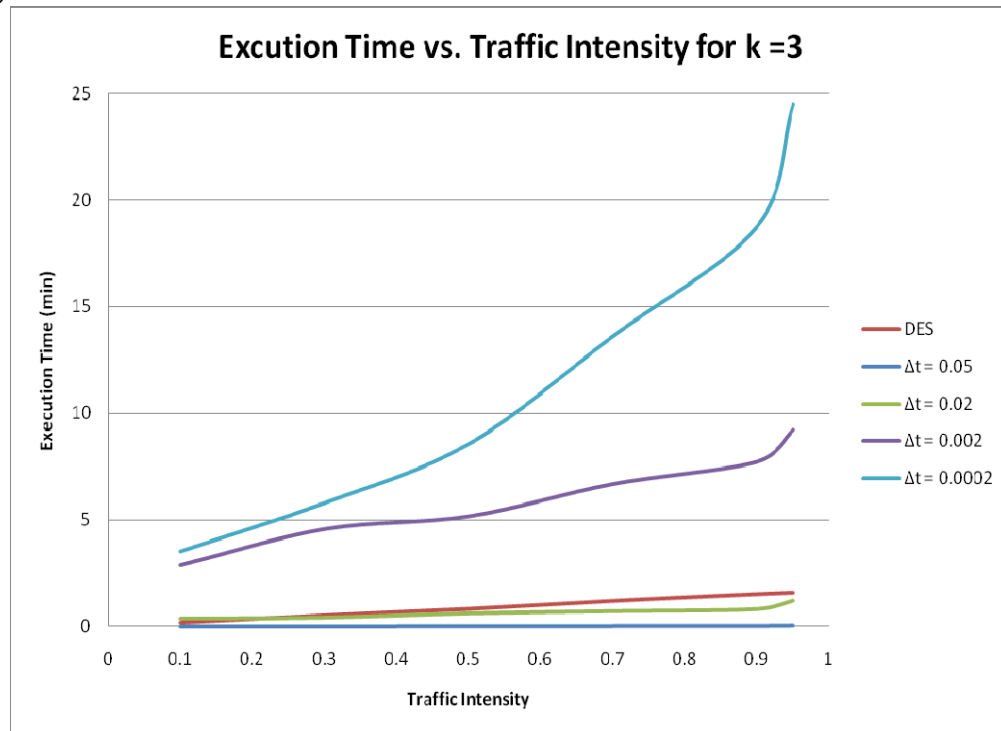
Since our measures of performance are observed at steady-state conditions, special care is required to eliminate initial conditions effect on the output measures. To eliminate this bias, the replication/deletion approach with the Welch method is used to specify the truncation limit of the observation (Alexopoulos and Seila 1998). For both measures of performance $\bar{L}$ and $\bar{W}$ we first specified the length of each observation to be 10,000 observations. Different numbers of customers (e.g., 500, 1000, 1500, and so on) are used each time with a 95% confidence interval to observe the value at which the system approaches a steady state level. A value of 500 observations was found to be a suitable truncation limit to avoid bias in the results, and those observations were removed from the proceeding simulations. A set of M/M/k experiments was run by varying the number of servers and the traffic intensity to test the impact of traffic intensity on the system (Lee & Strawderman, 2009). For these, the service time was fixed at $\mu = 4$, with the number of servers k varied between 3 and 5 and traffic intensities running from 0.1 to 0.95. Each simulation was run for 1500 time units with 400 replications and the results of both DES and DTS were compared.

		DES		DTS							
				$\Delta t = 0.05$		$\Delta t = 0.02$		$\Delta t = 0.002$		$\Delta t = 0.0002$	
k	ρ/k	Err%	ExeT	Err%	ExeT	Err%	ExeT	Err%	ExeT	Err%	ExeT
3	0.1	0.0	0.166	67.7	0.015	6.0	0.35	0.3	2.85	0.3	3.5
	0.3	0.0	0.51	43.8	0.02	1.2	0.4	0.3	4.55	0.5	5.77
	0.5	0.1	0.81	22.8	0.03	0.1	0.6	0.2	5.15	0.1	8.54
	0.7	0.2	1.19	5.3	0.033	4.0	0.7	0.6	6.67	0.3	13.6
	0.9	0.1	1.51	14.0	0.04	1.4	0.8	3.7	7.72	1.6	18.75
	0.95	0.5	1.58	19.2	0.05	3.7	1.2	8.2	9.22	3.2	24.5
5	0.1	0.0	0.28	97.4	0.025	5.4	0.38	0.6	3.24	0.2	4.8
	0.3	0.4	0.83	96.2	0.03	3.2	0.53	0.1	5.7	0.5	8.88
	0.5	0.0	1.42	55.4	0.042	2.0	0.71	0.0	8.61	0.1	14.65
	0.7	0.1	1.97	22.5	0.053	1.1	0.93	1.4	11.9	1.6	19.43
	0.9	0.1	2.58	3.8	0.06	11.9	1.12	8.1	13.5	0.7	26.1
	0.95	0.4	2.66	10.3	0.085	7.0	1.2	6.8	14.94	2.0	38.36

Table 5. Percent error with different values of k and ρ for the geometric arrival approach (“ExeT” = execution time in minutes)



a)



b)

Figure 18. Plots of percent error (a) and execution time (b) over multiple values of traffic density for k = 3

Looking at the system from a low-resolution perspective, we observe similar behaviors when using DES and DTS. The percent error between simulation results and exact solutions increases as a response to an increase in traffic intensity. With the rapidly changing environment for computing, simulation execution must be taken with a grain of salt as an absolute measure of performance. However, since all simulations were executed on the same computer, the relative performance gives useful information. The results illustrate that as the traffic intensity increases, the simulation execution time increases. Investigating the effect of time step specifically, we see that errors increase as a function of time-step size monotonically.

In particular, Table 5 and Figure 18 (a) even though the smallest time step gives relatively accurate results comparable to the DES model, there is a substantial performance penalty more than an order of magnitude in some cases, in which the familiar effect of execution time increases at a rate of $1/\Delta t$ compared to the linear increases observed in DES. Conversely, Figure 14 (b) shows that for large time steps, the DTS model executes quite rapidly (e.g., $\Delta t = 0.05$ and 0.02), much faster than the DES model. However, the errors in the output estimates for these time steps certainly outweigh any performance benefits.

		DES		DTS			
				Geometric Dist (Arr)		Poisson Dist (Arr)	
				$\Delta t = 0.002$		$\Delta t = 0.002$	
k	ρ/k	Err%	ExeT	Err%	ExeT	Err%	ExeT
3	0.1	0.0	0.166	0.3	2.85	0.0	0.35
	0.3	0.0	0.51	0.3	4.55	0.1	0.5
	0.5	0.1	0.81	0.2	5.15	0.4	0.6
	0.7	0.2	1.19	0.6	6.67	0.6	1
	0.9	0.1	1.51	3.7	7.72	1.5	1.2
	0.95	0.5	1.58	8.2	9.22	3.9	1.4
5	0.1	0.0	0.28	0.6	3.24	0.0	0.4
	0.3	0.4	0.83	0.1	5.7	0.1	0.6
	0.5	0.0	1.42	0.0	8.61	0.4	0.9
	0.7	0.1	1.97	1.4	11.9	0.4	1.6
	0.9	0.1	2.58	8.1	13.5	1.1	1.9
	0.95	0.4	2.66	6.8	14.94	1.9	2.3

Table 6. Percent error with different values of k and ρ for the Poisson (Batch) arrival approach and Geometric arrivals at $\Delta t = 0.002$ (“ExeT” = execution time)

Table 6 illustrates similar conclusions as observed in Table 5; however, all simulations were executed in less time than with the geometric approach. Also, it was noted that the errors in the output estimates for the batch approach were much lower than the geometric approach, but considerably higher than the DES model. The experiment was then run for smaller time-step size $\Delta t = 0.0002$. The error was reduced to 1.6% for $k = 3$ and 1.0% for $k = 5$; however, the execution time jumped to 12.3 minutes and 20.2 minutes respectively. An interesting point is that although the DTS model with Poisson type arrival increased the simulation accuracy and efficiency in comparison with the Geometric type arrivals, DES results are still better in terms of accuracy as well as efficiency.

2. Multi-server Queueing Model with Balking: M/M/K/C

Practical queueing systems with balking and reneging have been applied to many real-life cases such as impatient customers in hospital emergency rooms, restaurants, telephone services and inventory systems with new goods (Zhang, Yue, & Yue, 2005). This section discusses the simulation analysis used to model multi-server queueing systems with balking, known as the M/M/k/c queueing model.

a. Model Description

Multi server queueing systems with balking have the same characteristics as the M/M/k queueing model described previously. There are different ways to balk (not enter) in queueing systems; one such way is with a probability that follows distribution from observed data. Two types of customer balking are considered here, balking due to maximum number of customers in the queue e.g., an ATM queue and maximum number of customers in the system e.g., at a walk-in clinic system (Clarke, 2005). In the first type, balking occurs when an entity arrives and decides to leave the system without entering the queue because queue capacity has reached the maximum value c . This method is useful when modeling a multi-stage (multiple process) system where balking happens before each entity processes to the next stage based on the number in queue. The second type of balking takes place in situations where the entity arrives and decides not

to enter the queue when the number of entities in the system reaches maximum c . This method works in cases where entities in the system are part of the waiting process. The model assumptions are similar to the M/M/k models above with one exception, when there is a finite amount of waiting room in the queue, c . An arriving customer who finds fewer than c customers in the queue enters the queue. However, an arriving customer finding c number of customers already in the queue leaves the system without joining the queue and never returns. Assume the system is initially empty and all servers are idle to run the experiments.

(1) Analytical Model. Analytical solutions for this class of queueing models are extremely limited. The problem becomes more complex as the number of servers increases, queue capacity is amplified and balking strategy becomes complicated. The analytical closed-form expressions for the M/M/k/c queueing model with balking are described in equations (14) through (20). The purpose of using such models with existing closed-form analytical solutions is to validate simulation results.

(2) DES Model. Methodology similar to the one used to simulate the M/M/k models with the DES approach is also applied in this case. An Event Graph model was constructed to illustrate the conceptual model of the problem based on the procedure provided by (Buss, 2011a) in which the Event Graph was implemented in SimkitTM (see Appendix B). Figure 19 shows the Even Graph for the M/M/k/c queue model with queue/system capacity balking.

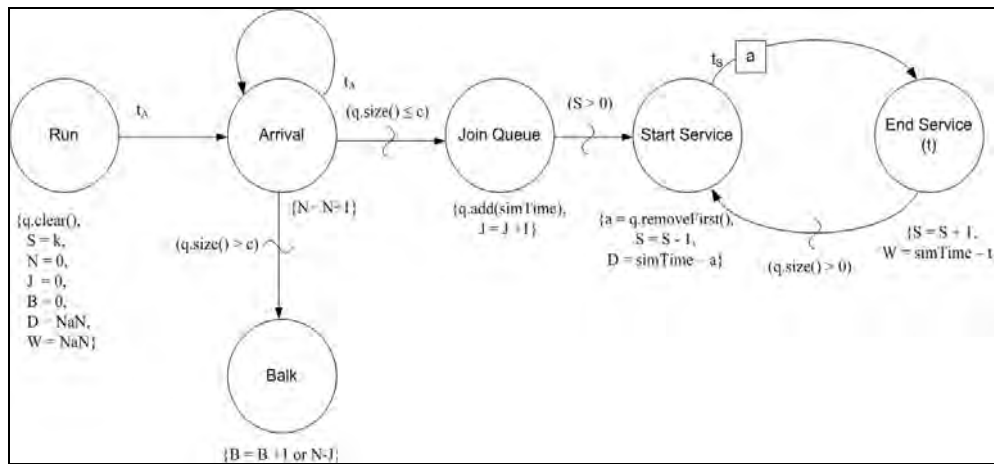


Figure 19. Event Graph for multiple server queue M/M/k/c with balking

The description follows through from a previous M/M/k Event Graph model. In addition, we have several state variables, the number of customer who arrived at the system N , the number of customers who join the queue to get service J , and the number of customers who balk from the system and never return B . On one hand, a Balk Event is scheduled by an Arrival Event if the queue capacity is exceeded (i.e., $q.size() > c$). On the other hand, Arrival Event schedules Join Queue Event in which customers enter the queue only if the amount of customers in the queue is less or equal to the queue maximum capacity (i.e., $q.size() \leq c$). The rest of the model is similar to the multiple server queue model described above. Balking due to the number of customers in the system exceeding a maximum limit is handled in the same manner.

Parameters

Similar to the previous case.

State Variables

Similar to the previous case with the addition of the followings:

N is the number of customers who arrive at the system (Initial value is 0)

J is the number of customers who join the queue (Initial value is 0)

B is the number of customers who balk (i.e., leave the system with no returns), (Initial value is 0)

(3) DTS Model. Batch arrival method is used, as in the previous case, with the M/M/k queue model where arrivals are randomly generated with a Poisson distribution at each time step. The service time is handled in the same manner in the M/M/k model, however, customers enter the queue based on a number in queue maximum capacity (in the case of balking due to queue capacity) or a probability that relates to the number in the system. Join queue probability is computed by normalizing the number of customers in the system $p_{join} = 1 - \left(\frac{\text{Number In System}}{c} \right)$. Once the probability is satisfied, customers enter the queue and follow the same procedures as in the M/M/k queue model (see Appendix B).

b. Results and Comparative Analysis

The simulation models were executed at an empty and idle state. For this system, we assumed the following parameters: inter-arrival time rate $\lambda = 15.5$, service time rate $\mu = 4$, and number of servers in the system $k = 4$, resulting in a high traffic intensity of $\rho = 0.97$, which helps evaluate the system under worst-case scenarios. Balking due to queue maximum capacity was first examined by varying the queue capacity c from 5 to 200. The simulations 100 replications were run for 1500 time units and the results of DES and DTS are compared along with analytical solutions in Figure 20.

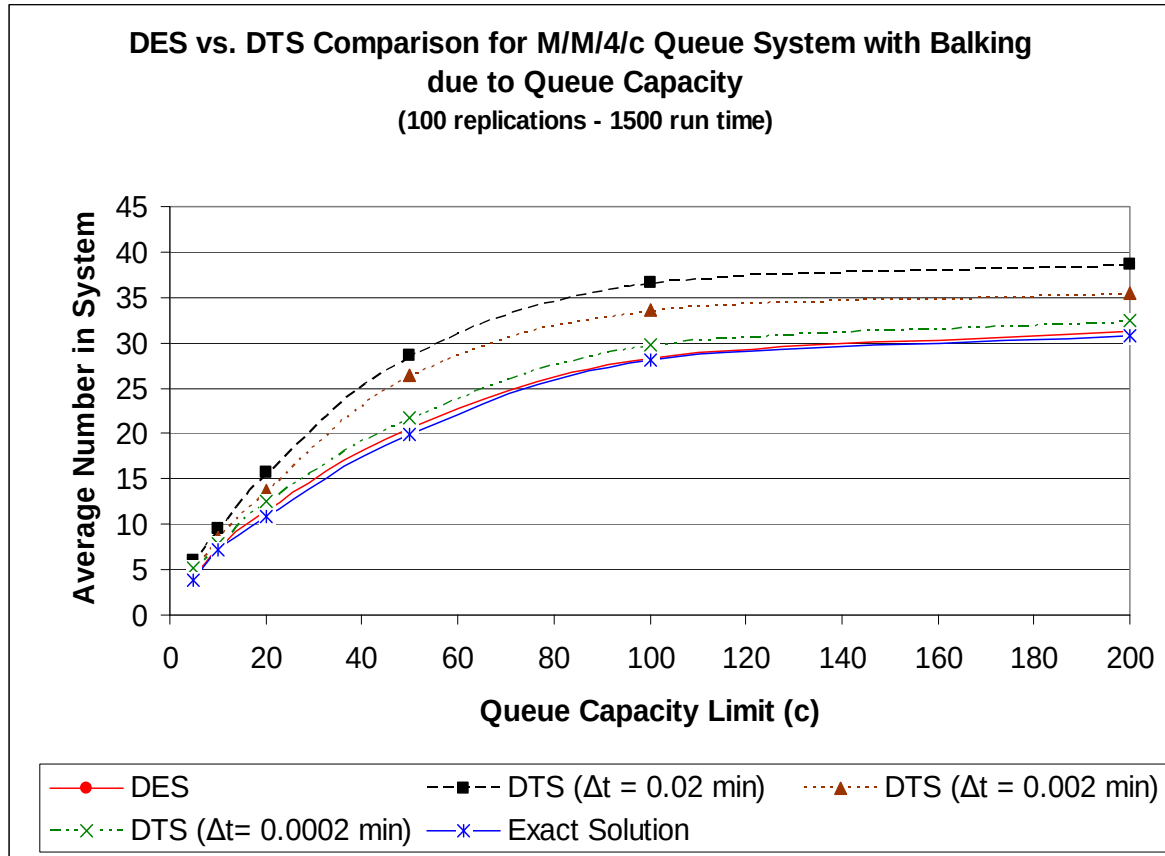


Figure 20. Output measures for DES, DTS and analytical solution for M/M/k/c queue model with different queue capacity.

Figure 20 shows that DES results are highly accurate for the exact solutions, where the DTS accuracy varies with the time-step size. Both DES and DTS show the same behavior of the system in response to balking that is the average number of customers in the system is proportionally related to the queue capacity. A queue capacity of $c = 100$ seemed critical in both simulations where the average number in the system is highly impacted. However, the provided details on the average number in the system can really help in the design of the system. As expected, when the time-step size is reduced to $\Delta t = 0.0002$ minutes (very small), the DTS results began to converge with the DES results and gain accuracy. The simulation executed in 3.4 minutes when the above time-step size was used, where the DES ran in just under a minute at 0.7 showing an advantage efficiency of approximately 5 times over the DTS approach.

Parameters similar to the first scenario were used in the second scenario. However, this time balking is due to the number in the system instead of in the queue. Therefore, the capacity of the number in the system was varied from 5 to 300. Figure 21 shows the results from running both approaches' simulations and compared to an analytical solution.

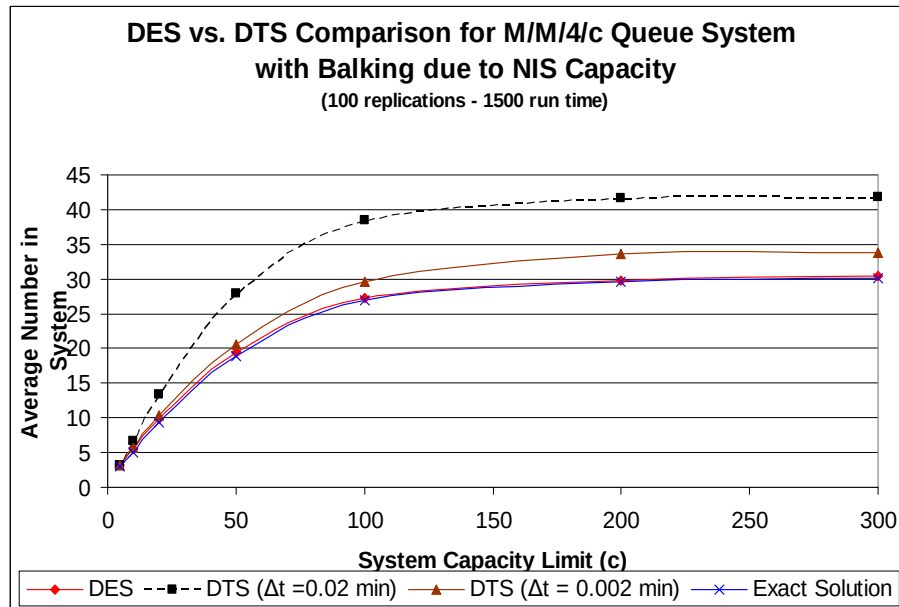


Figure 21. Output measures for DES, DTS and analytical solution for M/M/k/c queue model with different NIS capacity.

Results similar to the pervious case were observed in the second scenario with the exception that the DTS model showed approximately the same efficiency as the DES model for the smallest time-step size $\Delta t = 0.0002$ minutes. This is due to the design of the scenario in which the DTS method can easily check against the number in system at each time slice in comparison with the number in queue.

D. QUEUEING APPLICATION: TOLL PLAZA SYSTEMS

1. Model Description

One important class of queueing systems that most people encounter in their daily lives is in transportation service systems. One particular example this section covers is toll plaza systems, where vehicles are the customers and tollbooths are the servers. Toll plazas have been the subject of many recent studies in which this type of transportation system is experimented with to provide solutions to the traffic flow problem. Toll plaza problems centered on studying the behavior of this dynamic system to find optimal designs and policies to operate the system with efficient traffic flow and number of booths at minimum delays and costs (Wang, 2005). This section investigates a simple version of toll plaza systems in which a DES and DTS simulation comparison analysis is carried out for the purpose of our study. For the objective of minimizing travel time through toll plazas, typical measures of performance for this system are the average number of vehicles in the system for a period of time and the average time vehicles spend in the system. The two measures mentioned are used as simulation accuracy measures, where the simulation execution time is taken as an efficiency measure in the analysis. The method used to model toll plaza systems involves vehicles entering the facility by joining multiple queues waiting to get served by one server at each tollbooth. The dynamic behavior of Toll plaza system is agreed to follow a multiple single-server queueing system in which each toll booth is considered as an independent M/M/1 queue model (Morrow, 2005; Wang, 2005; Hoffoss, 2005). The assumptions considered below describe the toll plaza model under study.

1. The highway is generally free flowing on either side of the toll plaza to separate highway congestion from toll plaza.
2. Traffic flow at the toll plaza is much less than the flow on the mainline freeway.
3. The traffic flow arriving at a toll plaza can be modeled with a Poisson distribution, that is inter-arrival times must follow exponential distribution (Morrow, 2005).
4. Traffic flow rate is constant for the period under study.
5. All vehicles are considered homogenous (i.e., trucks $\approx$ trailers $\approx$ cars $\approx$ motorcycles).
6. Each toll booth is assigned one and only one queue.
7. Vehicles select the shortest queue, but lowest queue index is selected in the case of ties.
8. Once a vehicle joins a queue it remains in the same queue until service is completed at the booth (i.e., no queue switching).
9. All tollbooths are considered non-express electronic homogenous despite payment methods (i.e., cash, credit card, checks).
10. All booths are identical and the average service time to pay the toll at any of them is equal and follows exponential distribution.
11. Electronic toll collection is not considered as service time is considerably negligible.
12. The number of toll booths is greater than the number of highway lanes (lanes open up to equal number of booths).
13. At some point most vehicles must stop having either reached the booth or when a vehicle is in front.
14. Vehicle speeds when merging in and out of the toll system are not considered.

15. The merging phase after the tollbooths is not part of the study.
16. Drivers' different behaviors are not considered.
17. Queueing area is the area where the roadway has widened to a number of lanes equal to the number of booths.
18. The amount of time a vehicle spends between entering the queueing area and stopping at the end of the tollbooth line is constant regardless of the length of the lines or the booth the driver selects.
19. One traffic direction is considered in this analysis, and only section C and D of Figure 22 are under examination.
20. The system is operating under heavy traffic during peak hours (e.g., 0600 AM to 0830 AM or 0400 PM to 0600 PM)
21. Determining the optimal mix of booth types is not considered in this study.

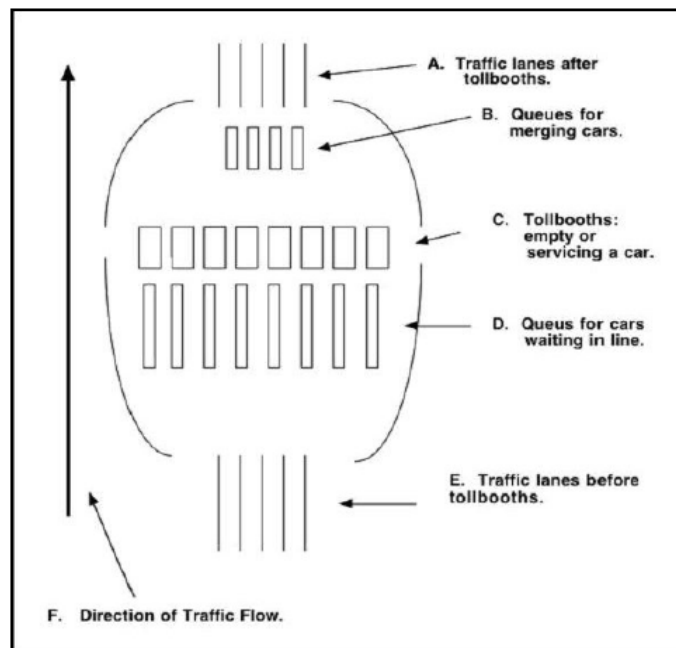


Figure 22. Toll plaza system (Amos, Galstad, & Higgins, 2005)

(1) DES Model. To develop the simulation model of the above toll plaza system using the DES approach, Event Graph conceptual model was first constructed then implemented in the Simkit software tool in the same manner described in previous sections. Figure 23 shows the Event Graph used to model the above toll plaza system in discrete event simulation.

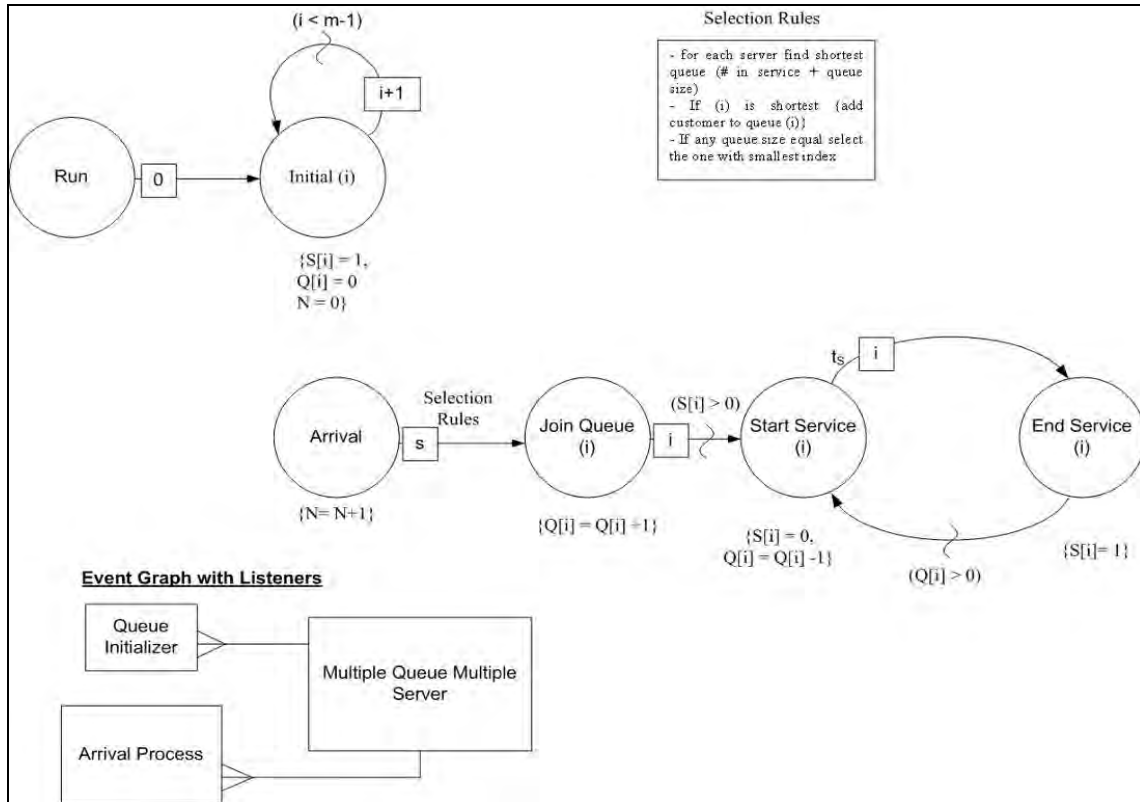


Figure 23. Event graph model of toll plaza with listeners

The Event Graph in Figure 23 is a modified version of the M/M/k Event Graph by using array state variables ($S[i]$ and $Q[i]$) to account for each tollbooth queue independently. Listeners were used to simplify the implementation of the model in SimkitTM and increase simulation efficiency.

(2) DTS Model. An approach similar to the M/M/k model was used to build the simulation model for the DTS approach. Batch arrival was selected to increase the efficiency of the simulation. At each time step, once arrivals occur, the queue selection rule is implemented to assign vehicles to each tollbooth queue. Amos, Galstad

& Higgins (2005), Wang (2005) and Hoffoss (2005) looping over all queues, toll plaza booths' services were then accounted and statistics were gathered (see Appendix B).

2. Results and Comparative Analysis

The simulation models were executed starting with empty and idle states. Unfortunately, simulation accuracy is not a direct measure in this part of the study since there are no exact solutions to compare against. Therefore, neither of the two simulation approaches could be claimed to be accurate, so all that can be shown are the differences between the approaches. Different scenarios were considered including variable service rates and variable numbers of booths. The first scenario was to test the impact of service time rate on vehicle waiting time in the toll system. This is a typical test used to assist decision makers to set strategies to improve the toll system operation. The typical question for this scenario is: What is the optimal booth service time rate required to reduce traffic delays to < 3 min?

The vehicles arriving rate λ was fixed at 25 vpm (vehicle per minute), five toll booths were considered to be operating, the simulations ran for 300 minutes for each of the 50 replications, and the service time rate μ was varied between 2 to 8 vpm. All of the above parameters are assumptions based on real toll plaza data (Wang, 2005).

Figure 24 shows the resulting average time spent in the system for various service times. Both DES and DTS illustrated that 5 vpm is the required service time rate to reduce toll plaza traffic contingency dramatically.

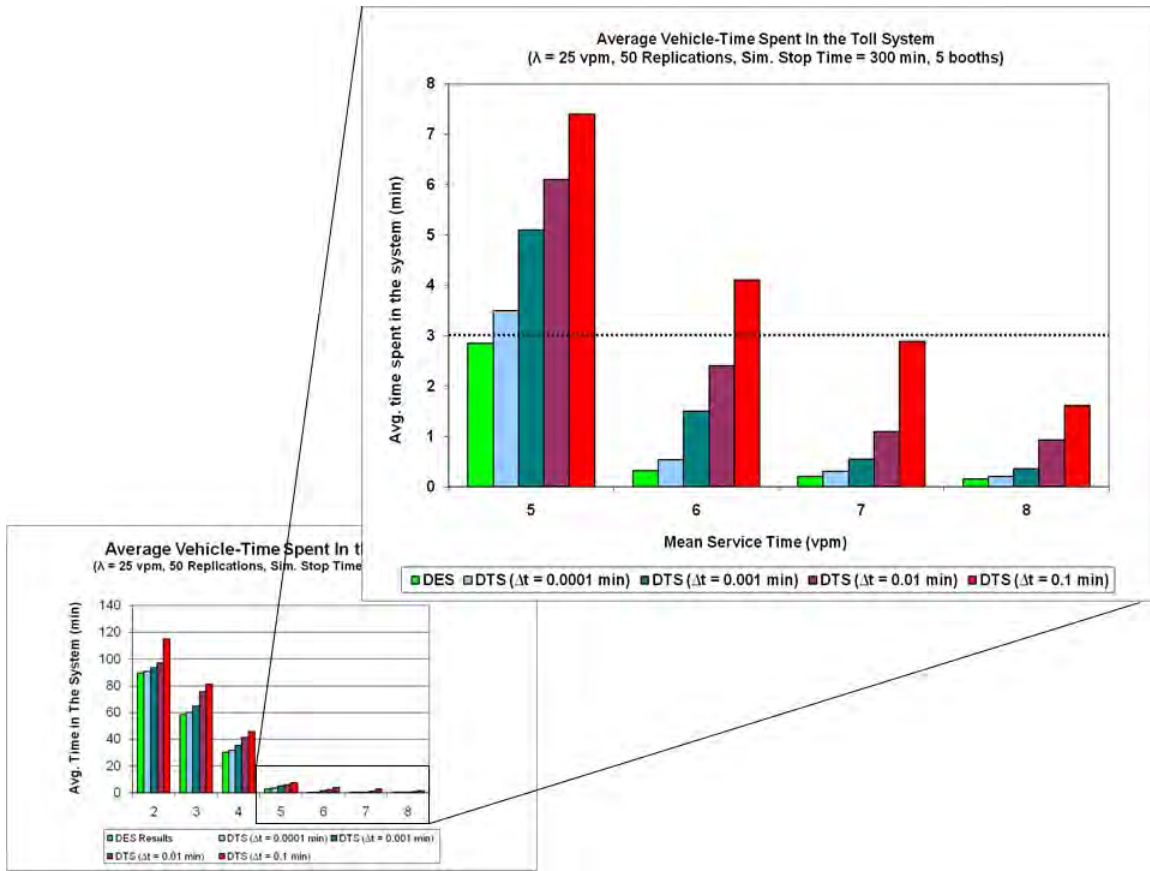


Figure 24. Toll plaza system wait delay for various service times.

Typically, many transportation system-related studies point out that the objective is to acquire an average delay time of less than 3 minutes when designing large toll plaza systems. Figure 24 clearly shows the advantage of the DES approach to highlight that a service time rate of 5 vpm is sufficient to arrive at a conclusion. However, the DTS approach showed different results with large time-step sizes. For example, DTS at $\Delta t = 0.1$ suggested a service time rate of 7 vpm is adequate to accomplish the objective. Even at $\Delta t = 0.0001$, the DTS approach recommended 6 vpm to be the decision. The method of increasing service time rate is not discussed in this study; however, any increase is considered proportional to the operation cost of the toll system. It is clear from the above observations that DES and DTS produce different results; furthermore the recommendations made by DTS would have significant consequences for the decision makers.

Simulation execution times were compared between DES and DTS approaches in Table 7, where DES demonstrated its efficiency advantage over DTS. At smaller time-step sizes DTS results began to converge with DES results and show reduction in the differences.

Type of Approach	Simulation Average Execution Time (min)	% DES vs DTS Differences	Time Comparison
DES	0.2		1 times
DTS with $\Delta t = 0.1$ min	0.05	> 100	- 0.2 times
DTS with $\Delta t = 0.01$ min	1.24	> 100	6 times
DTS with $\Delta t = 0.001$ min	3.99	77.0	20 times
DTS with $\Delta t = 0.0001$ min	6.16	23.3	31 times

Table 7. DES and DTS execution times and accuracy for toll plaza with various service time rates.

The second scenario was examining the impact of the number of toll booths on the time spent in the system. The vehicles arriving rate λ was fixed at 40 vpm, the service time rate μ was fixed at 4 vpm (i.e., 15 seconds per vehicle), the simulations ran for 300 minutes for each of the 50 replications, and the toll booth number was varied between 4 and 16. Figure 25 illustrates the simulation results of DES and DTS models for the toll plaza system under study at various booth numbers.

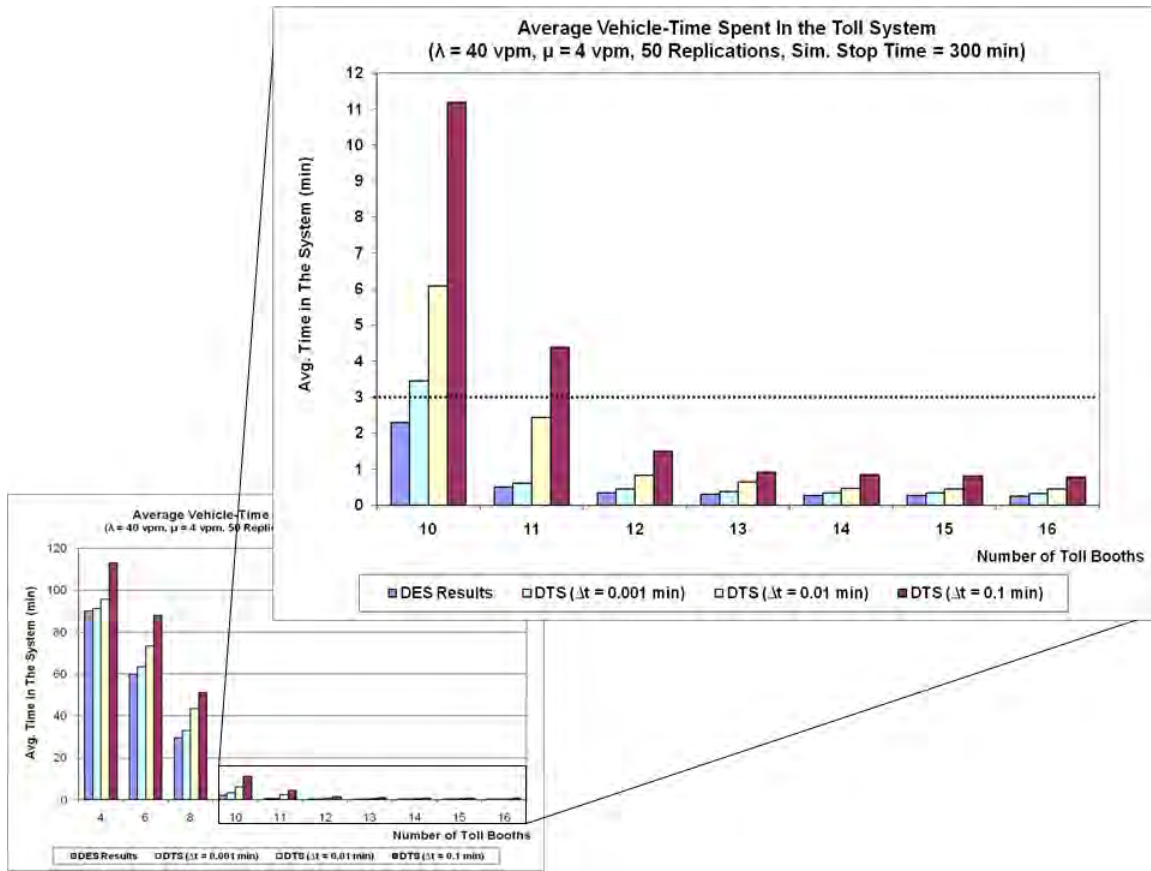


Figure 25. Toll plaza system wait delay for various booth number

It can be seen in Figure 25 that in both DES and DTS, operating 10 toll booths has a significant impact on reducing the average wait time in the system. However, when the objective is to reduce the average wait time in the system to less than 3 minutes, the results obtained from the two approaches differ. At $\Delta t = 0.1$, the DTS approach showed that 12 booths are required to arrive at the objective. DES recommended operating 10 toll booths to reach the target traffic delay. Both $\Delta t = 0.01$ and 0.001 suggested 11 toll booths are sufficient for optimal design. Although the difference of one toll booth seems small, it is of great significance when it comes to decision making at the design stage of toll systems. It was reported that the construction cost of a single booth was estimated to be \$1.5 million and approximately \$215,000 per year for operation and maintenance costs (Todd, 2005). This cost estimation clearly shows the disadvantage of using the DTS approach to these types of problems. Despite the difference, DTS recommendations

would certainly have cost consequences on the decision makers. Table 8 demonstrates the efficiency measures obtained from simulations.

Type of Approach	Simulation Average Execution Time (min)	% DES vs DTS Differences	Time Comparison
DES	0.224		1
DTS with $\Delta t = 0.1$ min	0.0218	> 100	0.097
DTS with $\Delta t = 0.01$ min	0.212	60.0	1.00
DTS with $\Delta t = 0.001$ min	2.35	19.3	10.5

Table 8. DES and DTS execution times and accuracy for toll plaza with various service time rates.

The DES approach clearly wins the efficiency test over the DTS approach, in this scenario, by being approximately 10 times faster than DTS at $\Delta t = 0.001$. The convergence of DTS to DES results illustrates the tremendous accuracy of the DES approach over DTS.

3. Summary and Discussions

Both of our simulation models produced interesting observations throughout the study. In the majority of the comparative analysis, DES demonstrated the capability to produce accurate and efficient results over the DTS model for queueing systems simulation. DTS shows that measures of performance heavily depend on the time-step size. We noticed that execution time of simulation can be a misleading metric when used to compare the two approaches in some scenarios. For example, DTS achieved better execution time for fewer entities in the system than DES where DES achieved faster execution times for dense systems. Although there are a number of time-step size selection algorithms in the literature that DTS modelers can apply to their simulation, model complexity can be extreme and costly when these are considered. The agreements between DES and DTS results come at the cost of simulation execution times where the DTS approach holds the longest. The assumption to vary time-step size is well

understood but rarely mentioned in any study in the literature. The fact that optimal time-step size selection is not defined in any DTS study raises the question of how much trust can be awarded the simulation results.

Although DES queue models can be harder to be built in computer programming language than DTS models, the Event Graph conceptual models made the implementation much simpler to proceed. Once DES models are built in simulation tools such as SimkitTM, they become more reliable for future modifications.

In conclusion, it does matter which approach is used to simulate queueing systems, especially when dealing with the common measures of performance stated above in the study. DES is preferred for this system when detailed analysis (e.g., finding critical values) is required for designing queueing systems, since it produces highly accurate and efficient results. DTS modelers need to be very cautious with the sensitivity of the results to the time-step size selection, even in steady-state conditions. The DTS approach could be used to model queueing systems in conditions where the required level of accuracy is accepted with moderate ranges, immediate system response to parameters change is essential and the system operation is examined. Both approaches are perceived as representative and provide realistic outputs in which queueing system behavior could be monitored. However, the two approaches differ in many ways. DES is more reliable when there are changes, provides detail operation and statistics, involves less conceptual model construction and more computer programming efforts with detailed complexity, and easy queue priority ordering modifications. On the other hand, the DTS approach is less reliable in terms of modifications, provides an overall picture of the system behavior, involves dense conceptual model construction and gives better initial cuts for performance of moderate-scale systems. The two models also differ in the concept of programming paradigms when dealing with the sequences of actions. Due to the discrete nature of queueing systems, DES illustrate a better understanding of the sequence of events (arrivals and services), where DTS relies on the time and makes it difficult to follow the system logic. Unlike DTS's time-step size control, simulation efficiency control is a major limitation in DES.

These experiments demonstrate some of the risks of incorrectly selecting time-step size in a DTS model. To deal with this issue in a simple manner, modelers may try multiple small time-step sizes to obtain better possible results. This chapter demonstrated that “small” does not necessarily mean “small” enough to arrive at accurate solutions, and that time-step size necessary to achieve accurate results depends on the system being simulated.

There is certainly a large area of overlap between the two approaches. Many problems could be modeled with either approach and produce results that would look very similar but with a price to pay. Both methods, if used appropriately, can help provide increased understanding and serve as an aid to decision-making. The DES and DTS queueing models could have been represented in many different ways. We were, of course, only able to choose one mode of display for each model. In the current study, we used the best possible sample that we had access to at the time. Of course the study could be improved with larger samples.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CASE II: COMBAT SYSTEMS SIMULATION

“Results of combat models and simulations directly impact lives...”

(Warhola, 1997, p. 1)

A. INTRODUCTION

Combat simulation models provide useful insights for many military decision problems (Caldwell et al., 2000). These models directly influence warfare tactics and strategies as well as the allocation of the limited Department of Defense (DoD) budget. It is almost impossible to study the behavior of military systems in combat such as entity movements, target acquisition, target engagement, command and control, communications, and resupply by recreating and testing real systems. For this purpose, combat simulation models are implemented to analyze the behavior of military forces. Unlike other models, combat model outcomes often directly affect human lives.

When a combat model predicts a high possibility of mission success for a naval ship with air support against enemy boats through only ship capability enhancement and larger use of surface-to-surface (SSM) instead of air vehicle capability enhancement or the air-to-surface (ASM), the decisions are guaranteed to be flawed. Demand for SSM with certain capabilities and ship warfare enhancement may direct the DoD to lose multi-billion dollars to SSM contracts instead of ASM contracts and cause budget allocation problems. For these circumstances, simulation accuracy is necessary to ensure effective decisions with minimum losses in lives and budget. In fact the DoD strongly relies on modeling and simulation (M&S) efforts to provide simulation models with timely and accurate results (M&SCO, 2010).

Decision makers typically rely on the simulation outcomes and analyst recommendation as inputs to their decisions. Due to the nature of decision makers responsibilities, they often have little time to discuss and understand the model internal activities. In fact, they lay a huge amount of trust in the modelers and analysts who build these simulations. Combat simulation models carry great risk and responsibility, and

modelers must take considerable careful steps in model construction stage because their choice of simulation approach can greatly influence the final decisions toward critical military situations.

When modeling large-scale combat scenarios, the view of warfare matters. The purpose of the model defines the levels of aggregation and resolution. For example high-resolution combat models that represent individual combatants as separate entities are desirable or necessary for some applications. Conversely, combining these individual combatants into groups of combatants to represent a hierarchical military command structure such as squad, platoon, and corps may be appropriate for certain purposes (Hartman, Parry, & Caldwell, 1992). Regardless of the aggregation level, it is always the actions of the individual entities in the battlefield that affect the results and battle outcomes. Therefore, modelers must understand the significance of their approach choices on model outcomes and decisions. Furthermore, they should be as confident as possible that the chosen simulation approaches do not produce flawed results.

The primary emphasis of this chapter is the internal time advance mechanism (TAM) present in combat simulations, specifically to test and describe its effect on simulation results and decision making. We discuss discrete time simulation (DTS) and discrete event simulation (DES) approaches to advance the simulation time in a number of combat scenarios. We also highlight consequences and limitations of each approach by addressing simulation efficiency and accuracy. Finally, we suggest guidelines for modelers to consider when selecting what simulation approach may be appropriate.

1. Background and Literature Review

Combat modeling is a discipline with a long history before the advent of modern computers. The military has always been interested in studying combat operations to assist commanders in critical issues such as force employment and structure, acquisition, tactics, doctrine, and training. In the past, sand tables have been used for military planning, wargaming and training for military actions. However, current military operations have increased in complexity. For example, it is not feasible to analyze a ship's weapon system alternatives by simply observing the number of casualties for each

system in a real life practice. There are numerous situations where it is not practical to analyze the behavior of a complicated military system by actually operating the system (Hartman, Parry, & Caldwell, 1992).

A model that represents the components of the real system and their behaviors and interactions is therefore required. Although, a model is never perfect, it can provide useful insights about the behavior of the system for the purpose of use. Building analytical models with the use of mathematical equations that show how state variables of the system change can be sufficient for simple systems. However, combat models seek to anticipate how an engagement may unfold under uncertainties and to assess the value of actions that the commander may take in each contingency. In these situations, it is necessary to implement simulation models of combat systems. Thus, modeling and simulation of combat has become an important element in the analysis of military strategy, tactics, policy, and training.

A combat simulation model can be classified in several ways. The model can be dynamic or static, continuous or discrete, deterministic or stochastic, high resolution or aggregated, and prescriptive and descriptive (Caldwell et al., 2000; Hartman, Parry, & Caldwell, 1992; Washburn & Kress, 2009). Combat simulation models are intensively used in situations including insufficient or imperfect real world data. Simulation, by its nature, works to address uncertainties. These models have the potential to improve test and evaluation decision making. That is, they can be used to investigate more operating environments, explore test field results, increase the extent of test scenarios, design field tests, and provide insights of operation tactics and strategies (Lucas, 2009).

a. Combat Simulation Time Advance Mechanisms

Caldwell et al. (2000) express the significance of simulation clock in combat model, “One important characteristics of a combat model is the simulation mechanism that is used to represent and to advance the battle clock.” A combat simulation is used to model entities or units of the system by changes in state variables to compute desired output values. Most combat simulation models are dynamic models, that is state variables change over time (Washburn & Kress, 2009). Therefore, it needs to

explicitly keep track of the simulation clock throughout the simulation run. Since combat simulations can involve many interactions between entities, management of simulation clock is an important aspect of simulation. A modeler's choice of TAM approach to represent a combat operation is a critical step as simulation outcomes are largely dependent on the selected mechanism. "Concurrent with the selection of a support mechanism is the decision of what form the underlying model will assume." (Warhola, 1997, p. 6). The management and update of the simulation clock is extremely crucial for accurate and efficient combat simulations.

Hartman, Parry and Caldwell (1992) discuss four different approaches to advance the simulation clock in combat simulation models; 1) fixed time step, 2) event scheduling, 3) process oriented, and 4) synchronized to real time. We have previously explained the first three; however, the synchronization to real time is advancing the simulation clock at the same rate as time in the real world. This TAM approach is useful for combat simulations used for training as the battle runs at the same time rate as the trainee, in "real time," often within virtual environments. Our studies here, however, deal exclusively with constructive type simulations that are used for analysis purpose to guide decision makers. Therefore, our focus is on two commonly used TAM approaches in constructive simulation, 1) Fixed time-step (i.e., DTS) and 2) Event scheduling (i.e., DES).

(1) Discrete Time Simulation. The fixed time increment or discrete time is the most straightforward time advancement method used in combat simulation. The simulation control logic is simply a repeated loop. At the end of each fixed time interval, the model must determine which interactions occurred between which entities and update the state variables to reflect these interactions. The simulation must test for battle termination conditions (e.g., allowed number of casualty) by evaluating all state variables at each time step. Typically, the size of the time step depends on the battle resolution and the force level being simulated, however, there are no explicit rules to identify thresholds for time-step size (Caldwell et al., 2000). Five seconds might be appropriate for fast moving ground vehicles but not for slow moving troops searching for enemy targets. A number of current combat simulation tools provide the modelers with

access to vary the battlefield time-step size. However, time step control in several combat simulation tools is invisible to the modelers (Surmit, 2004). Thus, potential difficulties in the behavior of combat simulation components may be invisible since the simulation clock governs all combat activities. Time step increment can be as a user input (Caldwell et al., 2000), and could be critical to the success or failure of the model; yet no step size selection methodologies have been widely accepted by the modeling and simulation community. Rather, modelers continue to use a trial and error to define what appear to be reasonable time-step sizes.

This chapter will illustrate how the choice of time-step size can directly influence combat processes such as force movements, target acquisition, engagement and termination conditions, communication and intelligence gathering, as well as indicate how an alternative approach can resolve these issues.

Some combat models using the DTS approach apply Lanchester equations at the end of each time interval to compute attritions between forces in the battlefield. The basic approach of Lanchester equations is to write differential equations that equate the loss rate of two homogeneous forces via two factors; 1) the size of opposing forces and 2) the effectiveness (attrition coefficient) of the killing power of each force (Washburn & Kress, 2009). Lanchester equations (Linear and Square Laws) were farther expressed in differences equations that facilitate the implementation of DTS methodology to describe the changes in force levels over the battle time. However, the errors might be large with various time-step sizes as will be shown in later sections.

A critical limitation of this type of approach is that deterministic outcomes have been obtained for probabilistic inputs. In a two force (blue vs red) battle, it was observed that only one side kept winning the battle over all simulation runs (McIntosh et al., 2007). The reason is the order of which forces change state variables was the same at the end of each time intervals. One way of avoiding this problem is to introduce randomness into the model. This produced more believable answers but rose

questions about how random is random. Numerous combat simulation tools such as MANA, Pythagoras, WARSIM, and JICM implement the DTS approach to advance the simulation clock.¹

Hillestad et al. (1995) studied the differences in combat results (losses and battle winners) predicted by models of different resolutions applied to identical combat situations. They considered time-step size as one degree of resolution and was varied over five different step sizes. The scenario considered three defender groups against nine attacker groups moving at five different speeds. At extreme speed values, the time-step size had no effect on the number of survivors. However, at intermediate speeds, changing time-step size had a great impact on the battle outcomes. They suggested that for corps and theater force level, long time-step sizes might be convenient where small step sizes are better for high resolution models. The study briefly discussed time-step size effects on combat outcomes but did not discuss causes or details such as effects on movement, detection, engagement, etc.

In a way to assist model validation, Dewar et al. (1996) examined the potential causes of non-intuitive results that take the form of non-monotonicities in combat simulation models. The model consisted of two opposing forces fighting until certain stopping conditions were reached. It was found in many cases, increasing the initial force of side resulted in a less favorable outcome for the same force. They observed that time step granularity was among the factors that caused problems. The time-step size was chosen to match the attrition coefficients that reflect the nominal values of the real attrition rates. A variety of time-step sizes in terms of attrition coefficient values were used to test the model and observe the area of a particularly non-monotonic region. It was clearly shown that step size had an impact on the result, and that smaller time-step sizes reduced these artifacts.

¹ There are several master theses conducted at NPS where the DTS approach used in combat simulation models (Hinkson 2010, Jacobson 2010, Yildiz 2009, Lalis 2007, SEED Center for Data Farming 2011).

Bletscher (2008) and McIntosh (2009) both developed separate combat simulation tools that use DTS as an approach to advance the simulation clock. Although automatic adjustments are asserted, modelers who decide to alter the time-step size for the purpose of their model were recommended to calibrate few modeling parameters to adapt to step size change. However, the effect of time-step size on system behaviors has never been explained, as well as which and how parameters can be adjusted not demonstrated.

Blais et al. (2010) examined the usefulness of the Battlespace Terrain Reasoning and Awareness Battle Command (BTRA-BC) Battle Engine (BBE) software tool which is used to assist commanders analyze friendly and enemy course of action (COA) towards a certain strategy. They used a statistical approach called Nearly Orthogonal Latin Hypercube (NOLH) for experimental design to determine the factors that have greatest effect on the outcomes. Terrain properties were the main focus and five maneuver corridors including road speed, attack maneuver, defense maneuver, attack fire support and defense fire support were explored as model factors. The study examined five time steps to investigate if the scoring of COA is effected by the time-step size. They conclude that “The results actually show that the important factors in the model changes as a result of changing the time slice selection.” Unanswered questions included which step size is correct, how time step affected the result and what can be observed from the study.

Recently, two methodologies in the literature have been suggested to improve the DTS approach in simulation including optimal time-step size; adaptive time-step size (Devenish & Thomson, 2006), and a hybrid approach (Wedemann, Barbosa, & Donangelo, 1999). However, our intensive research of the literature shows no implementation of these methodologies currently exist in combat simulation models.

(2) Discrete Event Simulation. The DES, or next event, TAM is relatively new to military simulation (Smith, 1998). In this approach the simulation clock is a continuous variable. Prior to the implementation of a DES approach, processes that define combat activities in the system must be first identified. For instance, a fixed sensor detection process involves identifying target movements (start and end), when an

enter range and exist range occur and sensor type in order to execute detection process over the simulation time. These instants cause changes in the system state variables at discrete points in time which we defined them as events. The same DES approach logic explained in Chapter II is therefore applicable to combat simulation models. Typically, the FEL of combat simulation include events such as move, stop, enter range, detect, classify, communicate, shoot, etc. The simulation clock is maintained throughout the simulated battle until stopping conditions are achieved.

The DES method has an advantage as it is able to skip quickly over timeperiods when no combat interactions or other events are occurring. This avoids unnecessary checks for updates to the system state variables and allow for precise real events timing to be taken into account.

Few studies have implemented the DES time advance approach in combat simulation models and demonstrated the ability of this approach to precisely model the interactions between force units combat battles (Havens, 2002; Grimes, 2005; Buss & Ahner, 2006; Hattaway, 2008). However, the purpose was to build combat simulation models without the discussion of reasons of approach selection.

(3) Comparison Studies. There are almost no studies that address in detail quantitative or qualitative comparison analysis between DTS and DES in the area of combat simulation modeling. Warhola (1997) quantitatively demonstrated the differences between DTS and DES in a small unit combat simulation for a single issue. Specifically, he addressed the problem of event sequencing for a high resolution, stochastic and two-sided combat models. Three event sequence orders were considered using the DTS approach, 1) Move-Detect-Shoot, 2) Move-Shoot-Detect, and 3) Shoot-Move-Detect. The DTS results were found to be affected by the length of the time-step size and change as the sequence of events changes. The smaller time-step size, the closer to DES results. However, the DES model proved to overcome the problem at the cost of simulation run time.

Buss and Sanchez (2005) developed a methodology to model simple motion (linear, uniform, two-dimensional) and simple sensing with a pure discrete event approach. Entity motion and sensing are essential element in combat simulation

models. With the use of equations of motion and the implementation of Event Graphs using the LEGO framework (Buss & Sanchez, 2002), they were able to accurately model the behavior of entities in a simple combat model. The purpose of the study is to illustrate the capability of DES to model combat elements; however, a discussion is made on the benefit of DES approach over DTS about the number of updates for movements and sensing. DTS tends to have higher computational complexity levels than DES when interactions between combating units occur. However, no empirical results were presented. We here extend the work of Buss and Sanchez (2005) to large-scale combat models to support our comparison analysis between DTS and DES time advance approaches.

2. Combat Simulation Software Packages

A number of software packages written in a variety of computer programming languages have been used to create constructive combat simulations. The following list of combat models in Table 9 is representative of the variety found in large-scale combat simulation. The list is by no means exhaustive, but rather represents a sampling of combat models with which DTS and DES are used as mechanisms to advance simulation clock (Lucas, 2009; Alexander, 2011).

DTS Software Tools	DES Software Tools
BTRA-B	COMBAT XXI
CBS	DAFS/ JDAFS
CEM	JANUS
COSAGE	JAWAR
DIAMOND	NSS
EADSIM	OneSAF
Eagle	Simkit
IWARS	
JAS	
JICM	
JTLS	
MANA	
PAX	
Pythagoras	
TACWAR	
THUNDER	
VIC	
WARSIM	

Table 9. TAM categorization of combat simulation models.

Our research started by exploring the possible use of combat simulation tools available at NPS. After considering a number of options and clearance limitation to some tools, MANA 4 and 5 were chosen as representations for DTS.

Map Aware Non-Uniform Automata (MANA), a time-stepped, agent-based model (ABM) tool is used for combat simulations, and has been intensively used in numerous military studies and Master theses at NPS. This intensive use of the tool makes it more attractive for investigation to attain valuable contributions that can impact the military M&S committee. MANA also implements the DTS approach in a similar way as other DTS combat simulation tools. MANA is user friendly, well documented and frequently updated by its developer. Combat scenarios can be quickly generated in MANA where agent personalities, movements, sensors, weapons, and other parameters are easily manipulated. Thus, MANA is considered to be our best choice as a DTS candidate.

MANA has chosen to favor simplicity and use a DTS approach. The developers of MANA claim that there is no advantage to highly detailed models as these never accurately capture every aspect of realism. Unfortunately, simplicity also entails limitations and specific problems may require an alternate approach. Our attempt of using MANA is not to criticize the tool itself, but, to conduct experiments and highlight some unintended consequences from the TAM used in MANA.

The focus of this section describes the characteristics of the combat models used for this research in terms that are easily understandable. Both MANA 4.4.4 and MANA 5.00.89 were used to model combat scenarios with the DTS approach.

a. MANA 4

MANA version 4 was released in 2006 and it is an ABM class in a subset called Cellular Automaton models. An element in MANA is the “squad,” representing one or several agents (McIntosh et al., 2007). An agent “squad” is described by sets of user defined parameters: 1) personality weightings, 2) move constraints, 3) physical

characteristics such weapons, sensors, communications and movements, 4) movement algorithms. For MANA 4 information, readers are advised to refer to the manual (McIntosh et al., 2007).

(1) **Battlefield Space.** The battlefield in MANA version 4 consists of a finite set of possible agent positions on a two-dimensional 200 x 200 cells grid pattern. Users define real maps by rescaling them on the grid pattern map. An agent cannot wander outside the battlefield boundaries. MANA provides several types of terrain such wall, bush and hilltops. Since our study is examine the impact of TAM on outcomes, terrain is not used. Agent locations (initial and waypoints) are defined in the battlefield map using (x,y) coordinates.

(2) **TAM Approach.** In MANA version 4, the time-step size is indirectly controlled by the size of the battlefield. That is, to represent certain time-step size, the battlefield size must be changed which leads to calibrating agent speeds, sensors, engagements, and other agent behaviors.

(3) **Movement.** Agent movement is governed by personality weightings for a squad defined by the user. Agents can only move from grid square to another grid square depending on the location of other agents and battlefield conditions. With personality weightings, penalties are defined based on the following equations (McIntosh et al., 2007):

$$P_i = 1.0 + \frac{\sum_{m=1}^M (D_N(m) - D_O(m)) D_L(m)}{100 \sum_{l=1}^M D_L(l)} \quad (21)$$

$$Penalty = \frac{1}{100} \sum_{i=1}^N W_i P_i \quad (22)$$

where, M is number of agents within range of moving agent, DN and DO new and old distances to other agents respectively. DL is a normalization factor to scale distances according to battlefield. Wi is weighting factors associated with each penalty. Figure 26 shows an agent tendency in movement.

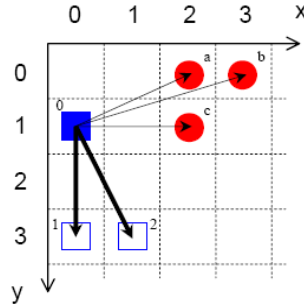


Figure 26. Agent tendency movement in MANA 4 (from McIntosh et al., 2007)

Agent speed is a tangible aspect or a physical property and defined in terms of number of cells an agent can move in one time step.

(4) Sensors. The sensor is based on a probability calculation within the sensor range of each agent. MANA uses two types of sensing models: simple and advanced. The simple type is a “cookie-cutter” sensor, that is if the agents within range then detection is with 100% probability. The advanced mode uses a range table to represent distances from the sensor and the associated probabilities. In the table, detection probability is defined in terms of the reciprocal of average time between detection. An agent sensor range is defined in terms of the number of grid cells from the agent. Classification in MANA is done per time step and determined by sensor’s Classification Rate per Turn. However, for the purpose of our study detection is assumed to occur at the same time as classification.

(5) Engagement and Weapons. Like sensors, weapons in MANA are modeled by user-defined weapons characteristics in terms of a probability of kill/range table, the type (direct fire or indirect fire), and the number of weapons. Since MANA is a time step model, firing rate is considered to match the step size. The user can define the weapon shot radius to be used as blast radius for indirect fire. Agents fire the weapon classified for specific targets once detection is registered. MANA models damage to agents as dead or alive, other damages can be modeled indirectly using triggers, but these are not considered in this research.

(6) Communication. Communication between agents in MANA is done through the use of Situational Awareness (SA) map. Each squad shares the SA map in which all information gained by an agent is put on the map for the accessibility of all agent members.

(7) Stop Conditions. The battle can stop when either Blue or Red force reach their final waypoint, or when a particular number of casualties had been sustained.

b. MANA V

MANA version 5 was released in 2009. It is a modified version of MANA 4 and it was developed to alleviate some issues with the battlefield grid to allow greater flexibility to add new features to the simulation of combat models (McIntosh, 2009). Readers can refer to McIntosh's (2009) manual for more information.

(1) Battlefield Space. The battlefield grid in MANA 4 was replaced by a vector-based movement scheme within a two-dimensional continuous coordinates system. The battlefield size can be defined larger and continuous ranges than in MANA 4.

(2) TAM Approach. The battlefield time interval is introduced in this version as a user defined element. The default value is one second $\Delta t = 1.0$ per battlefield time step, however, many modelers chose much larger step sizes to reduce the overall computational time. A caution is made that MANA should rescale automatically with changes in battlefield time-step size, but this is not guaranteed (McIntosh 2009). No farther explanations were given to when calibration is required and how parameters should be calibrated.

(3) Movement. Movements in MANA 5 are no longer grid-based as in the previous version. Each agent monitors all other agents and terrain features within sensor range and computes a force vector to each. Each of these vectors with the associated personality weightings are added to represent an overall force vector that drives the agent in its direction. Figure 27 provide an illustration of force vector methodology used in MANA V version 5.

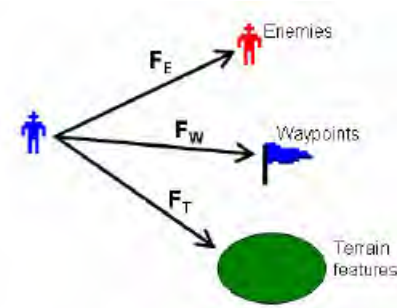


Figure 27. Agent tendency movement in MANA 5 (from McIntosh, 2009)

(4) Sensors. Similar to MANA 4, sensor implementations remain the same in this version with the addition of choosing particular classes of agent to be seen by a sensor. Sensor range can only be incremented by an integer value, limiting the flexibility to choose between ranges.

(5) Engagement and Weapons. Weapon and engagement tables are similar to what is used in the previous version. Each agent can be equipped with six different weapons with multiple rounds. The user defines a weapon firing rate in which weapons are selected in a specified order.

(6) Communication. Similar to MANA 4, communication implementations remain same in this version.

(7) Stop Conditions. Stop conditions are similar to MANA 4 with the addition of considering neutral agents and a particular squad.

c. DAFS and Simkit

Dynamic Allocation Fires and Sensors (DAFS) was found to be the best candidate to match MANA. DAFS was originally developed as part of an NPS Master thesis (Havens, 2002) and has undergone extensive revisions subsequently. The DAFS discrete event simulation framework is built on the concept of Schruben's (1983) Event Graph methodology (see Chapter I section C). DAFS is programmed in JAVA that utilizes several components of Simkit; a JAVA simulation toolkit developed by Professor Arnold Buss. Simkit implements the Event Graphs through interface classes and store them in a library of reusable software components (Buss, 2001 & 2002). The Simkit

library contains a variety of tools that can simulate agent personalities, movements, sensors, weapons, and other parameters making it easy to use. Simkit tool receives scenarios inputs from main classes, uses SimEntityBase class to implement most of the functionality for interaction with Future Event List (FEL), and report outputs. DAFS receives scenario inputs from Access or Extensible Mark-up Language (XML) files, implement all routine such as event scheduling and optimization routines in Simkit, and produce output Access files to be available for analysis. Simkit 1.3.8 and DAFS are used to model combat scenarios with DES approach.

(1) Battlefield Space. Locations on the battlefield are referenced by an (x,y) coordinate with units of distance. The battlefield size can take any continuous boundary values with any distance units in Simkit and with units of kilometer in DAFS. Altitude, terrain features have not been incorporated for the purpose of study.

(2) TAM Approach. DAFS and Simkit simulation tools implement the DES approach to advance the simulation clock. The simulation time advances from a current event time to the time of the next event on the top of the FEL and so on. This process is handled by a class called SimEntityBase that interact with the FEL. Time in Simkit can take any unit value, where DAFS consider the unit of time to be in hours.

(3) Movement. As in MANA, entities in Simkit generally move in uniform, linear motion by starting from a location and without relying on an explicit location state can end on another location. However, movement is not limited to uniform linear motion and any closed-form equation of motion could used. Buss and Sanchez (2005) claim that linear motion can model a wide range of motion possibilities using a piecewise linear approximation. An entity movement is not limited to a specific direction. Movement in Simkit is represented by a Mover component, responsible for maintaining movement state and a MoverManager component which is responsible for elementary maneuver types (Buss & Ahner, 2006).

(4) Sensor. Sensing in a pure DES approach is not focused on probability of detection, but rather on when the sensor acquires and loses contact of a target (Buss and Sanchez 2005). Events such as EnterRange and ExitRange trigger sensor

detections. Sensor range and probability of classification and detection are set as parameters in sensor class. Various types of sensor can be implemented in Simkit including: cookie-cutter, constant time and constant rate along with associated probability of detection. Three major components manage sensors in Simkit, 1) *SensorType*, 2) *SensorTypeMediator* pattern, and 3) *Referee* pattern. The *Referee* component is not a sensor component rather than a general component that plays an important role in the model structure of DES as discussed in the next section.

(5) *Engagement and Weapons*. In a pure DES approach, weapons are represented in similar way to sensing (Buss & Ahner, 2006). In simple two-shooter scenario in Simkit, weapon probability of kill given a hit is the primary concern. However, in DAFS the primary focus is on munitions rather than the weapon itself. A *Munition* in DAFS is represented by a fast mover that impact a target. Targets are either dead or alive, but other damage types can modeled. For a simple two shooters scenario, Simkit implement a shooter component linked the referee to represent engagement in the battlefield. DAFS uses two components 1) *MunitionTargetRefree* and 2) *MunitionTargetAdjudicator* to enable engagements. Weapon types, ranges, probability of kill, number of rounds and firing rates are input parameters used to define weapons.

DAFS also uses a simple linear optimization to conduct fire assignments. An optimization problem is solved in an entity called *Constrained Value Optimizer (CVO)* which supports forces to increase the near term probability of success (Buss & Ahner, 2006). The CVO controls weapon-target selection process, that is, which type of weapon is appropriate for certain targets.

(6) *Communications*. Communication in the DES approach is implemented using sources and listener protocol between the simulation components. Sources generate events which may or may not require actions and listeners receive this information and responsible to process it between components using the LEGO methodology (Buss & Sanchez, 2002). Agent or entity communication is simpler in the DES approach than in DTS. In DTS model all agents are checked for updates at every time step, where in DES only the occurrence of events trigger actions. Therefore, precise time of interactions are taken into to schedule events on time.

(7) Stop Conditions. Stopping conditions in Simkit are very flexible, as the simulation can stop according to; a simulation time limit, or reaching a specific event (e.g., Red or Blue killed), or reaching a specific number of events (e.g. the number of Blue casualties).

3. Combat Modeling Elements

This section discusses the primary elements of a high resolution, entity level combat simulation model. Typically, a simple combat scenario involves five essential elements; 1) entity/unit state, 2) movement, 3) sensing and target acquisition, 4) attrition and weapon engagement and 5) communication (Alexander, 2011). We give extensive descriptions on how these elements interconnect to form the structure of a simple combat scenario using DTS and DES time advance methodologies.

Battlefield representations for combat models take different forms. The size of the battlefield space depends on the resolution of simulations. Terrain representation may influence the process of some combat simulation elements. However, terrain features can be represented as averages of values to these combat elements without the present of terrain. For example, an entity moves slower in bush area than on streets. The entity speed can be altered to slowed down for the bush area and made faster on streets without involving terrain features. Since the focus of our study is to investigate the impact of TAM on the simulation outputs, terrain is not included.

a. Entities/Squad units/Platforms

(1) MANA – DTS Approach. The main player in MANA scenario is the squad unit. A squad is a group of agents of any size including a single agent. An agent in MANA can represent an individual, a vehicle, and military devices. Agents in a squad share the same properties and can change states in time step increments. An entity/agent properties are defined by parameters such as allegiance (e.g. 1-Blue and 2-Red,), personality weightings to control agent behaviors (e.g., 100% towards enemies or waypoints), speed, locations, etc. Personality weightings and other agent characteristics are defined as Default state but may change to other states by using

“Trigger State” feature. This feature puts agents in a specific state for a predefined period of time expressed in terms of time steps. An agent characteristics such as movement, sensors, weapons and communication networks are built separately but linked to the agent.

(2) Simkit/DAFS – DES Approach. Typically, agents in combat-oriented models are moving entities rather than stationary. Simkit incorporates a dedicated interface named “Mover” to depict a moving entity’s functions. A squad unit of a number of homogeneous agents can be constructed by modeling a group of same individual movers using an ArrayList instance in the Mover class. Agents’ behaviors such as moving towards waypoints, or towards an enemy, speed, and initial locations are input parameters in the main class.

In DAFS, platforms represent the foundation structure of any Unit of Action (UA) involved in the scenario such as soldiers, tanks, and air vehicles (Havens, 2002). Platforms can also represent stationary entities such as radar antenna, or any other physical elements. The responsibility of each platform is to know its location and speed and report this to the requesting source. Like any physical entities in combat simulation, platforms may have associated with them any number of sensors, weapons and communication elements as shown in Figure 28.

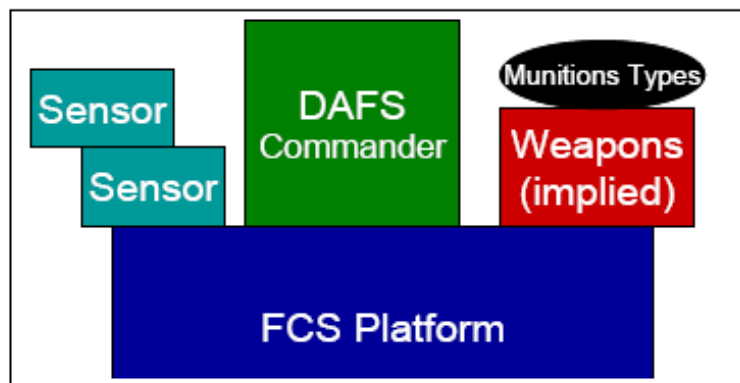


Figure 28. An example DAFS entity structure (from Havens, 2002)

b. Movement

Entity movement is one of the most essential element in combat models, “Movement processes are part of the basic bookkeeping in any combat model.” (Caldwell et al., 2000, p. 26). The movement process involves three aspects: the choice of movement objective (where to go), the travel route (the path) and the speed of the agent (how fast to go). Typically, movement processes in the battlefield should answer questions regarding movement path, factors that affect movements like terrain, location update methodology, and speed determination. In general, movement destination and path for each moving entity is part of the simulation input. The movement process then manages the update of entity position as time progresses.

(1) MANA – DTS Approach. Agents movement in MANA 4 is simply done by relying on the personality weightings defined by the user. The overall penalty value calculation is carried out for every agent in the scenario according to equations (21) and (22) to decide which adjacent square it should move to in the following time step. With a small random component being introduced, the square which receives the lowest penalty value will be the most likely that the agent moves to at the next time step (McIntosh et al., 2007).

The cell/grid-based scheme has been replaced by a vector-based scheme in MANA V version 5 (McIntosh, 2009). Each agent monitors all other agents within sensor range and calculates a vector to each. Vector addition is applied to all individually calculated force vectors F_i , weighted by associated personality weightings W_i defined by the user that yield a resultant force vector F shown in equation (23). Index i represents the agent's elements of interest to move towards to, these are enemies (e), waypoint (w), and terrain features (t).

$$F = F_e W_e + F_w W_w + F_t W_t \quad (23)$$

Entity movement is calculated from Newton's second law where, $F = ma$ and standard kinematic equations for constant acceleration as shown in (24) and (25):

$$\bar{x} = \bar{x}_0 + \bar{v}\Delta t + \frac{1}{2}\bar{a}\Delta t^2 \quad (24)$$

$$\bar{v} = \bar{v}_0 + \bar{a}\Delta t \quad (25)$$

where, $\bar{x}$ is the current location vector in 2-dimensional coordinates (x_{01}, x_{02}), $\bar{x}_0$ is the initial location, $\bar{x} - \bar{x}_0$ distance traveled in time interval Δt , $\bar{v}$ is the current velocity vector and $\bar{v}_0$ is the initial velocity, $\bar{a}$ acceleration vector, Δt time-step size.

The user defines movement input parameters such as time-step size, speed, initial locations, and waypoints for each agent prior to simulation execution. For the purpose of our study, all entities are considered to have small masses ($m < 0.1$) to give them almost instantaneous acceleration and produce movement behavior similar to Simkit. The equations above show that time step plays an important role in agent movement. Several combat scenarios will explore this role in more details in the forthcoming sections.

(2) Simkit/DAFS – DES Approach. Similar to MANA, entity movement in Simkit is a simple uniform and linear motion movement type. A moving entity continuously change its position over time, but explicit state is not necessarily required to express movement. In fact, Buss and Sanchez (2005) modeled entity movements with pure DES approach by implicitly determining state variables. A moving entity starts moving from an initial position (defined by the user) at a certain start time and begin moving with a velocity in the direction of a destination waypoint. Therefore, the implicit state of position is determined from three explicit state variables: 1) initial position, 2) start time of movement, 3) velocity vector. The current location of an entity can be computed from applying the kinematic equations of motion shown in (26) and (27) at any needed time instead of explicitly storing it at all time.

$$\bar{x}(t) = \bar{x}_0 + \bar{v}(t - t_0) \quad (26)$$

$$\vec{v} = \frac{\vec{d} - \vec{x}_0}{\|\vec{d} - \vec{x}_0\|} \cdot s \quad (27)$$

where, $\vec{x}_0$ is the initial location vector in 2-dimentional coordinates (x_{01}, x_{02}) , $\vec{v}$ is the velocity vector, $\vec{d}$ is the destination vector in 2-dimentional coordinates (d_1, d_2) , s is the entity speed.

The equations of motion represented above can be considered absolute motion in the base coordinates. It is possible to represent an entity motion relative to other entity by transferring coordinates of one in terms of the other. Furthermore, acceleration and turns can be easily modeled using a piecewise linear approximation to smooth curved motions (Buss & Sanchez, 2005).

In Simkit and DAFS, an entity movement is represented by a mover component. Figure 29 illustrates a common Event Graph of the mover component that implements equations of motions. Only events that cause the state of an entity to change are included in this component. Additionally, when an entity movement is changed like a direction change in any way, a property change is fired that may be received by other components listening for the change to perform actions. These actions then may execute events and schedule events in the registered listeners which may be heard by other components and so on.

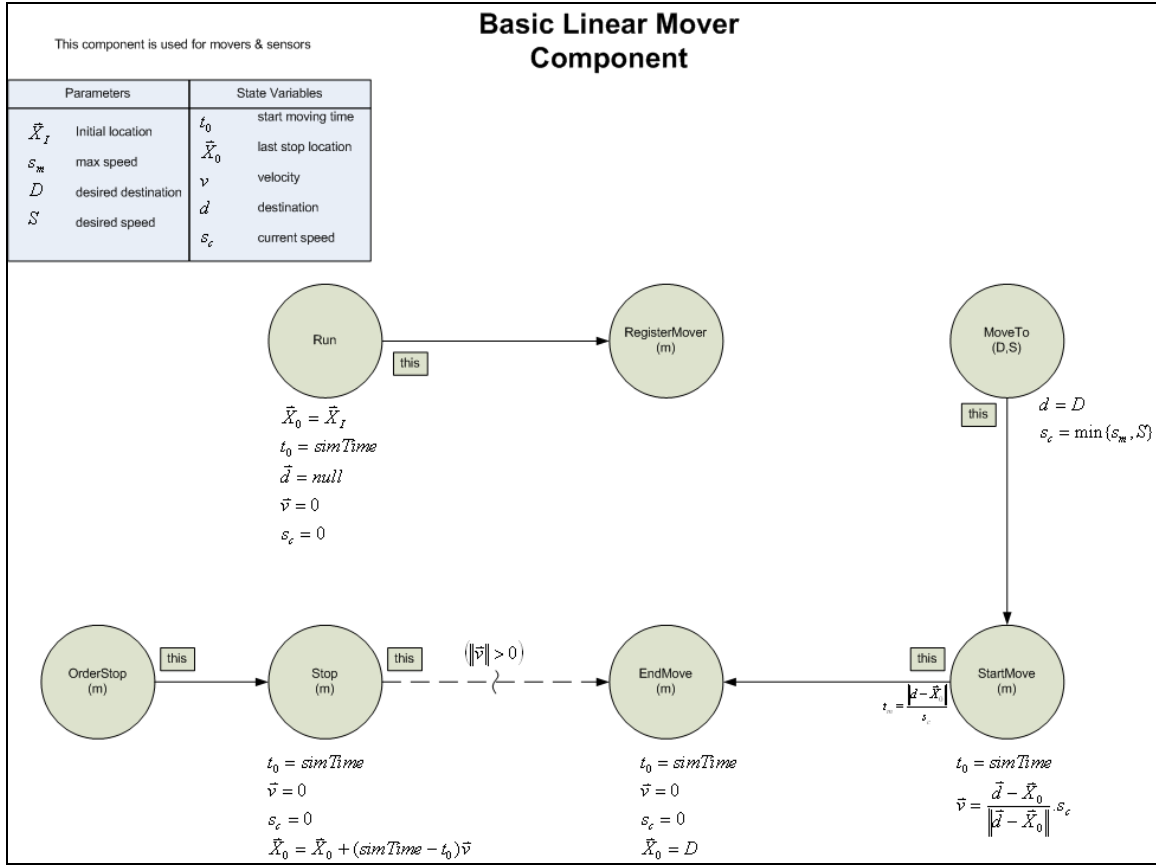


Figure 29. Basic linear mover component Event Graph

Movers (or platforms in DAFS) may also represent non-mobile entities like radar station. Each Mover has a single MoverManager component that implements the rule of maneuver and control its movement at any time. The MoverManager listens to its Mover for EndMove event and choose certain movement actions based on its type (e.g., RandomMoverManager or PatrolMoverManager). The movement to a predetermined list of waypoints is controlled by the PathMoverManager component. This component is responsible for directing the entity to the next waypoint when each waypoint is reached until the collection of waypoints “path” is traversed. Figure 30 shows an example of a PathMoverManager component used in controlling entity movements.

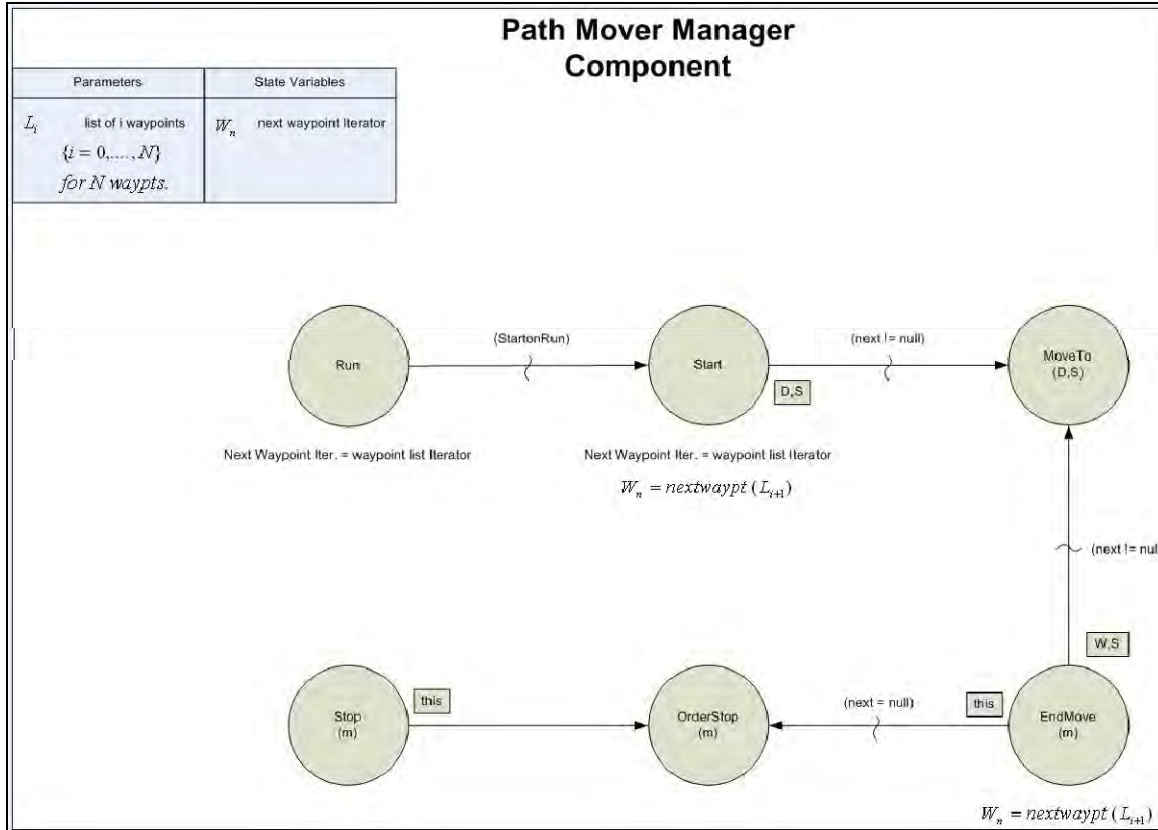


Figure 30. PathMoverManager component Event Graph

It is important to understand that the time of each event is pre-calculated in advance to determine when a moving entity will reach a destination. Events are then put on the FEL in the order of event with least time first.

c. Sensors

The acquisition of individual targets in the battlefield is an essential element of combat simulation models. Some entities are interested in long range acquisition of unengaged target units, other entities are interested in only short range acquisitions and other critical intelligence information. For simplicity, only circular-type sensors are utilized in all of the study combat scenarios.

(1) MANA – DTS Approach. Each entity in MANA 4 has an associated sensor that is attached to it. MANA has two types of sensing models: the

Simple “cookie-cutter” and the Advanced models. In the cookie-cutter model, the sensor range defines when detection and classification occur. Sensor ranges are expressed in term of the number of grid-cells from the entity center point. Grid-cell representation of the real map can cause limitations to the sensor range. That is, if a single cell represents a large area in the real map, the sensor range cannot take any value within that cell range. Thus, the sensor range can only take certain values the user is forced to use. The idea behind “cookie-cutter” sensors is simple, if the target is within the sensor range, detection occurs with 100% probability (i.e. $P_D = 1.0$), else no detection should be considered. Once detected, a target can be classified at the same range or at less range. For the purpose of our study, classification is assumed to occur at detection range.

In the Advanced mode, a range table can be set up to represent the degradation of the sensing capabilities with distance from sensor. In this mode, once a target is within range, detection occurs with a user define probability with some time delay.

Sensor implementation in MANA 5 remains very similar to MANA 4. Although sensor range is limited to integer values selection, it is more flexible than the grid-based feature in the previous version.

(2) Simkit/DAFS – DES Approach. Entity detection in DES approach depends on events generated from sensor-target interactions rather than probability of detection as a primary measure of detection. Buss and Sanchez (2005) developed a methodology to model target detection in a combat scenario with DES approach. Sensor-target interaction events and computation of time to first interaction are primary measures in the methodology. Consider a single stationary sensor and a moving target. The first interaction between them is when the target enters the sensor range, thus an EnterRange event has occurred. A detection event may occur at any time depending on sensor behavior and target cross section area. A sensor may also lose a target due to internal or external factors that influence detection such as target exit range or bad weather condition. At this point two events could occur simultaneously or with time delay, these are ExitRange and Undetection events. Assume a cookie-cutter type sensor with range R is located at the origin in a two dimensional space and a target moves at

constant velocity starting from point x at start time 0. EnterRange event only occurs when the target distance is at exactly sensor range R from the sensor. EnterRange event time is then required for the event scheduling process in the DES approach. It is computed in equation (30) by finding the solutions to equation (29) (Buss, 2011a):

$$\|\bar{x} + \bar{v}t\| = R \quad (28)$$

$$\|\bar{x}\|^2 + 2(\bar{x} \cdot \bar{v})t + \|\bar{v}\|^2 t^2 = R^2 \quad (29)$$

$$t = \frac{\bar{x} \cdot \bar{v}}{\|\bar{v}\|^2} \pm \frac{\sqrt{\|\bar{v}\|^2 (R^2 - \|\bar{x}\|^2) + (\bar{x} \cdot \bar{v})^2}}{\|\bar{v}\|^2} \quad (30)$$

The solutions to above equation (30) determine the time (if any) of sensor-target interaction events such as the time of enter range or exit range event. Figure 31 shows sensor-target interaction events for a circular range sensor.

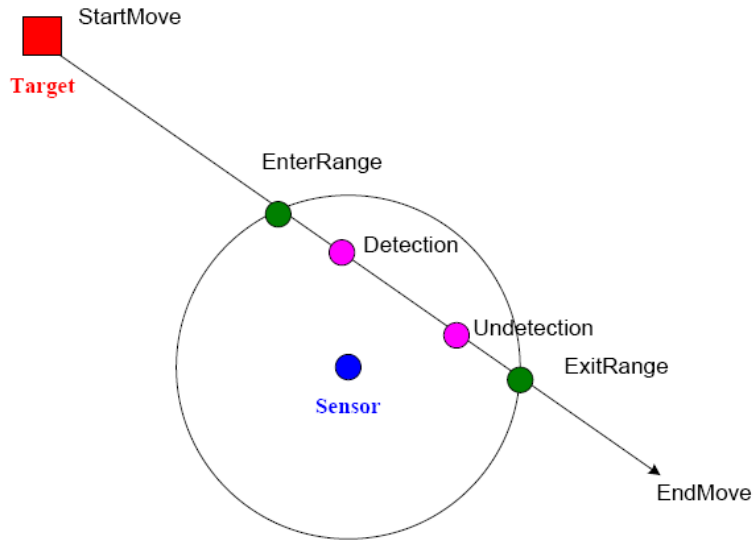


Figure 31. Sensor-target interaction for a circular sensor (from Buss & Sanchez, 2005)

Detection and undetection events are then scheduled based on the type of sensor used in the combat model. For a cookie-cutter type sensor, a target detection event is scheduled immediately (no delay) after the target EnterRange event occurs. Otherwise, time delay that follows a distribution may be implemented towards other sensor types.

Buss (2011a) and Buss & Sanchez (2005) also implemented the above methodology in Simkit using the power of Event Graphs. A Sensor component is created to maintain a container of its detections by registering detected targets and hold their parameters for detection algorithm. Figure 32 illustrates a basic Sensor component used in combat scenario.

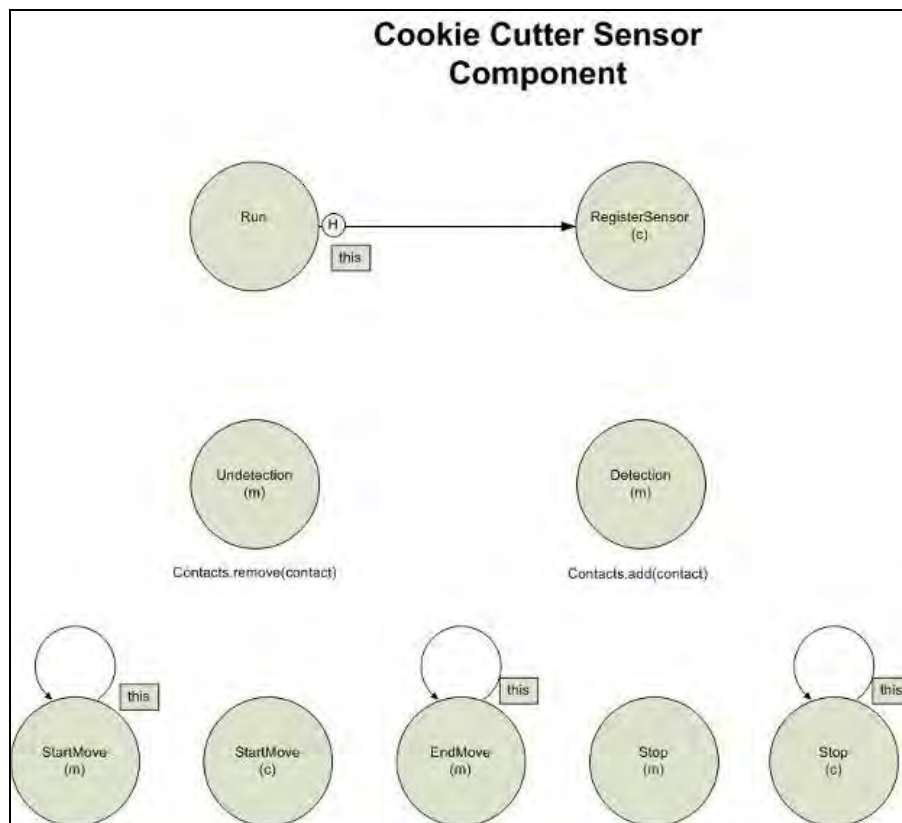


Figure 32. A typical Sensor component Event Graph

The capability of a sensor to process detections is accomplished through the use of the functional object called Mediators and Referees along with the LEGO process. Mediator/Adjudicator and Referee components are used intensively in DAFS to manage the sensor-target interactions (Havens, 2002).

The Referee component is the central heart of sensor-target interaction process that takes place in almost all combat simulation models. The Referee maintains a list of targets and sensors, and has a primary responsibility to schedule EnterRange and ExitRange events. The Referee uses the LEGO process to listen to events like StartMove from any moving entity (sensors or targets) and uses equation (30) to schedule EnterRange events. It keeps track of all moving entities and their changes in movement to schedule target detections and link them to the correct sensors that generated detections. Figure 33 shows an example of a Referee component used in combat scenarios.

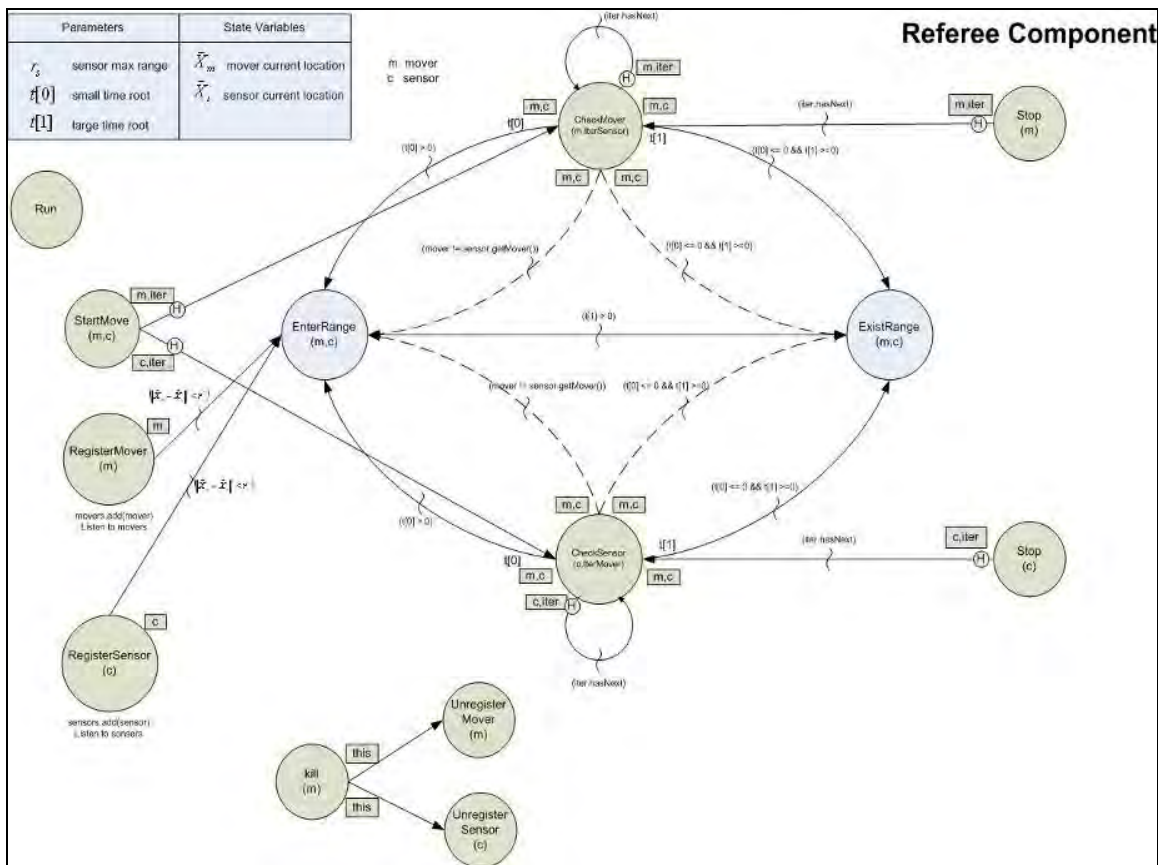


Figure 33. An example of Referee component Event Graph

In Simkit and DAFS, Mediator components are responsible to schedule detection and undetection events. Mediators listen to the Referee for EnterRange and ExitRange events to implement the detection algorithm associated with the type of sensor used in the combat scenario. The LEGO process facilitates the information passed to the Mediator components to keep track of the type of targets and sensors (Buss & Sanchez, 2005). Figure 34 provides an example of a cookie-cutter sensor Mediator component.

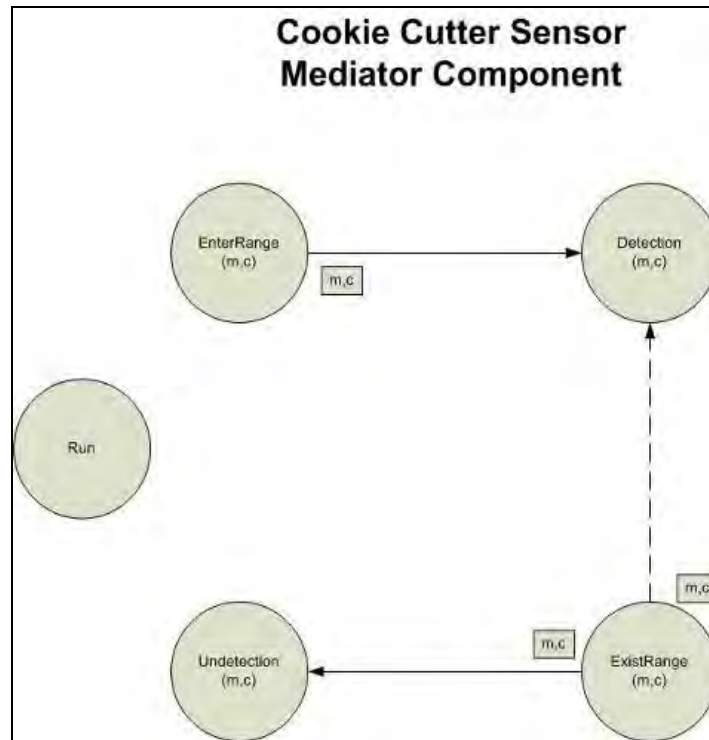


Figure 34. Cookie-cutter sensor Mediator component Event Graph

d. Engagement and Weapons

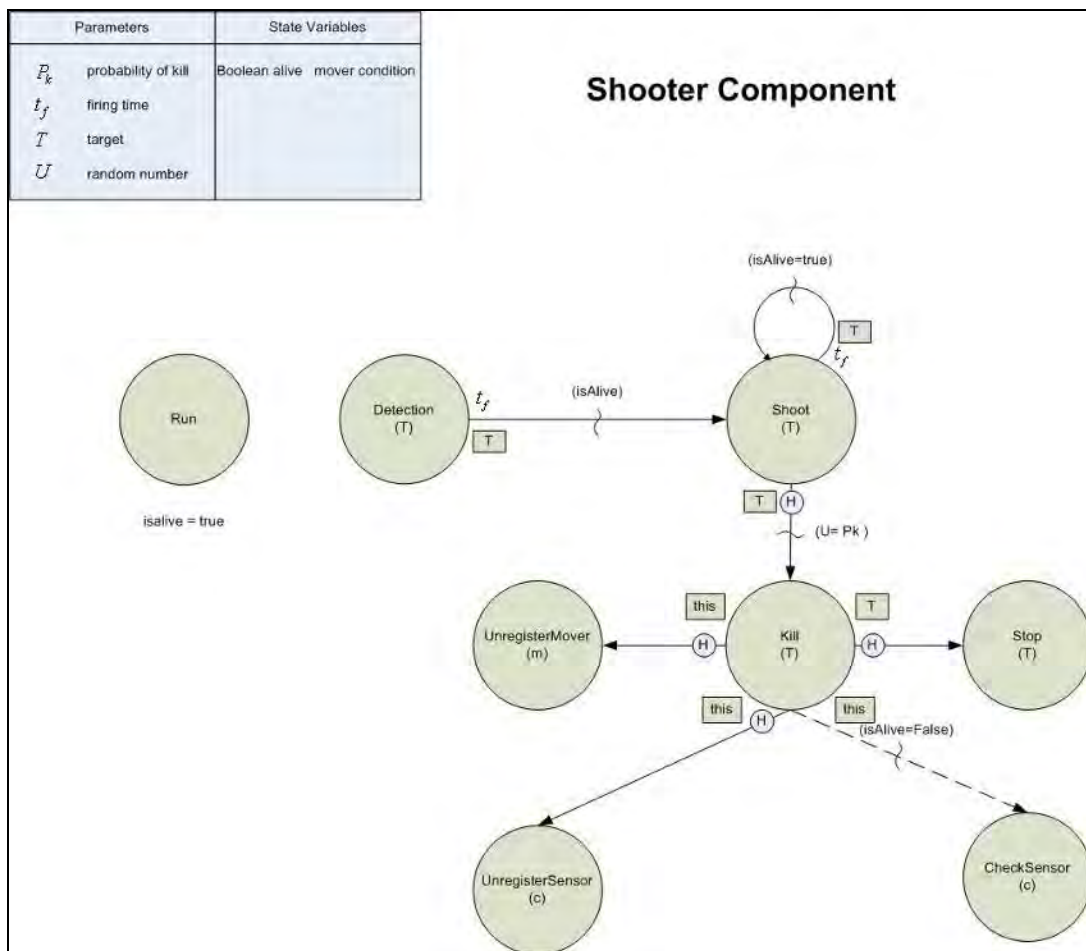
Combat models are built to engage in combats, “Engagement and attrition processes are at the heart of any combat model.” (Caldwell et al., 2000). As with real combat, entities in combat simulations are present to fight in the battlefield. When a weapon system on the battlefield knows about any enemy system, an engagement decision must take a place. If an engagement is planned, a number of elements must be

determined including; the ammunition to be used, the method of delivery, probabilities of hit and kill, effective range, engagement termination method, firing rate, and the number of rounds. Typically, these are user defined inputs and combatants make their decisions during scenario execution.

(1) MANA – DTS Approach. MANA offers the possibility of defining several weapons per agent. These can be direct fire weapons like machine guns or area fire weapons like high explosive weapons. If an agent with loaded weapons registers target detection from its sensor, available weapon to that target is fired. Firing rate in MANA 4 occurs once at each time step, where as MANA 5 allows for flexible firing rates to be defined per time unit. A fired weapon travels at a very large velocity to cause immediate effect on the target. Calculations are then made to determine if target is hit, and killed given a hit. The results are reported immediately at the same time step. Weapons are fired at targets as long as no target kill is registered. The probability of kill in MANA is assumed to be piecewise constant over the weapon range.

(2) Simkit/DAFS – DES Approach. Weapon representation in DES is similar to sensors representation. The only functionality of a weapon is to launch munitions (Havens, 2002). A Munitions object is represented by a fast moving Mover whose EndMove event schedules an Impact event. Both direct and indirect fire munitions are modeled using the same approach. In DAFS, a MunitionTargetReferee component is responsible for determining the targets that are impacted by the munitions and selecting an appropriate MunitionTargetAdjudicator. The MunitionTargetAdjudicator component listens to these impacts and decide which targets in the munitions blast area are affected. The only function of the MunitionTargetAdjudicator is determine whether a shot/impact did or did not kill the target. Like the Mediators in sensors, MunitionTargetAdjudicator implement different algorithms to associate probability of kill with munitions type, weapon range, and target distance from the blast. DAFS considers linear probability of kill by user defined minimum and maximum ranges along with associated probabilities. Upon a target kill, a Kill event is heard by other component to eliminate the target or possibly end combat. Otherwise, available munitions will keep firing based on a user defined firing rate until the target is killed.

In DAFS, the fire assignments are updated using the Constrained Value Optimizer (CVO) to solve a simple linear optimization problem. The CVO is responsible to provide a solution that enables the forces to revise their collective engagement tactics to increase the short term probability of success in the simulation model.



The Shooter component hears Detection events from the Mediator, then schedules a Shoot event at the desired target. Munitions launch at a firing rate with a time delay t_f until a Kill event occurs (i.e. when probability of kill PK is satisfied).

e. Communications

(1) MANA – DTS Approach. Communications and sharing information between combatants is a key element in combat model. MANA represents Situational Awareness (SA) in a map shared and accessible by all members in each squad. Part of the function of the SA map is to represent communications within the squad, whereby agents share information as one would expect in a combat scenario. Information such as enemy locations passes from a sensor station to all friendly squads via the SA map. The squad's Inorganic SA map controls the flow of information between squads. The parameters defined by the user represent delays in the flow of information, the rate of information flow, and if all information should be sent to squads or only part of it. The delay in receiving a single message between squads is defined by the number of time steps.

(2) Simkit/DAFS – DES Approach. Buss and Sanchez (2002) proposed the LEGOs method that is useful for discrete event modeling. This method links small models represented by Event Graph components using a design pattern from Object Oriented Programming called “listener pattern” to produce new modules of greater complexity. The method starts by first encapsulating small models Event Graphs into components, then link them with listeners. LEGO is an effective method as it prevents breaking model elements that are tightly integrated in a single component and provide a mechanism by which events occurring in one object trigger events in another object. This type of interaction between model components using LEGO is called source-listener interaction (Buss and Sanchez 2002). Sources are the source of a trigger event that may or may not require actions on the part of another entity. Sources are responsible to send the information. Listeners receive the information and responsible to process it in the desired way.

Communication between entities and objects in DES combat models are primarily done through the implementation of source-listener template. This template allows for passing information to elements within the simulation to act based on the property changes of other elements. It also allows the monitoring of property changes due to event occurrences throughout the model as a data gathering medium for analysis. For example, a combatant's Shooter component would be registered as a listener to a particular sensor station, the source. Every time the sensor station component "fires" a detection event, the Shooter component will hear it and schedules a shoot event.

With the availability of source-listener interaction protocol, the user has a great flexibility to introduce parameters that represent delays in the flow of information between combatants, rate of flow of information through communication links and likelihood that information gets lost en-route to a destination.

Figure 36 illustrates the complete system of a combat simulation model for the two shooters scenario. Each sub-model is shown to be encapsulated in a component and linked by using the listener pattern to other components.

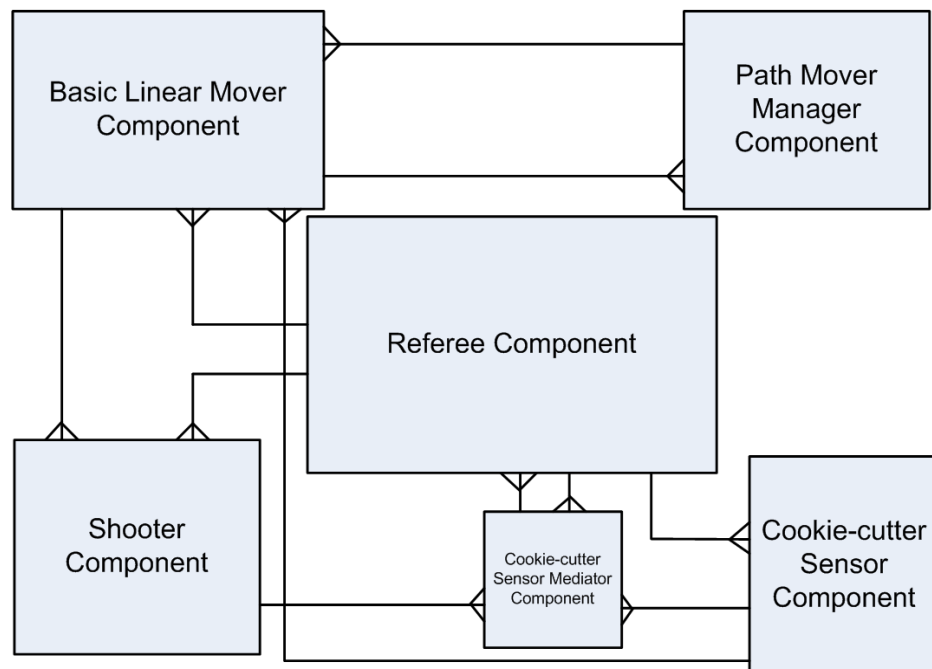


Figure 36. Illustration of a simple shooter simulation model in DES using the LEGO framework

Figure 37 shows a graphical representation of the input scheme used by DAFS (Havens, 2002). The blocks on the left of the figure represent a self contained XML document, however, these are created in Access data base file. The diagram shows that each entity in must contain a list of information to participate in constructing the combat simulation model.

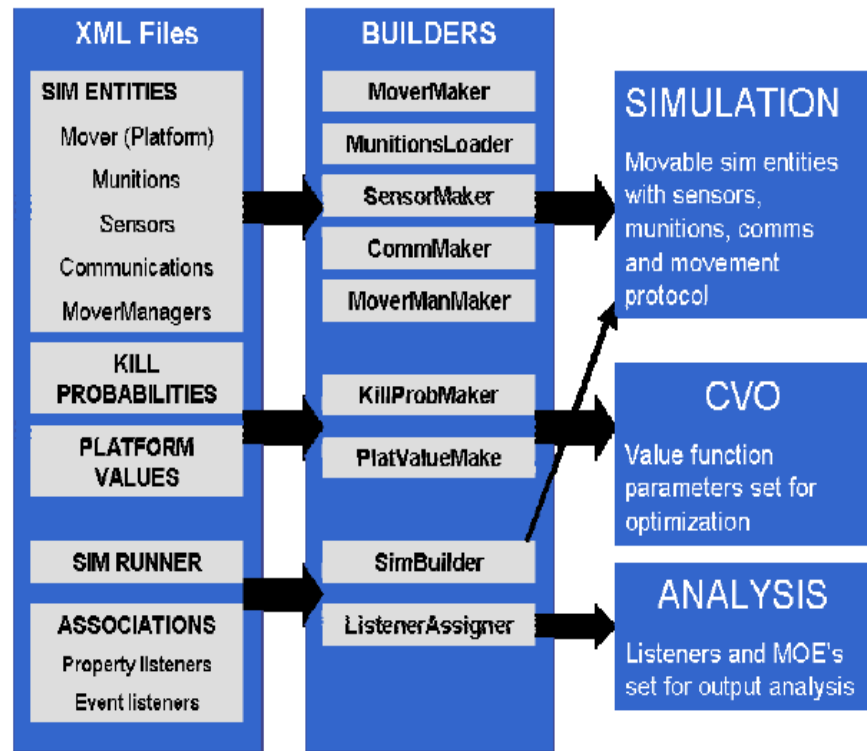


Figure 37. DAFS input design structure (from Havens, 2002)

4. Chapter Organization

This section provides a brief description of the combat simulation scenarios used in our comparison study between DTS and DES time advance approaches. Both of these simulation approaches modeled combat with a dynamic, stochastic, descriptive, high-resolution (i.e., individual to squad aggregate level), and constructive simulation form, but differ in their handling of simulation clock.

We analyzed eight scenarios of increasing complexity, which will be presented in four subsequent sections. First, we studied the simplest scenario for agent movements

(Scenario 1). We noticed that there were errors with movements to a single waypoint; consequently, we tested multiple waypoints to examine behaviors of an entity states change (Scenario 2). Multiple waypoint scenario was put in practice in a simple agent engagement (Scenario 3). Sensors detection capability was investigated under both of the time advance mechanisms (Scenario 4 and 5). Two emergent behaviors in combat models were then explored, namely the skipping phenomenon and the event ordering effect (Scenario 6 and 7). Finally, we combined all of the previous work into a more realistic combat simulation involving a naval combat model built in a Master thesis at NPS (Jacobson, 2010).

The modeling and simulation community understands that both DTS and DES approaches differ in the way models are constructed but believes that consistent results are attained and comparable recommendations are placed. All scenarios are simulated in a computer with processor speed 2.0 GHz with 2.0 GHz of RAM. Combat scenarios' input data are loaded in MANA models using GUI objects and XML files. The same input data are coded in JAVA for Simkit and Access data base for DAFS.

B. SECTION I: AGENT MOVEMENTS

As described in previous sections, combatant's movement is an essential element in combat models. The purpose of this section is to investigate the effect of TAM on this critical element by modeling simple combat scenarios using DTS and DES time advance approaches. Our objective is to answer the following questions:

1. Does the TAM affect the accuracy of agent movement and error "missed distance" of agents arriving at target waypoints?
2. Are the problems associated with movement in DTS related to the grid?
3. Are these problems addressed by the change to vector based movement in MANA 5, or are they more fundamental aspects of the TAM?
4. What other effects TAM has on the outcomes of agent movement scenarios?

1. Scenario 1: Agent Moving to A Single Waypoint

a. Simulation Methods

We used a single agent moving in a uniform and linear motion toward a single stationary target away from the agent's initial location at the opposite end of the simulation space. This simulation was built in MANA 4, MANA 5, and Simkit. The MANA and Simkit simulation environments are described in detail in the above section. In the MANA 4 environment, all movement is grid-based where the agents move directly from one cell to the next cell towards a destination waypoint. Thus, agent speed is partially determined by the size of the grid cells. Two speeds were tested in the MANA 4 environment: 25 meters per second, and 32 meters per second. Given the grid-based restrictions, it was necessary to alter both the grid cell size and the overall number of cells in the simulation "battle-space" to map changes in time-step size with agent speed. The starting location and target location distance were varied along with the cells in order to maintain a constant 10,000 meters distance traveled. The is equipped with a simple cookie-cutter sensor with a detection and classification range of 10,100 meters to ensure the agent acknowledgement of the target prior to movement. The time-step and event-driven models are compared varying the time-step size in increments of 2.0 seconds from 2 to 10 seconds with the addition of exploring smaller steps at 0.5 and 1 second to validate the model accuracy.

Time Step Size	Each cell is	# of Cells		Real World Max Range		Home location		
Δt (sec)	m	x	y	x (m)	y (m)	$x0$ (cells)	$x0$ (cells)	$y0$ (cells)
0.5	12.5	1000	1000	12500	12500	500.0	500	0
1	25	500	500	12500	12500	250.0	250	0
2	50	250	250	12500	12500	125.0	125	0
4	100	125	125	12500	12500	62.5	63	0
6	150	83	83	12500	12500	41.7	42	0
8	200	62.5	62.5	12500	12500	31.3	31	0
10	250	50	50	12500	12500	25.0	25	0

Table 10. Table used in time-step size conversion to grid-based in MANA 4 for agent speed of 25 m/s

Similar scenario in MANA 5 used a single agent movement speed of 12 meters per second. This speed was chosen to eliminate the effects of simulated mass in the MANA environment. In MANA 5, agent speed must be kept below certain thresholds otherwise the explorations of time-step size in the current simulation are affected by uncontrollable conflating factors that are inherent to the MANA modeling environment. In all simulation conditions, the agent moved at a constant fixed speed toward a fixed destination waypoint at 10,000 meters from the agent's starting location. Similarly to MANA 4, the agent is equipped with a cookie-cutter sensor with range of 10,100 meters to ensure target acknowledgement prior to movement. We compared the time-step and event-driven modeling environments by varying the time-step size Δt in the DTS simulations in increments of 0.2 seconds from 0.1 to 1.3 seconds to explore the impacts agent movements toward the target waypoint.

Since this simulation experiment is completely deterministic, we report results for single-run scenarios. In both the DTS and DES environments, the agent knows the exact location of the target destination in advance prior to movement. In the simplified physics model explained equations above, we do not account for acceleration or mass. In MANA 5, an agent mass of 0.1 is equivalent to instant move (McIntosh, 2009). In this model, agent start movement at their origin waypoint and moved directly towards the destination waypoint at the opposite end of the simulation space.

b. Results

For all simulation runs, we collected statistics about the time it took for the agent to reach the target waypoint "trip time," and the number of meters by which the agent passes or overshoots the target destination "miss distance" when this occurred.

Since the size of the time step is reflected on the size of the cell in MANA 4, agent movement was affected. We observed with both agent speeds (25 m/s and 32 m/s) that as the time-step size increases the greater the miss distance between agent final location and the target waypoint. This relationship is demonstrated in the Figure 38. Running the model with the DES approach produced the exact results as the analytical

solution since the same equations of motions are used to precalculate the agent final location and the time of event occurrence.

Altering the number of cells the agent can move in a single time step produce similar trip duration results, but smaller miss distances. The agent end location of the agent at the end of simulation was of the range of 20 meters away from the target waypoint location.



Figure 38. Two speeds agent miss-distance to a target location at various time-step sizes in MANA 4

For the scenario where agent movement set at 12 meters per second, the DES environment provided the shortest trip time to the target destination at 13.88 seconds, identical to the analytical solution for this movement. In the time-step environment, each increased time step window Δt extended the trip time for the agent. This is illustrated in Figure 39.

As the size of the time step increases, we observe an emergent phenomenon of agent movement in discrete time simulations. We see that the agent must “backtrack” to reach the target waypoint as the time step increases, although the agent will eventually reach the destination for time-step sizes under a critical threshold.

However, starting at a Δt of 1.3 seconds, the agent never stops at the target destination, instead oscillating back and forth between points on either side of the target waypoint forever.

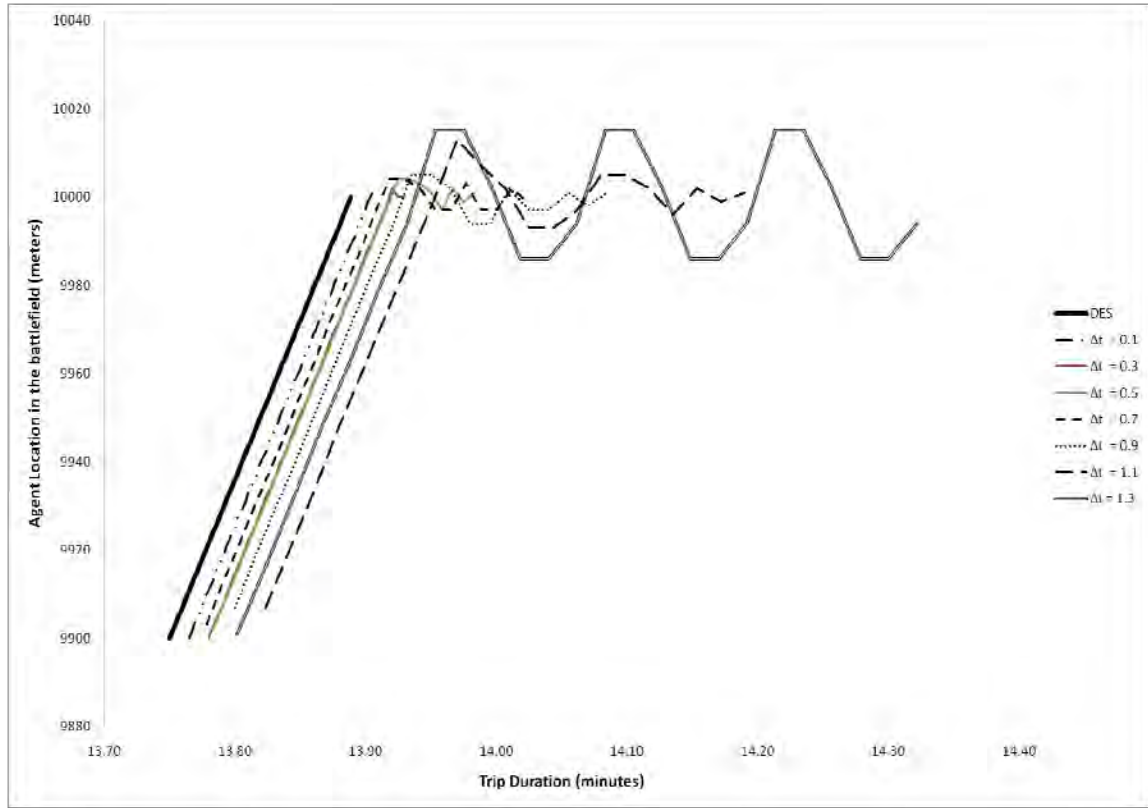


Figure 39. Agent locations and trip time over model runs from DES and DTS models in MANA 5 environment.

We also observe a monotonic increasing relationship between the time-step size Δt and the miss distance of the agents. Figure 40 below reports the cumulative miss distance for each DTS agent as a function of time-step size. The cumulative miss distance is calculated by adding together all miss distances travelled by the agent before reaching the target waypoint. That is, in simulations with the time step window below the critical threshold described above, the agents overstep (oscillate) to either side of the target at decreasing intervals before they reach the target destination. Both the number and distance of these oversteps increase as a function of increase time-step size Δt .

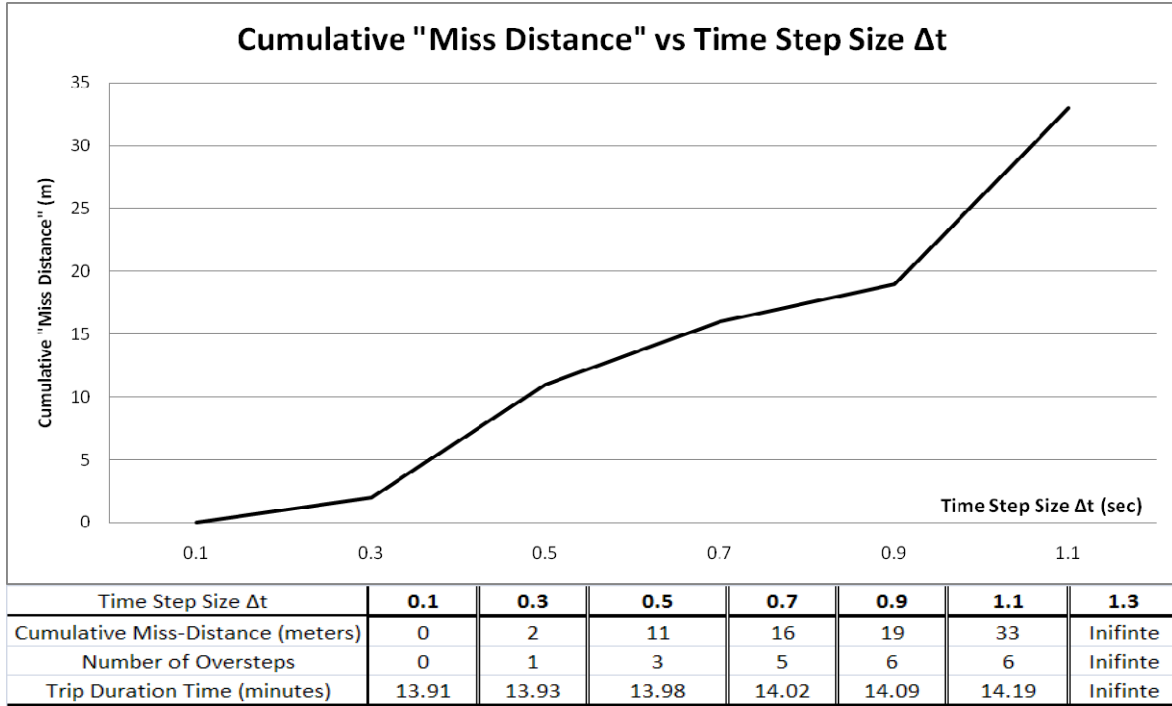


Figure 40. Agent cumulative miss distance from target location for multiple time-step sizes in DTS model in MANA 5 environment.

c. Discussion

There is always a miss distance between the agent end location and the target waypoint in when using the DTS approach. The size of miss distance depends on the size of the time-step size used in the model as well as the technique used to reflect the time-step size in MANA 4 version. Altering the cell size restrict the agent to be a cell distance from the next moving location, thus increased the miss distance as the cell size grows. Altering the number of cells for the agent to move at each time step reduced the miss distance. However, due to the penalty method used in MANA 4 the agent equation of motion was violated as the agent trimmed a number of cells immediately before the last movement to move to the target waypoint.

The choice of time advance mechanism in combat simulations greatly impacts agent movement as well as target location recognition. Moreover, the size of the time step Δt has important implications for the outcomes of combat simulations,

including the ability to generate emergent simulation behavior when Δt exceeds a critical threshold. In the current simulation, that critical threshold is $\Delta t = 1.3$, but this depends upon all time-sensitive elements of the simulation and almost certainly depends also on inherent properties of the simulation environment that are not under the direct control of the modeler. MANA 5 the vector-based version demonstrated that the errors in agent movement to a single waypoint are not due to grid approach limitations only, but also time step approach limitations

There is a direct relationship between time-step size and the amount of time it takes for an agent to reach a target destination. This is true even though the agents are moving at the same speed, and the destination is at the same distance throughout all simulation scenarios. Although the equation of motion is the same throughout, the trip time is impacted by the time-step size that is an element of the DTS environment. Given the effects of the time advance mechanism on even the simplest combat simulation scenario depicted here, we propose that the impacts of TAM choice could produce even greater and less understood affects on the more complex scenarios that are common in military M&S.

It is understood that in some agent-based simulation tools, an agent movement is terminated based on penalty decision of where is the target grid location. That is similar to agents movement in MANA 4 where the agent stops at the target location despite its speed. This might help to illustrate agents reaching locations but surely contains avoidable errors and generally is not used in combat simulation tools.

2. Scenario 2: Agent Moving to Multiple Waypoints

a. Simulation Methods

This scenario extends Scenario 1 by adding multiple waypoints to the experiment design. In this scenario, a single agent moves from a starting location to four waypoints distributed over the battlefield space. As in the last experiment, this scenario was modeled in the MANA 4, MANA 5, and Simkit environments in order to investigate the effects of TAMs. In all environments, the agent moves from the starting location to

the first waypoint at a speed of 32 meters per second, then moves to the second, third, and fourth waypoints at a speed of 8 meters per second. The agent slows movement speed solely for the purposes of simulating a standard search operation across multiple locations. The battlefield model is a square 8000x8000 meters, with 1000x1000 cells representing 8x8 meters for each cell in MANA 4, and all four waypoints were fixed at the same x,y-coordinates for all environments and all runs.

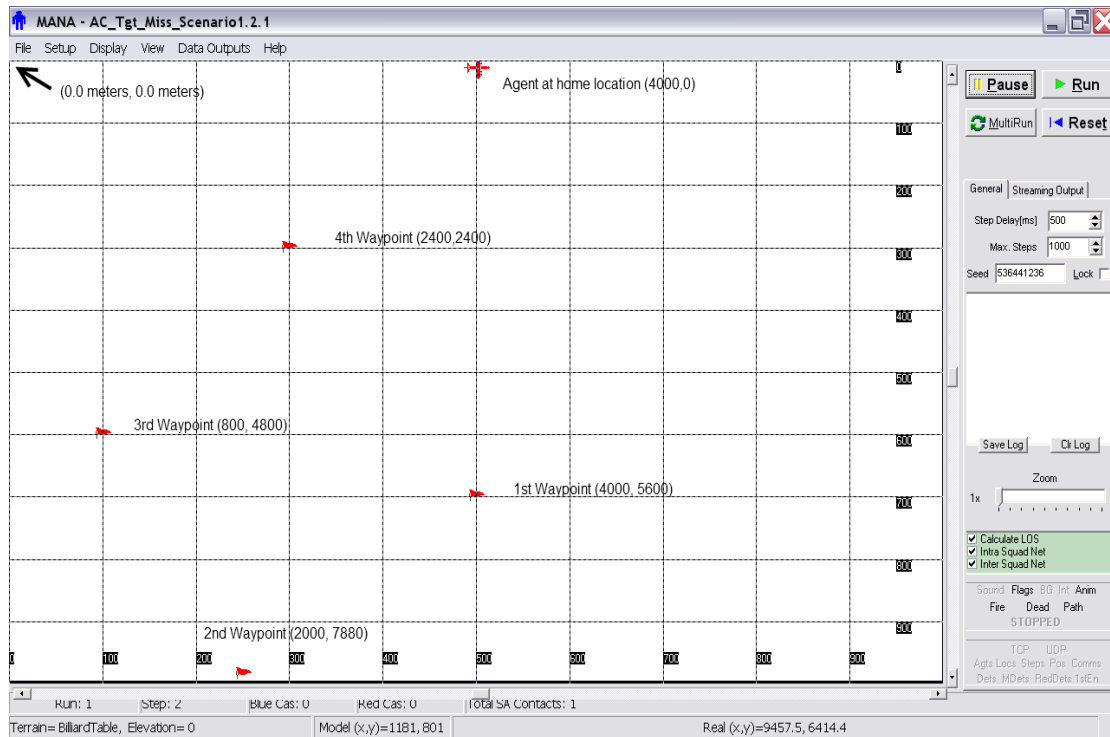


Figure 41. Illustration of Scenario 2: a single agent and four waypoints in MANA 4 environment

Like scenario 1, we used a simplified physics model that does not incorporate weights, drag, friction, or acceleration. Only one grid size setting was used in MANA 4 because testing grid size variations produced identical results for this model. In MANA 5 the time-step size was varied in increments of 0.2 seconds from 0.2 to 1.6 and results were averaged across 50 runs for each condition.

b. Results

For this scenario, we collected two categories of measures related to the effects of TAM on agent movement. We measured the time it took for the agent to reach each waypoint, and focused on the total time it took for the agent to reach the final waypoint, the ‘trip duration’. We also collected data related to the location and timing of state changes of agent speed. In this scenario, the agent is supposed to stop movement after reaching the last waypoint, concluding the simulation run. All agents in Simkit and MANA 4 stop moving after reaching the last waypoint. With larger time steps in MANA 5, the agent did not stop and rather oscillates around the initial target waypoint. In these cases, the simulation run continues until stopped by the experimenter. This oscillation behavior is explained in depth above in the results section of Scenario 1.

The analytical solution using the equations of motion described above produced a trip duration time of 22 minutes and 7.85 seconds (22.13 minutes). The results obtained within the simulation environments are compared to this solution.

Every simulation run in MANA 4 produced a trip duration time of 23 minutes and 57.4 seconds (23.957 minutes). The change of 1 minute and 49.55 seconds (1.826 minutes), represents a significant 8.2% difference from the analytical solution. We further noticed several phenomena related to the use of grids in MANA 4. The agent did not follow the prescribed path precisely as the grid limits movement in the 8 cardinal and ordinal (intercardinal) directions, choosing one at each time step. The MANA 4 manual claims that the errors introduced by these movement restrictions are corrected by approximation algorithms that should ameliorate errant velocities (McIntosh et al., 2007). However, these problems persist as demonstrated below.

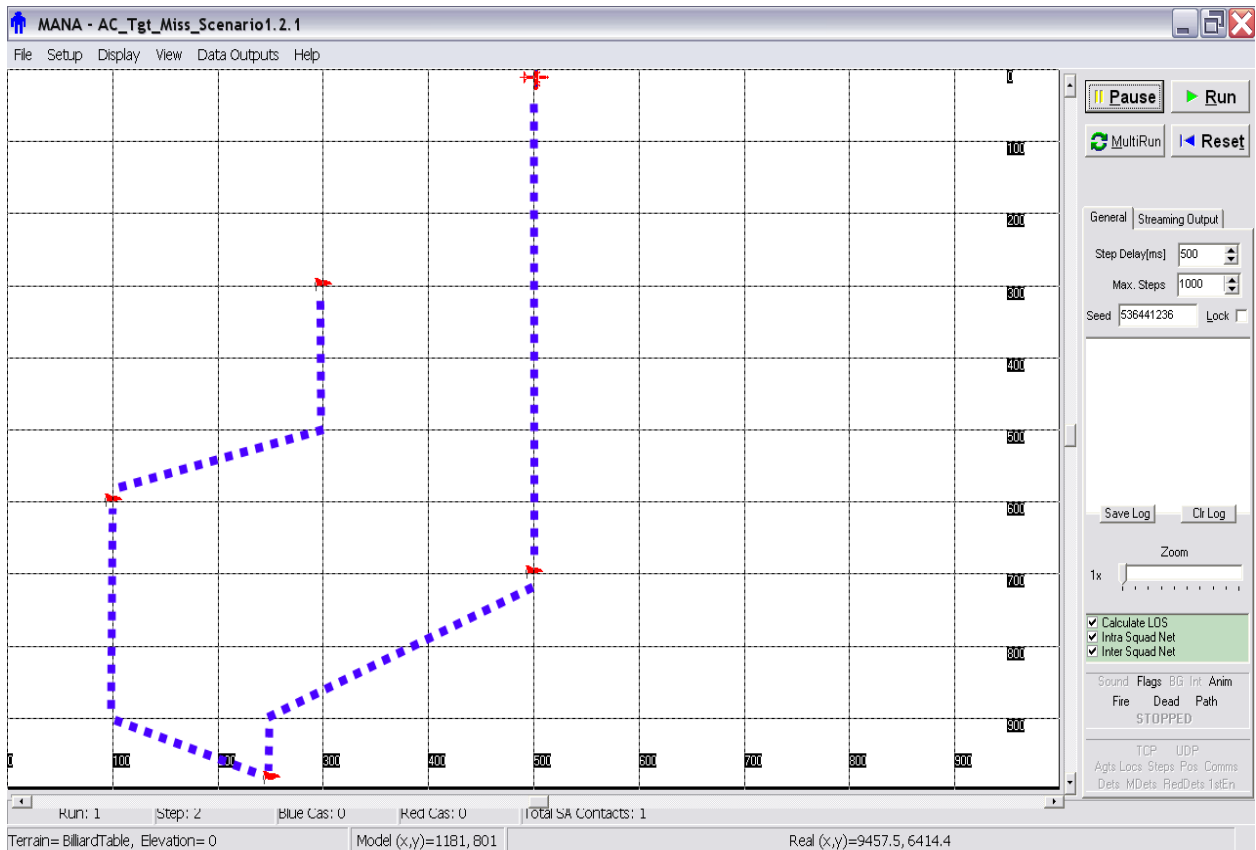
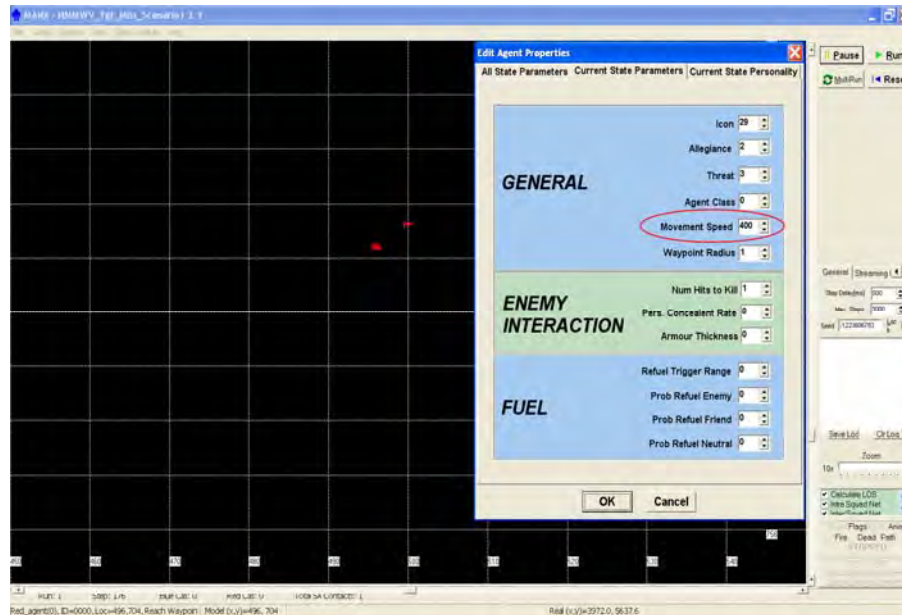


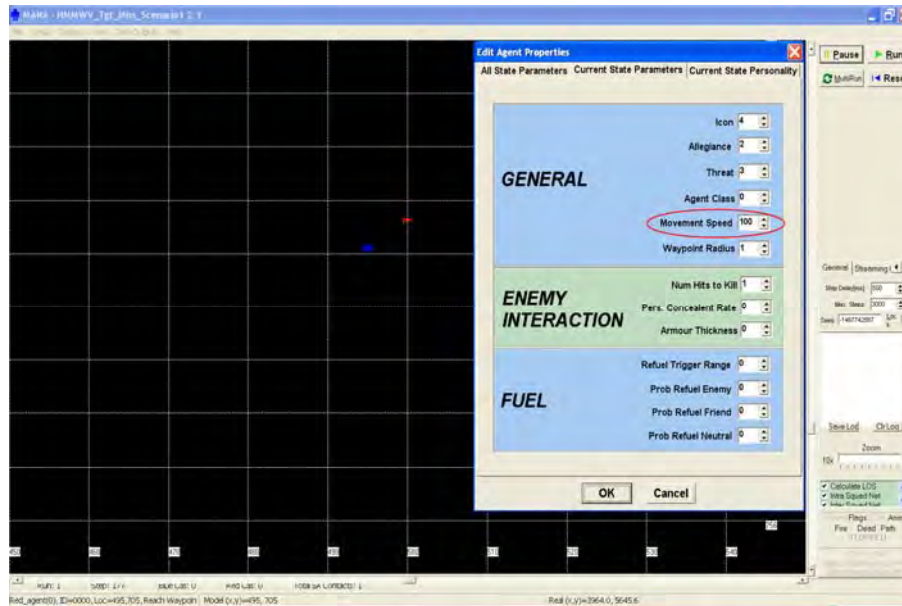
Figure 42. The agent's path traversed during Scenario 2 simulation execution in MANA 4

The movement restrictions introduced by the grid structure result in unrealistic paths, such as the “ladder movement,” that cause delays in both waypoint arrivals and speed state changes. The grid structure coupled with the discrete time method introduces additional errors to the timing and location of state changes, as compared with the analytical solution and other simulation environments. Figure 43 demonstrates the effects of these errors that are visible in both time and location. Because of these conditions, when the agent arrives at the waypoint location, the simulation does not recognize the arrival until the next time step. As a consequence, the agent reaches then overshoots the location of the waypoint in space, and changes all resultant states with a corresponding temporal delay. When an agent misses the waypoint via overshooting, the state transition is not triggered until a later time and different location. In most cases, the

arrival is recognized approximately one cell away from the target waypoint. Note that MANA 4 agent movement speeds are represented in terms of number of cells the agent can move across in a single time step. For example, a movement speed of 100 means 1 cell (100/100) per time step and 400 is 4 cells (400/100) per time step.



a)



b)

Figure 43. Illustration of state transition delay caused by the time step method in MANA 4 environment; a) agent speed is not changed after one time step from the first waypoint, b) agent speed changed after two time steps from the first waypoint

In MANA 5, the trip duration increases monotonically with time-step size increases until $\Delta t=1.6$ at which point the agent oscillates around the first target waypoint endlessly until the simulation run is stopped by the experimenter. Furthermore, the agent in MANA 5 oscillates around each of the three target waypoints for some duration before recognizing it and moving to the next. The amount of waypoint oscillation also monotonically increases with time-step size. This result is described in detail in the results section of the previous scenario.

The time-step size correlation with trip duration can be partially explained by the increasing amount of oscillation experienced by the agents at each target waypoint. The trip duration also increases monotonically with time-step size because the agent must wait for the next time step after reaching the waypoint in order for the arrival to be registered. Thus, these two effects combine to dramatically increase trip duration with each increase in time-step size. Figure 44 demonstrates this effect.

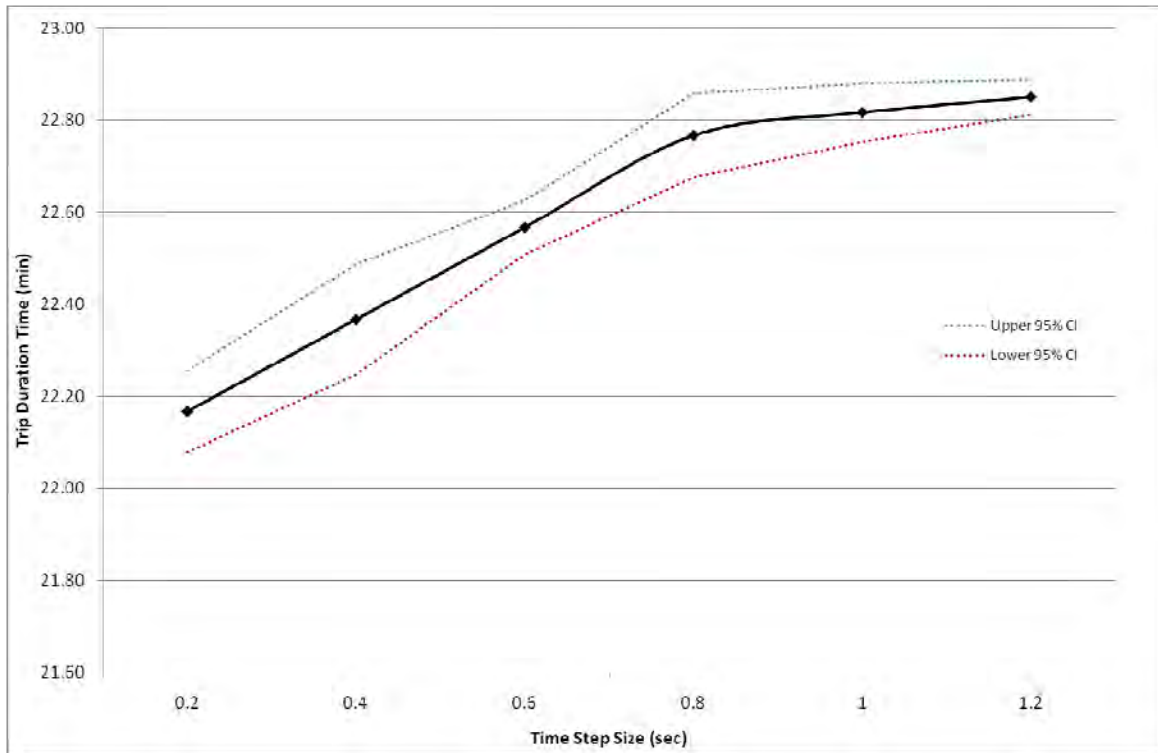


Figure 44. Trip duration versus time-step size results from MANA 5 implementation to Scenario 2

The results obtained by the DES environment are the same as the analytical solution, with a trip duration of 22.13 minutes. Likewise, the location and timing of state changes of agent speed take place without delay.

c. Discussion

While the grid structure in MANA 4 introduces compounding movement restrictions that affects the simulation results, the grid structure alone is not responsible for the observed errors in the DTS environment. Rather, the time-step mechanism also introduces compounding errors (as seen in the MANA 5 results) that combine with the errors generated by grid movement in MANA 4. Since MANA 5 uses vector-based movement, we can differentiate between the errors resulting from a grid structure and those resulting from the time-step mechanism in the simulation when these effects are absent. We did not directly test for compounding error effects generated by the vector-based movement, but any errors introduced by this movement technique are a product of the equation of motion directly and will thus be equally present in the analytical and DES solutions.

The nature of the errors resulting from grid-based movement suggests that increasing path complexity will generate increased errors, regardless of the time-step size. That is, a greater number of target waypoints will lead to greater overall error in the simulation with discrete time advance approaches. Furthermore, many simulations are designed such that state changes are dependent on the confirmation of arrival of a moving agent at a target waypoint. This scenario demonstrates that DTS approaches cannot be relied upon to realistically describe these conditions given the delays and errors introduced by the TAM. Changing agent or simulation states at the wrong time or wrong location will affect other components waiting for these updates. In particular, as the complexity of the simulation increases and more state changes need to be represented, the errors introduced by this TAM compound as each dependent arrival confirmation and state change become delayed. This ‘cascading delays’ phenomenon can dramatically affect the final outcome of the simulation, and the recommendations or decisions made as a result, as demonstrated by the next scenario.

3. Scenario 3: Simple Agent Engagements

a. Simulation Methods

Scenario 3 builds upon the results from Scenarios 1 and 2 by extending the simulation design to the operational environment on a 2000 x 2000 meter battlefield space with 3 heterogeneous agents and 4 waypoints in a simple agent engagement. A single blue agent moves in a uniform linear motion and proceeds in sequence to two target waypoints to engage with an agent at each location. The blue agent moves from a starting location to engage with a stationary agent at the first target waypoint and determines the identity of this waiting agent as friendly, enemy (red), or neutral (yellow). The blue agent will fire on a red agent until one is destroyed, and inspect a yellow agent for 30 seconds before moving on. In all cases in this scenario, the agent waiting at the first target waypoint is neutral (yellow), and the blue agent then progresses toward the second target waypoint

A red agent located at the second target waypoint will fire on the blue agent upon detection. As the blue agent moves toward the red agent, both have the opportunity to detect, classify, and fire on the other. Both the blue and red agents have the same speed, weapons, fire rate, sensor range, sensor type, probability of detection, probability of hit, and probability of kill shown in Table 11. The red agent, unaware of the blue agent, waits at the second target waypoint for fixed duration of time, and then proceeds to the third target waypoint (towards south of the red agent path) in the simulation. The waiting duration of the red agent is used to simulate operations such as receiving information, delivering supplies, or refueling.

Entity	Speed	Sensor Type	Detection & Classification Range	Weapon Type	Weapon Range	Probability of kill hit
Friend (blue)	10 Km/h ~ 2.78 m/s	Cookie-cutter	20 meters	Direct fire	500 meters	1.0
Enemy (red)	30 Km/h ~ 8.34 m/s	Cookie-cutter	20 meters	Direct fire	500 meters	1.0
Neutral (blue/yellow)	0 m/s	NA	NA	NA	NA	NA

Table 11. Agent input parameters used in Scenario 3 simulations

In the discrete time environment the time-step size was varied from .5 to 4.5 seconds in increments of 0.5. All movement takes place in uniform linear fashion along predefined paths and the simulation does not account for terrain or other environmental effects. Like the other scenarios, these simulations utilize a simplified physics model. Scenario 3 was modeled using MANA 5 and Simkit. We performed 50 runs for each time-step size, as well as the DES model.

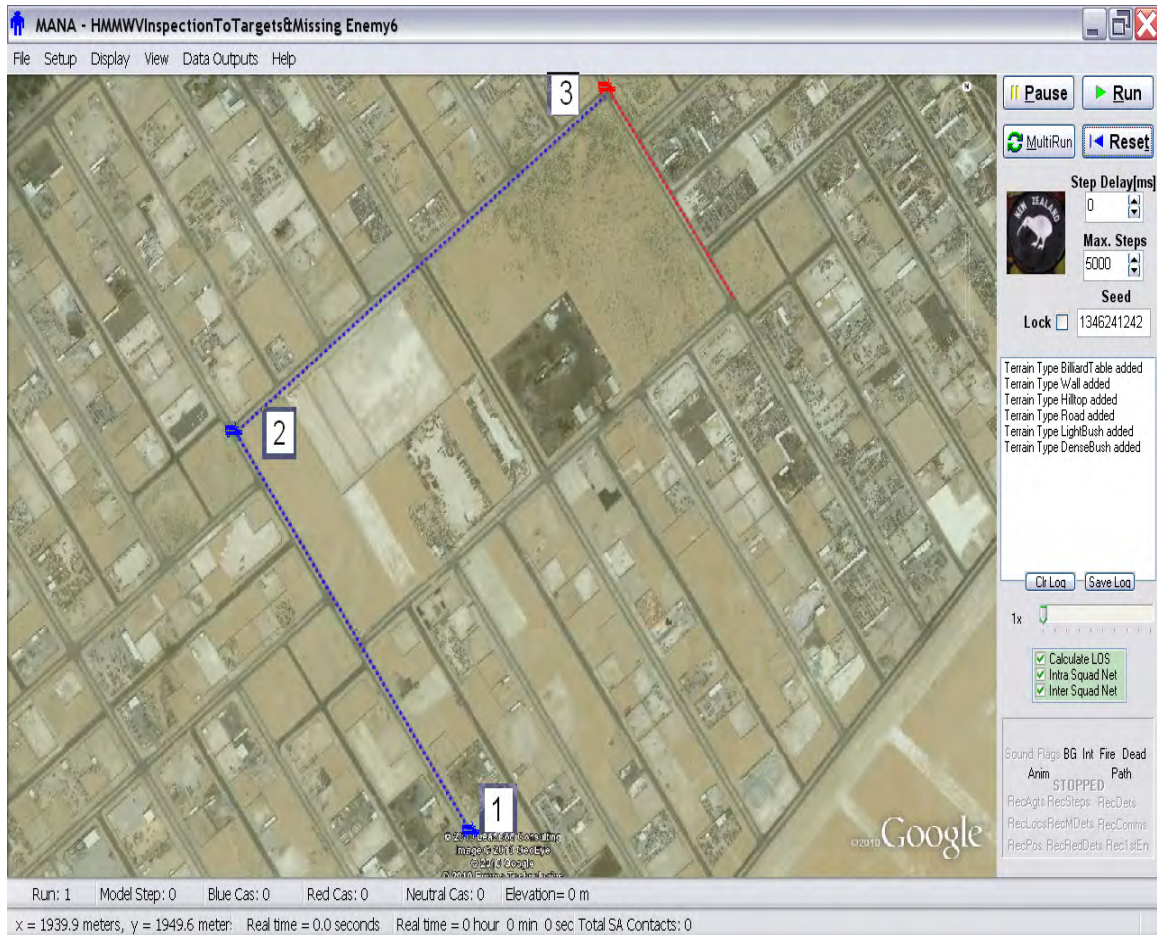


Figure 45. Simple agent battle initial settings in MANA 5 GUI display, where 1 is the blue moving agent, 2 the neutral agent and 3 is the enemy (red) agent. Dotted lines show moving paths [best viewed in color].

b. Results

We measured the blue agent and red agent casualties for each simulation run. In the DES model, the blue agent moved with a constant speed of 7 MPH (2.77 m/s), detected the first target (yellow agent) at 20 meters range, and arrived at this target waypoint at 6.579 minutes (394.75 seconds). Following the 30 second inspection time, the blue agent then proceeded to the next target waypoint at the same speed. The second target was also detected at 20 meters range and combat started between blue and red agents at 13.667 minutes (820 seconds). With 50 replications, each agent in the DES environment had an equal (50%) chance of win.

In the discrete time MANA 5 environment, we obtained similar results to the DES model for time-step sizes between 0.5 and 2 seconds. Time-step sizes larger than 2 seconds, however, introduced a new phenomenon into the simulation where neither the blue nor the red agent was killed during some simulation runs. At a time-step size of 3 seconds, 4% of the simulation runs resulted in no casualties. At a time-step size of 4.5 seconds, there were no casualties in 100% of the simulation runs. The results obtained at time-step sizes of 5 and 5.5 seconds were identical to those obtained at 4.5 seconds. This phenomenon is displayed in Figure 46. At time-step sizes of 6 seconds and greater, the blue agent oscillated around the first target waypoint and the simulation run had to be stopped by the experimenter. This behavior is explained in detail in the discussion accompanying Scenario 1.

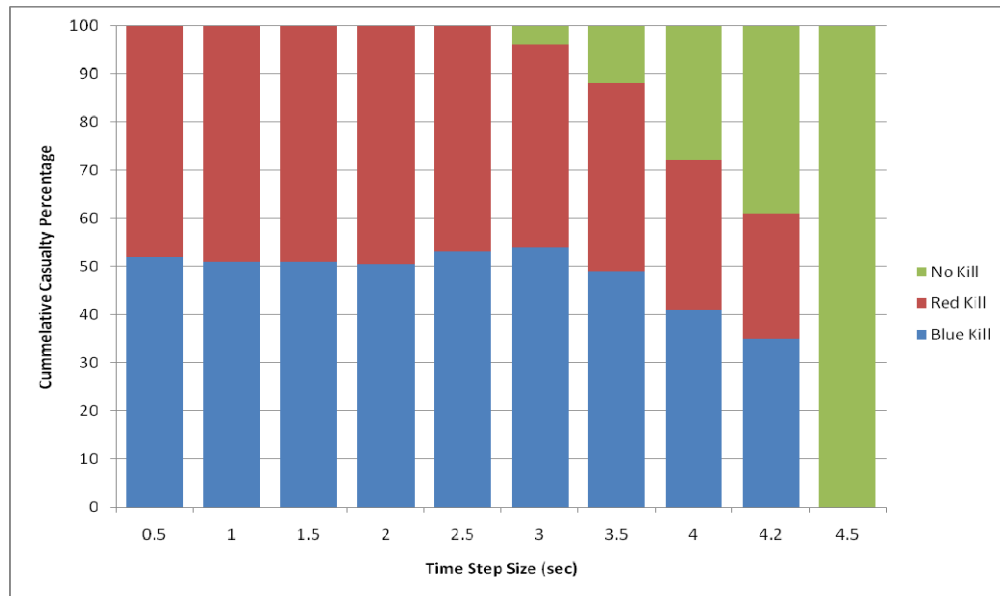


Figure 46. Illustration of the cumulative casualties between the blue and red agent

c. Discussion

For all time-steps sizes the detection of the first (yellow) agent was identical in time, 6.42 minutes (385 seconds) in to the simulation run, and occurred at the same location in the battle space. However, the arrival time at the first target waypoint does significantly differ as a result of time-step sizes. This effect is explained in depth in

Scenarios 1 and 2. As a result, the detection of the red target is also increasingly delayed as time-step size grows. However, since the red agent waits a fixed amount of time before leaving the second target waypoint, these delays sometimes resulted in the blue agent not engaging the red agent.

Many combat simulations are used to estimate and suggest improvements to the blue probability of success. This experiment demonstrates that the time advance mechanism utilized in such a simulation can drastically impact the subsequent estimates and suggestions. The results obtained from the DES approach suggest that improving sensor and weapons capabilities to enable the blue agent to engage earlier (or at a greater distance) are key factors to improving the chances of success. Conversely, the DTS approach here suggests that the overall speed of the agent, rather than the engagement capabilities, are key. The results from MANA 5 also suggest that reducing the inspection time of the blue agent will benefit the ratio of success for this scenario whereas the DES approach does not point to this change.

C. SECTION II: SENSORS AND DETECTION

In this section, we explore the TAM effect on another fundamental combat modeling element. Sensor detection behavior is examined in two simple combat scenarios modeled using both approaches DTS and DES. This section is tend to answer the following questions:

1. Will the modeling of sensor range and type differ between TAM approaches?
2. How sensitive is the response of sensor range changes between TAMs?
3. Will each simulation respond equivalently to the changes in sensor type?

1. Scenario 4: Changes to Sensor Ranges

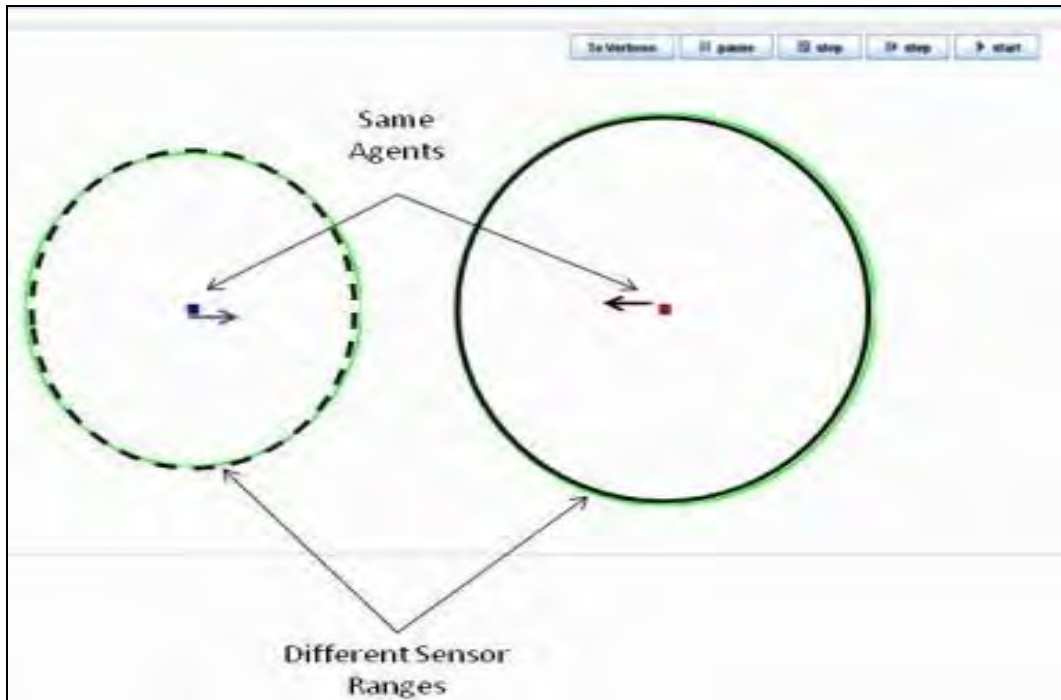
a. *Simulation Methods*

In Scenario 4 we extend the previous scenarios to investigate the effects of changes to the sensor range of agents, specifically looking at the impacts of these changes on model accuracy across TAMs. This experiment uses two agents, a blue agent and a red agent, that are identical in all respects except for sensor range. These agents move in a uniform linear fashion at 6 meters per second towards each other and engage in direct fire when the other is detected. For the red agent, we varied the sensor range in increments of 1 from 30 to 60 meters, while for the blue agent the sensor range was kept at a constant 30 meters in all simulation runs. Thus, there were 30 scenario conditions, each of which was replicated 500 times in both the DTS and DES frameworks, for a total of 30,000 simulation runs for this scenario. We collected metrics on the number of casualties for each type of agent across all 30 scenarios conditions to elucidate the impact of sensor range changes.

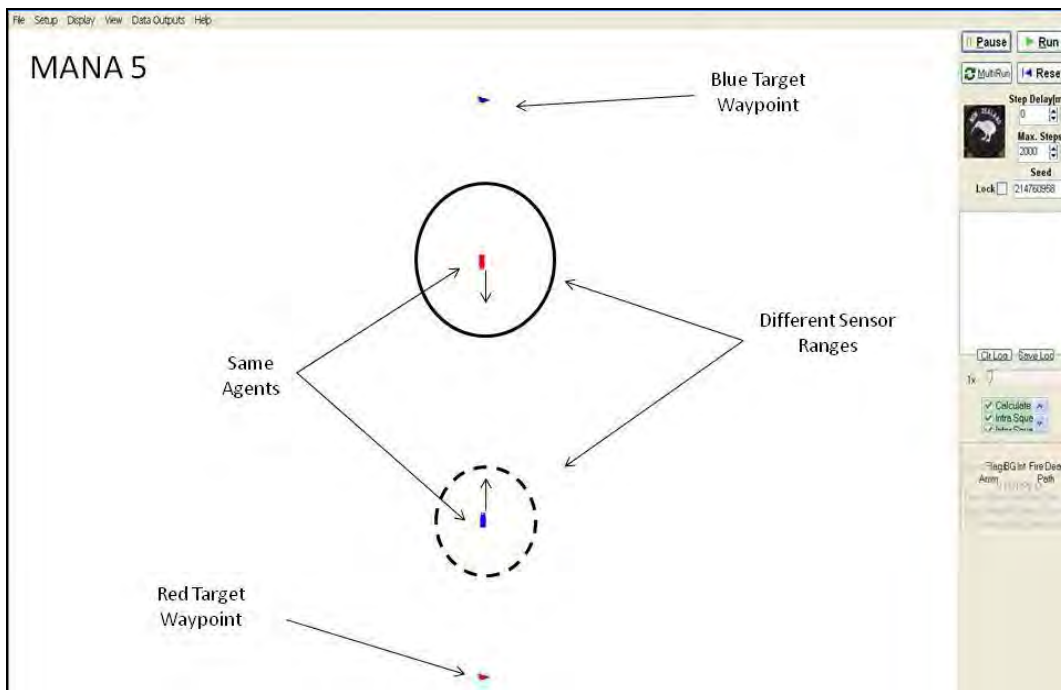
This experiment was performed in the MANA 4, MANA 5, and Simkit environments. For MANA 4 and MANA 5, the time-step size was varied from 0.1 to 1.1 seconds in increments of 0.5. The initial time-step Δt of 0.1 seconds was selected because it is the smallest Δt available within MANA 5 (McIntosh, 2009), and was specifically selected to minimize the likelihood of DTS discretization error. In all simulations, the agents move from their origin waypoint directly towards the other agent's origin waypoint at the opposite end of the simulation space. Thus, agents directly encountered each other during every simulation run without exception. In every run, either the red agent or the blue agent was killed. Recall that this was not the case in Scenario 3. Likewise, although both agents fire their weapon in some runs, there were no runs in which both agents were killed.

The effective probability of kill P_k for both agents is 0.1 (10%) with a firing rate of 5 shots per second (0.2 seconds between each shot). The low P_k allows us to observe more closely the agent behavior at every sensor range value. The determination of a firing rate in Simkit was handled by introducing a time step delay generated from a

uniform distribution with a mean equivalent to the time step in MANA 5. This method allows for a direct comparison by compensating for disparate rates of fire.



a)



b)

Figure 47. Simple agent combat in a) Simkit environment and b) MANA 5 environment

b. Results

In both the DTS and DES environments, when both agents had the same sensor range (i.e., 30 meters), they had an equal chance of killing the opposing agent, as seen in Figure 48. For all DES simulation runs, a change in the red agent's sensor range had a corresponding effect on the probability that the red agent would engage and kill the blue agent. In the event-driven model, even a modest increase in the red agent's sensor range had a positive impact on the probability that the red agent would successfully engage and kill the blue agent. For example, with the DTS approach the impact of increasing the red agent's sensor range on the red agent winning probability was only noticeable after 10 % change.

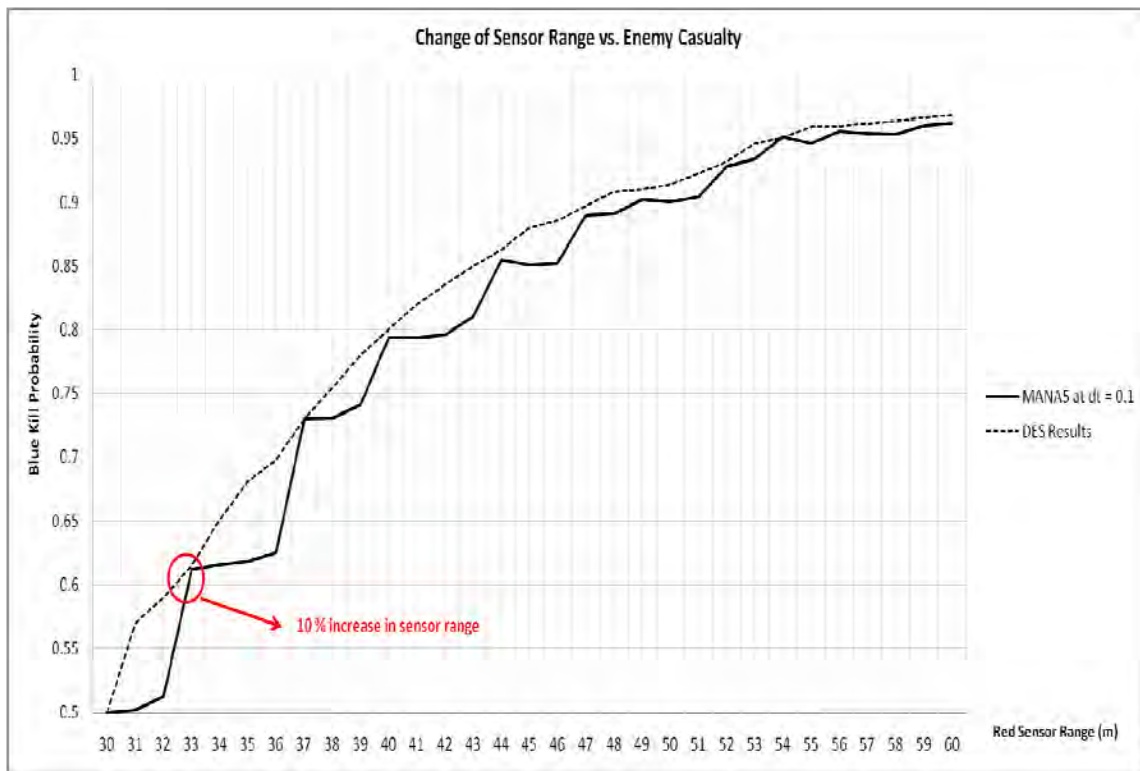


Figure 48. Time-step size impact on sensor range change in two agents combat simulation model

It was also observed that as the time-step size increases, the area difference between DES and MANA5 (MANA4 follow the same pattern) results increases showing a delay response of the sensor detection. The accumulative difference between these increases monotonically resulting in greater inaccurate simulation outcomes even at small time-step sizes.

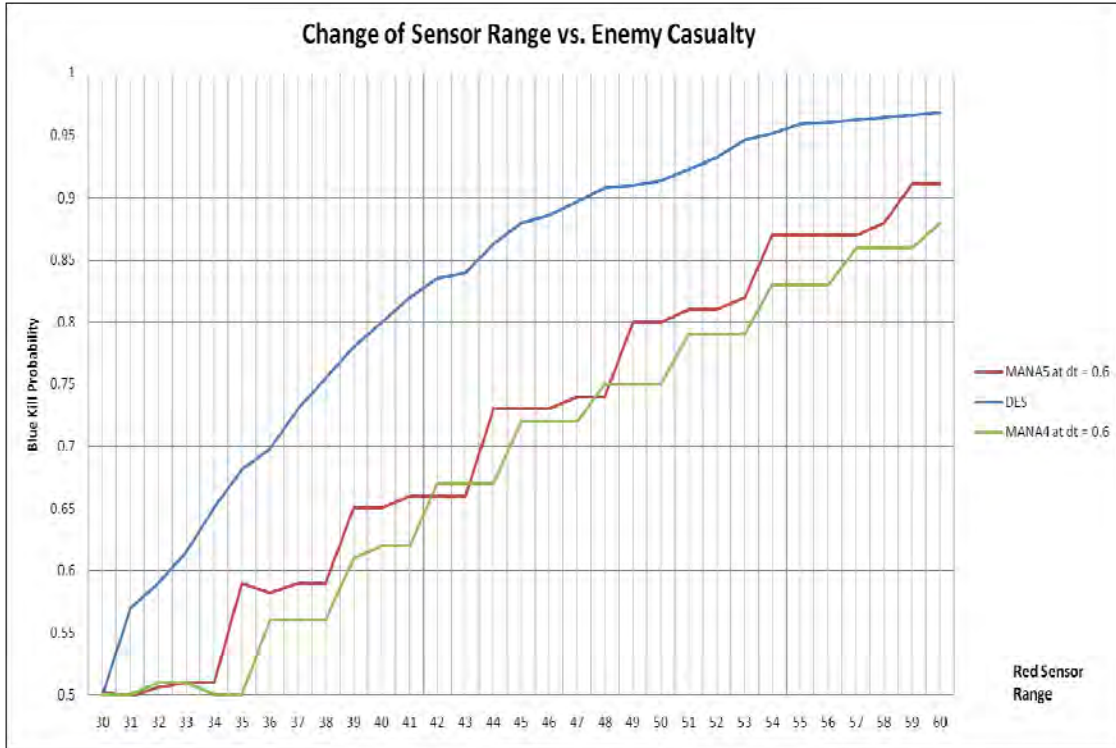


Figure 49. Comparison of red agent's sensor range change effect on blue kill between Simkit (DES), MANA5, and MANA4 (DTS) results

In MANA 5, the relationship between sensor range and kill probability was much more erratic. Sensor range changes appeared to impact kill probability in the DTS environment in a more step-wise fashion. The incremental changes to sensor range exhibit a non-monotonic relationship where increases in relative sensor range had no impact and sometimes actually decreased the average effectiveness of the agent.

c. Discussion

The time advance mechanism in combat simulations clearly affects the accuracy of modeling results having to do with changes in sensor ranges. Given these

results, it appears that discrete time simulations often fail to record important impacts of sensor range changes on the behavior of agents in a combat simulation. This has significant implications for the use of combat simulations in decision-making contexts. Recommendations from simulation results made regarding changes to sensor ranges in operating environments may be flawed depending on the time advance mechanism chosen.

As an example, suppose that an operational commander would like to increase the probability of detection for a combat element by 25%. The results here would indicate that using a DES model would suggest that small increases in sensor range would achieve the improvements. Although this may not necessarily be the case given the true complexity of the operational environment, the discrete time model clearly indicates that such improvements are not possible to obtain from small sensor changes. Instead, the DTS model is likely to suggest that a potentially suitable sensor range change would have little or no impact on combat effectiveness. Although simulation models are typically just one element in a decision-making process, it is important that they convey as accurately as possible potential effects. Nevertheless, since these types of simulations are being increasingly relied upon for operational and acquisition decision-making, the choice of time advance mechanism in simulation may have a large impact on combat effectiveness as well as the final costs of military operations.

2. Scenario 5: Sensor Type Changes

a. Simulation Methods

Scenario 5 investigates the influence of implementing DTS and DES approaches on the simulation results testing 2 different sensor types, the standard “cookie-cutter” (CC) sensor and the “cake” type of sensor. A CC sensor is modeled as a circle where there is a constant 100% probability of detection P_D for all entities that enter the radius. A cake sensor is modeled as multiple concentric circles where each region has an individual variable P_D that exponentially increases as entities move closer to the sensing agent. This can be thought of as overlapping concentric constant-rate sensors. In

all runs for this scenario we use one stationary “target” agent, and one mobile sensing agent that moves toward the target agent at a fixed rate of 10 meters per second. The scenario was implemented in Simkit and MANA 5 where we varied the time-step size from 0.5 to 10 seconds in increments of 0.5 seconds.

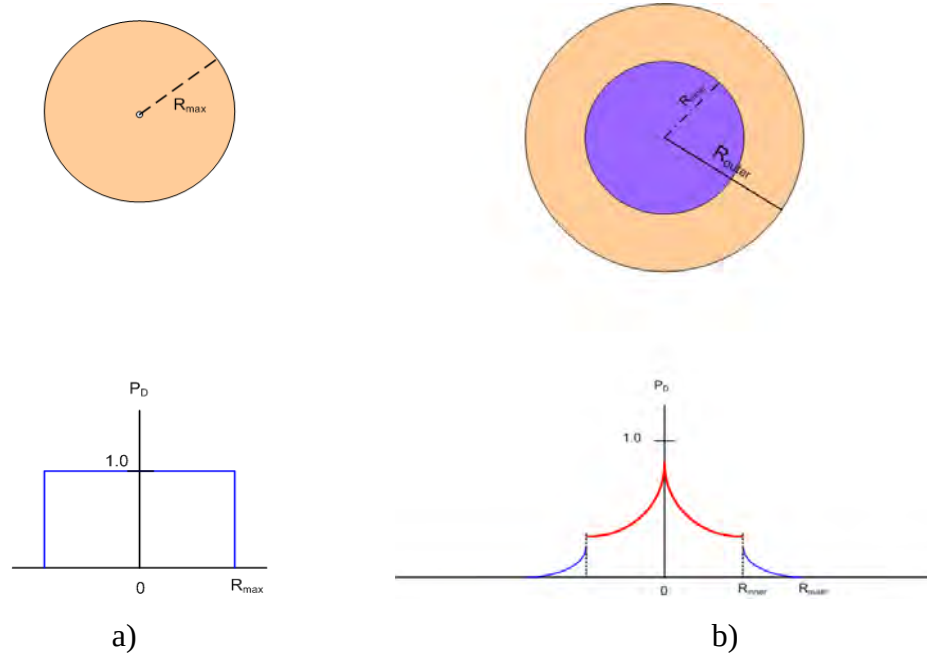


Figure 50. Typical sensor types used in combat scenarios, a) Cookie-cutter and b) two ranges constant-detection-rate Cake sensor

First, we considered an agent with a CC sensor with a 100 meter range. Distance between target and sensor is fixed at 1140 meters. We examine two conditions testing the effects of target-sensor intersections and computed the expected time to detection as measure of performance. We examine the effects of a “heads-on” approach where the target is directly in the search path (Figure 51-a), and a “lateral range” geometry test where the target intersects with the agent sensor at a set distance from the sensor center (Figure 51-b).

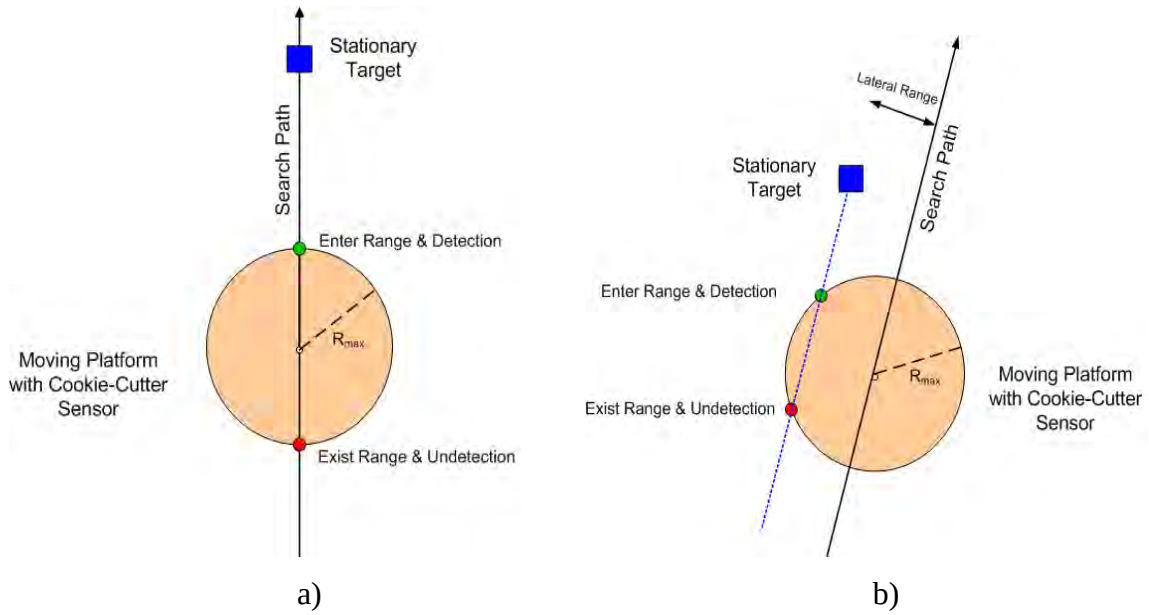


Figure 51. A demonstration of sensor-target interactions for a Cookie-cutter type sensor and a single stationary target, a) without target lateral range and b) with target lateral range

In the second experiment, we replace the CC sensor with a cake sensor, keeping everything else constant. Each “ring” of the cake sensor has an independent probability of detection that follows an exponential distribution. This type of sensor model is used in combat simulations to better approximate real sensors where the probability of detection increases as targets move closer to the center of the sensor (Figure 52). In this scenario, all targets were placed exactly 1 meter outside of the sensor range. This allows us to report the exact time of detection for a simplified structure. Here we examine both the “heads-on” and “lateral range” tests as before.

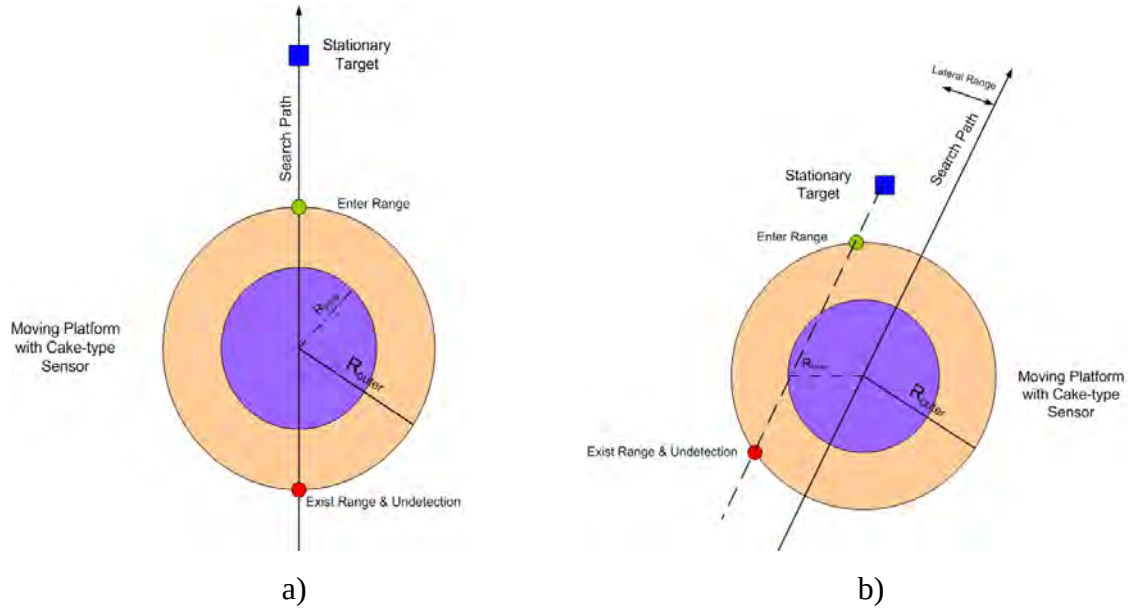


Figure 52. A demonstration of sensor-target interactions for a 2-ranges Cake type sensor and a single stationary target, a) without target lateral range and b) with target lateral range

	Inner	Outer
Range (meters)	400	600
Average Time Between Detections (seconds)	2	10

Table 12. Two-ranges “Cake” type sensor input parameters

Typically, the expected time to detection $E[TTD]$ for a constant rate type sensor is computed from determining the probability of detection for a given constant rate of detection (i.e. time between detections) using the following equation:

$$P_D = P \{t_D < t\} = 1 - e^{-Ct} \quad (31)$$

where, C is the expected time between detection.

For the DTS model, the $E[TTD]$ is computed by sampling from a Bernoulli trial with P_D as the probability of success (Washburn and Kress 2009). The analogy in the DES model is to sample from an exponential distribution with mean C to

determine time till detection given that a detection will occur (Buss, 2011). The two rings “cake” sensor Mediator (see Figure 53) was implemented in Simkit and linked by the LEGO methodology to the rest of the combat components.

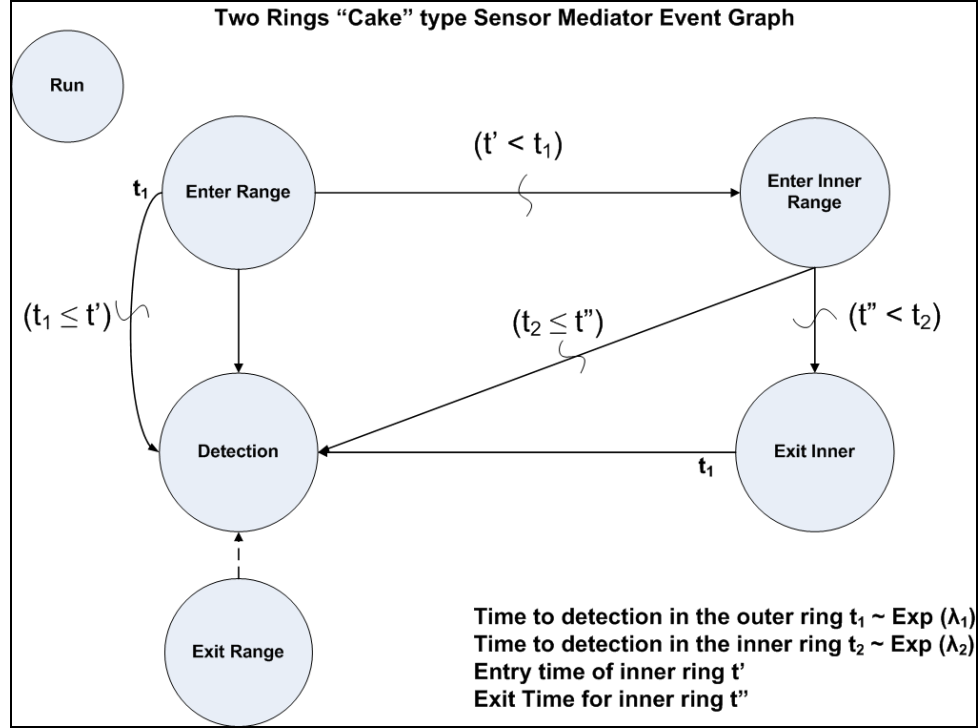


Figure 53. Two ring “Cake” type sensor Mediator Event Graph

For both experiments, we performed 200 runs of all TAM conditions for both sensor types for both the “heads-on” and “lateral range” tests. This resulted in 8800 total simulation runs, 8000 MANA 5 runs, and 800 Simkit runs. Every simulation stopped once the moving agent detected the target.

b. Results

Our results show that the expected time to detection $E[\text{TTD}]$ varied significantly as a condition of both sensor type and time-step size, as well as TAM approach. The DES result shows $E[\text{TTD}]$ at 104 seconds for the CC tests, an exact match with the analytical test, and this is used as a benchmark number for the comparison of TAMs and time-step sizes. In the MANA 5 runs, the CC sensor $E[\text{TTD}]$ increases as the

time-step size increases identically for both the “heads-on” and “lateral range” tests. Figure 54 shows the increasing difference between the DES result and results obtained from the DTS environment at the complete range of time-step sizes tested. At the smallest time-step size of 0.5 there was a detection delay of exactly 1 second.

Interestingly, the increasing difference is not linear with time-step size increases. Rather, changes appear in a step-wise fashion where $E[TTD]$ differences are only affected following the increase of multiple time-step size increments. The maximum difference between the DES and DTS results plateaued at 4 seconds for the time-step sizes between 8.5 and the maximum 10 seconds. All standard deviations for MANA 5 results ($E[TTD]$) were small, between 0.06 and 0.11 seconds. As explained before, the deterministic DES environment runs resulted in a standard deviation of 0.0, because the CC sensor P_d is always 100%. The same behavior observed when the target placed at a lateral distance from the center of the sensor with a detection delay of 1 second from the previous settings.

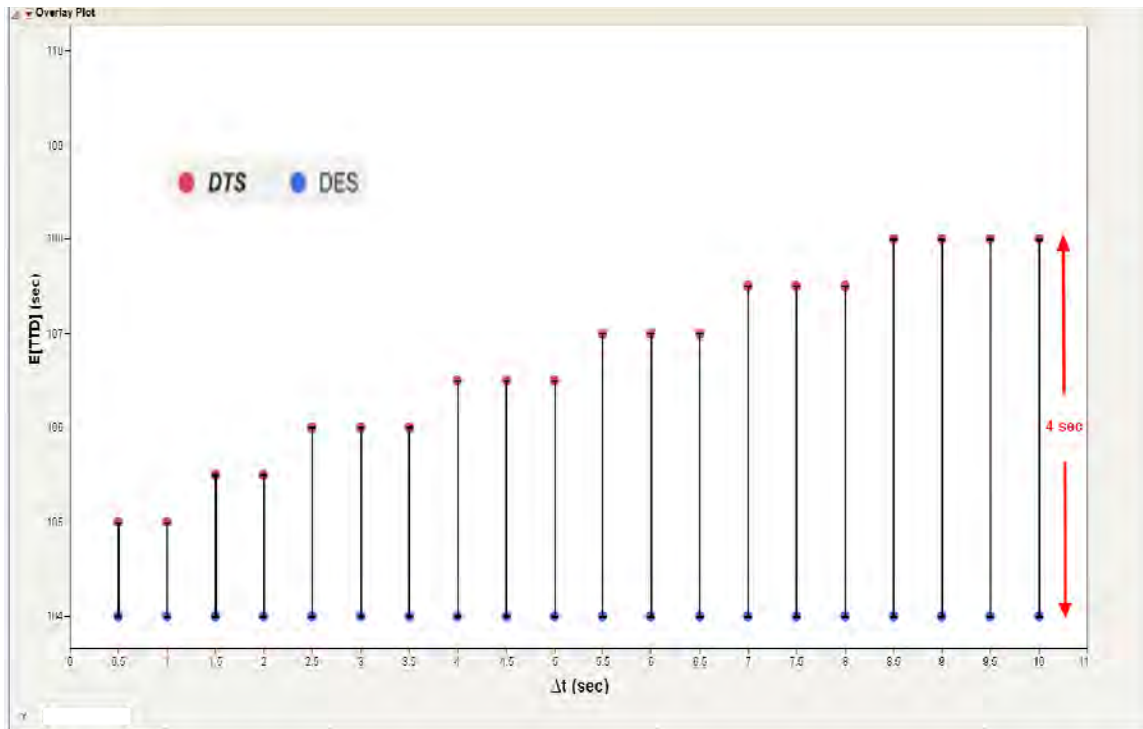


Figure 54. Cookie-cutter type sensor results with 4 seconds difference between DES and DTS approaches

Likewise, the $E[TTD]$ difference between the DES results and the MANA 5 results increase as time-step size increases in both the “heads-on” and “lateral range” tests. The DES result is $E[TTD]$ at 9.12 seconds with a standard deviation of 5.66 seconds. Again, we use the DES result as a tool for comparison with results derived from the MANA 5 runs across varying time-step sizes. At the smallest time-step size of 0.5 seconds, the DTS environment produces a $E[TTD]$ at 11.94 seconds with a standard deviation of 5.69 seconds. This difference of 2.82 seconds falls within the standard deviation of the DES results. At a time-step size of 2.0 seconds, the DTS results are increasingly beyond 1 standard deviation of the DES results reaching to a difference of approximately 26 seconds at time-step size of 10.0 seconds. Even if we assumed the difference between time-step sizes 10 and 2 seconds, there still be approximately 21 seconds glitch in sensor detection time. This is shown in Figure 55 below.

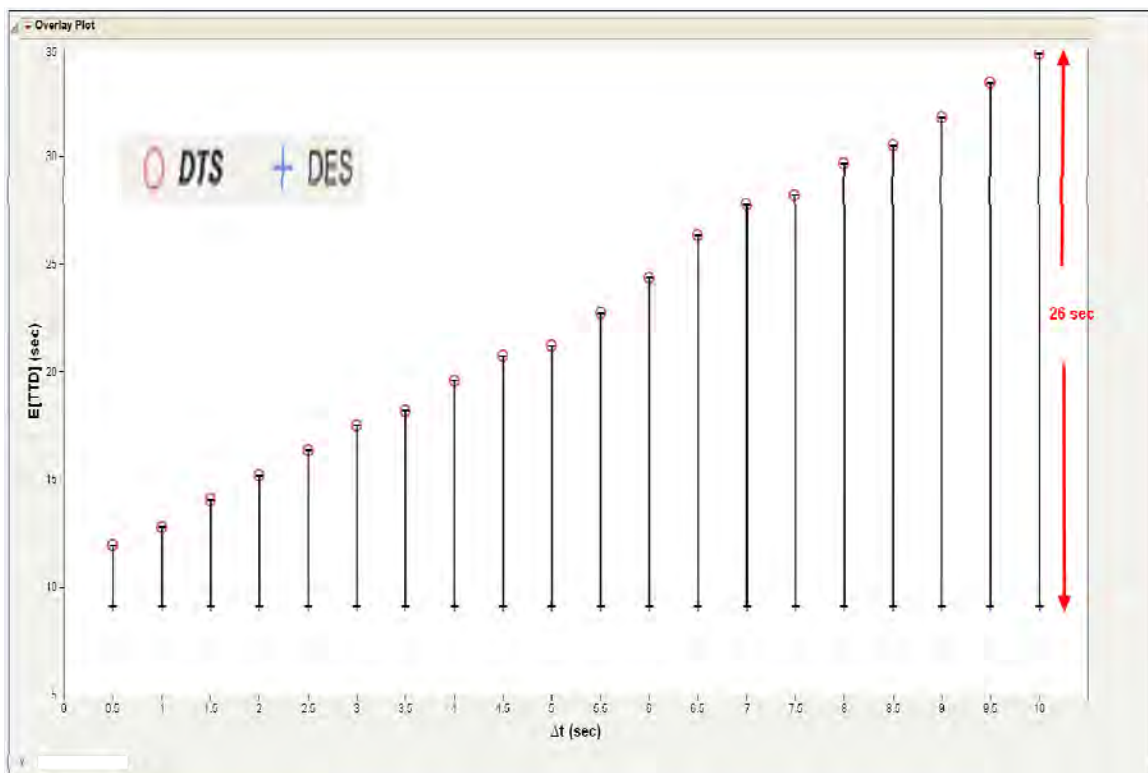


Figure 55. A two-ranges “Cake” type sensor results of the difference between DES and DTS approaches

The same behavior observed from the geometry test when the target lateral distance existed, however, with greater detection delays than the results observed above.

c. Discussion

Since the DES environment uses deterministic processing, the target is always detected at the moment it enters the range of a CC sensor. Given that it is often unrealistic to assume that a target will be detected with 100% accuracy at the absolute edge of the sensor range, we argue that the probabilistic DTS environment provide a more believable representation of reality for these sensors. This does not mean that MANA 5 produces accurate results, however. When a more realistic sensor is used, such as a cake sensor models here, there are numerous problems encountered with the DTS approach. In general, DTS fails to capture the behavior of more complex sensors. Specifically, DTS cannot effectively deal with sensor probabilities of detection other than 100% because of the conflating factors affecting simulation timing.

For instance, with a single constant-rate sensor $E[TTD]$ of 6 seconds, further tests show that the average result in the DTS environment fails to converge to the accurate $E[TTD]$ with time-step sizes over 9 seconds even after a large number of runs. The MANA 5 manual claims that the software will automatically adjust the necessary components of the simulation, such as time-step size, to account for changing sensor probabilities. However, this claim does not appear to be supported by the data. The problem here appears to be the lack of a defined method to calibrate cake type sensors. In particular, the deficiency is apparent with multiple range sensors acting in environment with larger time-step sizes. In the DTS approach, there is no established method for keeping sensor detection probabilities the same given differing time-step sizes.

Caution: In principle, many parameters such as agent speed, turning rate, weapons firing rates and sensor detection probabilities should rescale automatically with change in battlefield time step. However, exact rescaling is not guaranteed and it is recommended that scenarios be re-calibrated if the battlefield time step is changed.

(McIntosh, 2009)

As a simple example of this phenomenon, consider a sensor with a probability of detection equal to 50% at a time-step size of 1 second. This results in a $E[TTD]$ of exactly 2 seconds. When the time-step size increases to 2 seconds, the $E[TTD]$ of that sensor doubles to 4 seconds because the sensor is given twice as many opportunities time to detect a target within the same simulated amount of time. Thus, the sensor does not maintain its fundamental property when the time-step size changes. See equation below to demonstrate how the detection probability adjusted for a change in the time-step size in the case of a “head-to-head” sensor-target interaction.

$$E[TTD] = \frac{\Delta t}{P_D} = \frac{\Delta t'}{P'_D} \quad (32)$$

$$P'_D = \frac{\Delta t' \cdot P_D}{\Delta t} \quad (33)$$

where, P'_D is the adjusted probability of detection for the adjusted time-step size $\Delta t'$.

Even though the P_D of a single constant rate sensor can often be rescaled without much difficulty, rescaling a sensor with multiple P_D at different ranges, such as the simple cake sensors presented here, proves to be impossible in a general sense. This problem is compounded when moving target, overlapping sensors, or a “lateral range” type of situation is encountered. Consider the rescaling problems introduced by the simple situation presented in Figure 56:

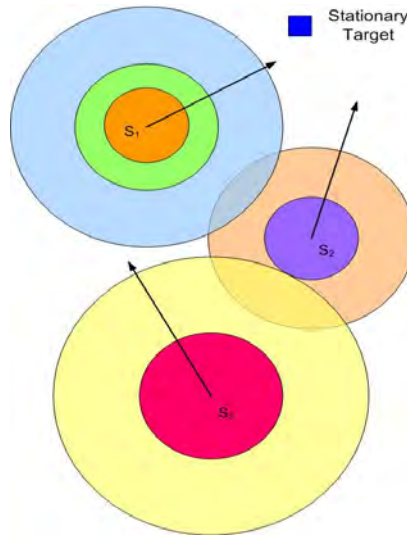


Figure 56. An example of a complex situation in which the DTS approach encounter difficulties to adjust for sensor detection

The problem is that modelers working with complex constructive type simulations typically with little animation might not recognize that this phenomenon is happening. In complex simulation models where many components' interactions are present, delays in sensor detections may cause the inadvertent sequencing and directly impact decisions.

D. SECTION III: EMERGENT OUTCOMES

To support the purpose of this study, a number of emergent modeling behaviors associated with the choice of TAMs and their related consequences is explored in combat simulation models.

1. How do different TAMs and time windows affect the agent sensing and movement? Specifically, we document the “skipping” phenomenon.
2. How do the different TAMs affect the recognition of scheduling consequences and task ordering?

1. Scenario 6: Skipping Phenomenon

a. Simulation Methods

Implementing certain TAM approaches in simulation could produce unrealistic and undesired behaviors of the system being modeled. We conducted two tests to more deeply explore a phenomenon introduced in our previous experiments. Given variations in time-step size and sensor ranges, agents can miss detections and waypoints. That is, agents will “skip” detections and locations. We saw this especially in the agent movement scenarios above. Our designs here were constructed specifically to examine this phenomenon across MANA 4, MANA 5, and Simkit. We return to the use of MANA 4 to test for the difference in skipping between grid-based and vector-based movements, and find some interesting results.

In our first test, a blue and a red heterogeneous agents start from their origin waypoints 1600 meters apart and move linearly towards each other’s origin waypoint in a simple “capture the flag” setting. When one agent encounters the other, they engage in combat. Each agent has a CC sensor and shoots with a probability of kill equal to 100%. Following the battle engagement, the remaining agent proceeds to its target waypoint “capturing the flag.”

Parameters	Blue Agent	Red Agent
Speed (meters/second)	10	15
Sensor Type	Simple Cookie-cutter	Simple Cookie-cutter
Sensor Detection Range (m)	13	17
Weapon Range	50	50
Probability of Kill	1.0	1.0

Table 13. Input parameters for Scenario 6 combatants

This scenario was modeled in Simkit and MANA 5 with time-step sizes increasing in increments of 0.5 seconds from 0.5 to 4 seconds. We also varied the firing

rate in SimKit to match the effective firing rate in MANA 5 for each time step condition, and completed 200 replications of each condition for each environment. As described in detail below, this initial set of timing conditions demonstrated that skipping starts to occur when the time-step size is larger than approximately 2 seconds. To further specify the effects of time-step size variations on this phenomenon, we then conducted the same experiment varying the time-step size by increments of 0.1 between 2 and 4 seconds. For consistency and clarity, both are presented together in the results.

In the second skipping phenomenon test, we examine the emergent behavior in a more complex combat model involving a single agent searching through 20 stationary targets randomly distributed over a 2000x2000 meter battlefield. The agent is outfitted with a CC sensor with 14 meter range and moves at a fixed speed of 7 meters per second in a linear fashion sequentially between 8 fixed waypoints as shown in Figure 57. We performed this test in MANA 4, MANA 5, and Simkit with the same time step increments and replications as in the first test.

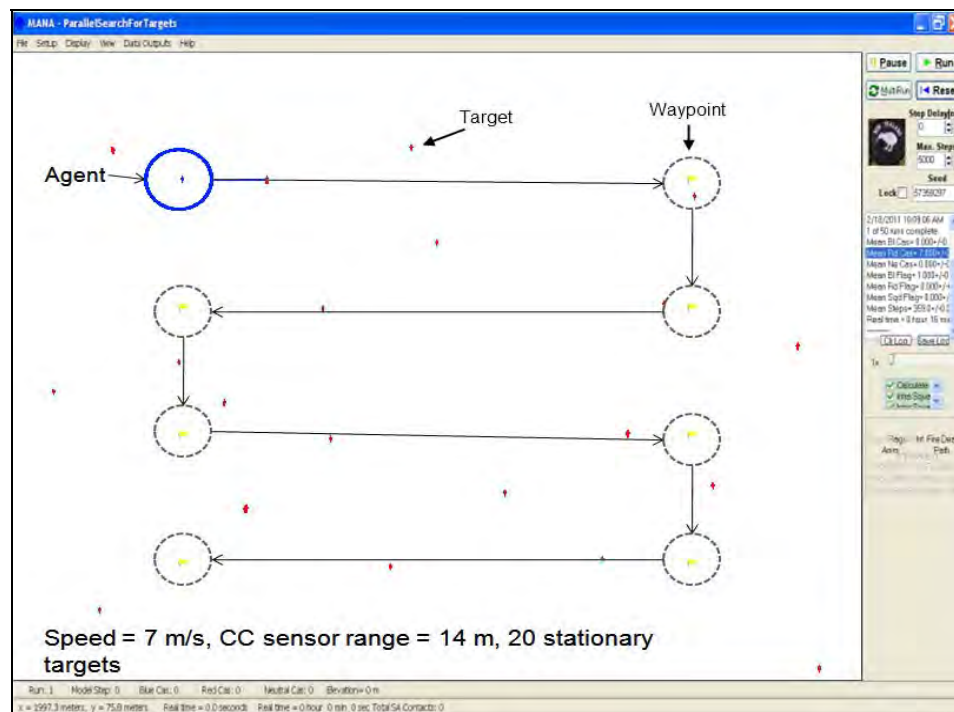
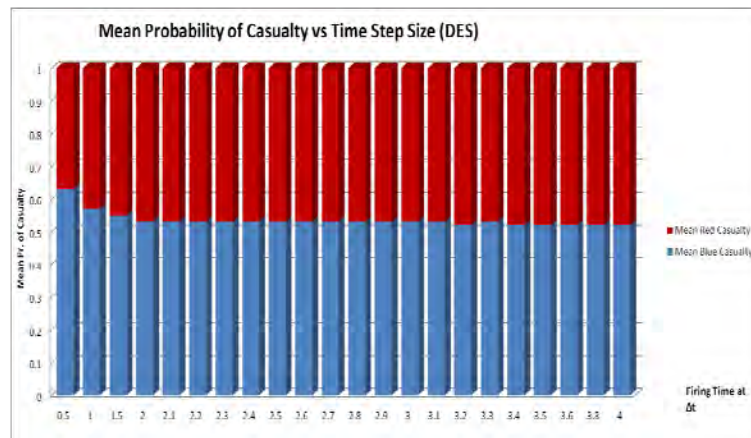


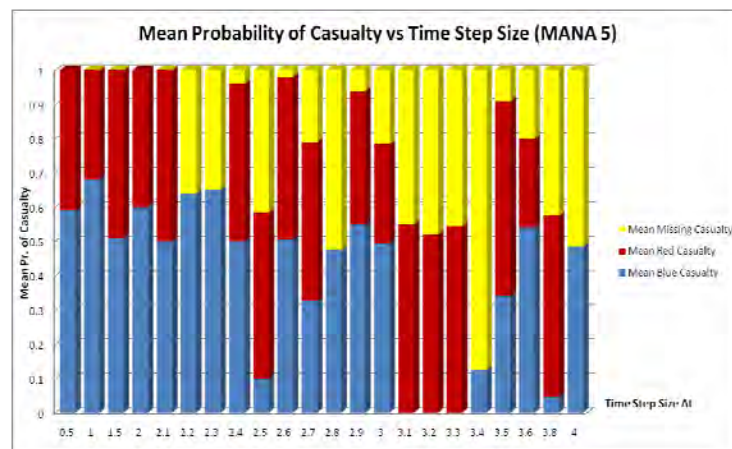
Figure 57. A simple parallel search operation scenario of one agent and 20 stationary enemy targets in MANA 5 GUI display

b. Results

In the head-to-head capture the flag test, the DES environment produced no agent skipping as the overall mean probability of both sides casualty is always 100%, while the DTS environment produced skipping at the time-step size increased beyond 2.2 seconds. Additionally, the DES results are relatively consistent whereas the DTS results are extremely erratic. Figure 58 shows the results from DES and DTS side by side. Figure 59 illustrates the randomness of the error and effects on the results introduced by the skipping phenomenon. The red line shows the differences between the mean of red casualty of DTS results and DES results and similarly for the blue line.



a)



b)

Figure 58. A demonstration of skipping phenomena in a simple two heterogeneous agents combat; a) DES model results and b) DTS (MANA 5) model results

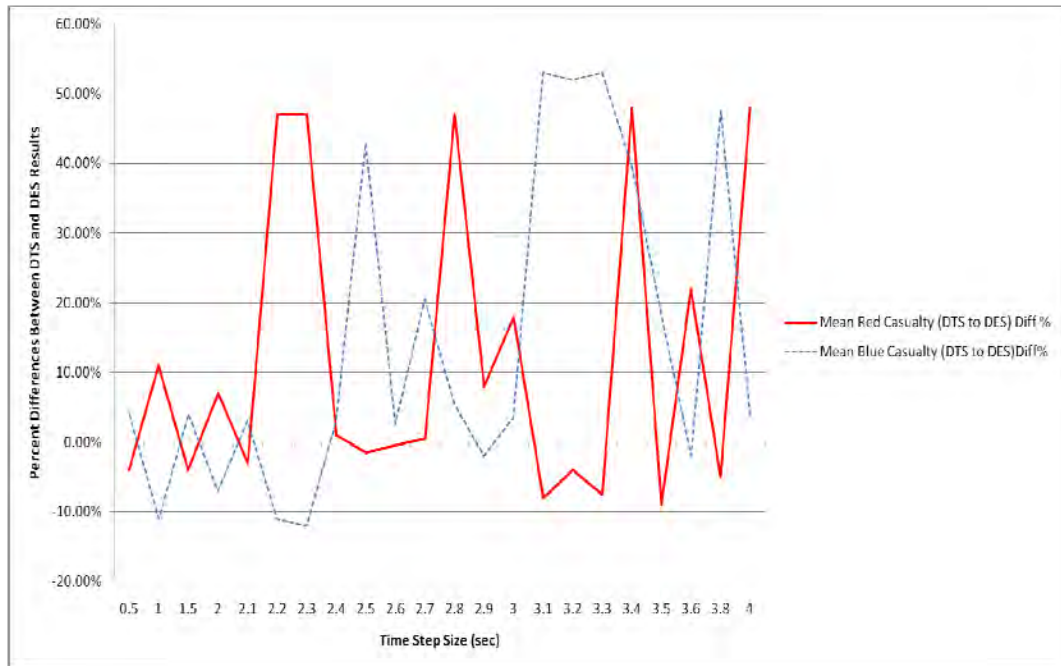


Figure 59. Illustration of randomness in the differences between the DTS and DES results for the simple two heterogeneous agents combat.

In the search operations test, the DES environment identified 9 of the 20 randomly distributed targets in all runs. The MANA 5 agent detected the same 9 targets for time-step sizes less than 2 seconds. As the time-step size grew beyond 2 seconds, the number of detections drops monotonically. The effect was exaggerated in the MANA 4 environment due to the grid-based movement paradigm. While the MANA 4 agent detected 9 targets for time-step sizes less than 1.5 seconds, the number of detections decreased as a more rapid rate than that of the MANA 5 results. The MANA 4 agent identified fewer than half of the agents detected by the DES agent at a time step of 4.5 seconds. Since these scenarios are completely deterministic, Figure 60 shows the results of one run for each condition.

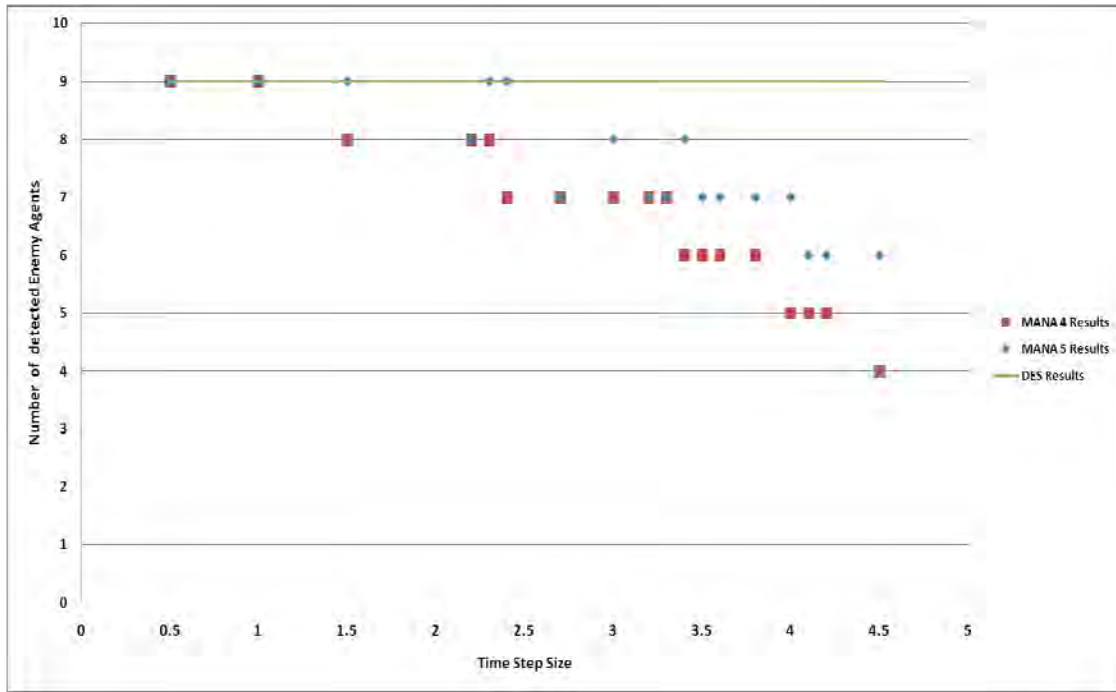


Figure 60. Parallel search operation results showing the number of detected enemy targets for various range of time-step sizes [best viewed in color]

c. Discussion

It is clear from these results that the skipping phenomenon can be a major contributing factor to output in DTS models. These errors could accumulate behind the scenes of constructive simulations, resulting in deceptive outputs that can lead to faulty decision-making. Regardless of whether the movement structure is grid-based or vector-based, the effects of the TAM are strikingly similar. This indicates that the issues lies not with the movement mechanism, but rather with the time advance mechanism behind the simulation. That is, the functioning of the simulation clock in a “global, logical” approach fails to effectively discriminate between entity movements (and other state transitions) to avoid skipping. This is true regardless of the other factors affecting the entities.

The skipping phenomenon is a result of the failure to detect events. In the DTS environment, the failure is a function of the TAM. However, the DES environment can also experience the phenomenon but only if the modeler fails to build the event

recognitions in to the model itself. Because the construction of a DES model relies on event and state transition definitions, this is unlikely to happen in practice. In DES models the only way that a skipping would occur is if that skipping is explicitly included in the calculations, either by design or by error. In other words, in DES skipping will occur only in an incorrectly constructed model, whereas in DTS it may occur even in a “correctly” constructed model.

The MANA results in Figure 58 demonstrate that the skipping phenomenon defies straightforward characterization. It has a random signature of impacts on the simulation over the course of time-step size variations. Thus, this phenomenon produces significant errors that cannot be predicted or accounted for in the current modeling techniques. While our current tests use only two agents, the impacts of this phenomenon take on much more significance in the context of more realistic combat simulations that use many agents.

For instance, consider a platoon-level combat scenario containing 100 to 200 agents on each side. The patterns of movement and detection for each individual agent are not visualized or generally explained discretely in the simulation output. Thus it would be impossible to identify instances of skipping in general, and one would not know if this phenomenon has affected the output measures, even if the results were affected in a significant way as is the case here. Even reducing the time-step size does not solve the skipping issues because there is no established approach in DTS environments. These concepts are explained in depth in the conclusion section of this chapter.

2. Scenario 7: Event Scheduling and Ordering Phenomenon

a. Simulation Methods

The ordering of state transitions is a major deciding factor in most combat simulations. In any scenarios involving multiple agents, the order at which state changes take place in a simulation will affect the outcome, often in dramatic ways. For instance, when combining movement with engagement for two agents, it is often the case that an arrival at a location must precede an engagement. We noticed that the proper ordering of

events is sometimes compromised in combat scenarios as a result of the TAM chosen. Here we developed a test to explicitly characterize the event ordering phenomenon in a relatively simple combat scenario.

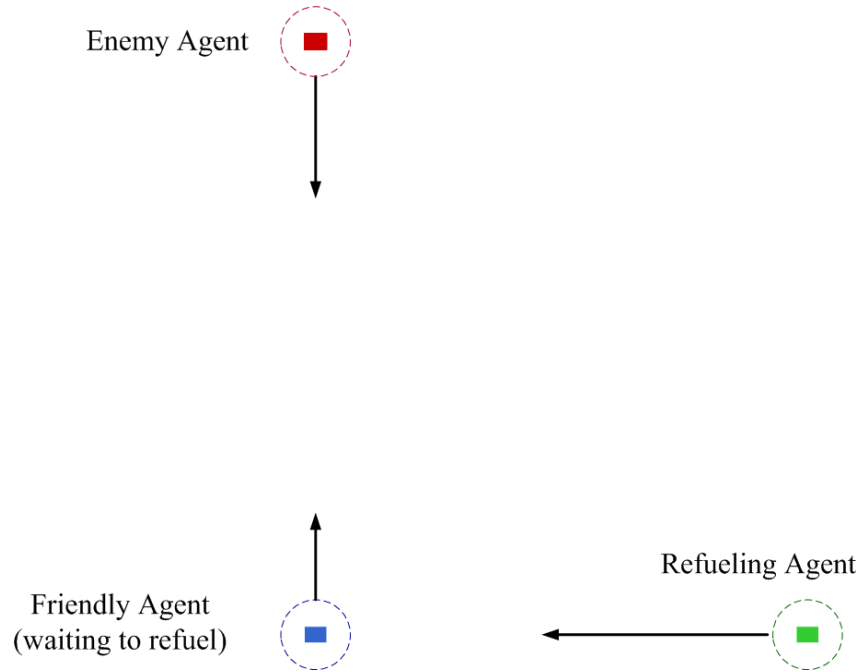


Figure 61. Illustration of initial setup of agent refueling Scenario

In this scenario two homogenous agents, red and blue, engage in combat on a 2000x2000 meter battlefield. These agents have the same speed, sensor type, sensor range, direct fire weapon, and probability of kill (P_K) for all simulation runs. This is shown in table 14. The blue agent must wait for refueling from a third green supply agent before moving towards the red agent to engage in combat. When the simulation starts, the green agent immediately moves towards the blue agent at a speed of 5 meters per second, and must travel a total of 966 meters to meet and refuel the blue agent. Once the yellow agent meets the blue agent, refueling takes exactly one time unit. Following this time unit, the blue agent may immediately engage upon detecting the red agent. If the red agent is not in range, the blue agent immediately starts moving towards the red agent at a speed of 7 meters per second.

When the simulation starts, the blue agent is waiting for refueling, but the red agent immediately starts moving from its home location towards the blue agent to engage. The red agent moves at a speed of 7 meters per second and is initially 1600 meters away from the stationary blue agent. Once the blue agent is within sensor range, the red agent engages. This is true if the blue agent is moving, or if the blue agent is still waiting for refueling. All in conditions, the simulation stops when either the blue or red agent is killed.

Parameters	Blue Agent	Red Agent	Refuel Agent
Sensor Type	Cookie-cutter	Cookie-cutter	Cookie-cutter
Sensor Range (meters)	250	250	200
Weapon Type	Direct fire	Direct fire	NA
Weapon Range (meters)	500	500	NA
P_K	1.0	1.0	NA

Table 14. Agent refueling Scenario input parameters

We simulated this scenario in both MANA 5 and Simkit environments and compared the number of casualties of both sides. In the DTS approach, we varied the time-step sizes in increments of 0.5 seconds from 0.5 sec to 2.5 seconds. As in the previous scenario, we further tested finer time-step size increments in secondary tests to achieve greater definition of key transition points. We varied the time-step size in increments of 0.1 seconds between 1.0 and 1.5 seconds. All conditions were replicated 100 times, and the average results from both sets of tests are shown together for clarity.

b. Results

For every time condition, we calculated the mean casualty percentage and 95% confidence interval for both the blue and red agents. We also calculated the total difference between the red/blue DTS results and DES results for casualties at each time step. These results are shown together in Figure 62. At small time-step sizes (less than 1.4 seconds), the difference between the DTS results and DES results is within the 95%

confidence interval. However, at time-step sizes larger than 1.4 seconds we see a noticeably increasing and significant trend of the ordering phenomenon affecting the results.

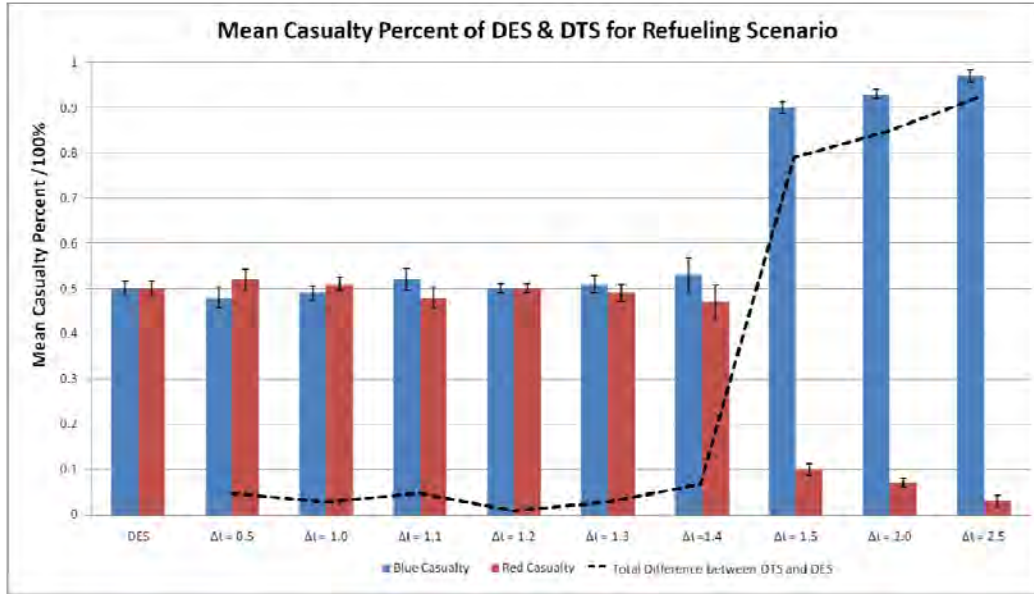


Figure 62. Mean casualty percentages of red and blue agents for the refueling Scenario

c. Discussion

As Law and Kelton (2000) state, obtaining accurate data using the time step approach requires the assumption that “all events *actually* (sic) occur at one of the times $n\Delta t$ ” (Law & Kelton, 2000, p. 117), and this is clearly not the case in combat simulations such as the one presented here. At smaller time-step sizes, each event in the blue agent’s process- refueling, detecting, engaging- takes place at a separate time step. This event ordering represents the reality of the continuous system we are modeling. As the time-step size grows, the ordering of events becomes increasingly compromised. Multiple events that should occur separately (in sequence, at different times) are recorded together in the simulation. Thus, the sequencing of events is no longer represented by the activities that take place in the simulation environment. For instance, in this scenario medium sized time steps can cause both the refueling and detection events to effectively

occur at the same time. In the large time-step sizes, all three events are recorded and reported together. This is shown in Figure 63.

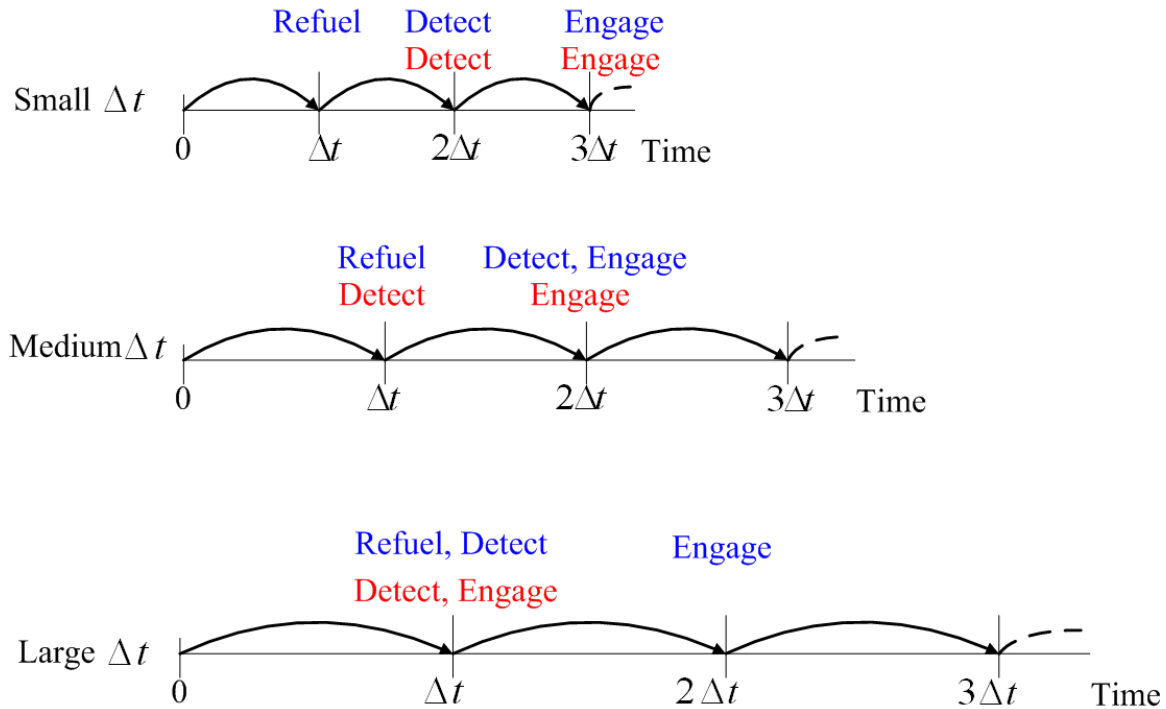


Figure 63. A diagram showing the effect of increasing time-step size on the order of state transition occurrence [not to scale]

At larger times steps blue casualties increase because the agent cannot engage in combat until the refueling is complete, and the blue agent state change is not reported until the following time step. As Figure 63 above illustrates, at large time steps the red agent will be able to engage before the blue agent can engage. Although the refueling event only takes 1 second, time-step sizes larger than 1 second effectively delay the state change.

The secondary exploration of time-step sizes between 1 and 1.5 seconds shows that the probability of casualties is a result of agent movement timing. For these time-step sizes, the red agent is not yet within range to engage. As a result, the red agent does not always kill the blue agent even though the blue agent refueling state change is

delayed by the difference between 1 second and the time-step size. Thus, the results between 1.1 and 1.4 seconds time-step sizes are not significantly different than the results obtained at a time-step size of 1.0 seconds.

The scheduling of state transitions (events) in the DTS model becomes inaccurate as the time-step size increases. For example, the end of refueling state is suppose to schedule a state transition that makes the friendly agent (blue) starts to move towards the red agent. This occurs at small time-step sizes, however, at large time-step sizes, end refueling state transition is delayed to occur at the end of the time interval. The delay in end refueling state occurrence along with the time of the next interval causes major delays to the blue agent readiness for combat and leads to inaccurate results and conclusions.

E. SECTION IV: PUTTING IT ALL TOGETHER

With the exploration of combat elements in our comparison study, it is now valuable to create and analyze a more complex maritime combat scenario. This section addresses the following questions:

1. What effects do TAM choices have on the outcome of an LCS simulation?
2. Specifically, does the choice of TAM have a material effect on the results of the simulation keeping all modeling elements the same? Put another way, does the outcome change solely as a result of changes to the TAM or time window, keeping all model elements constant?

1. Littoral Combat Ship (LCS) Battle

a. Simulation Methods

In this scenario, we combined the significant elements of the previous 7 studies to explore the effects of TAM on an established relatively complex combat scenario. This study leverages a MANA 5 scenario recently developed and completed for a Master's Thesis in Modeling and Simulation at the Naval Postgraduate School

(Jacobson, 2010). The study investigates the effectiveness of missile types for the Littoral Combat Ship (LCS) with the Surface Warfare mission module (SUW). In particular, the original study built a simulation model to determine what surface-to-surface capability the LCS should have as well as what air-to-surface capability the LCS helicopter/unmanned aerial (UAV) should have.



Figure 64. U.S. Navy envision of the LCS mission (from Jacobson 2010)

Specifically, the simulation includes the LCS, a supporting MH-60R helicopter, and 20 enemy boats engaging on a 40,000 x 40,000 battlefield. The LCS moves in a uniform linear fashion from its home waypoint towards a single destination waypoint on the opposite side of the battlefield approximately 32,500 meters away at a speed of 35 knots (18 meters per second). The helicopter starts from its home waypoint and moves towards the enemy boats at a speed of 125 knots (64.3 meters per second). The 20 enemy boats are randomly distributed within a rectangular box approximately 5x6 meters in size positioned approximately 29,000 meters from the LCS home waypoint.

The 20 enemy boats move at a speed of 45 knots (23 meters per second) toward an engagement target waypoint approximately 27,000 meters away. All movement is initiated at the start of the simulation

Both the LCS and the helicopter will fire at the enemy boats if they are within range. The enemy boats will fire only at the LCS if it is within range. The LCS is outfitted with a single constant rate sensor with a detection range of 32,000 meters and probability of detection between 50% and 100%, depending on design point. The helicopter also has a single constant rate sensor with a range of 25,000 meters and a probability of detection between 50% and 100%. All red enemy boat agents are each equipped with a single CC sensor with a range of 29,000 meters and a 100% probability of detection.

The main point of the original MANA scenario was the investigation of different weapons on the effectiveness of the LCS and helicopter to neutralize enemy boats. The original Master's Thesis scenarios involved a full design of experiments analysis and changing factors that are concerned with the missile capabilities for the blue forces (LCS and helicopter). These factors are the type, number, Maximum Effective Range (MER), probability of kill (P_k), and firing rate (FR) of various missiles. The probability of sensor detection for the blue forces was also among these factors. No variables were changed for the enemy boats. These variables are displayed in full in Table 15.

Platform	Element	Factor	Value Range
LCS	Sensor	Detection Range	32,000
		Classification Range	26,000
		Probability of Detection (P_D)	0.5 – 1.0
	NLOS-LS	Probability of Kill (P_K)	0.16-0.72
		Rate of Fire (shot/sec)	0.04 - 0.2
		Number of Loads	60
		Max. Effective Range (MER)	40,000
	Harpoon	Probability of Kill (P_K)	0.8075-0.9405

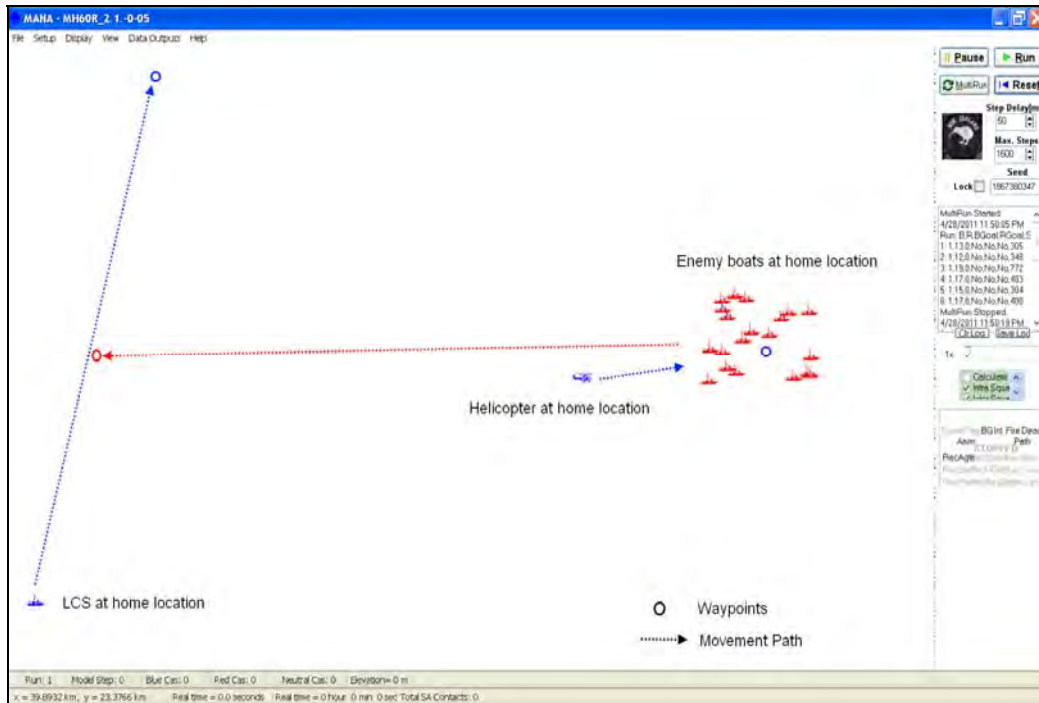
		Rate of Fire (shot/sec)	0.04 – 0.1
		Number of Loads	8
		Max. Effective Range (MER)	85,000
MH-60R	Sensor	Detection Range	25,000
		Classification Range	6,500
		Probability of Detection (P_D)	0.5 – 1.0
	Hellfire	Probability of Kill (P_K)	0.6375 – 0.7125
		Rate of Fire (shot/sec)	0.05-0.13
		Number of Loads	4-8
		Max. Effective Range (MER)	6,000-8,000
	LOGIR	Probability of Kill (P_K)	0.4225-0.585
		Rate of Fire (shot/sec)	0.06-0.17
		Number of Loads	14-38
		Max. Effective Range (MER)	4,350 -5,600
	APKWS	Probability of Kill (P_K)	0.4875-0.6175
		Rate of Fire (shot/sec)	0.05-0.13
		Number of Loads	14 -38
		Max. Effective Range (MER)	4,350 – 5,000
	DAGR	Probability of Kill (P_K)	0.4875 -0.6175
		Rate of Fire (shot/sec)	0.05-0.13
		Number of Loads	8 - 24
		Max. Effective Range (MER)	5,250- 7,000
Enemy Boat	Sensor	Detection Range	29,000
		Classification Range	24,000
	C-802	Probability of Kill (P_K)	0.45
		Rate of Fire (shot/sec)	0.17
		Number of Loads	4
		Max. Effective Range (MER)	120,000

Table 15. Variable factors used in the original study, all distances are in meters (after Jacobson, 2010)

Jacobson's study used only the DTS approach using the MANA 5 environment where "each time step was equal to 10 seconds of real world time" (Jacobson, 2010 p. 22). He used a Nearly-Orthogonal Latin Hypercube (NOLH) design to test the weapons and sensor variables, generating 512 distinct design points. Then he ran each of the design point at 40 replications, collected data and performed statistical analysis. In our study, we select illustrative design points to further our analysis and discussion of time advance mechanisms.

Jacobson's main findings related to the firing rates and types of missiles selected for the blue forces. Based on his simulation results, he recommended that the LCS should be equipped with a Non-Line-of-Sight Launch System (NLOS-LS) rather than a Harpoon only if the firing rate of the weapon is less than 9.1 seconds between shots. He also recommended that the helicopter be equipped Advanced Precision Kill Weapon System (APKWS) or Low-cost Guided Imaging Rocket (LOGIR) missiles if the firing rate of the weapon is less than 9.1 seconds between shots. Overall, he stated that the "most statistically significant factor is the rate of fire" (Jacobson, 2010, p. 45) and "when looking at future missile systems, the rate of fire should be the most important deciding factor" (Jacobson, 2010, p. 46).

We duplicated Jacobson's results in MANA 5, and also simulated the scenario using the DAFS Simkit DES environment extension package and focused on a subset of the design points in the original thesis. Here we explore the effects of TAM on the simulation results and resulting recommendations. We also highlight the effects of TAM on simulation accuracy for a range of time steps applied to the original MANA 5 model. While the original MANA 5 model used only a time-step size of 10 seconds, we varied the time-step size from 1 to 10 seconds in increments of 2 seconds for Jacobson's key design point, as explained below. We also applied the small 0.5 seconds time-step size in MANA 5 for 5 additional important design points to develop a more complete analysis of TAM differences.



a)



b)

Figure 65. a. LCS combat scenario in MANA GUI, b. LCS combat scenario in DAFS GUI [not to scale]

Our purpose is not to replicate all of the original study design points at various time-step sizes, but rather to illustrate that the design points' outcomes contain errors caused by the selection of the time step. We focus here on key design points in which his results was showing a fairly high number of red kills. We found more than 80 design points with adequate conditions for comparison testing, and selected 5 of these as samples for the current study. For all environments, each simulation run continued until either the LCS reached final waypoint or it received a hit from an enemy boat.

b. Results

We collected statistics in two broad categories: number of kills, and number of weapons used. Specifically, we measured the number of shots fired by the helicopter weapon, and number of shots fire by the LCS for all time conditions. We also measured the number of enemy boats killed (out of 20), and the percentage chance that the LCS was killed throughout the conditions. As we see in the figures below, in the DES and $\Delta t = 0.5$ seconds DTS simulations, the LCS is never killed, and all enemy boats are killed by the end of the simulation run. However, as the time-step size increases, these trends reverse. The chance that the LCS is killed increases monotonically with time-step size, and the number of enemy boats killed decreases monotonically with time-step size. These statistics hold true when we compare various time-step sizes' outcome differences with the DES model results.

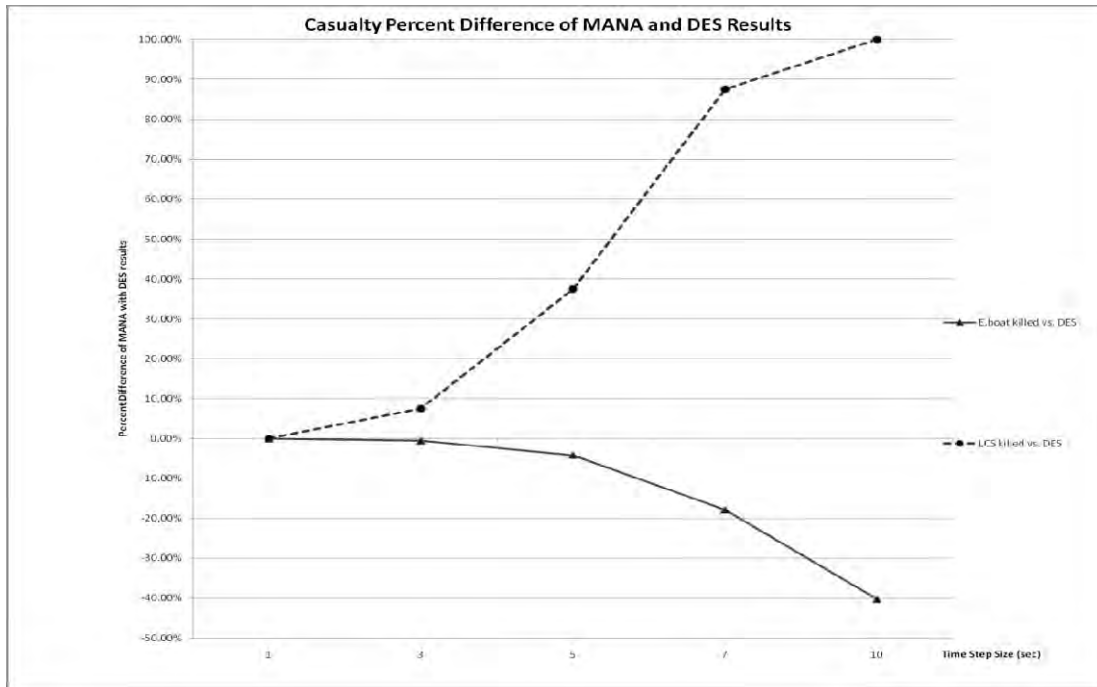


Figure 66. Illustration of percent differences between MANA results at various time step sizes and DES results for the mean number of kills ²

The second major trend that we see is with respect to the number of weapons used by the helicopter and the LCS during the simulation run. In the DES environment and smallest time-step size DTS simulation, the LCS never fires. Moreover, the helicopter effectively kills all enemy boats before there is a chance for the boats to fire on the LCS. As the time-step size increases, the number of weapons fired from the helicopter decreases as the helicopter becomes unable to kill all of the enemy boats before they engage with the LCS. Likewise, the number of weapons fired by the LCS jumps significantly as the time-step size reaches $\Delta t=3$ seconds. This dynamic can be seen clearly in Figure 67 below.

² These results compare Design Point 236 from the original study (Jacobson 2010). See Appendix C.

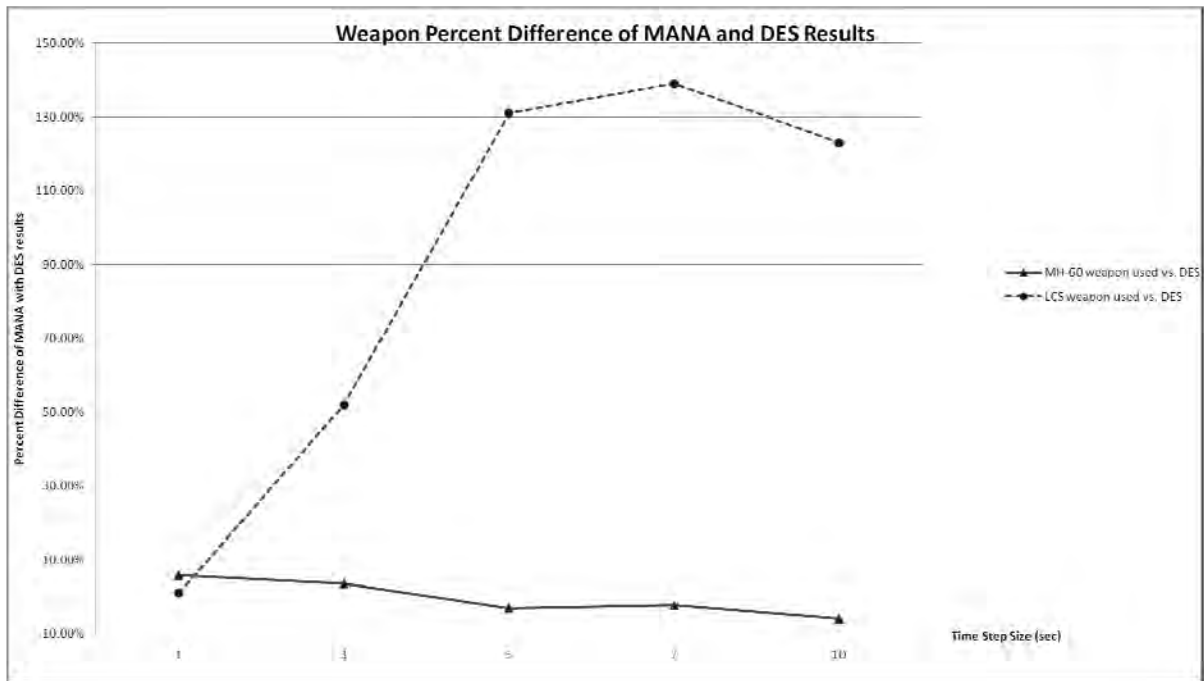
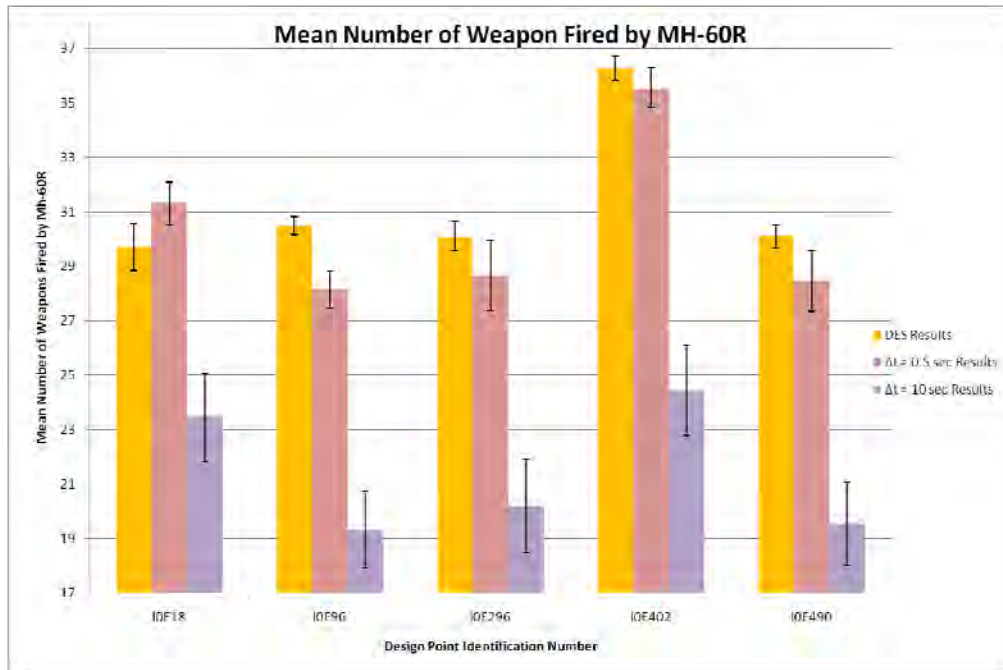


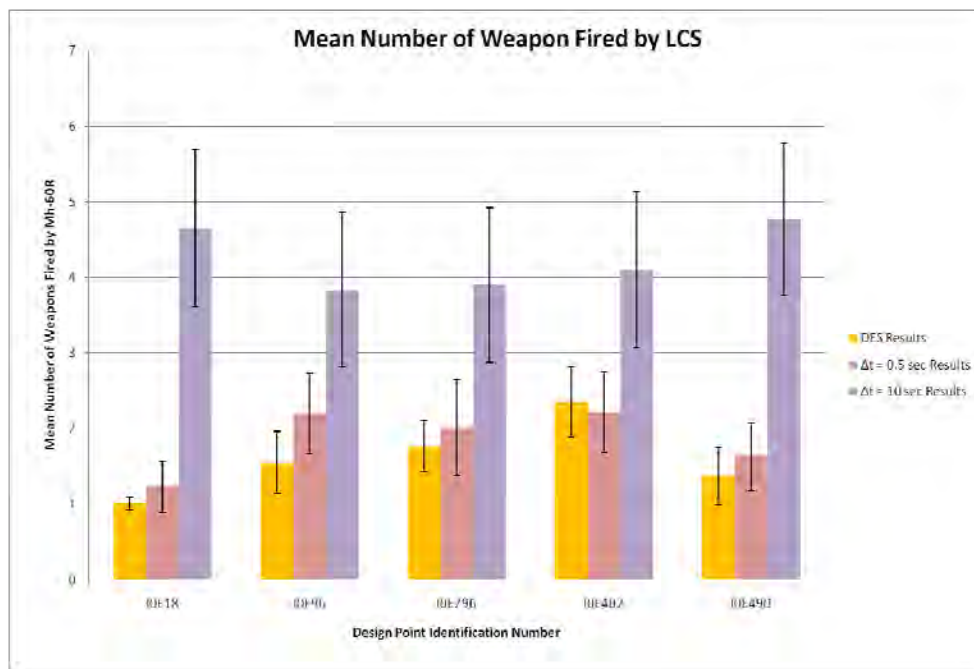
Figure 67. Illustration of percent differences between MANA results at various time step sizes and DES results for the mean number of weapons used

To ensure greater coverage and further delineate the effects of the difference TAMs on the results defined above, we followed our initial comparison with 5 addition design points³. Each of these design points tested various missile configurations and firing rates. These results are reported in depth in the figures below. In general, we found virtually identical behavior to the above-described trends. In these tests we utilized three time conditions: 1) the DAFS DES environment, 2) the MANA 5 environment with a small time-step size of 0.5 seconds, and 3) the MANA 5 environment with a large time-step size of 10 seconds. The large time-step size condition is identical to the original MANA 5 study. Thus, these results can be compared to the initial outcomes obtained with the “baseline” approach.

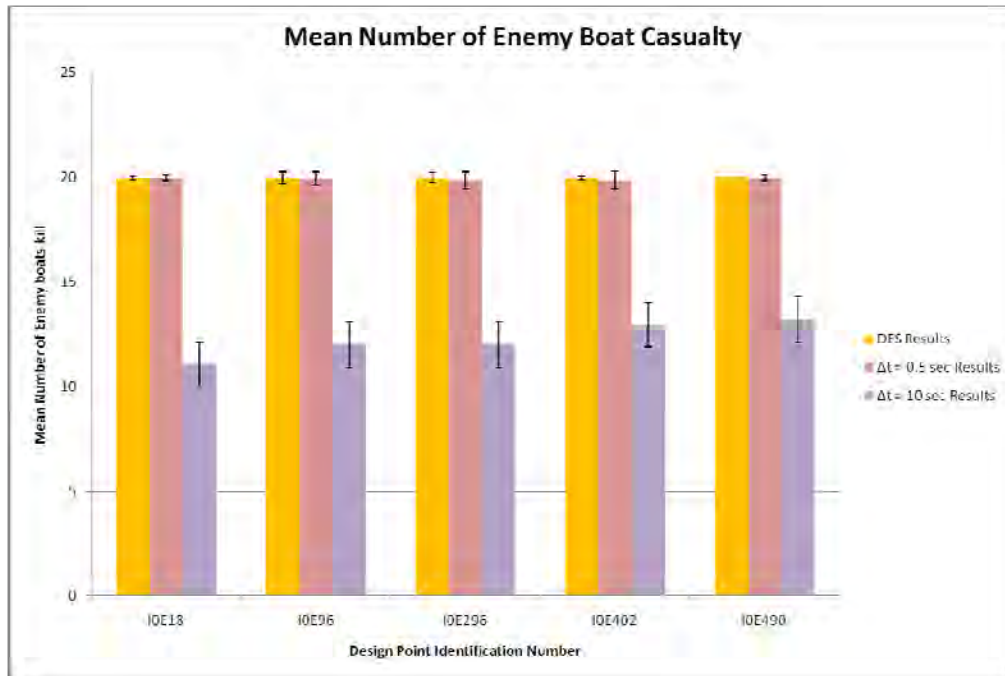
³ These results compare with Design Points 18, 96, 296, 402, and 490 from the original study (Jacobson 2010). See Appendix C.



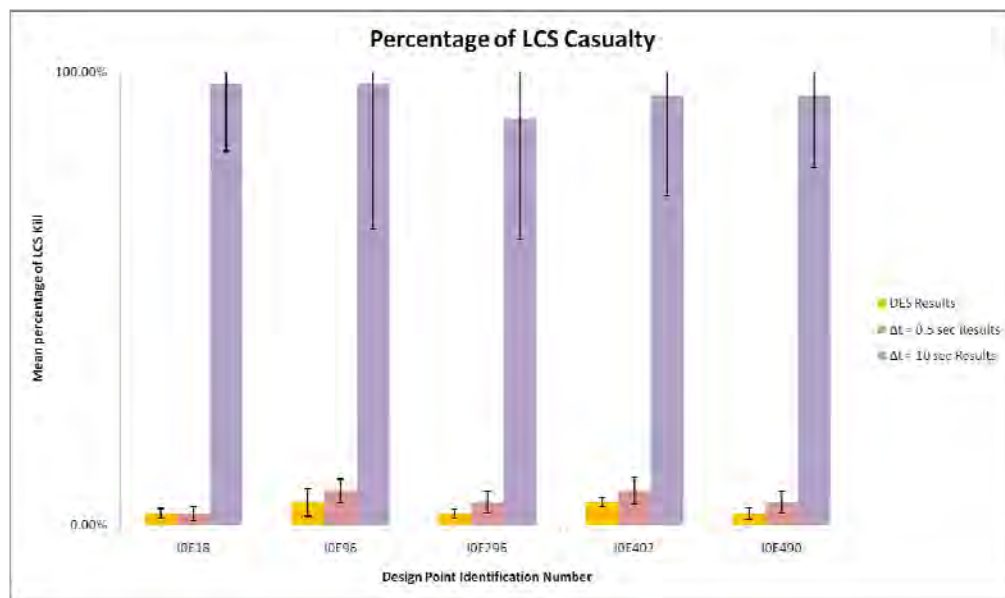
a)



b)



c)



d)

Figure 68. MANA and DAFS simulation results of 5 selected design points for; a) the mean number of weapon fired by MH-60R, b) the mean number of weapon fired by the LCS, c) the mean number of enemy boat casualty, and d) the percentage of LCS casualty

Simulation run time was not considered as a measure of performance due to the various options used to conduct the experiment. However, DAFS' run time for a single design point at 40 replications noticeably outperformed MANA execution time of the same number of replications when both are executed using the batch mode.

c. Discussion

The mean total number of enemy boats killed decreases with an increasing the time-step size Δt . This is significant for several reasons. Notably, although the MH-60R helicopter is loaded with enough missiles to kill all enemy boats, it does not detect the some boats as they move towards the LCS. Thus the helicopter cannot and does not kill some percentage of the enemy boats, enabling them to attack the LCS directly. As the time-step size grows, more enemy boats move beyond the helicopter's detection region, resulting in fewer helicopter weapons fired. Our results show that the helicopter fired more weapons at the targets, as compared with the original results. With larger time-step sizes, some enemy boats "skip" past the helicopter sensor range.

The DTS simulation uses standard equations of motion with the Δt to position the enemy boats in their estimated correct locations at each time. This movement strategy is appropriate for position finding, but clearly is not robust for complex situations involving multiple agent detections and interactions. This miss detection occurs at time-step sizes as small as 3 seconds. The mean total number of enemy boats killed in the original study ($\Delta t = 10$ seconds) is significantly fewer than our small time step and DES results. This can affect various simulation results as shown below and dramatically change the original recommendations.

Since the helicopter engages fewer enemy boats at large time-step sizes, the LCS becomes more likely to engage them. This engagement increases as the time-step size grows leading to two significant results. First, more rounds are fired from the LCS during the simulation run. Secondly, as a result of engaging in combat, the LCS is more likely to kill enemies and also get killed by the enemy boats. In contrast results with DES and small time-step sizes produced very small number of LCS weapons being used, and very small percentage chance of the LCS being killed.

The scenario is designed to test various configurations of weaponry and support affecting the chances that the LCS is damaged or killed. The use of helicopter air support is meant to increase the protection level and minimize LCS kills. Having a large number of enemy boats engaging with the LCS at large time-step sizes, as represented by the original study, increases the probability of killing the LCS well beyond the range identified by the other TAM conditions studied here.

In the original scenario, the LCS could be loaded with either 60 NLOS or 8 Harpoon missiles. Regardless of the LCS weapon firing rate, if the enemy boats can engage with the LCS without proper helicopter mitigation, then the LCS weapons used are not sufficient to eliminate the increasing threats. Thus, LCS weapon choices need to be reconsidered. This is significant to the study because the simulation terminates once the LCS gets killed. We noticed that as the time-step size increased, the LCS engages early with the enemy boats and gets destroyed with less average number of weapons used by both the LCS and the helicopter. The termination conditions prevent the reporting of accurate statistics related to the number of helicopter weapons used, as well as the number of enemy boats killed. When the simulation stops, it is assumed that the helicopter fires no more weapons and that no more enemy boats are killed.

Prior to implementing the scenario in the batch mode (i.e. no visual display) to generate multiple replications, the scenario was visually explored in the display mode setting for MANA 5. The display mode is what MANA modelers initially use to verify their models as it provides visual understanding of the model behavior. However, complex models replicated a large number of times can take incredibly long simulation execution times and are typically run in batch mode. A large number of replications is necessary to identify the convergence points in the stochastic elements of the simulation.

While running the original scenario, we examined the helicopter detection data and noticed that numerous enemy boats were not detected. The locations of these boats at each time step showed that these boats never entered the helicopter sensor range, thus no detection was reported. Further analysis showed that the DES approach scheduled detection for all enemy boats, indicating they were within helicopter range at the initial

stages of the simulation run. Testing the model at smaller time-step sizes illustrated that all enemy boats get detected by the MH-60R. The skipping behavior is present in all large time-step size conditions where a number of enemy boats skip the helicopter's sensor range. This phenomenon is explained in depth in Scenario 6 of Section 3.

State transition updates and the "Event Ordering" phenomenon also play a significant role in this study. The effect of DTS approaches on state transition updates is explained in Scenario 7 of Section 3. In MANA, even after the enemy boats are within the helicopter sensor range and detection occurs, the helicopter must wait for the correct integer time step to fire its weapon. This causes a firing delay and can lead to enemies leaving the helicopter's range without being killed. State transition delays are a natural phenomenon of the DTS approach (as explained in Chapter 2). As a result, enemy boats slip through the helicopter engagement and threaten the LCS in larger numbers, resulting in a much larger probability that the LCS is killed. As shown above, our results illustrate pointedly different observations, challenging the results and recommendations of the original study.

The original study stated that the firing rates of the helicopter and LCS weapons are the main contributing factor to LCS protection. In contrast, our studies clearly show that the helicopter sensor range and helicopter weapon features (range, probability of hit, probability of kill, blast radius, *and* firing rate) are key factors to mitigating the enemy boat threat and protecting the LCS. The large time-step size of the original study also lead to skipping and, with the original simulation termination conditions, resulted in fewer helicopter engagements with the enemy boats. Taken together, the original simulation lead to erroneous recommendations that the helicopter needs fewer weapons than necessary and the LCS needs *more* weapons than necessary to effectively engage with the enemy. Consequently, budgets are likely to be misdirected to enhancing unnecessary features at the cost of overall mission success. This is especially compounded by the fact the long range missiles for the LCS are extraordinarily costly compared to the weapons used by the helicopter.

F. SUMMARY AND CONCLUSIONS

Through a series of eight scenarios, our results in this chapter clearly demonstrate that the choice of Time Advance Mechanism (TAM) can have a very large impact on combat simulations. This impact is visible in the results of each simulation run, and the overall outcomes of difference scenarios, as well as the resultant decisions and recommendations that are made concerning the simulations. Importantly, the differences in results and outcomes between the time advance mechanisms are, in general, different enough to have a major impact on the germane conclusions and recommendations derived from the simulations. This is plainly seen in Scenario 8, where the divergent TAM results suggest strikingly different approaches in the real world.

In fact, the effect presented here is even more striking when one considers the fact that the DTS and DES results do not converge to the same answer for the vast majority of cases. Non-convergence seems to be the rule, rather than the exception. This is true even when a multitude of difference time-step sizes are tested in a comprehensive fashion. While reducing the time-step size in DTS models can, in some cases, lead to convergence between the DTS and DES models, this reduction comes at a substantial cost in terms of both simulation execution time and the probability of compounding accumulative time step errors. Thus, a time-step size reduction approach to minimizing overall simulation error will not necessarily be effective.

While it may be possible to reduce the total error by reducing time-step size in select cases, most simulations of any complexity will not benefit from this tactic. This is underscored by the fact that there exists no established methodology for choosing the appropriate time-step size in complex simulations. Although substantial work has gone in to sampling and estimation methodologies regarding the underlying differential equations used in simulations, the complexity of even the most straightforward instructive combat simulations render much of this work inapplicable. Note, for instance, the “simple” scenarios involving the movement of one or two agents presented at the beginning of this chapter.

We have presented here a wide-ranging quantitative and qualitative comparison of the behaviors of DTS and DES time advance approaches in combat simulations across a range of variables and model complexities. Throughout this investigation, we have found that the TAM has a significant impact on the movements, sensing, detection, and combat engagement of simulation agents. Moreover, we have demonstrated that the discrete time (time step) approach causes a number of emergent “anomalies” in the behavior of all of these fundamental combat simulation elements. Problems such as entity miss detection due to skipping, state transition delay, simultaneous event and event ordering can be present at all time-step sizes in discrete time simulations. Conversely, these issues do not arise in the corresponding discrete event simulations.

For instance, the movement of agents is handled differently in both DTS and DES, and these differences (although mostly invisible to the modeler) can have significant impacts on the simulation. Specifically, all entities in a DTS environment share the same global logical simulation clock and must be polled at regular intervals (time steps) to update positions. This is true whether the agents are moving or stationary. We have shown that the anomaly of “skipping” and the resulting miss distance can grow dramatically as the time-step size grows, leading to anomalous conclusions regarding system behavior. This is particularly germane to complex simulations where all of the interacting entities cannot be visualized.

Being in the right location at the right time is essential in most combat models, as engagement depends on time and location accuracy. In scenarios where precise locations have to be recorded and realized, DES outperforms the “equivalent” DTS model. This has to do with the fundamental difference in the way entity movement is handled between the methods. With the DES method, entities can be at any location in the battlefield at any specified time, according to the underlying equations and theories dictating agent behavior. However, an entity location is dependent on the size of the time step in DTS environments, and this has nothing to do with the underlying theories of behavior in the system. As a result of location inaccuracies, it is possible for a DTS approach to produce oscillations around waypoints even with small time-step sizes. Indeed, this is shown to be the case in several of the scenarios presented in this chapter.

The impacts of TAM on sensor detection ranges and types provide another example of the significant differences and problems introduced by varying approaches to simulation time. In general, the DTS approach lacks the necessary sensitivity to changes in sensor ranges. The size of the time step directly affects the precision of sensor detections, causing diverse and sometimes unpredictable simulation results as sensor ranges change. One of the most notable results is unexpected detection delays that, like the movement anomalies, having nothing to do with the underlying system. Instead, the observed impacts on the simulation are a direct result of the time-step size within the DTS environment. In cases where combat models involve examining detection sensitivity to changes in sensor ranges, and the corresponding impact on the simulation outcomes, the DES approach again outperforms the “corresponding” DTS models. Furthermore, our results clearly demonstrate that sensors involving time and range dependent detection (“Cake” sensors) are more accurately modeled with DES approach since DTS produces compounding errors as greatly complexity is introduced to the simulation.

To review, time-step errors can accumulate every Δt through entity movement, detection and engagement. This can result in higher overall errors even when the DTS time-step size is small. Conversely, DES mathematical errors are constant. These “state transition” errors are stable, predictable, and do not result from the specifics of the time advance mechanism chosen. In many cases, these errors are known in advance and explicitly stated by the models. This is not an option for the DTS approach since the accumulative time step errors have yet to be fully characterized.

The only significant impact of time-step size reduction that can be fully characterized is the time-step size impact on simulation execution time. The simulation execution time will increase as the time-step size decreases, and this is especially true when there are a large number of complex agents. While the significance of this measure should not be overlooked, metrics of execution time should not be confounded with measures of simulation accuracy or fidelity. As Buss and Sanchez (2005) illustrate, the computational complexity for the DTS system will always be $O(n^2/\Delta t)$ and for the DES system $O(n \cdot e)$ where n is the number of entities and e is the number of events.

If simulating agent sensing, detection, or movement with high accuracy and precision is not necessary to understand the behavior of a combat model, then one could argue that the DTS approach could provide results as satisfactory as the “equivalent” DES approach. However, this will rarely be the case as even the simplest scenarios modeled here are negatively, and substantially, impacted by the discrete time methodology. If one absolutely must use a DTS environment, then extra care is required when building combat scenarios and must include the full characterization of the effects of a full range of time-step sizes, to the greatest extent possible.

An adaptive time-step size technique has been recently used for single-element type dynamic system simulations such as fluid motion and objects collision (Joukhadar, Laugier, & Alpes, 1996; Huang, Duo, & Li, 2010). The objective with these models is to choose a time-step size that insures numerical stability and computational efficiency. Typically, the adaptive method involves determining a common measure (CM) such as the system energy (in the case of rigid-body collision) that integrates all of the model measures (e.g., position and velocity) in the simulation. The adaptive time step algorithm is applied during the time intervals and only time-step sizes that insure the CM value does not exceeds certain limits are chosen. The challenge with adaptive time-step size is to determine the safety measures and the desired accuracy for each component in the system. This task becomes impossible when dealing with complex systems such as combat models that contain a large number of entities and components that interact with each other. To our knowledge, this technique has not been implemented to military combat simulations or agent based simulations. One possible reason is that these systems are more complex and typically involve many confound elements such as movement, sensing, and cognitive systems, and their interactions, making it mathematically impossible to find global adaptive step sizes that suits all elements.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CASE III: HUMAN BEHAVIOR REPRESENTATION

“DoD simulations must include authoritative and accurate representations of human behavior so as to be sufficiently credible and variable to provide valid analytical results, effective training and supportable acquisition decision.”

(McClanahan et al., 2001)

A. INTRODUCTION

Human behavior representation (HBR) is an essential element in representing political, military, economic, social, infrastructure, and information (PMESII) aspects of an operational environment undergoing stability operations (Ferris, 2008). Current military organizational and societal modeling methods are “lacking for complex, dynamic, self-organizing, and adaptive systems in general, and modeling human behavior is particularly problematic.”⁴ As well, data collection, knowledge acquisition, and behavior representation methods for organizational and societal models are inadequate at best. The goal behind modeling HBR is not to produce models capable of predicting responses of the population to specific actions, but rather to provide a tool useful for gaining insight into the effects that certain actions by other actors in the environment, such as coalition, government, and insurgent forces, have on the population. Recently, a number of models have been developed to meet the need to understand the contemporary battlefield. Currently these models fall into multiple categories such as agent based, system dynamic (SD), or analytical. These models differ in their underlying designs used to generate outputs. However, they are similar in that they are a step towards modeling the asymmetric battlefield where the main objective is to gain the civilian satisfactions towards the government, security level, services, infrastructure, and so on. Although

⁴ United States Department of Defense, Modeling & Simulation Steering Committee (M&SSC), DRAFT, “Modeling and Simulation Steering Committee Common and Cross-Cutting Business Plan,” 1 November 2006, p. 39.

gaining insight from such models can be difficult, a simulation model can nevertheless aid decision makers in achieving a better understanding of the effect of their actions on the local population.

Representing human behavior and cognition from individuals to societies present a range of challenges to the modeling and simulation (M&S) community. Among these challenges is formulating an authentic representation of time in a model simulating irregular warfare (IW) operations (Pew & Mavor, 1998). The methodology used to represent time in these complex IW simulation models can produce inaccurate representation of the population behavior and affect the stabilization operations. Recently, combat models and training simulations have begun to include some HBR, but often these have fallen short of what is truly required and can actually reduce the usefulness of the model (RTO, 2009). Our purpose in this chapter is to address the following questions:

1. Can TAM choices in complex simulations involving HBR significantly influence the simulation outcomes and effect decisions?
2. What are the advantages and limitations of DTS and DES approaches when used in HBR simulation models?

To answer the above questions, we studied the effects of time advance mechanisms (TAMs) on military M&S into an emerging domain of increasing importance in modern military operations. Namely, we touch on the potential effects of TAMs on the complex simulations involving human behavior. We extend our study of the effects of TAM from the combat scenarios described previously to the complex HBR scenarios involving many agents. Unlike traditional combat simulations, HBR in military M&S is not as developed or well defined. The National Academies of Science (Zacharias, MacMillan, & Van Hemel, 2008) review of the state of the art indicates that the field is currently fractured with no simulation theories obtaining major purchase in the military M&S community.

The vast majority of HBR extend the traditional combat simulation environments that we have explored previously, and provide a good frame of reference for further

exploration. We view HBR as an extension of greater complexity than the typical combat models given the number of interactions that are taking place in most simulations, as well as the sheer number of agents in large-scale HBR. The TAMs used in HBR are the same as in combat simulations, namely either DES or DTS. Like combat simulations, the majority of HBRs use the DTS approach. As a result, we examine changes in the time-step sizes in HBR, finding the same general trends as those found in combat simulations.

To date there has not been a study that looks at the comparison of TAM choices and their effects on the accuracy of HBR in IW type of models. There are also no studies in the literature that conduct sensitivity analysis to demonstrate the effect of time-step size on human behavior values.

B. METHODOLOGY AND LITERATURE REVIEW

1. Methodology

We studied two environments that use the DTS approach for HBR, Pythagoras⁵ and the Peace Support Operations Model (PSOM)⁶. We utilize one HBR DES environment, the cultural geography model (CG)⁷. Like PSOM, CG was designed specifically for irregular warfare applications. It was built on top of Simkit, which is a general modeling and simulation framework that was explained in depth in the chapter on combat simulations. While the use of Pythagoras for HBR is in the early stages of development, both PSOM (Marlin, 2009) and CG (Alt et al., 2009) have been used in both decision-support and training applications.

We explore these three HBR simulation environments (Pythagoras v2.0.0, PSOM v2.2.3 and CG v0.4.0) by looking at the changes in population behaviors (e.g. security stance and consent towards coalition forces) as a response to events and actions caused

⁵ Northrop Grumman Space and Mission Systems Corps., *Pythagoras User Manual Version 2.0*, 2007. Project Albert Website, <http://www.projectalbert.org/>, last accessed on 12 June 2011.

⁶ Developed by the Defence Science and Technology Laboratory (Dstl), <http://www.dstl.gov.uk>, last accessed on 3 June 2011.

⁷ In development by TRAC-MRY, <http://www.nps.edu/research/TRAC/>, last accessed on 4 June 2011.

by political entities such as coalition or insurgent forces as well as events resulting from communications between actors and their surrounding environment.

Pythagoras is an agent-based, time-stepped model primarily used in combat simulation. However, it has been extended for use in HBR. We generate a population from random distributions to create a social network in which agents communicate and influence each other to change their behaviors toward a stance (security or consent to coalition) over a period of time. Weapons category in Pythagoras is used to represent communication and all weapons are made non-lethal. At each time step, agents communicate with each other based on their closeness in the social network at a specific rate that is adjusted by variables such as “firing rate” and “weapon selection” which allows for cognitive and social factors. An agent behavior or attitude towards a stance such as security is represented in Pythagoras by color values (0-255 for Red, blue or green). Triggers with thresholds are the events that causes an agent to change his/her behavior.

For the purpose of our study, we will use the Iraq scenario that is built by the developers of PSOM. We narrow the scope by considering two key measures of effectiveness that PSOM recommends, the Sunni population consent towards coalition forces and the overall security level for the Iraqi population. This study extends the work explored by Marlin (2009) to farther investigate to the sensitivity and impact of time-step size on the simulation results. Marlin (2009) carried a quantitative analysis using design of experiment method to explore the capability of PSOM by varying several input factors and examined as a response the civilian population changes in security and consent towards coalition forces.

The CGM is also explored for the same measures of effectiveness used in PSOM to investigate the use of DES approach in representing the population behavior changes to events and actions in the agent’s environment.

2. Comparison Studies in the Literature

There is no generic approach for including human behavior in military simulations (RTO, 2009). The implementation of HBR in military modeling and

simulation is still at its initial phases with currently evolving methodologies and approaches to produce accurate representations. There are several studies in the literature focusing on building agent based social system on the discrete time based paradigm (Zacharias, MacMillan, & Van Hemel, 2008; Marlin, 2009; Ferris, 2008; Seitz, 2008) and the discrete event based paradigm (Dubiel & Tsimhoni, 2005; Seck et al., 2005). Discussed below, we found few studies in the literature that relate to our comparison analysis in the HBR simulations from a general view but not directly related to the field of IW modeling.

Qiu and Hu (2010) studied the discrete time and discrete event models of agent-based crowd behavior simulation and compared their performance results. They simulated a pedestrian crowd to exploit the crowd system's spatial-temporal heterogeneity resulting from agents' non-uniform movements. Three scenarios were experimented using discrete event system specification (DEVS) framework. The first experiment involved a single agent moving uniformly to target destinations with obstacles. They extended the experiment to account for non-uniform movements with agents moving at different speeds. The final experiment simulated evacuation of pedestrian crowds under emergent situations such as fire alarm.

Qiu and Hu's experiments' results showed that the DES model can achieve better performance results (small position errors) than the DTS model for uniform moving agents. Furthermore, non-uniform moving agents in the DES model showed to achieve fewer decision makings than in the DTS models. It was found that agents in the DTS model make the same number of decisions despite their speed differences. This is because all agents make decisions at every time step. The number of decisions made by agents in the DTS model increased as the time-step size decreased. However, in the DES model, the number of decisions made between the slow and the fast moving agents were observed to be different. All DES models had less execution time than the DTS models (small time-step sizes) and showed to have the ability of tracking the crowd system's activities in both space and time.

Matsopoulos (2007) developed an agent-based behavioral model using the DES approach in Simkit tool. He examined the ability of DES approach to model situational

awareness and represents human behavior by simulating a number of homogenous neutral agents (100 to 1000 agents) packed in a small area, moving randomly with uniform speed and a fast moving searcher looking to detect and classify a slow moving target. The DES model results were compared with the DTS model results obtained from the same scenario implementation in MANA 4. The study showed that the simulation execution time (the time it takes the searcher agent to detect and classify the target agent) of the DES model increased proportionally to the square of the number of neutral agents in the search area (see figure below). In the DTS model, the execution time increased linearly with the number of neutral agents in the environment area (see figure below). The study demonstrated that the DES model is more efficient than the DTS model reaching an order of magnitude of 32 times higher in performance level. The study strongly suggested creating agent-based model (ABM) components using the DES approach rather than the commonly used DTS approach to provide more efficient results in such HBR models.

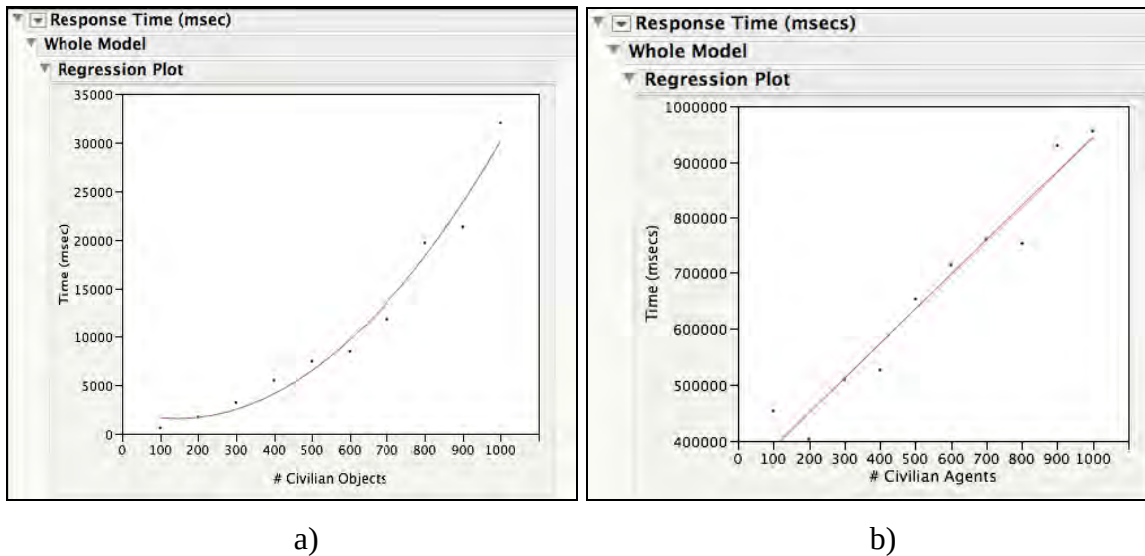


Figure 69. a) Simulation execution time as a function of the number of agents in a DES model, b) Simulation execution time as a function of the number of agents in a DTS model (from Matsopoulos, 2007)

Abdul Majid, Aickelin and Siebers (2009) investigated the output accuracy of DES models and ABM when studying the human reactive behavior. The study involved

several experiments that analyzed the fitting room operation of a women-wear department where two measures of performance were used, the customer waiting time and staff utilization. Actual data from the real system was used to assist in model validation process. They study concluded that there are no differences between using any of the two modeling approaches as very similar results would be obtained in modeling this aspect of the human behavior.

C. SCENARIO DESCRIPTIONS

1. TAM Effects in Pythagoras Simulation Environment

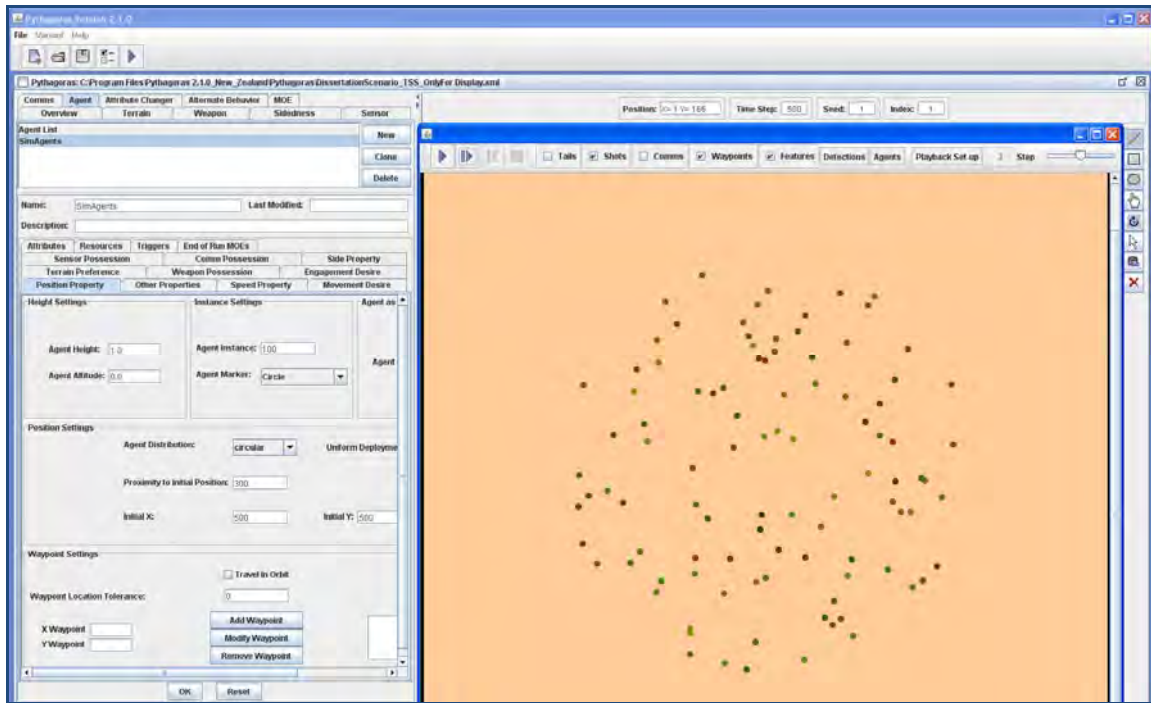
Pythagoras is a time step ABM designed to provide modelers with the ability to investigate the nonphysical characteristics of combat as well. The development of Pythagoras 2.0.0 modified these nonphysical-modeling capabilities specifically with respect to modeling human behavior. We decided to investigate the Pythagoras simulation tool because it has the flexibility and ability to explore agent complex interactions and has been used to search for emergent behaviors within a civilian populace resulting from actions targeting their beliefs and issues structures (Ferris, 2008; Seitz, 2008).

a. Simulation Method

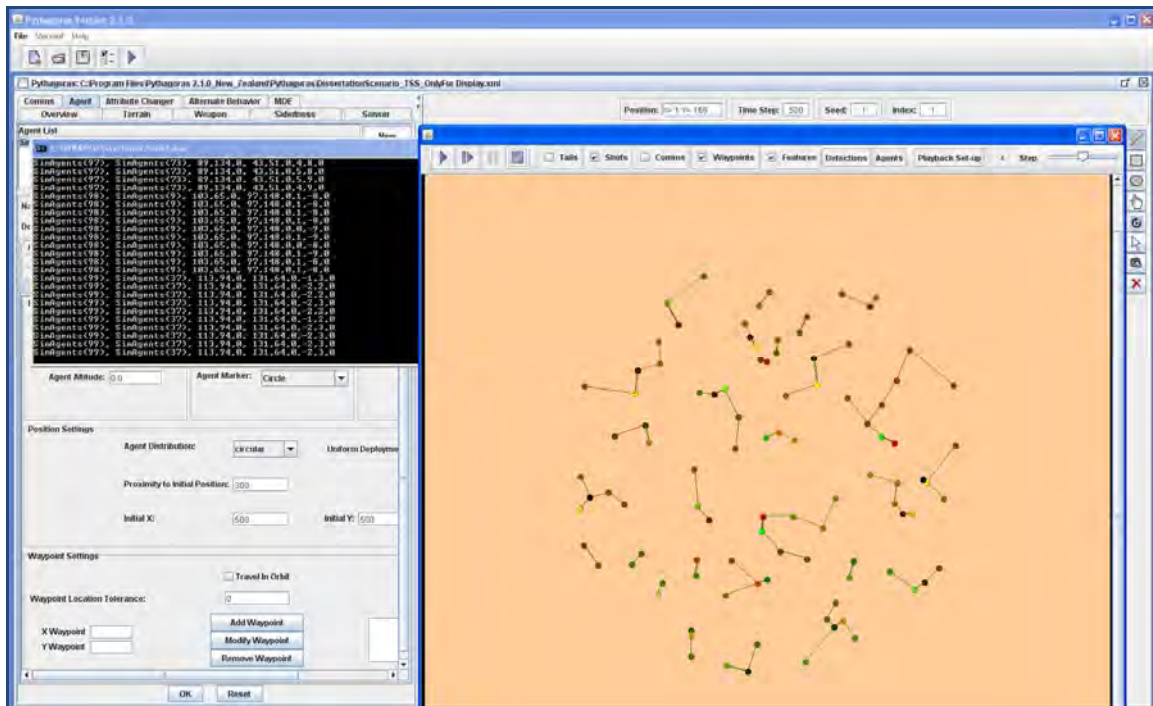
We modeled a simple human behavior scenario in the Pythagoras software involving 100 agents that can communicate with each other to influence changes along two lines of belief. In Pythagoras, cognitive and behavioral changes are modeled as “movements” across either physical space or latent (belief) space that represents dimensions of thought or behavior. For instance, we look at the dimensions of “Support for Coalition Forces” and “Satisfaction with Security,” which are modeled as changes in either red or green color. Color is the feature in Pythagoras that expresses the agent’s affiliation. Red, green, and blue (RGB) are the three colors implemented; each color can take a value from 0 to 255, and a three-way-combination is valid. So an agent may have 15 red, 152 green and 250 blue, and his color on the monitor will be the corresponding

mix. We chose these factors in order to compare these results with CGM in a later section. It is worth noting that since Pythagoras is the least developed of the HBR environments that we explore, having been built only by changes to a preexisting combat simulation, the dimensions are completely abstract. As we show below, PSOM and CGM have more solid and specific definitions of what dimensional changes mean.

In the Pythagoras scenario, agent communications are modeled by each agent using “weapon” with the only effect being changes to the targeted agent’s red and green colors. When agents “communicate,” they effectively share their ideas and move closer to each other in the red and green spaces. Since the only actions taking place in the scenario are the firing of the “communication” non-lethal weapon, we varied the effective time-step size by altering the firing rate of the homogenous agent weapon directly where, for instance, a firing rate of 1 equals a time-step size of 1, and a firing rate of 5 equals a time-step size of 5.



a)

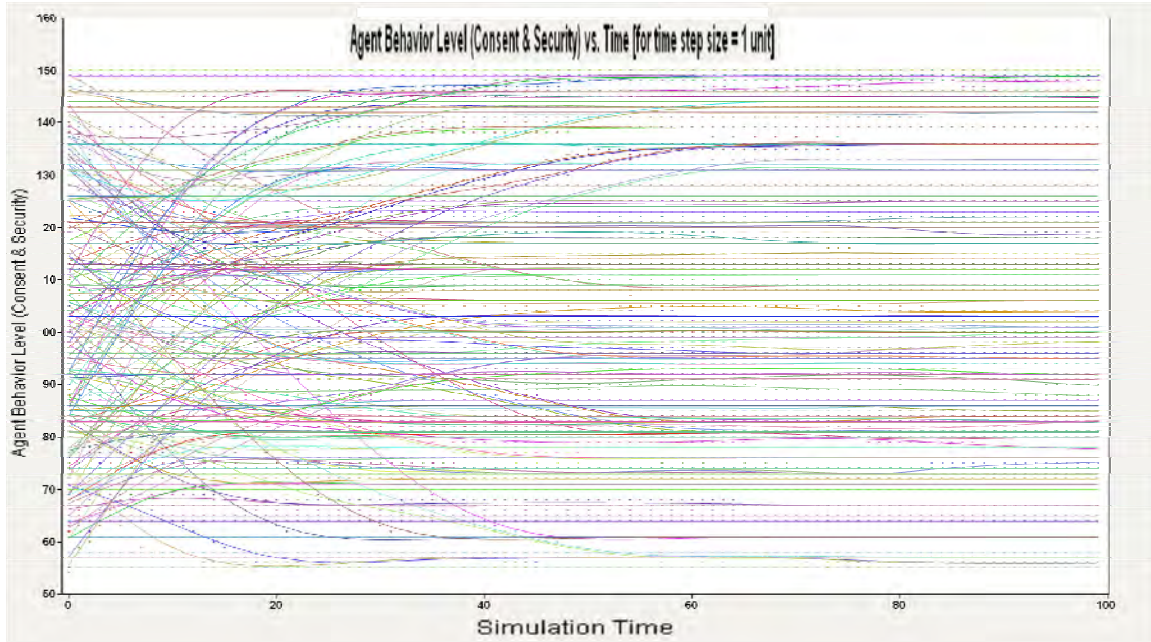


b)

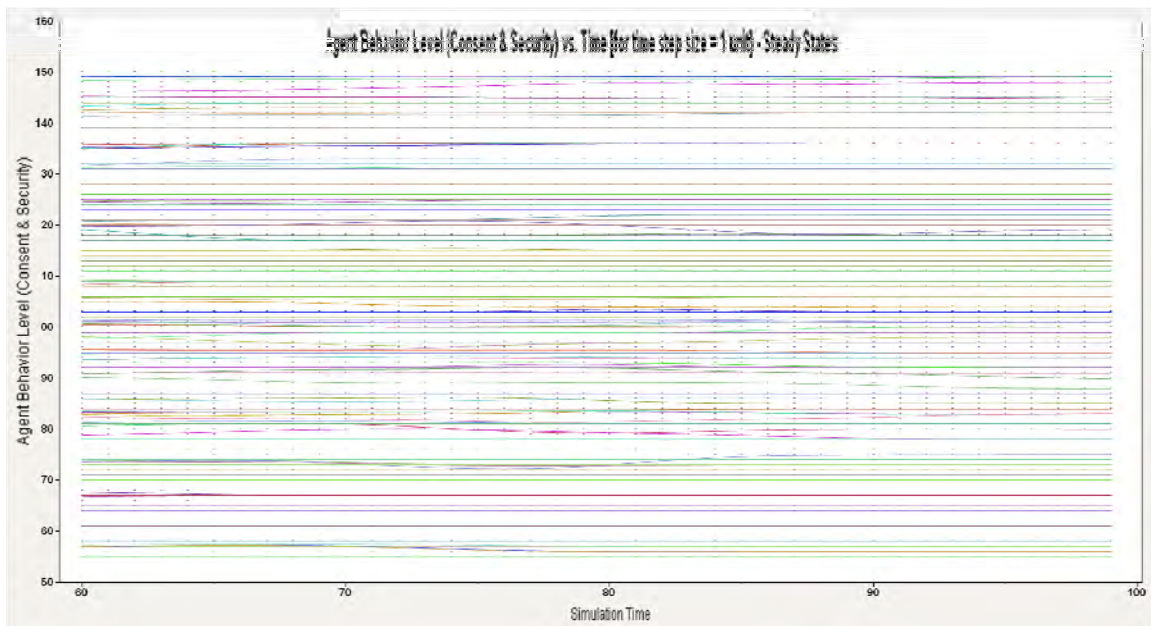
Figure 70. A social network representation in Pythagoras 2.0.0 GUI screen with 100 agents, a) initial network setting before agent communication and b) final network setting after agent communication [best viewed in color]

b. Results and Discussions

In our initial study of Pythagoras, we tested 4 time-step sizes: 1, 3, 5, and 10. Figure 71 demonstrates the belief changes for all 100 agents at time-step size 1.0. As we can see, all agents maintain steady state behavioral trajectories after some duration of time, where many converge on central values. The figure also shows that there are a few agents in the network with no change in their behavior levels due to no communications existence. We found that skipping and oscillations among the agents start to occur around time-step size 3 where some agents would fluctuate between two “belief values” on the red and green dimensions. Whereas many agents would function normally and converge on intermediate values, some agents would oscillate indefinitely and never converge. This is the same finding as above in the combat simulations where we observed the oscillation and skipping behaviors.



a)



b)

Figure 71. a) Illustration of two behavior changes (consent to coalition and security) over time of 100 agents obtained from the social scenario in Pythagoras at $\Delta t = 1.0$ [best viewed in color, but colors do not represent behavior-color]. b) close view to show steady state levels.

Figure 72 shows the small amount of oscillations at a time-step size of 3. To further investigate the phenomena, we then tested time-step sizes of 2 and 4 in an attempt to pinpoint where the oscillation behavior would start. We found that there is some oscillation behavior even at a time-step size of 2, where a small minority of approximately 1 or 2 agents will exhibit the behavior. Because there are 100 agents in the scenario, the direct visualization of the oscillation behavior exhibited with a time-step size of 2 is difficult to see on the graph and not included here.

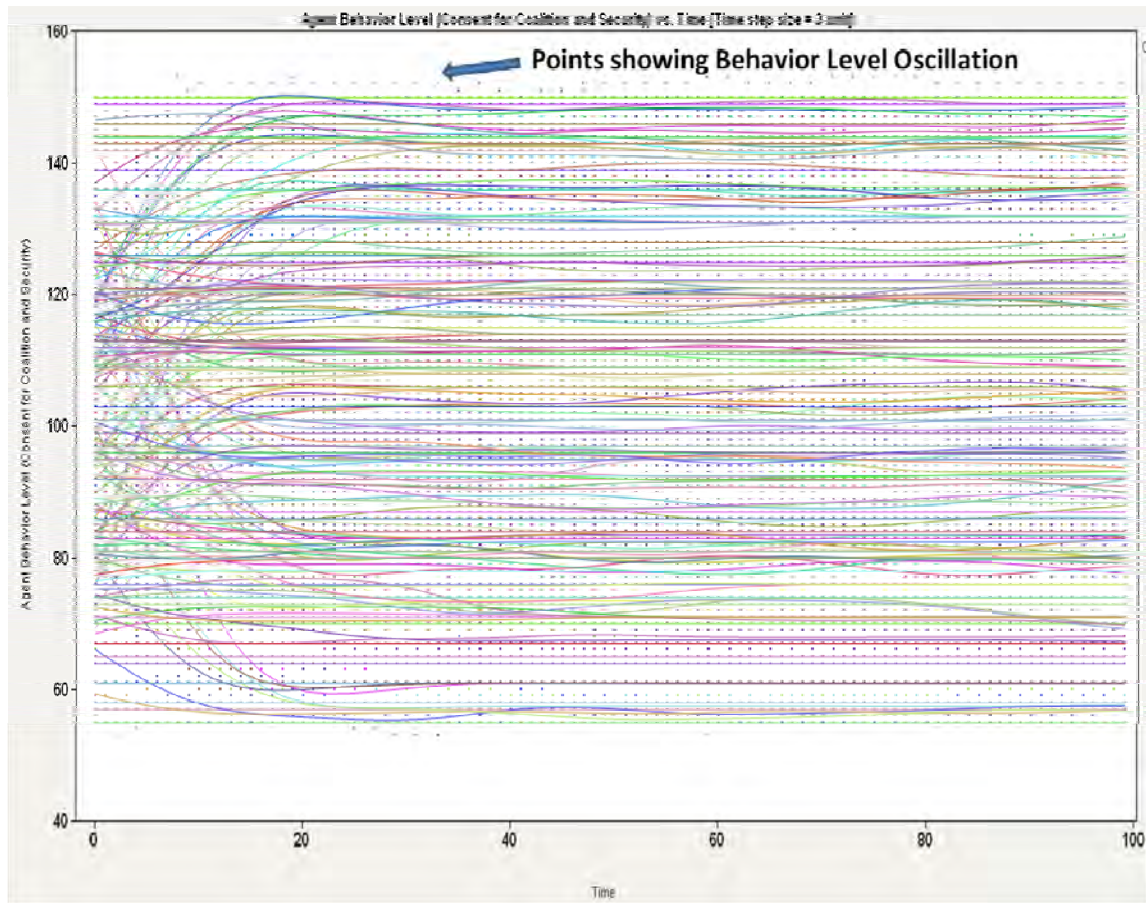


Figure 72. Illustration of oscillations in few agent behavior levels for the 100-agent social scenario at $\Delta t = 3.0$ [best viewed in color, but colors do not represent behavior-color]

As the time-step size grows, the convergence in agent belief occurs earlier in the simulation. That is, the changes in agent stance become asymptotic at earlier times as the communication frequency “firing rate” increases. We observed that the rate at which these belief level converges is not proportional to the size of the time-step size. As the time-step size increases, however, there is another effect. Whereas the agent stances generally become flat after reaching their asymptote in small time-steps, the stances at larger time-step sizes become more unstable. Figure 73 of the simulation at time-step size 5 shows these effects clearly.

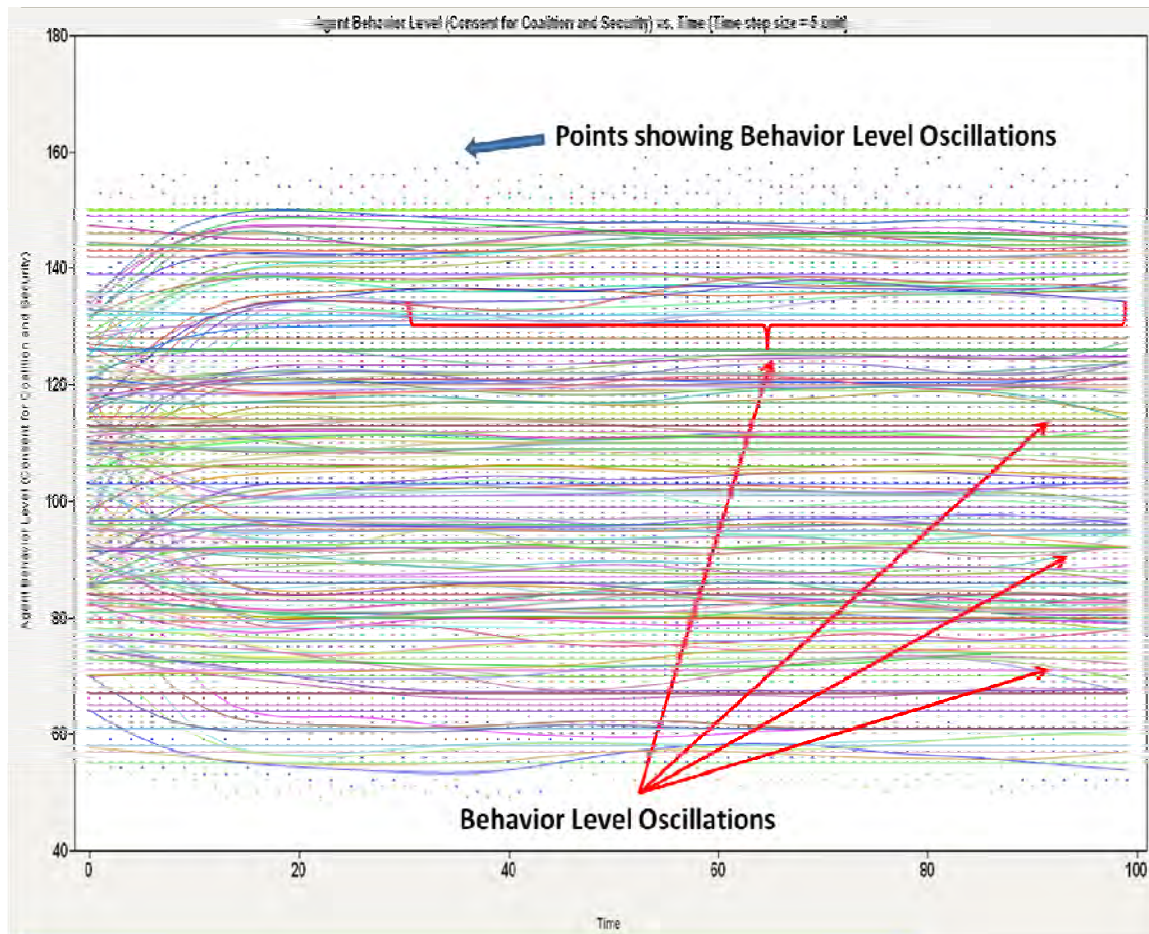


Figure 73. Illustration of oscillations in agent behavior levels for the 100-agent social scenario at $\Delta t = 5.0$ [best viewed in color, but colors do not represent behavior-color]

Another important observation about this study is that the amount of oscillations and skipping increases monotonically with the time-step size. These results are completely analogous to the results obtained in the combat simulations for the same phenomena. We posit here that the skipping occurs because of the same underlying causes as explained in the previous sections. Since Pythagoras is an extension of the combat simulation DTS environment, it utilizes the broad concepts of movement to affect changes in the cognitive and behavioral dimensions. This “movement across space” is subject to the same influences from the DTS approach as the physical movement and sensor detection in combat models. Indeed, as we can see in Figure 74, at a time-step size of 10, the agent behaviors become very unstable and the simulation experiences a very large number of oscillations where most agents never converge on intermediate values. Also, we see that at a larger time-step size, the agents that do oscillate, tend to oscillate completely among the most extreme values of the dimension, namely 0 and 255.

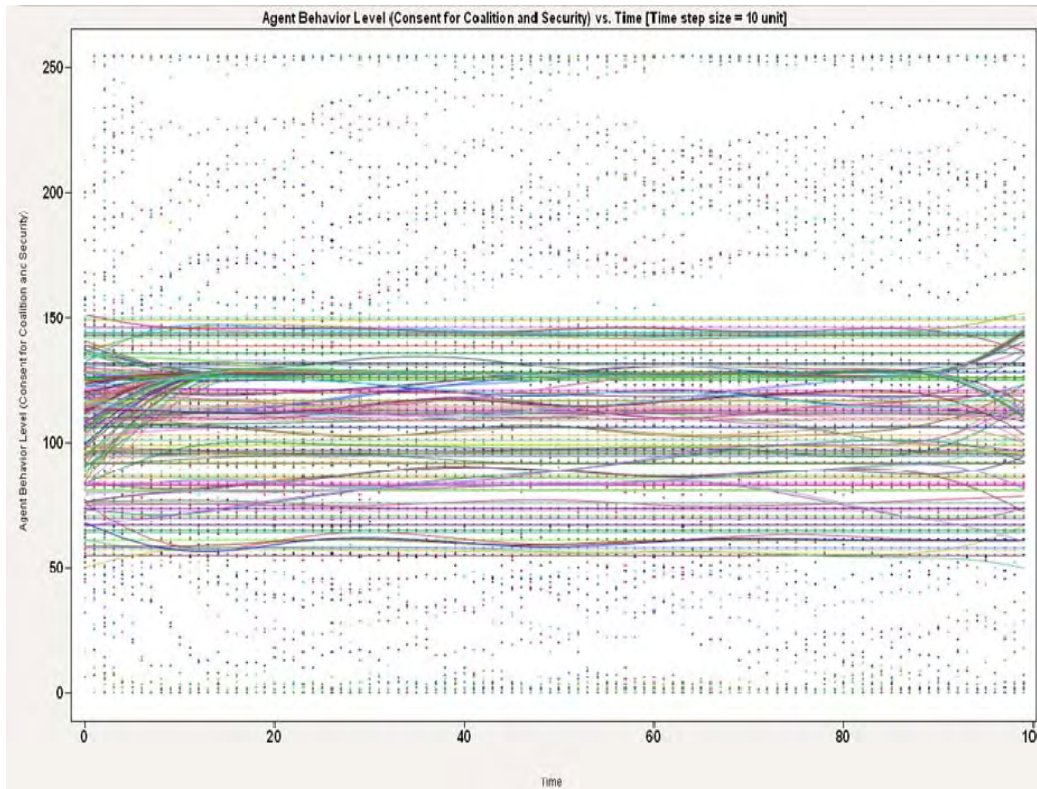
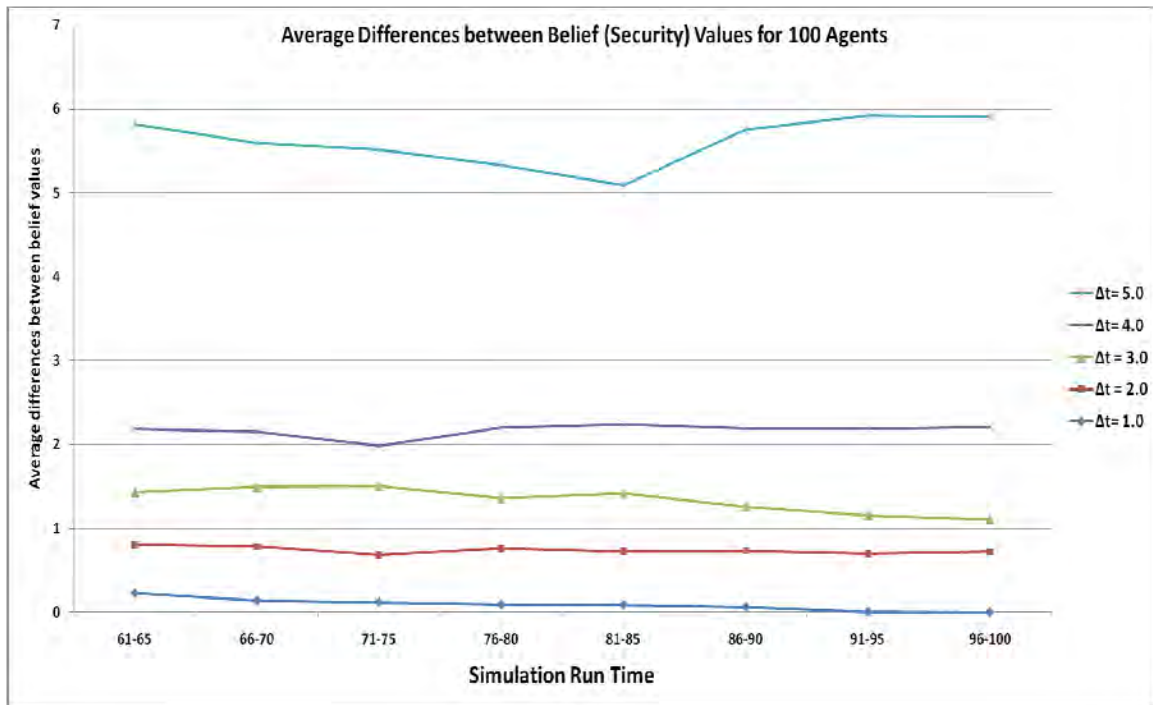
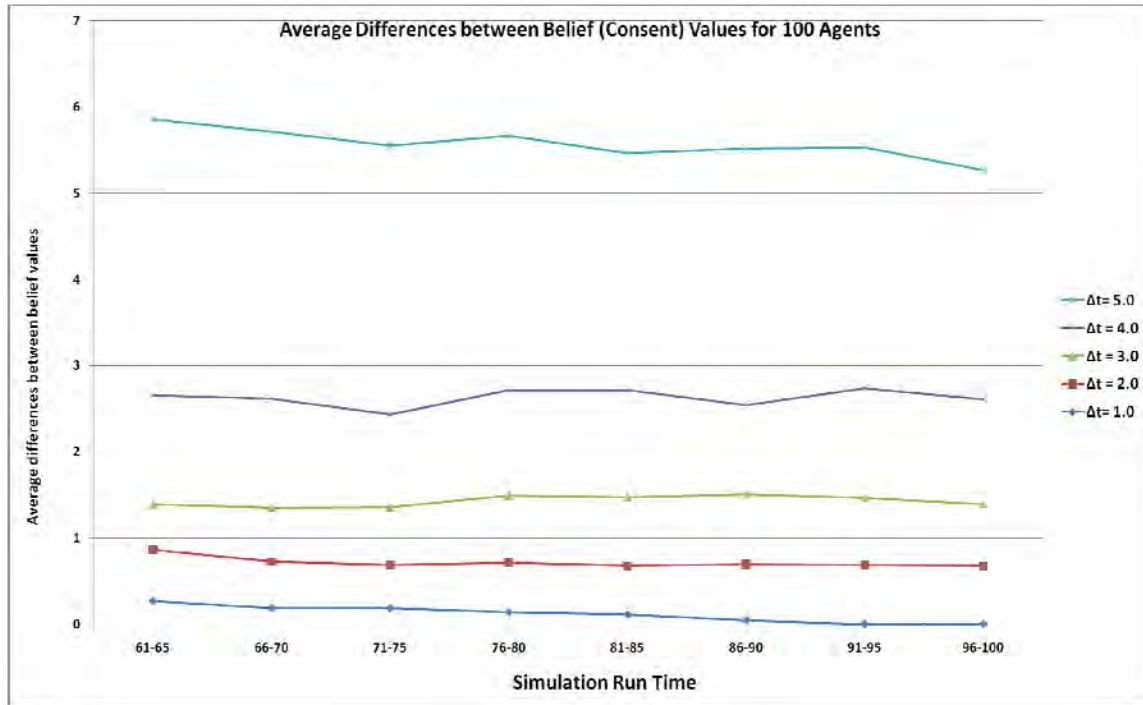


Figure 74. Illustration of oscillations in agent behavior levels for the 100-agent social scenario at $\Delta t = 10$ [best viewed in color, but colors do not represent behavior-color]

Another approach to illustrate the instability in the population behaviors at different time-step sizes can be achieved by computing the average differences between belief values for observations at every five units of simulation time. Figure 75-a shows the average differences in security belief changes for 100 agents for time-step sizes 1.0, 2.0, 3.0, 4.0 and 5.0. The figure shows that for small time-step size, there are no differences in the belief-value changes as the steady states are reached. However, as the time-step size grows, the belief values become unstable over a long period of time where there are always changes between observations and no steady states can be reached. Figure 75-b shows the same phenomenon appears with the population beliefs towards consent.



a)



b)

Figure 75. a) Illustrations of average differences in the population first belief (security) changes over the simulation time for multiple time-step sizes. b) average differences in the population second belief (consent towards coalition forces) changes over the simulation time for multiple time-step sizes.

It is easy to see how these phenomena could have a major impact of other elements of a more complex model that includes HBR as a component. This is especially true when the HBR is not visualized or analyzed alone, as we have done here. It is also clear that the time-step size can have major impacts on the final results or recommendations resulting from simulation outcome. For instance, a HBR could be used as a decision support tool to help influence information operations strategies. It would be very misleading for the decision maker to have a simulation output that indicates there would be small or no changes in the “support for coalition forces” at one time-step size, and a major change in this dimension with another time-step size. Moreover, since most analysts often look at only the beginning and ending states of the simulation, for these types of models there might be no attention paid to the oscillations or other emergent

phenomena that occur during the middle of the simulation. The resulting “final state” reported in the simulation results could have nothing to do with the actual behavior of the system being modeled.

2. TAM Effects in Peace Support Operation Model (PSOM) Environment

a. Simulation Method

We next explore the Peace Support Operations Model (PSOM) that was developed by UK Defense Ministry to support decision making in irregular warfare environments at the campaign level (Parkman & Hanley, 2008). PSOM is a time-stepped model, was built specifically for IW operations in Iraq, and has a 2004 Iraqi IW scenario built in to the environment. It was designed with the intention of supporting human-in-the-loop exercises, as well as more generalized decision support. As such, the population of each cell is visualized on a map of Iraq. Static populations are modeled on a grid representing the communities, neighborhoods, and provinces in Iraq, where the proportions of each modeled population (across a total of 135 grid cells) are purportedly equivalent to the underlying population in terms of basic demographic factors. Here we are concerned primarily about the belief stances of the Sunni populations in each grid cell, which predominately populate 65 of the 135 grid cells. Figure 76 shows the basic PSOM environment window. As we can see, statistics for different belief “stances” are shown in the grid cells on the map, and the PSOM output data references these grid cells. Stances are reported from 0 to 10, where a value of 10 represents the most “favorable” position for the stance from the perspective of the coalition forces.

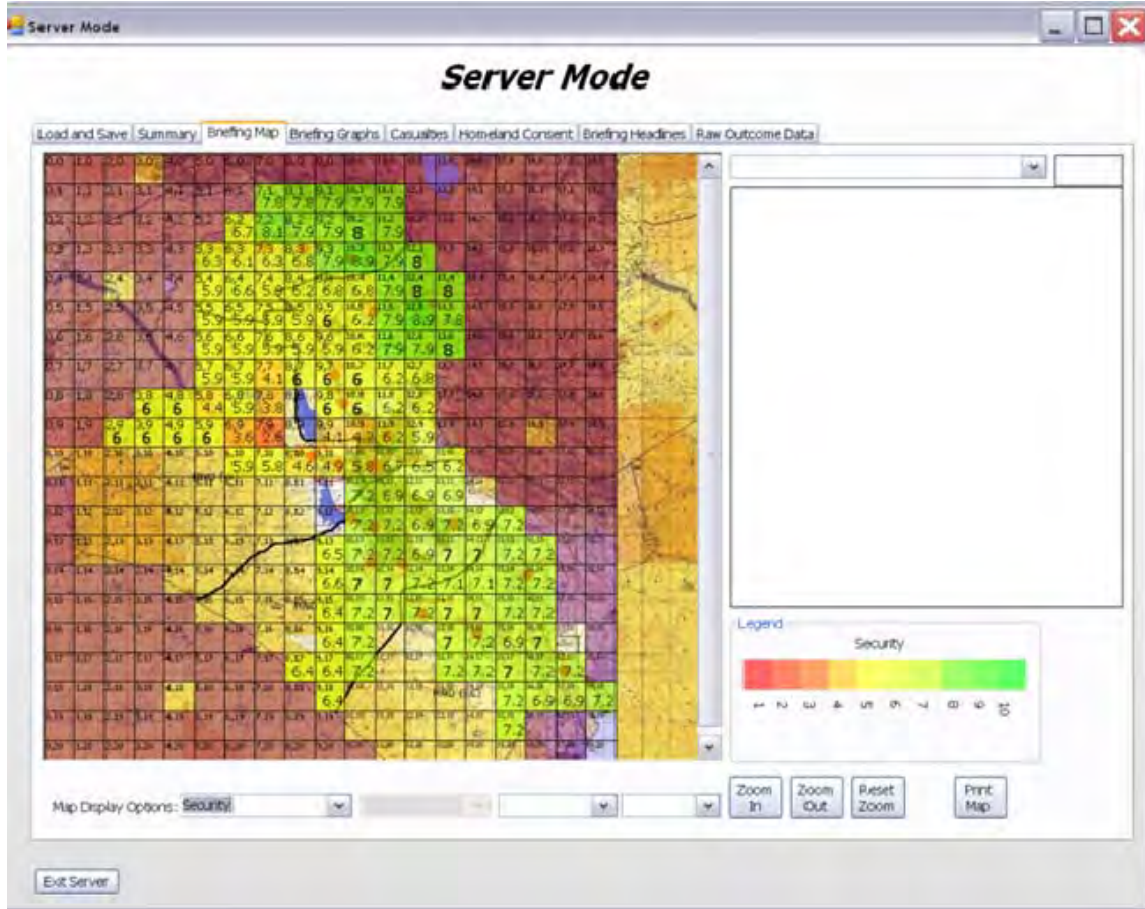


Figure 76. Graphical display of the Iraqi population security level from PSOM [best viewed in color]

We performed a series of tests to determine the effects of time-step size on two stances for the Sunni populations. Keeping with the general theme of the Pythagoras scenarios, we investigate “Consent to Coalition Forces” and “Perception of Security” for all Sunni populations in Iraq. These MOEs go along with the military doctrine (U.S. Army, 2008, pp. 1-33 and 1-77). We also test the time-step size effects on the “Perception of Security” for the entire Iraq population. The default time-step size for PSOM is 30 days, meaning that changes in the values of belief stances are reported for every population demographic in every grid cell at 30-day intervals. The PSOM simulation ends after 1 year of simulation time in all cases. The time-step size can be changed directly in PSOM by changing the time intervals which effectively change the subsequent calculations. Thus, this follows the same methodology as explained in detail

above in the combat simulation chapter, and we can posit corresponding effects on the event ordering and other emergent phenomena that result from time-step size changes. Our measures of effectiveness (Sunni consent towards coalition forces and the Iraqi population security) are collected from the output data by grid square across the area of concern. The estimator of each of these values is an aggregate of the mean response of each region based on coalition forces' area of responsibility and the entire country. The following equations are used to compute the mean of consent of the Iraqi Sunni population towards the coalition forces, where the security level for both the Sunni population and the entire country is computed in the same way:

$$\text{Mean of Sunni Consent}_{\text{Coalition Forces}} = \frac{\sum_{i=1}^n \sum_{j=1}^g \text{ConsentValue}(i, j)}{n.g} \quad (34)$$

where, n the number of simulation runs

g the grid identification (ID) number

i grid index, typically involves (x,y) in PSOM environment

PSOM is mostly a deterministic model with only few stochastic terms, therefore, the variance is expected to be small, but nevertheless worth computing:

$$\text{Variance of Sunni Consent}_{\text{Coalition Forces}} = \frac{\sum_{i=1}^n \sum_{j=1}^g [\text{ConsentValue}(i, j) - \text{Mean of Sunni Consent}_{\text{Coalition Forces}}]^2}{(n.g) - 1} \quad (35)$$

b. Results and Discussions

We tested PSOM with seven time-step sizes at 5, 7, 14, 30, 60, 90, and 120 days. Attempts to test the model at a time-step size of 1 day failed as we obtained consistent memory errors at these small time-step sizes⁸. Our results show that the time-step size has a major impact on belief stance changes in PSOM. There are significant

⁸ This error was apparently due to the PSOM software rather than the computing environment, which has performed all of the other tests in this manuscript without error.

differences between the results obtained by the default time-step size of 30 days, and time-step sizes both smaller and larger than this default size. For the smallest time-step size we were able to test, 5 days, the average stance of “Consent to Coalition Forces” for the Sunni populations in Iraq at the end of the simulation run was almost 50% different from the results obtained at the default time-step size. Since most modelers would support the notion that, in general, a smaller time-step size leads to more precise results, one could argue that the default time-step size in PSOM can lead to misleading output.

In all cases, the average Sunni stance on coalition forces decreased during the course of the simulation run. From the initial average stance of 2.5 (out of 10), the ending range was between 1.16 for the smallest time-step size, and 2.38 for the largest (120 days). The output of 2.38 represents a sizeable 8.6% difference from the results obtained by the default time-step size, although it was not as extreme of a difference as the output of 1.16 obtained with the 5 day time-step size, which is over a 53% change from the initial stance value of 2.5.

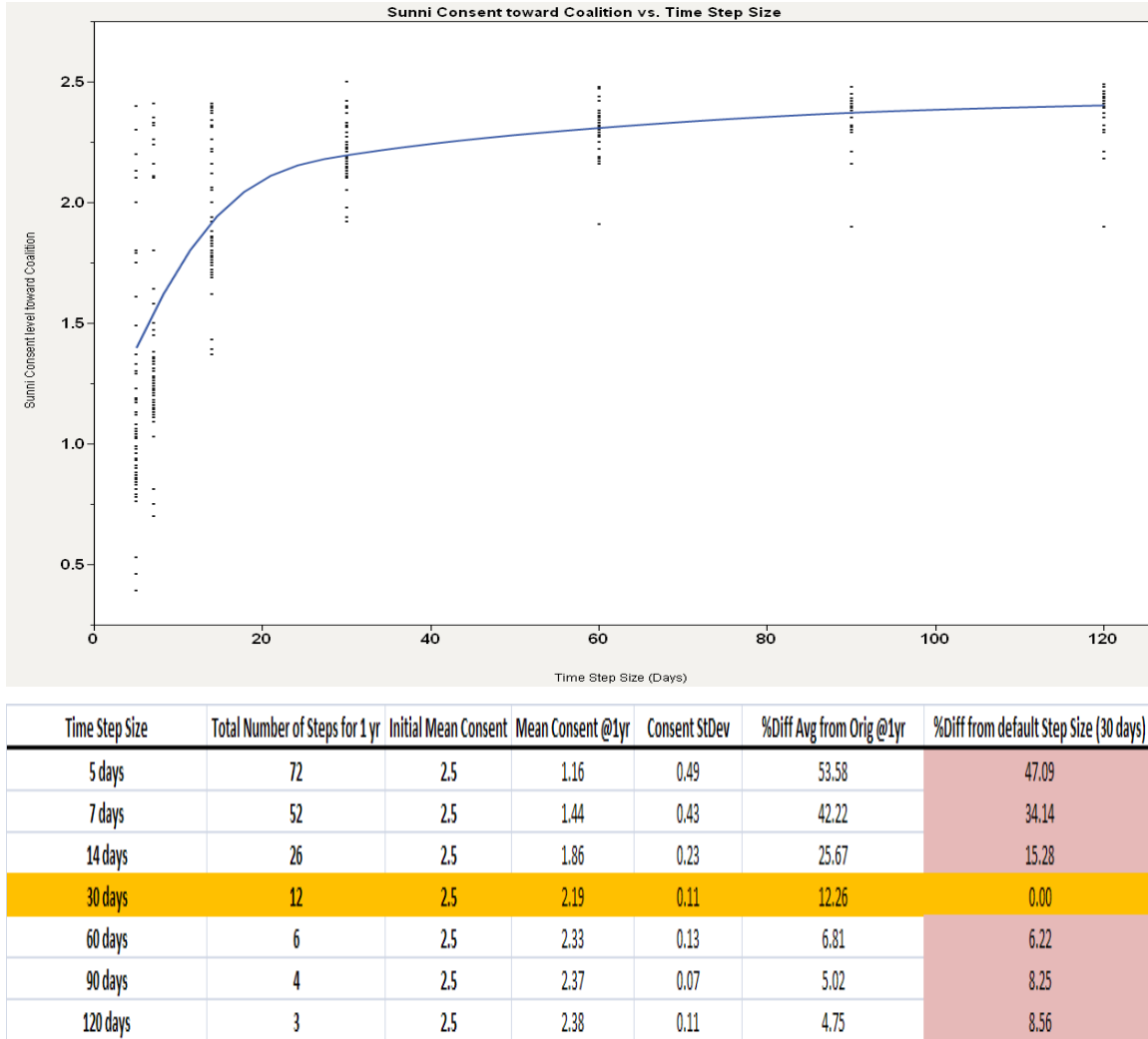


Figure 77. Plot and table of the Mean Sunni Consent values towards coalition forces at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days

Our results also show that the impact of time-step size changes increased more rapidly when the Δt decreased from the default compared with increases of Δt over the default. This suggests that the default time-step size of 30 days may be closer to the extreme large Δt range than a middle-ground Δt . This is supported by the fact that the mean difference in the consent value from 2.5 for all time-step sizes was approximately 0.53, which would result from a time-step size of just over 14 days. This can be seen in Figure 78 where the “Average Outcome” is marked on the stance change curve.

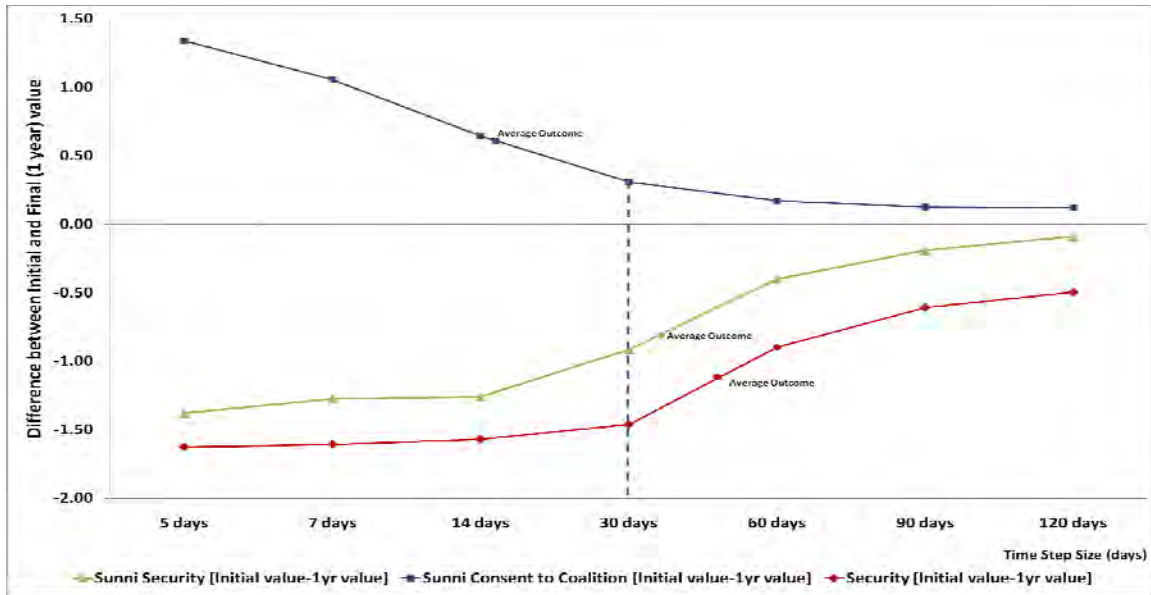


Figure 78. PSOM's Iraqi scenario results for the differences between initial value and the final value at 1 year for the Sunni Consent towards coalition forces, overall security level and the Sunni security level [best viewed in color]

The results above are even more interesting in the context of comparison with the result we obtained for the changes in the Sunni stance on security for different time-step sizes. The Sunni stance on security also changed in significant ways as a result of the time-step size in PSOM. However, unlike the stance on coalition forces, the stance on security increased from the default of 30 days as the time-step size decreased, and decreased from the default when the time-step size increased. The average Sunni stance on security started at 5.8 (out of 10) and, at the default time-step size of 30 days, ended at 6.72, representing a 15.8% change. The value obtained at the smallest testable time-step size of 5 days at the end of the run was 7.18, a 23.8% change from the initial value of 5.8. This represents an approximately 7% increase from the end value obtained at the default time-step size.

At the largest tested time-step size of 120 days, the ending value was only 5.89, representing a minimal 1.55% difference from the original value, and a 12.3% decrease from the end value obtained at the default time-step size of 30 days. While the largest time-step size only resulted in a 1.5% increase from beginning to end value, which

may or may not be significant, the over 12% change in the end value from our largest tested time-step size versus the default time-step size is noteworthy. In other words, with the default time-step size, there was a somewhat impressive increase of over 15% in the Sunni perception of security. The smaller time-step sizes resulted in even larger changes to this stance, and the larger time-step sizes resulted in very minimal changes to the stance over the simulation run. Importantly, this trend is the opposite of the effect that was seen for the Sunni consent to the coalition.

Although we did not perform a deep analysis of the internal mechanisms of PSOM, our impression of the environment would suggest that both of these stances- support for the coalition, and perception of security- are presupposed to trend together. In other words, since the goal of PSOM is to model the effects that “blue forces” have on perceptions in the population through various irregular warfare operations, being successful along one line of support, such as increasing the perception of security, should be reflected in corresponding stances, such as support for the coalition. The results obtained here would conversely suggest that support for coalition activities falls dramatically as the coalition becomes increasingly more effective at securing the area of operations. We do not claim that these trends could not co-occur in the actual system; however we posit that these results potentially present enormous difficulties for users of HBR simulations such as PSOM.



Figure 79. Plot and table of the mean sunni security values at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days

Our investigation of the effects of time-step size changes on the perception of security for the entire Iraqi population followed the same general trends as the changes observed for the Sunni perception of security. The initial average value in all cases was 5.8, and the end value for the default time-step size of 30 days was 7.26, representing a 25% increase. As the time-step size decreased down to 5 days, the average end value increased to 7.43 which is only marginally higher than the end value obtained at the default time-step size. However, as the time-step size increased the average end value for the stance decreased. At the largest tested time-step size of 120 days, the average end value was 6.3. This is an 8.6% increase from the beginning stance value, and 13.3%

decrease from the average end value obtained at the default time-step size. The table below shows the changes in security stance for the entire Iraq population along with the average differences between the initial values and PSOM default value of 30 days.

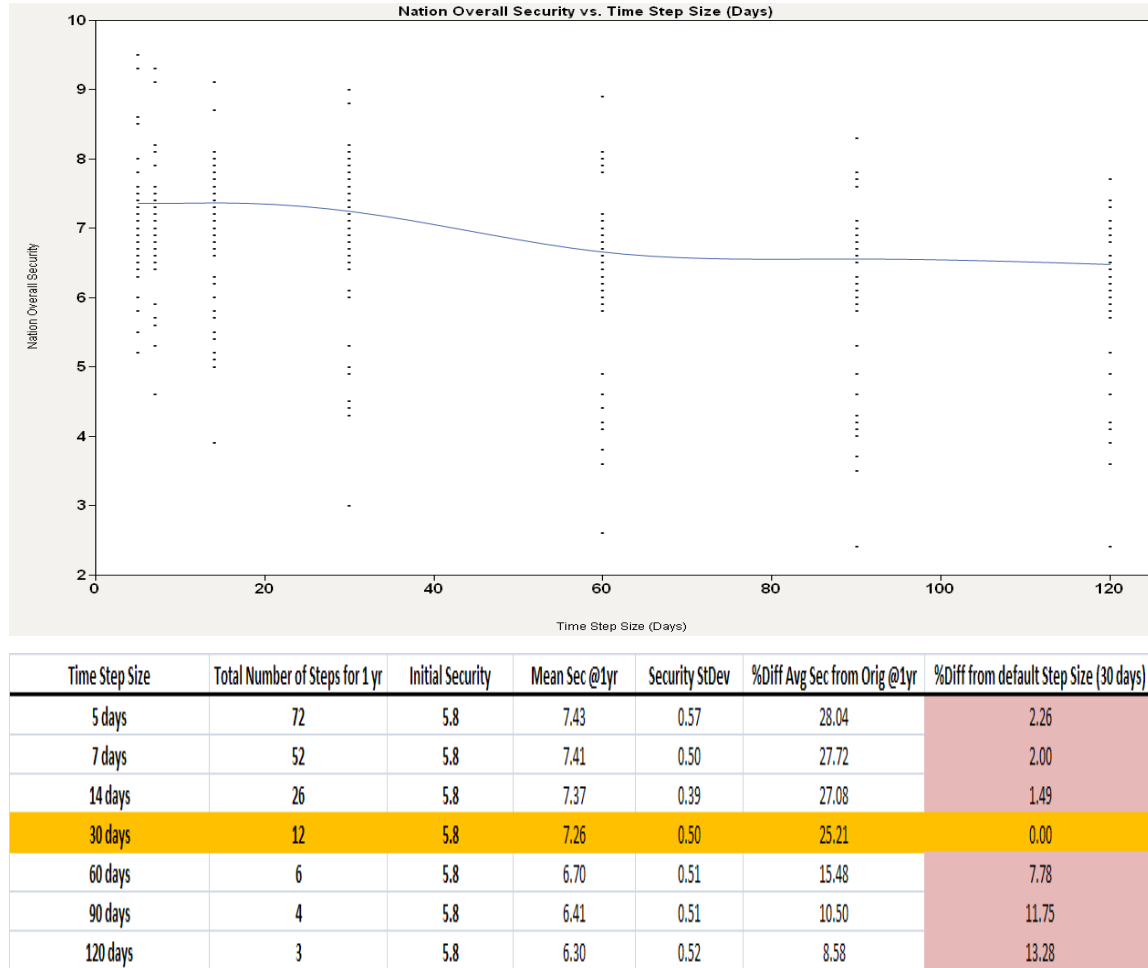
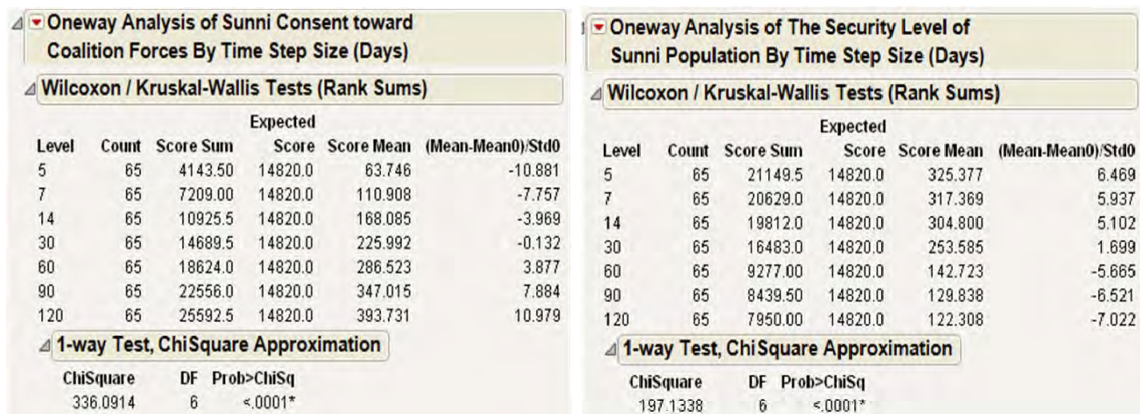


Figure 80. Plot and table of the Iraq population mean security values at the end of 1 year for multiple time-step sizes, with PSOM default time-step size value of 30 days

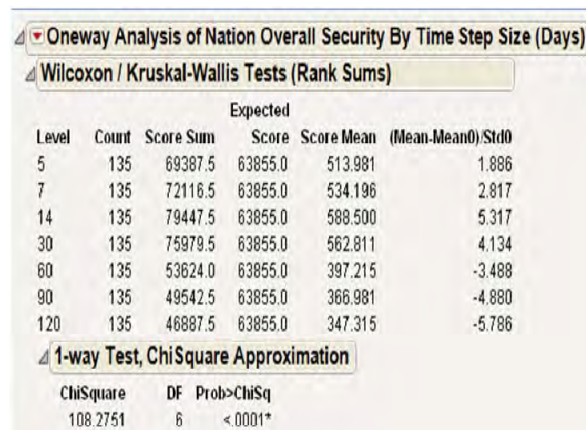
To further support our analyses, we performed a Kruskal-Wallis Ranked Sum Test on the output obtained from PSOM in order to determine if the outputs from difference time-step sizes were statistically different. We use this non-parametric test to compares multiple population MOEs where the null hypothesis is that all population MOEs are equivalent. We performed the test on all three variables, Sunni stance towards coalition forces, Sunni perception of security, and overall perception of security for the

population of Iraq. In all cases, the p-value was less than the set level of 0.01. Therefore, the results obtained by the different time-step sizes are significantly different for all variables measured. Figure 81 shows the ranked sums, chi squared approximation and p-values for each variable.



a.)

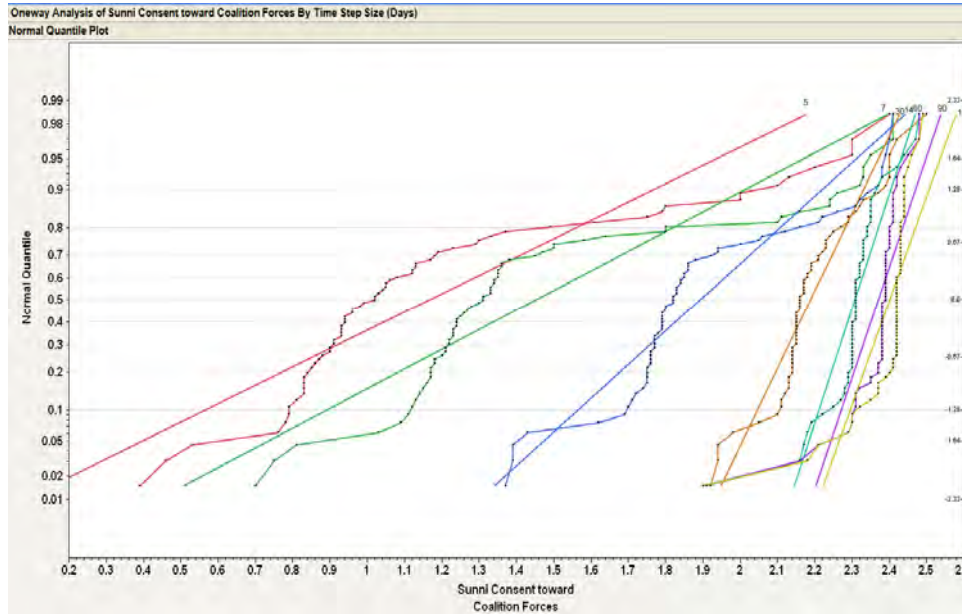
b)



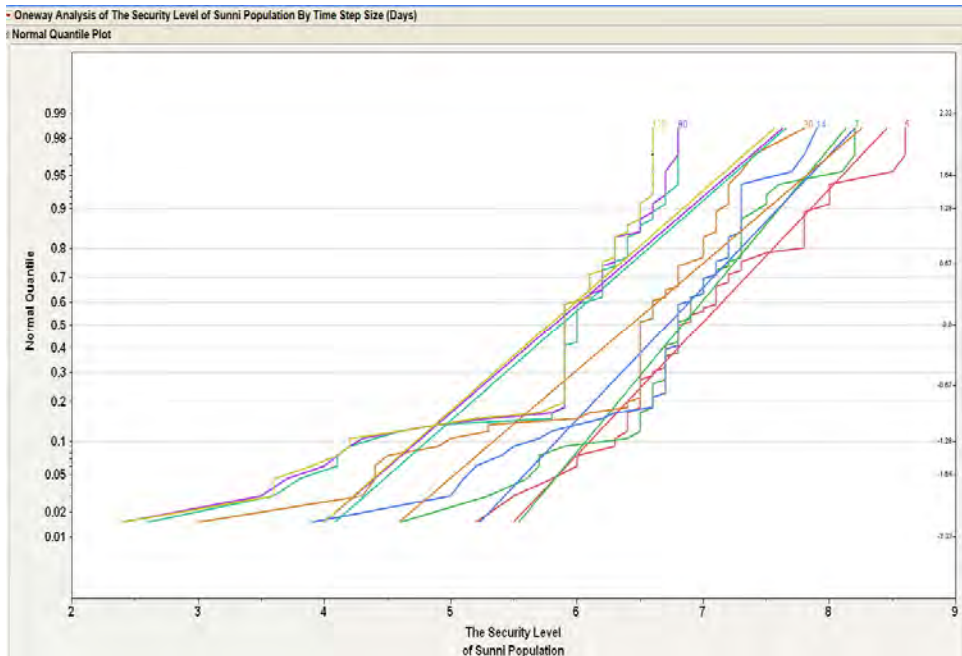
c)

Figure 81. Kruskal-Wallis rank sum test for a) sunni consent, b) sunni security, c) Iraq security for different levels of time-step sizes

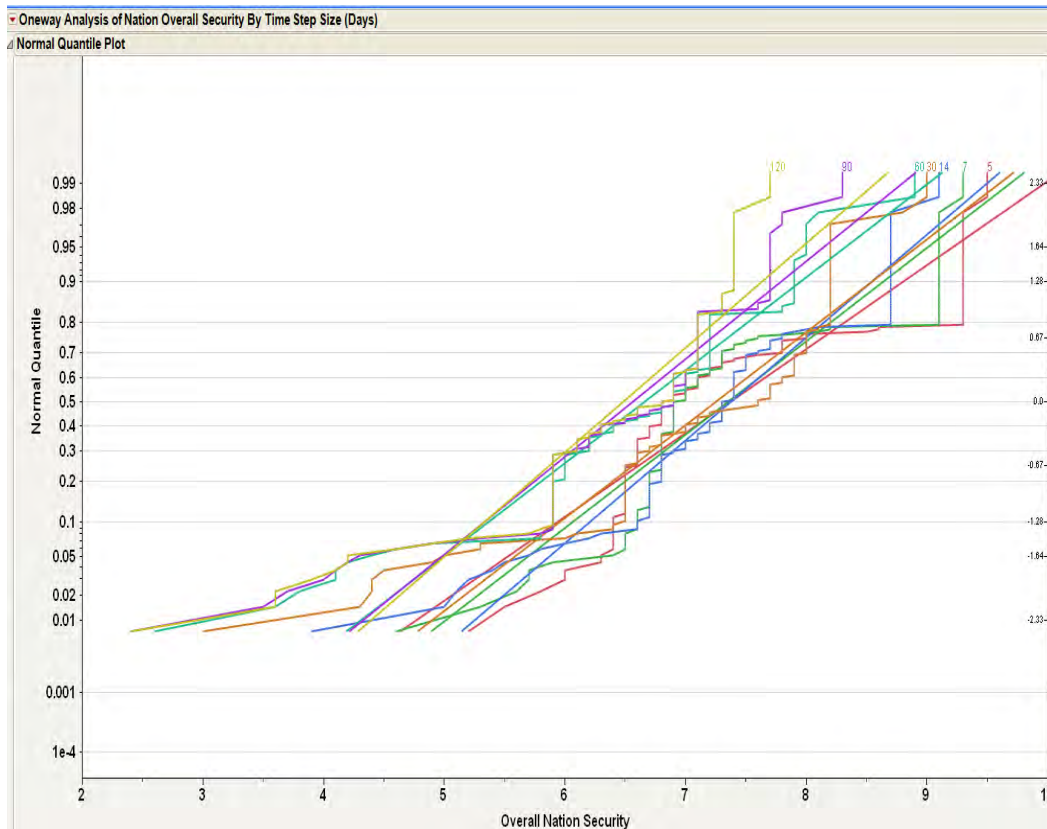
We constructed quantile plots for the same MOE results to a number of randomly selected data set obtained from simulation runs. Although these plots did not show that PSOM results statistically follow a normal distribution, the gradients provide great insights of how similar or different these results are when put together for comparison.



a)



b)



c)

Figure 82. Normal Quantile plots for a) sunni consent to coalitions, b) sunni security, and c) Iraq security with multiple time-step sizes (5,7,14,30,60,90,120 days), [best viewed in color]

As can be seen from the above Figures, there are significant differences between the Sunni Consent results in which the slopes are noticeably not equivalent. For both of the Security results (Sunni and entire Iraq population), the lower group of small time-step sizes (5, 7, 14 days) seem to be slightly different from each other. Similar observations were obtained with the upper group of large time-step sizes (30, 60, 90, 120 days).

As seen from the results above, PSOM as a DTS model is very sensitive to changes in time-step sizes. This makes decision making and implementing the “right” strategies to win the civilian population a more complex task in IW military operation. The model sensitivity to step size can be confounded with other factors in the model,

which makes it hard for analysts to distinguish the cause and effect of behavior changes. We acknowledge that there is no valid answer to these types of model, but when models produce confounding results depending on time-step sizes, resource allocation can be at great risk.

3. TAM Effects in Cultural Geography (CG) Model Environment

This section addresses the qualitative aspects of the Cultural Geography (CG) model to demonstrate the capability of a DES approach to simulation human behaviors. CG is a java-based model of human behavior that has been in development by TRAC-MRY since 2006. CG is a discrete event, agent-based model design to provide insight into the behavioral response of a civilian population and their stances on issues regarding PMESII in IW environment. The CG model shows the capability of the DES approach to simulate civilian responses to blue force actions to provide a better representation of human behavior changes. The model is implemented in Simkit, and the use of event listeners and data loggers in which any event that occurs within the model can be captured. The CG model is used in this study to support our qualitative comparison analysis between DES and DTS approaches. The analysis to identify main benefits and disadvantages of the DES approach in comparison with the DTS models (Pythagoras and PSOM) in the area of IW is discussed.

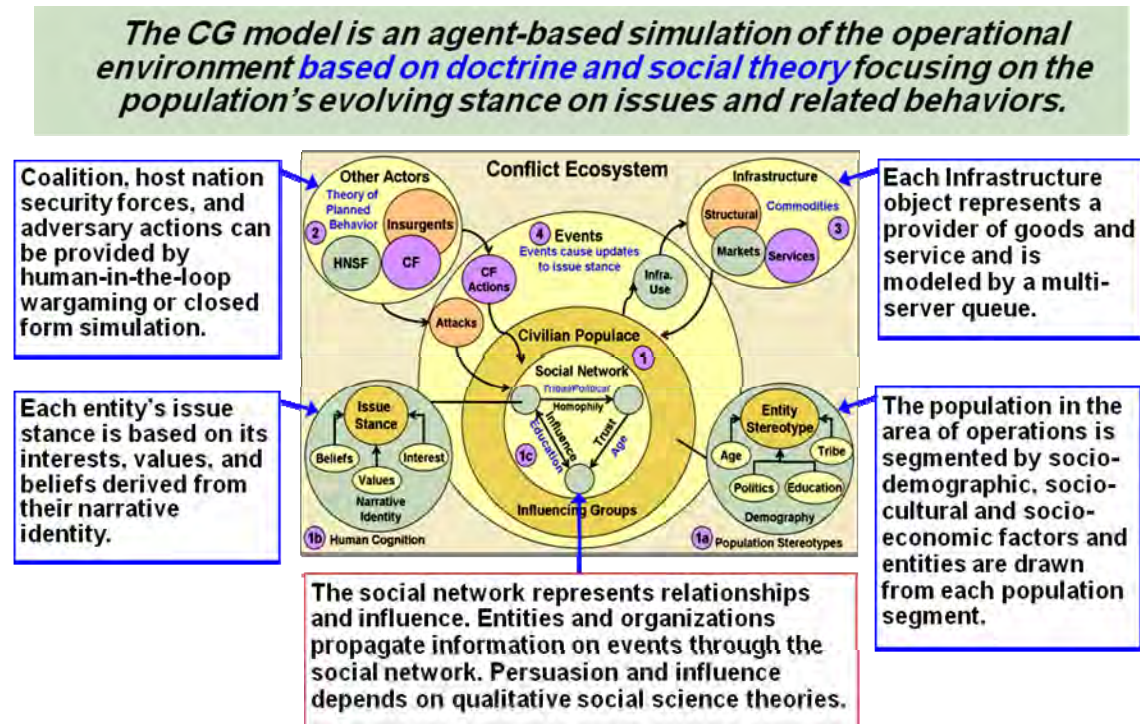


Figure 83. Cultural Geography (CG) model description Chart (TRAC-MRY, 2011)

a. Simulation Method

The work is built on unclassified scenario set developed by TRAC- MRY in a Middle Eastern major city. In the scenario stabilizing force agents (government, coalition, and local forces) seek to influence the civilian populations satisfaction on a number of issues related to security, election and infrastructure. While the opposing insurgent forces seek to destabilize the region by reducing the population satisfaction on these issues. Our purpose is to go along with the goal of the CG model, that is not to predict population responses to specific actions, but to use it as a tool for gaining insight into the effects that certain actions by other actors in the environment (stabilizing forces, insurgent forces or civilian agents) have on the population.

The CG model is built on Simkit and works on the basis of implementing and integrating social science and behavioral theories. At the beginning of the simulation run, agents are instantiated by using real survey data about beliefs, values and interests

(BVIs) and a social network for the population is generated by connecting them according to social factors (e.g., age, education and tribe). Keeping with the theme of this chapter, we look here at the “Satisfaction with Security” BVI. The CG model framework is composed of “plug and play” modules that allow researchers and users to implement and integrate different social theories in order to address precise and novel research questions.

In the CG model, population response is in the form of a change in agent’s stance on the relevant issues using methods such as a Bayesian belief network. As in PSOM, we consider two major measures of effectiveness, the population stance towards security and election (as a result of consent to coalition forces). Similar to Pythagoras, the CG model allows for communication among the agents of the population through the use of a social network and population stances to certain issues can change as a result of these communications. Specifically, the agents are able to pass messages in the social network about events occurring within their region to other agents in the scenario. An arc in the network represent a connection link between agents, and the connection strength is based on the concept of homophile.

We explored the output of a single run from one design point to analysis the effect of different events occurrence on the city population stance towards the security and election issues. The data used in our study is obtained from several CG model output files provided by TRAC-MRY.

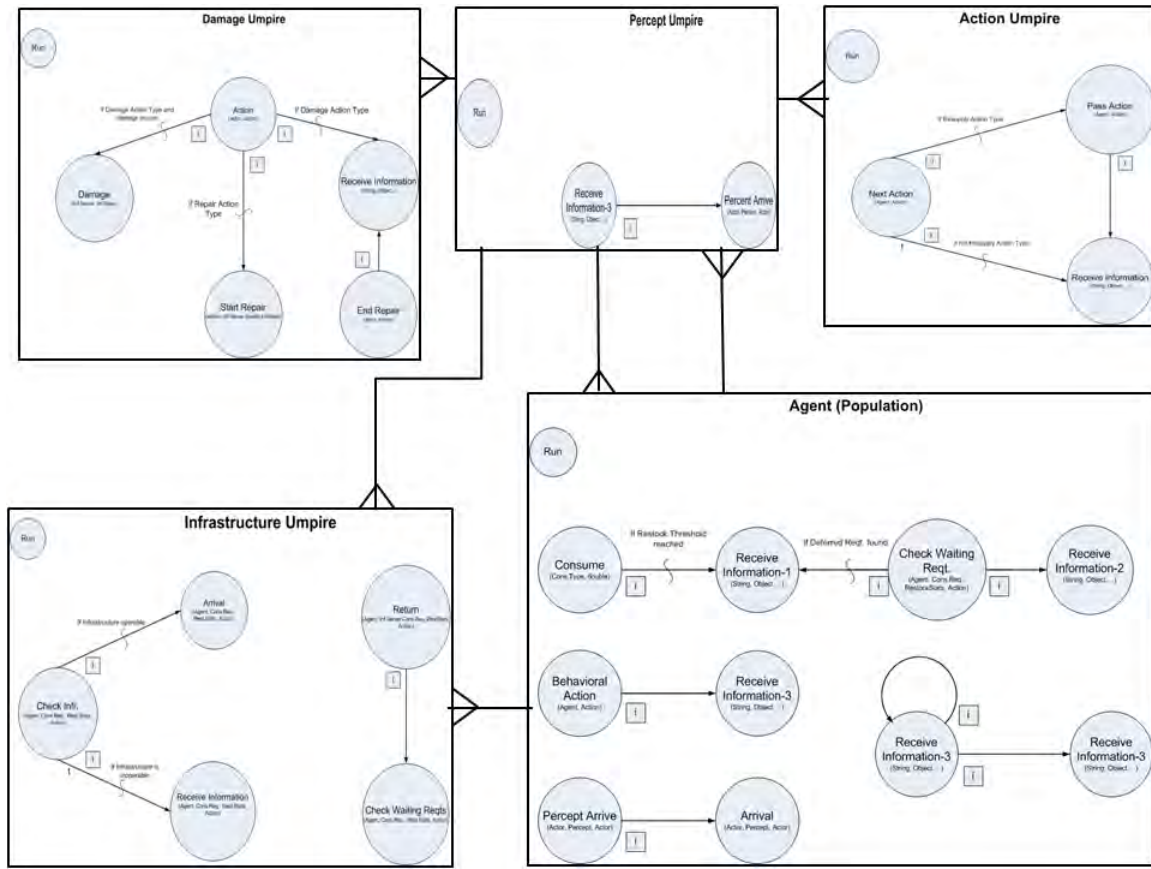
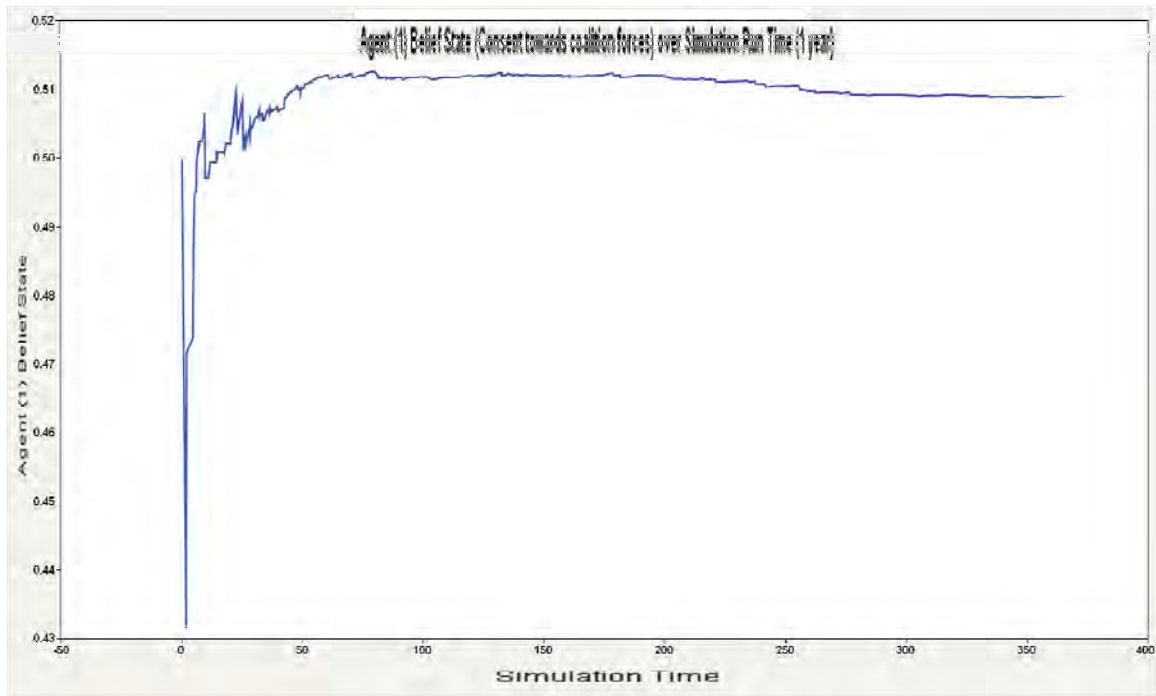


Figure 84. A sample illustration of Event Graph methodology used in the CG model (after TRAC-MRY, 2011)

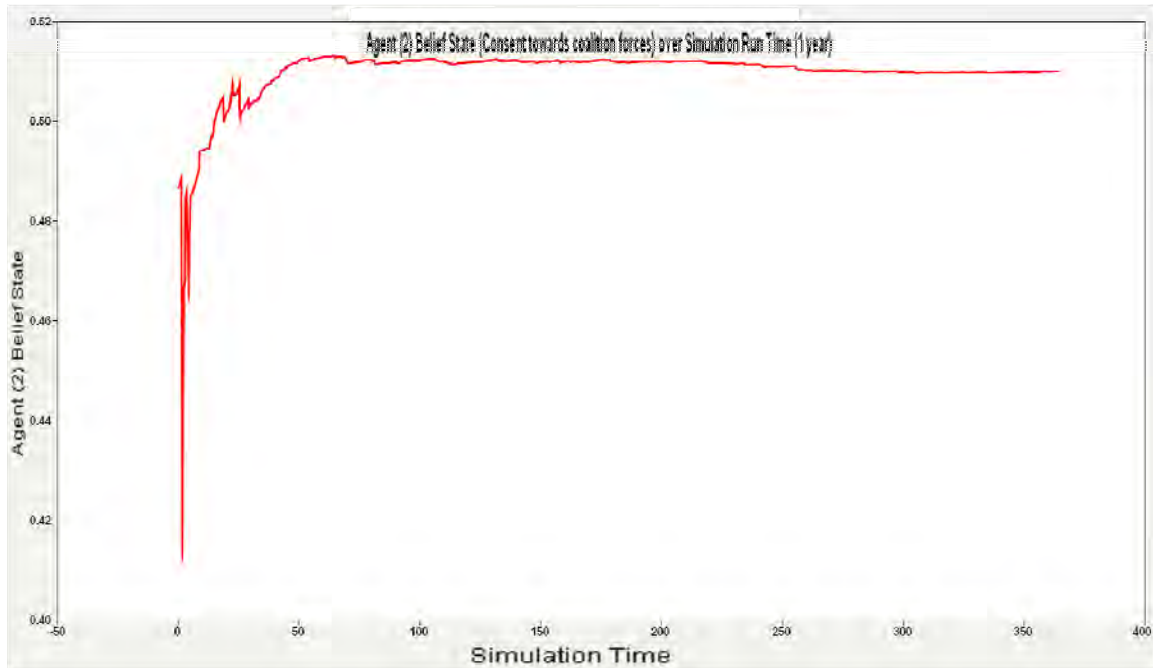
b. Results and Discussions

The results of the analysis for the population stance towards security in response to actions by forces in the population environment show convergence in each agent security value to a steady state levels. These results cannot be validated, but can provide decision makers with valuable insights to the amount of improvement in security and stability in the region. Despite results accuracy, and unlike the results obtained from Pythagoras and PSOM environments, the CG results are more stable with small noise factors. This tends to provide a better understanding of the sensitivity of the response to changes in population behaviors. The change in belief state over the simulation time (e.g., 1 year) for two arbitrary agents are demonstrated in Figure 85. Since CG simulation is event driven, both agents are responding to frequently occurring events at the start of

the simulation with frequent changes in their belief states. However, the state-change rate reduces or reaches a steady state levels in the case of fewer or no events occurrence. Despite the accuracy of the results, the behavior of human in IW operations seem to have better representations of real human behaviors using the DES approach rather than a DTS approach.



a)



b)

Figure 85. Political belief state changes over time for two arbitrary agents a) agent 1 and b) agent 2.

Another demonstration of the CG model results is shown in Figure 86 below where the mean of security satisfaction level for few agents within the populations is plotted over time. We observed that the population consent towards coalition forces produced similar results converging to steady state levels.

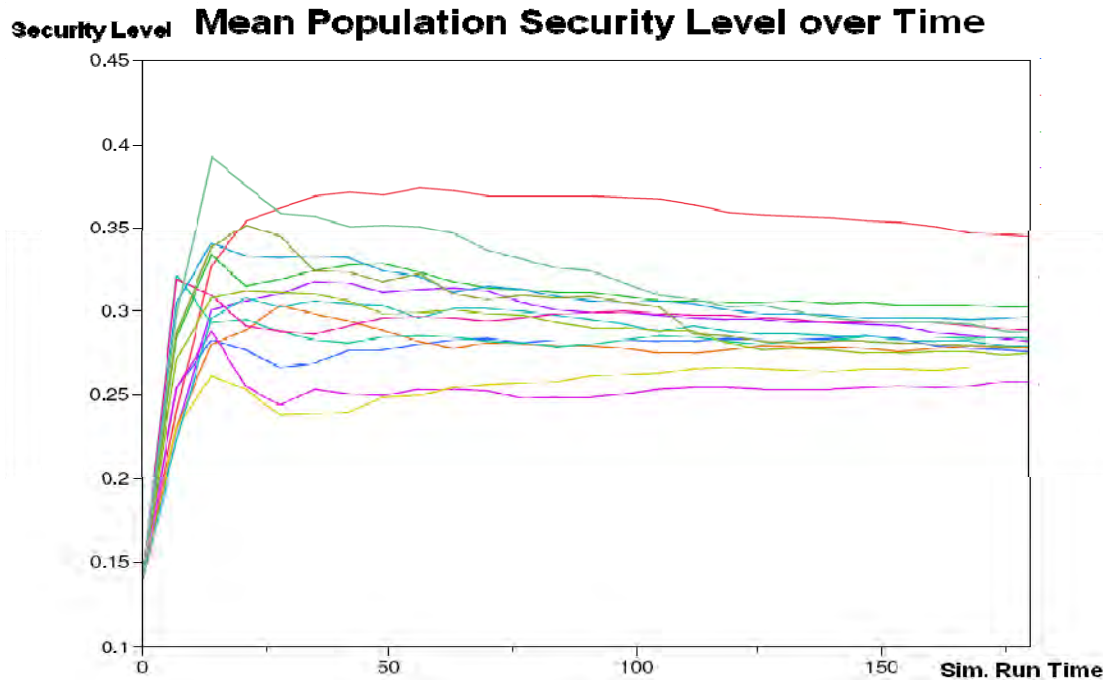


Figure 86. A demonstration of CG results of multiple agents' stances towards security over time [best viewed in color]

As we can see in Figure 86 above, agent belief-state changes are qualitatively different than the output derived from the DTS approaches. Agent states change in response to events occurring in the environment, either by direct observations or interactions (communications) with other agents. The state changes are a direct reflection of what is occurring in the simulation environment at any point in time. Changes are represented without artificial delays that could be introduced by ordering phenomena (simultaneous event representation) or other emergent effects resulting from time-step size in the DTS approach. As events occur within the model, information about the events is passed on to entities as observations about the world. This observed information is mapped to the appropriate belief and impacts the entities stance on the relevant issue by updating their prior world view. Individual agents report changes in their internal states, while simulation events, including communication events between agents, are managed and reported by the centralized event manager.

As we can see from the results, HBR DES output has two important characteristics that differ from the HBR DTS output. Firstly, agent state changes are reported individually and this allows for user-defined levels of aggregation, all the way from single agent analyses to whole population analysis. Secondly, the use of the event list in the DES framework allows for the analyst to draw conclusions about event and state traceability as explained below.

The discrete event simulation (DES) in the CG model provides a well-documented alternative to time-step simulation (DTS) and shows potential for applications across the domain of HBR. Human behavior representation in DES, as in the combat simulations, is centralized in a notion of time that tracks changes to the agent population via a central future event list (FEL), leaving the individual states of agents tracked and managed internally.

One of the benefits of DES in the CG model is that agents not only keep track of their own internal state and can report/output state changes individually but DES allows the flexibility to link state changes to events. This can be very challenging in DTS models in which events occurrence is simultaneous at the end of “large” time-step sizes. Likewise, communications between agents (communication events) are reported with or without regard to the internal states of agents. As in the other DES examples in this manuscript, this TAM allows users to focus analytic efforts on specific questions based on, for instance, specific times, specific events, or specific groups of (heterogeneous) agents. Analysis can range from very high level measures, such as general changes in population sentiment during a long period of time, to tight granular analysis of one specific agent across a short period of time. This allows agents, and groups of agents, to be compared with one another, either at the same point in time, or different points in time for simulation-length longitudinal analysis. These features are difficult to capture in DTS models, specifically at large time-step sizes for many reasons. First, when events occur at high frequency, the DTS model typically report multiple events at the end of each time interval. In this situation, specific questions on agents behavior change as an affect of specific events occurrence are not feasible. Second, events occur in certain region of the country should only effect agents within the region. In the DES model only agents within

the affected region will response to these events and change their behavior accordingly. However, in the DTS model, all agents' behaviors are recomputed at every time step to checks for update and only effected region agents change behaviors in response to these events. For example, suppose only a single event (i) occurs in region (k) during one week period and only at most (m) agents out of the entire N population are affected (through observation and communication). In the DES model, there are only (m) agents behavioral changes. While in the DTS model, there are N agents behavioral checks every time step.

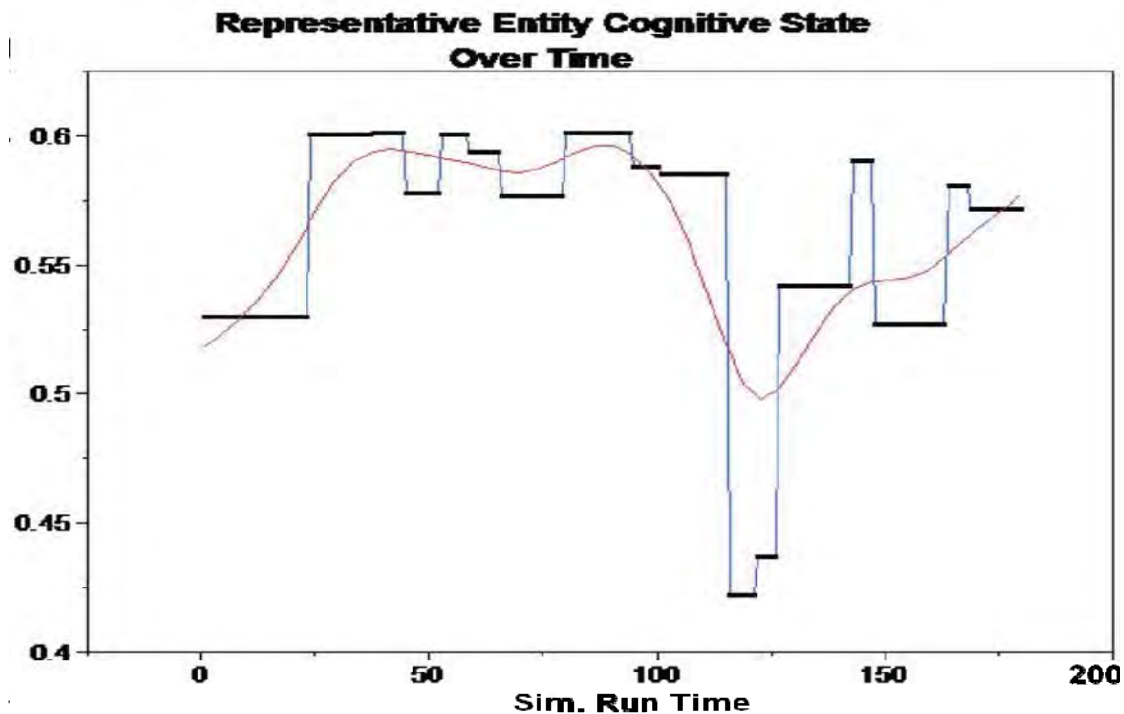


Figure 87. Discrete state level of an entity cognitive state over time (Alt, Lieberman & Alrowaei, 2010)

The information preservation in DES means that deeper analysis is available by using the dichotomy of a centralized event list in coordination with endogenously managed agent states. Information preservation lets us look specifically at the effects of actions through time. We can track and analyze the series of events, actions, and agent communications that lead to a single agent's change of state, or changes in the simulation that affect multiple agents. Traceability of single or group quantities permits

useful time series analytic techniques that can help simulation users and analysts gain insight in to how the BVIs of individuals and groups of various sizes change over time.

Figure 87 shows a sample of output from CG model. The use of the event list facilitates a traceable view of cause and effect within the simulated environment. DES requires that the actions within the model be decomposed and sequenced. Notice how each change in agent state is reported along with the simulation time, reason for the change, and origin of the change (i.e., observation or communication). Traceability allows for the capability to examine, specifically, what events in the simulation lead to observed outcomes. We can investigate what range of impacts each event has on the agent population. The real power of DES in HBR rests in the ability to view data as both emergent (e.g., changes at the agent population level), and deterministic (e.g., at the individual agent level), and this benefit is derived through information preservation and data output tactics.

By comparison, time step models generate multiple simultaneous actions within the simulation environment. In most cases all entities within the environment represent all planned behaviors and the model adjudicates the outcome of those behaviors for each step through time. The ability to trace cause and effect in this type of model is often lost. In PSOM, for instance, we only receive information about the beginning and end states (in the standard output configuration). Furthermore, we can only investigate effects at the aggregate level in the form of changes to demographic grid cells, without the ability to analyze individual or arbitrary group agent behavior more closely.

In CG model, agents communicate with each other about events they experience in person (firsthand knowledge), and events that they know about through previous communications with other agents (secondhand knowledge). Both firsthand and secondhand knowledge change an agent's BVI, and hence the homophily network. While the homophily network represents opportunities for contact, the actual communications that take place between agents generate the communication networks in the simulation. Communication networks are simulation output that can represent the actual set of communication instances between agents that occurs 1) leading up to an event, 2)

following an event, or 3) over a user-defined period of time. This is almost infeasible to be obtained with DTS models unless the time-step size is made small enough to capture single events at each time interval.

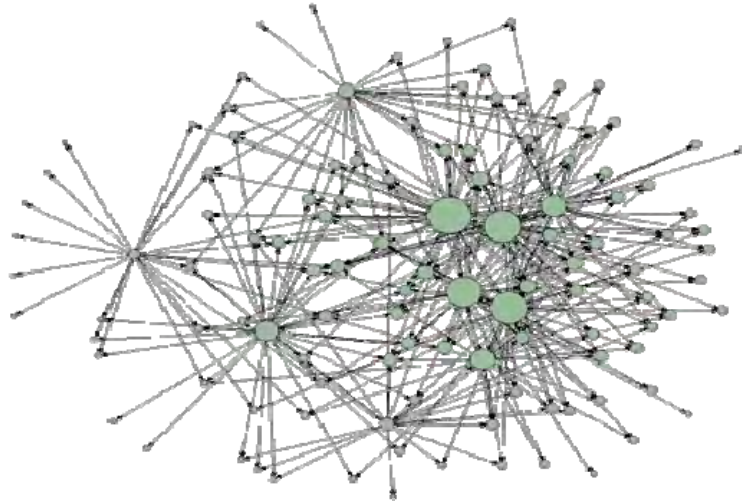


Figure 88. A CG model agent network of communication following an event (from Alt, Lieberman, & Alrowaei, 2010)

Multiple time steps are in effect when representing human behavior. Stimuli from the environment in the form of preceptors generate both immediate responses and responses might not generate observable behavior until some future point in time. DES preserves the authenticity of both cases through the scheduling of events on the event list. However, in DTS models any response delay would depend on the size of the time step and in most cases cannot naturally be represented.

Representing events that occur on competing timescales within models and simulation can be challenging in DES particularly in the domain of human behavior representation. Communications within the CG model are generated in response to the receipt of new information, as opposed to belief revision, which occurs over a longer period of time. When an entity receives information and determines it will communicate that information to other entities within its social network it schedules those events on the event list at some point in the future, usually the next period of time. Belief revision occurs on a much longer time scale. Based on thresholds or rare events impacting entities within the model, belief revision might be scheduled at some distant point in the future.

DES accommodates either case. We feel that the literature on time step models does not adequately address the representation of these phenomena.

The ability to document input into human behavior representations through the use of event graphs and the ability to trace state changes back to events on the event list to determine causality serve as powerful tools for verification, validation and accreditation (VVA). The validation of theoretical underpinnings and conceptual models for use in this domain is an open question. DES, however, provides this transparency and requires clear documentation of the underlying processes and uses of the data.

4. TAM Effects on Agent Behavior in Continuous Systems

The military is also extending these HBR simulations by involving human entities with some cognitive abilities in order to capture more of the human nature behaviors to reach a higher level of fidelity and go beyond the current behavioral models (Zacharias, MacMillan and Van Hemel 2008). Recent studies have been focusing on adding the psychological aspects to the sociological aspects to provide theoretical and a methodological framework to build better HBR simulations (Elkosantini and Gien 2007). The HBR psychological aspects can include factors such as stress, fatigue, malaise, motivation and emotion. These factors would not necessarily improve the model prediction, but it would at least improve the realism, model quality and make the simulation more engaging. Such factors are typically modeled as continuous systems using System Dynamic (SD) or Agent-Based (AB) simulation models (Seck, Giambiasi et al. 2005). The difficulties with these simulation approaches is that for policy or decision makers to comprehend, they must have complete faith in black box.

For the purpose of our study, we will qualitatively discuss the challenges that DTS models encounter in simulating such continuous systems and an alternative DES approach. Specifically, we focus on the main difference between these modes in how they deal with changes over time.

Typically, continuous system simulations (CSS) require that each operation be performed at every “tick” of the system clock, that is the DTS approach (M&SCO, 2010).

Usually, these types of simulation involve differential equations (DEs) to describe the system by presenting the relationships for the rate of change of the system's state variables with time (Ossimitz & Mrotzek, 2008). Simple DEs or difference equations can be solved analytically to give the values of the state variables for all values of time. However, due to the complexity of CSS models, analytical solutions are infeasible and numerical techniques are used to integrate the DEs. Mathematically, there are several numerical integration algorithms to solve DEs that range from simple algorithms such as explicit and implicit Euler and Midpoint, to a more complex algorithms like Runge-Kutta (RK), and Adams. There are many superior algorithms to solving DEs in the mathematical world (Hairer & Wanner, 2010). However, with the appearance of modern computers numerical integration of DEs have been only implementing the simple techniques in order to reduce model complexity and increase simulation efficiency. This is believed to be the origin of the emergence of DTS approach in the M&S field.

The majority of CSS models use numerical integrations of DEs to represent the dynamic behavior of the system over time. Modelers have been implementing these algorithms explicitly in various software simulation packages such as game simulation engines (e.g., Delta 3D⁹) and Simulink¹⁰, or implicitly in simulation methods such as System Dynamic and Agent-based. Recent studies have been heavily implementing SD and AB methodologies in various simulation fields including military simulations that involve human entities. More recently, the DoD has also taken interest in system dynamic modeling representation, most particularly to analyze the dependencies, interdependencies, strength and weaknesses of a country's PMESII system (Zacharias, MacMillan and Van Hemel 2008). For example, a system dynamic model called SROM was used to model the Iraqi population's influences and responses to services and critical infrastructure operations (Zacharias, MacMillan, & Van Hemel, 2008).

In SD, stocks/blocks represent the state variables of the system (e.g. people, knowledge) and flows between these blocks represent the rate at which these state

⁹ Delta 3D is an open source gaming and simulation engine developed by MOVES at NPS, <http://www.delta3d.org>, last accessed on July 13 2011.

variables change over time. Just like in any DTS model, time in SD models is an independent variable and the main organizing principle. In SD models, time is discrete and advanced by a specified fixed increment used in integrating all the DEs used in representing the system (Ossimitz & Mrotzek, 2008). A commonly used SD software package (Vensim¹¹) shown in the Figure 89 provides an illustration of how differential equations are integrated and time step is handled in a typical SD model. There are few choices of time-step size and only two integrator type selections (Euler and RK4). As shown in an upcoming example, these limited selections can have a major impact on the accuracy of the simulation results that are hardly visible to the modeler.

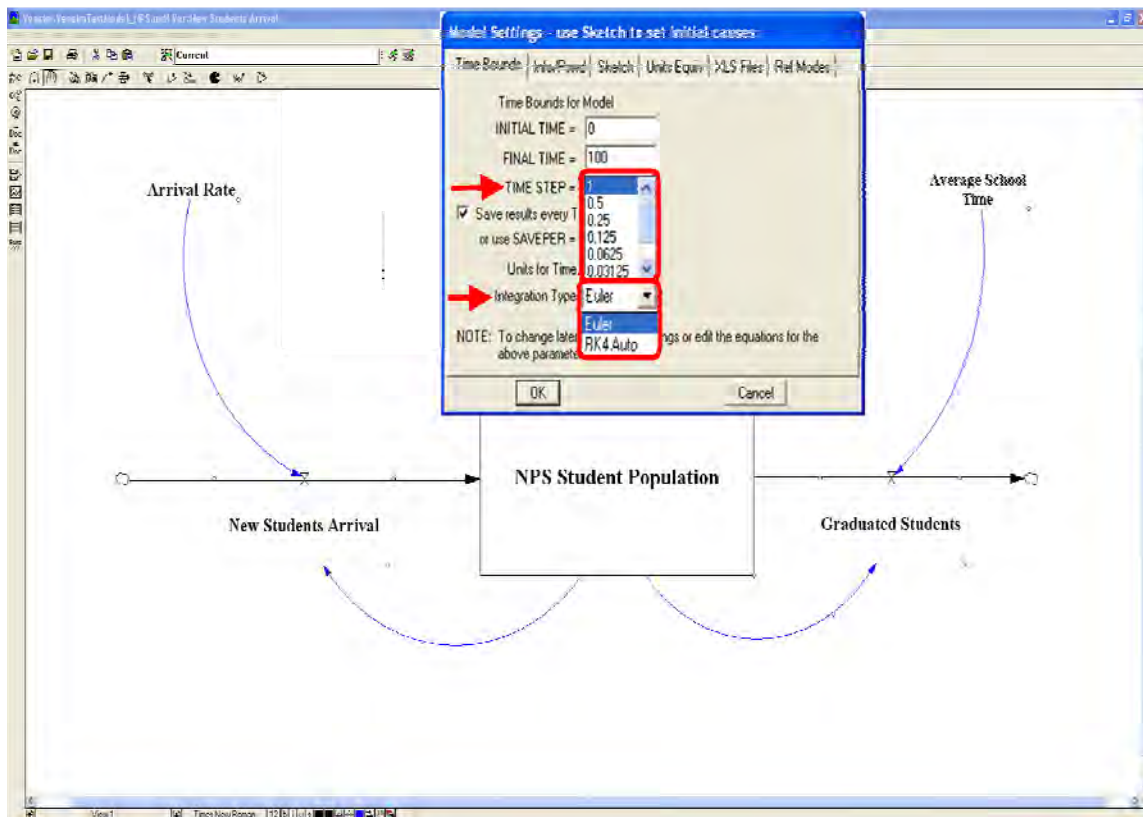


Figure 89. A display of a simple SD model showing the time-step size and integration solvers used in Vensim®

¹⁰ Developed by MathWork to simulate dynamic systems, <http://www.mathworks.com>, last accessed on June 27 2011.

¹¹ Developed by Ventana System Inc. <http://www.vensim.com>, last accessed on June 29 2011.

In previous chapters, we discussed few problems related to using the DTS approach in constructive type simulation. Similarly, the effect of TAM on the accuracy of CCS simulation results surely exists. In fact numerous studies in the mathematical literature point out the disadvantages of large time-step sizes and their impact on decreasing the accuracy of the solution approximation. In the M&S field the problem remain the same. Modelers still have no methodology to select an optimal time-step size that fits their models and have no way to validate the accuracy of the simulation results. Therefore, we investigate the TAM effect on CSS models' results to provide modelers with the benefits and limitations of DTS and DES approaches in such models.

In spite of the several differences between the mentioned DE solver algorithms used in dealing with CSS, all share a property; they are based on time discretization. That is, the simulation time in a CSS model advances by a fixed and equal increment "step size (Δt)" which is initialized at the start of the simulation. Changes in any state variable value are only permitted to occur at the end of each time interval and the value is normally calculated by solving the differences equations.

Zeigler and Lee (1998) introduced a completely different approach for solving DEs in CSS where the time discretization is replaced by state variable quantization. As shown in the Figure 90, this resulted in models that are not discrete time (DTS) but discrete event (DES). In another word, the discrete time approximations of continuous systems use discrete time and continuous states, while the discrete event approximations of continuous systems use discrete states and continuous time (Zeigler, Praehofer, & Kim, 2000; Nutaro, 2005; Zaft, & Zeigler, 2002). The approach suggests we calculate how long it takes for a given amount to flow past a particular point instead of using the rate equations to calculate how much flows past that point in the system in a given time. That is, the question to be asked is at which time in the future the system will be in a certain state?, instead of which state is the system going to be at a given future time?

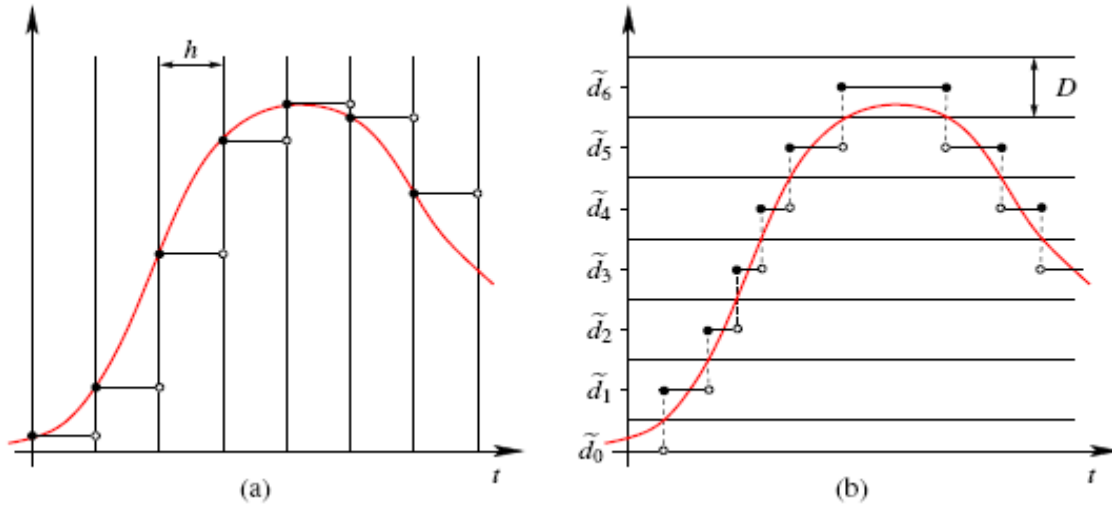


Figure 90. a) Time discretization in DTS and b) State quantization in DES, (from Bolduc, 2002)

The above idea was introduced in Zeigler's DEVS model and proved to produce strong stability, convergence and error bound properties in a very efficient fashion (Kofman, 2006). The quantization approach to continuous systems still puts boundaries on state values (i.e., no state value exists between identified states) but tend to produce less errors than the DTS approach as shown later in this section. Similar to the uniform time-step size in the DTS approach, this approach applies uniform quantum step size (D) between each quantum (state value).

DEVS formulation (atomic and coupled) for solving ordinary DEs in discrete event system has been only described in a mathematical form as expressed in appendix A (Kofman, 2006). In order to reduce the simulation complexity, we introduce a new representation of DEVS formulation in terms of the Event Graph (EG) theory explained in previous chapters. To illustrate this new procedure, we look at two classical approaches to numerically solving differential equations, the Euler and the 4th order Runge-Kutta (RK4). Although this is simple, and perhaps more naïve approaches, they are appropriate to study in this context because they are the ones that are most analogous to a standard time-step simulation. In one dimension a simple linear differential equation has the form.

$$\frac{dy}{dt} = f(y, t) \quad (36)$$

The Euler method turns the differential equation into a difference equation by discretizing time into intervals of size Δt , thus turning Equation (36) into Equation (37) below (Golub & Ortega, 1992).

$$y(t + \Delta t) = y(t) + \Delta t \cdot f(y(t), t) \quad (37)$$

Equation (37) can be thought of as a first-order Taylor series approximation to Equation (36). Note how Equation (37) is analogous to a typical time-step simulation model. The new state value is the old state value plus an increment. The Euler method starts with an initial value $y(0)$ and then uses Equation (37) to update the value of the state y at successive time increments. A commonly used member of the Runge-Kutta methods is the fourth-order Runge-Kutta (RK4) described in the following equations:

$$\left. \begin{aligned} y(t + \Delta t) &= y(t) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ \text{where,} \\ k_1 &= \Delta t \cdot f(y, t) \\ k_2 &= \Delta t \cdot f\left(y + \frac{1}{2}k_1, t + \frac{1}{2}\Delta t\right) \\ k_3 &= \Delta t \cdot f\left(y + \frac{1}{2}k_2, t + \frac{1}{2}\Delta t\right) \\ k_4 &= \Delta t \cdot f(y + k_3, t + \Delta t) \end{aligned} \right\} \quad (38)$$

It is well-known that difficulties arise with the Euler and Runge-Kutta methods for stiff equations even with a very simple linear equation. Stiff equations are DEs that for which certain numerical methods for solving the equations are numerically unstable unless the step size is made extremely small (Eberly, 2008). Typically, these equations are used in control theory, physical behaviors, chemical kinetics and electrical circuit theory. These equations if not handled with care can cause complete non-physical behaviors of the simulated system (i.e., causing the system to “explode” for no reasons). For example, taking $f(y, t) = -100y + 100$ and $y(0) = 2$, the resulting numerical solution is wildly different for different sizes of time step Δt (See Figure 91).

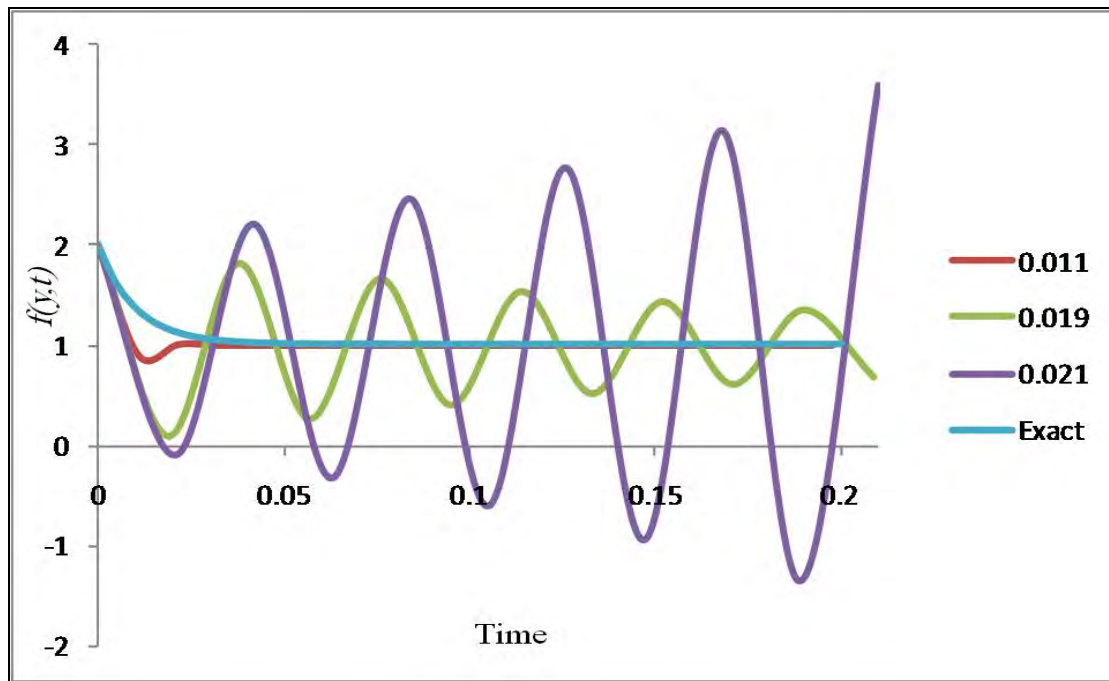


Figure 91. Euler solutions to Stiff equation [best viewed in color], (from Buss & Alrowaei, 2010)

For values of Δt less than 0.02, the steady-state value of 1.0 is approached, although for $\Delta t = 0.019$ it oscillates about the value, gradually converging. However, for $\Delta t = 0.021$ there is a qualitative difference in the “solution” and it winds up oscillating ever more wildly, ultimately diverging instead of converging.

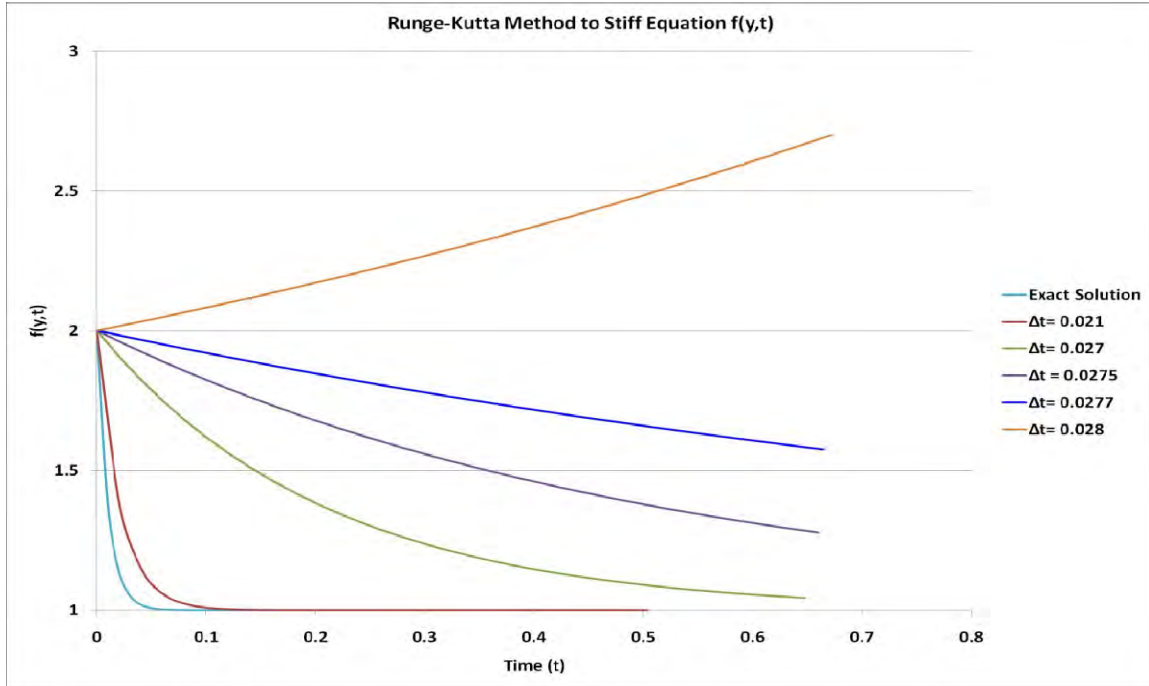


Figure 92. Runge-Kutta Solutions to Stiff Equation [best viewed in color]

We observe that whenever the time-step size is a little too large, the numerical solutions either go too far beyond the equilibrium or violent oscillations occur.

From a certain perspective, such a disparity between results based only on the size of the time step should be considered a red warning flag for all time-step models. If there can be such an effect for this simple, one-state linear model, what unknown effects might the size of time steps be having on more complicate models? How confident can a modeler be that the results obtained from a simulation (or agent-based) model are qualities inherent in the model itself versus anomalies introduced by the discretization of time and a “bad” size of time step chosen?

For a simple differential equation model such as discussed in the previous section, the simplest approach is to quantize the state space to be multiples of some fixed quantity called “quantum size” (D). Where $D = |y(t+\Delta t) - y(t)|$ is a fixed quantity and the approximated time for $y(t)$ to change to by D with time $t_a(y,t) = D/f(y,t)$. Depending on the sign of $f(y,t)$, the next level of state will be one higher or one lower than the current state. The time for that transition to occur can be approximated by solving Equation (37)

for Δt . Although this approach was originally formulated in terms of DEVS (Zeigler, Praehofer, and Kim 2000), our simplified version can be compactly expressed as an Event Graph (Schruben, 1983), as indicated in Figure 93.

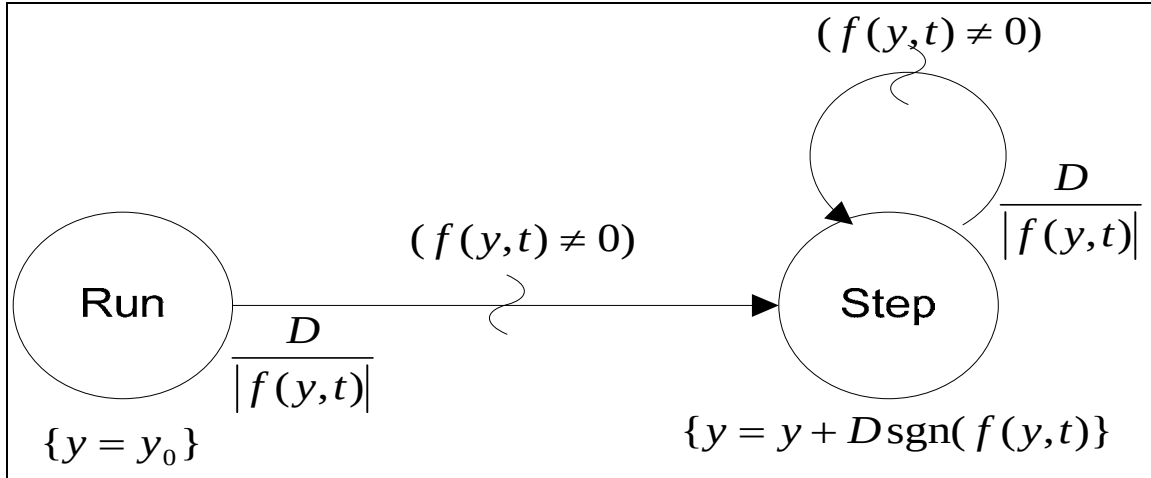


Figure 93. Event Graph Quantized Model for 1-dimensional DE (from Buss & Alrowaei, 2010)

The parameters for the model in Figure 93 are the initial value, y_0 and the state space quantum, D ; the state variable is y and the sgn function that takes the sign of $f(y, t)$, [0 if $f(y, t) = 0$, -1 if $f(y, t) < 0$, and +1 if $f(y, t) > 0$].

When this approach was applied to the stiff linear equation of the previous section, the results were much more satisfactory than the Euler method (See Figure 94). For one thing, there was little deviation in behavior for vastly different sized quanta D . All solutions converged quickly to the neighborhood of the steady-state value of 1.0. Those for which the steady-state value was in fact a multiple wound up staying at the correct value, whereas those which were not a multiple of the quantum ended up oscillating around the steady-state value. Even then, the state values were never more than D from the “true” value.

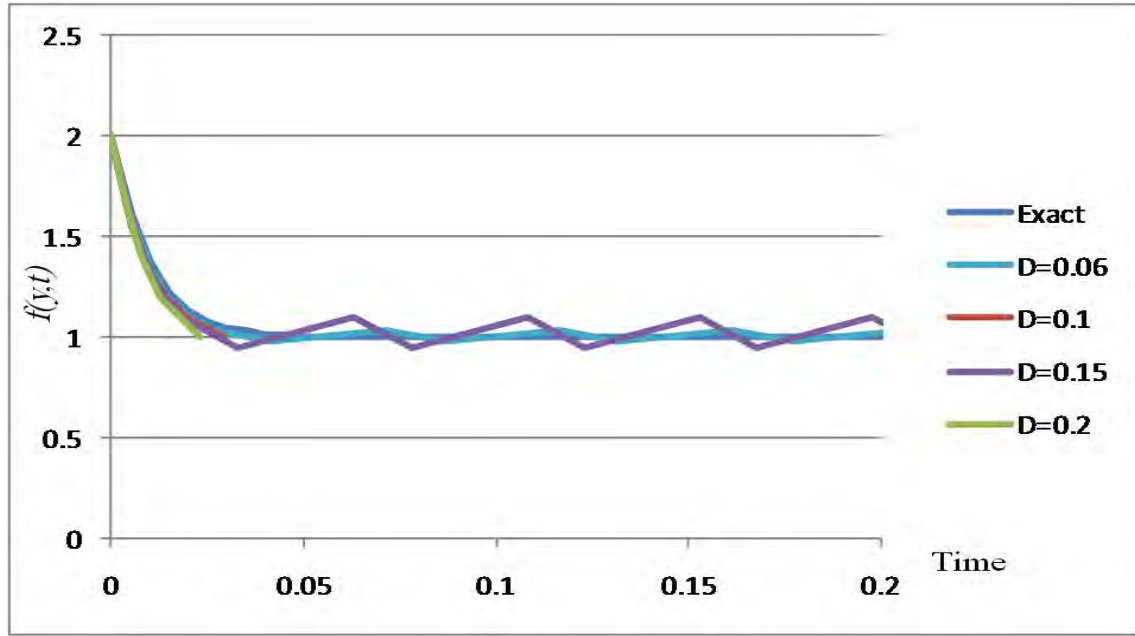


Figure 94. Quantized solution for various state quantizations [best viewed in color], (from Buss & Alrowaei, 2010)

The relative ease with which the quantization approach can be implemented along with its superior robustness with respect to the Euler method suggests strongly that a DES approach may be generally preferable to a time-step approach in many more areas than previously thought.

Figure 95 shows the relative speedup between the DTS and DES models assuming that the discrete time scheme would require a time step equal to the time advance of the discrete event system to resolve the same set of events. That is the DTS time step should equal the smallest of the $t_a(y,t) = D/f(y,t)$. The relative speedup is computed in terms of simulation execution time of both models as $[Execution\ Time_{DTS} / Execution\ Time_{DES}]$ with the use of RK4 for the DTS model.

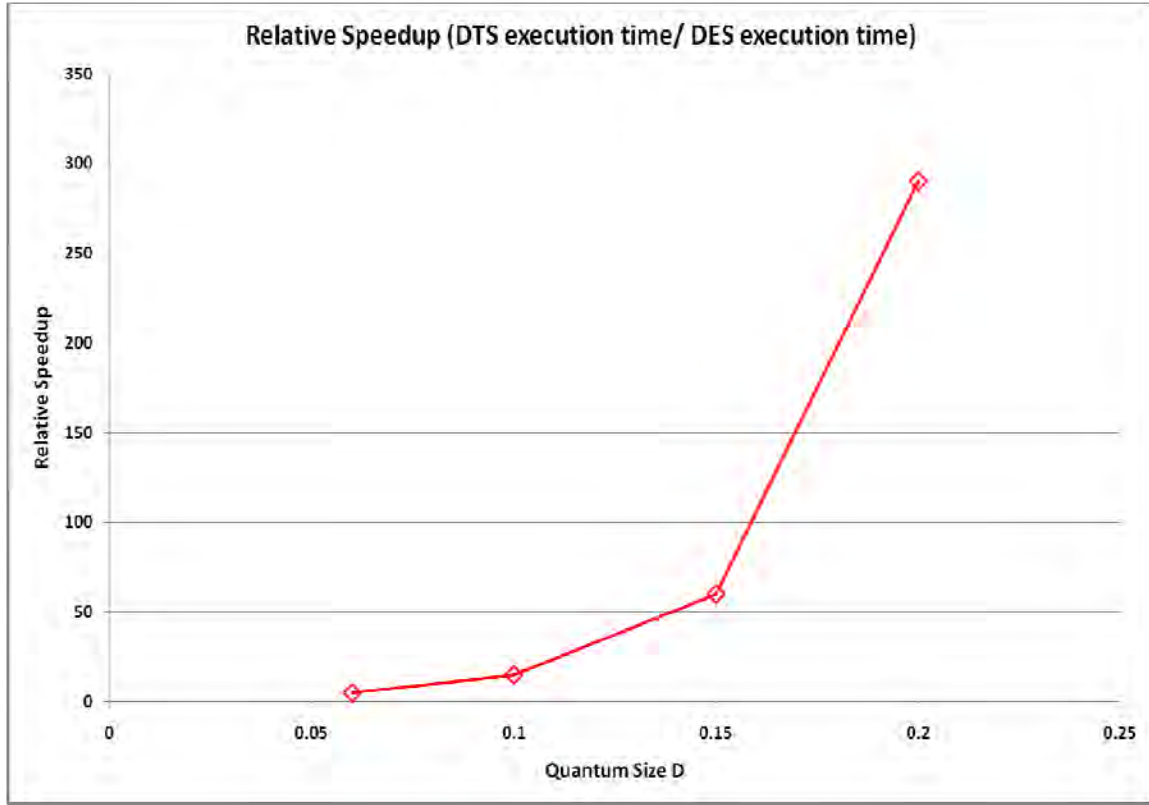


Figure 95. Relative Speedup between DTS and DES models with different quantum sizes

We extend our version of expressing DEVS approach using Event Graph methodology to solve multi-dimensions differential equations. The one-dimension example discussed above shows the quantization approach or DEVS-Euler integrator (Nutaro 2005) in its simplest form. There is one limitation shown in the Figure above and that is if the solution (equilibrium state) lie in between two quantum states (q_i), the simulation results will oscillate but with controllable error size of not greater than D . Error control is certainly desirable and advantageous to many fields related to the M&S. We illustrate this in a simple 2-dimentional DEs in which we want to simulate a time-invariant system such as:

$$\frac{d(x_1)}{dt} = f(x_1, t) \quad (38)$$

$$\frac{d(x_2)}{dt} = f(x_2, t) \quad (39)$$

Modern combat models such as advance Lanchester's equations that involve movement of forces on the battlefield, the nonhomogeneous character of the modern army, the importance of the principle of command and control, and the impact of logistics and intelligence, are new and more comprehensive mathematical models that are based on multi-dimensional partial differential equations (PDEs). The amount of errors produced using the conventional numerical algorithms to solve multi-dimensional DEs can be large enough to raise questions about the validity of the results, particularly at large time-step sizes in which the errors can be enormous. A detailed and more complex combat simulation such as the advanced Lanchester's equations can be modeled only if the spatial grid is made fine enough to account for the local situation on the battlefield. This has as a consequence the increase of the number of DEs to be solved, and thus generates an increase of the computational time. At the same time there are indications that for the same problem, the increase in the number of DEs through the increase of number x_1 -dimension and x_2 -dimension DEs tends to transform the DE system into a stiff one making its integration difficult task and increasing the computational time even more (Rusu, 1988).

In the following example, we demonstrate the ability of providing approximate solutions with minimum errors for the 2-dimensions DEs using the EG methodology.

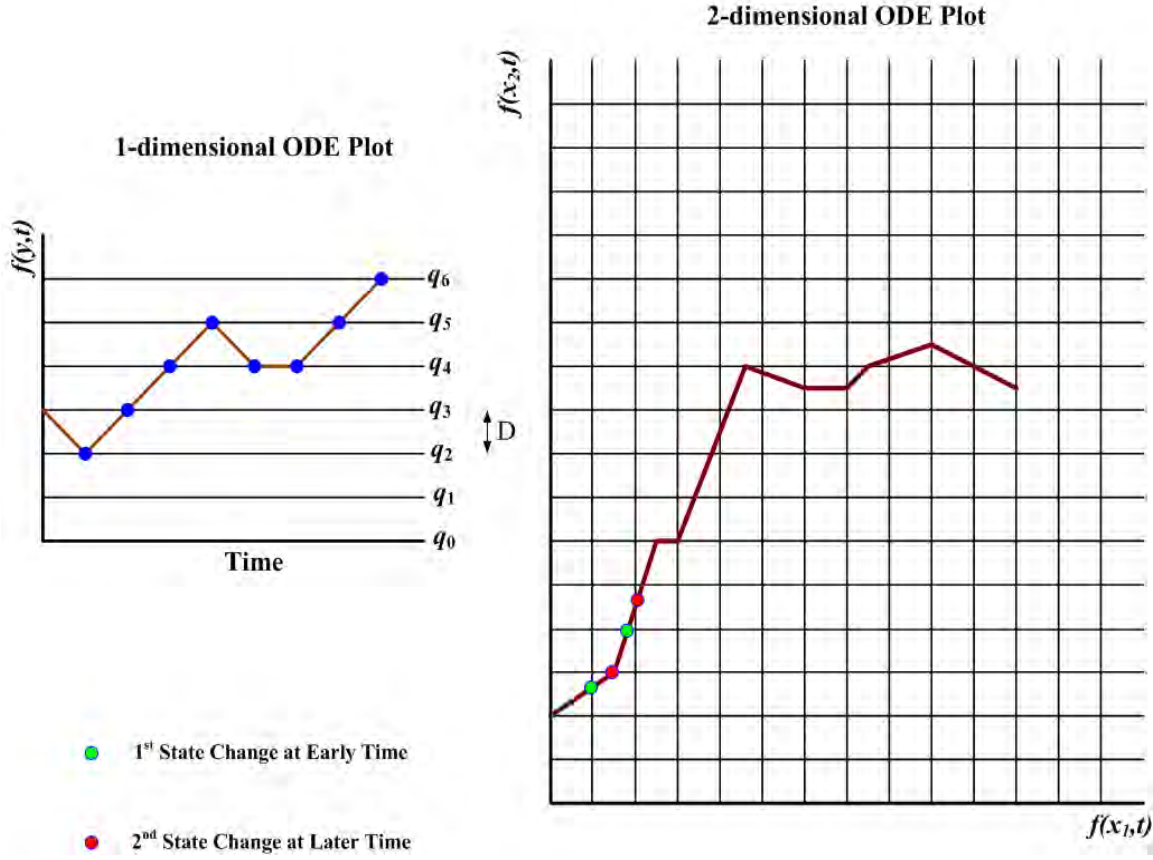


Figure 96. Sample plots of 1-dimension and 2-dimensions grid of DEs using DEVS methodology [best viewed in color]

The same methodology can be followed in the grid points are regularly spaced by a distance equal to the quantum size D . To simulate this system with EG we use similar method to the one introduced by Nutaro (Fishwick, 2007) in which four variables need to be identified for each dimensional state space. These are the followings:

x_i the position of the state variable i on its space axis, where i is dimension index

y_i the last point occupied by the variable x_i

$t_{next(i)}$ the time until x_i reaches its next point on the i th space axis

$t_{last(i)}$ the last time at which the variable last modified

As previously discussed, the quantum size D is chosen by the modeler according $D = |x_1(t+\Delta t) - x_1(t)|$ or using the x_2 dimension. The time required for the variable x_i to reach its next point on the grid is computed from:

$$h_i = \begin{cases} \frac{D - |y_i - x_i|}{|f_i(y_1, y_2)|} & \text{if } f_i(y_1, y_2) \neq 0, i=1,2 \\ \infty & \text{otherwise} \end{cases} \quad (40)$$

$$t_{next(i)} = t_{last(i)} + h_i \quad \forall i \quad (41)$$

The following Figure shows the event graph of the above system that simply can be implemented in any programming language.

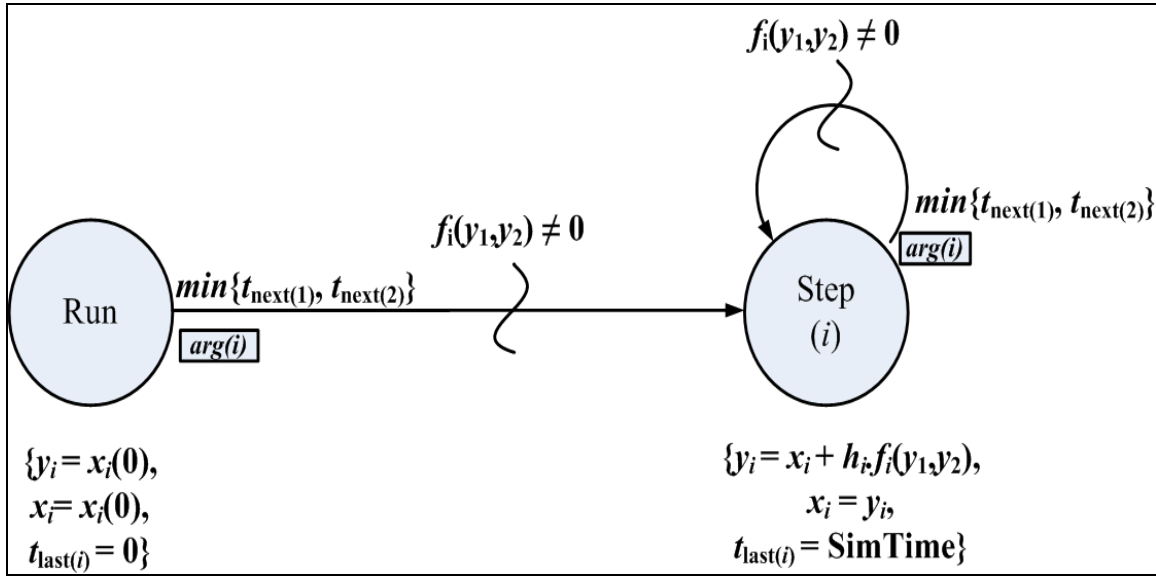


Figure 97. Event Graph of 2-dimensional ordinary differential equations using DEVS algorithm

Barton and Tobias (2000) demonstrated a discrete quantity approach (DQA) to show how continuous systems can be modeled in discrete event. We observed that conceptually the DQA is identical to the state quantization approach introduced earlier by Ziegler. They implemented the DQA by simulating the predator-prey model and showed that the results are matching with the system dynamic results obtained at a small time-step size (0.5) using Euler integration method.

Similar methodology can be followed for m-dimensional state space in which the grid points are regularly spaced by a distance D along the state space axes. We believe that the amount of errors obtained from this approach can be more smaller than with the errors obtained from the one-dimension approach. The authors believe this can enhance the accuracy level as well as increase the simulation efficiency in comparison with the traditional DTS approach.

As mentioned in Chapter II, recent studies have started to compare simulation approaches such as SD with DES across a wide range of simulation fields not related to the military combat (Brailsford & Hilton, 2001; Tako & Robinson, 2008; Lin et al., 2008; Özgün & Barlas, 2009). The goal of these studies, however, was to demonstrate the ability to model the system with different approaches, instead of performing a detailed investigation of the TAM effects on the simulation results.

D. CONCLUSIONS

Well constructed social simulations possess the potential to provide great insight to decision makers seeking to allocate scarce resources in order to maximize the benefit to a given population. However, as we have seen in this chapter, the construction of such a simulation, as well as the usability of the output, is extremely dependant on the chosen TAM and the concepts of time present in the simulation. Time and sequencing of events play a critical role in developing a realistic and traceable representation of human cognition and the resulting observable behavior. Both the time stepped and discrete event methods of implementing simulation possess strengths and weaknesses in regards to the representation of human behaviors and cognition. In Pythagoras, we have seen that changes in the time-step size can introduce anomalies and that these can cause problems in our interpretation of simulation output. In turn, the TAM can undermine our ability to understand human behavioral change over a period of time.

The same is true for the time-stepped PSOM environment. In PSOM, the rate at which security perceptions changes can be misleading with different time-step sizes. In all complex and credible social and behavioral simulation, events that change behaviors can take place at different rates. For example, sometimes 100 or more important events

can occur within the period of one day, while at other times the systematically important events can take place at a much slower rate of, perhaps, 1 or 2 events per week. The rate of important events may differ between periods of the simulation and the effects could be extremely hard (or impossible) to capture in detail utilizing a DTS framework. While there may be no universally accepted validated answer to the complex questions introduced by the irregular warfare environment, intuition and assessments of face validity should help us determine when different time-step sizes may be not appropriate.

With the discrete time TAM approach, questions about belief or behavior change sensitivity, including the various political, military, economic, social, infrastructure and information (PMESII) aspects can be difficult to answer. Conversely, in a DES environment where the TAM does not introduce anomalous conditions, one can look to the main issues and controllable factors that influence the simulation outcomes. Social simulation with DES approach provides improved flexibility and the ability to examine potential futures that impact individuals and the society. The main advantage of DES over time-stepped model implementations in HBR from an analytic point of view is very similar to the benefits outlined for combat simulations in the previous chapter. It allows the modelers and M&S users to understand causal relations as represented in the model through the event list. Our analysis shows that DES models are more analytically sound and provide an emphasis on detail complexity within the system where the model resolution can be on individual entities, attributes, events and individual behaviors. DTS models, however, tend to focus on the system as a whole and on the dynamic complexity of how the system progress over time with resolution on homogenized entities and overall emergent behaviors.

There is an increasing need for accurate, reliable and efficient IW simulation models that are capable of adequately present the human behavior. This tends to increase the HBR model complexity particularly in situations where high resolution and fidelity are desirable factors in which models include continuous behaviors such as stress, emotions and fatigue. The DES approach with DEVS's quantization theory provides better stable solutions than the DTS models and showed its promising potentials to model continuous systems with better error control.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION, IMPLICATIONS AND FUTURE WORK

“We can only see a short distance ahead, but we can see plenty there that needs to be done.”

(Turing, 1950, p. 433)

A. CONCLUSION

This work consisted of empirical studies that compared the discrete time simulation (DTS) and discrete event simulation (DES) time-advance mechanisms (TAMs) based on experimental scenarios involving currently used military simulation tools including MANA, Pythagoras, PSOM, DAFS, Simkit, CG, and Vensim. We provided quantitative and qualitative descriptions of the two TAMs and their outcomes. With these analyses, we studied the results obtained from each approach as applied to the same scenarios in various military applications involving queueing systems, combat models and human behavior representations. The comparisons presented in this study therefore contribute towards understanding issues regarding the choice between the two simulation approaches and may ultimately help in the selection of the appropriate simulation approach to model a particular problem situation.

We found that DTS models involve greater time at the conceptual stages for queueing systems relative to DES models, but less time in model coding. This could be because event graphs simplify the design of how the system is illustrated in DES models. However, transitioning the model to code is more complicated and requires state transitions specifically linked to certain events as well as the interaction with the event list. Both DES and DTS models presented problems with the occurrence of simultaneous events. However, introducing event priorities allowed the DES models to address this problem, unlike in DTS models where such an approach is generally infeasible to set up. Identifying events and the associated state transitions is a challenging task to DES models, where DTS models are often simple and based on well understood mathematical expressions that describe the system.

We observed that the DTS approach is capable of representing the general behavior of queueing systems but came up short in providing accurate and detailed results. Decreasing the time-step size resulted in closer approximations to analytical solutions but caused the simulation execution time to reach extreme levels in comparison with the equivalent DES models. Naturally, there is a tradeoff between fidelity and computational error and computational time in DTS models. Determining the appropriate compromise depends on the scenario and goals of the modeler. In general, decreasing the time step size will lead to convergence with any available analytical solution, but for some mathematical problems decreasing the time step size beyond a critical value will cause dramatic or chaotic behavior instead of convergence. Complex queueing systems such as those with high traffic intensity and balking tend to exhaust the DTS model and had low accuracy even at small time-step sizes. We observe that TAM effects on queueing systems can lead to costly decisions when it is put in practice. These results make it clear that DES is the preferable approach to modeling queueing systems.

Our study demonstrated that the choice of TAM can have a very large impact on the results of combat simulations. We have shown a series of eight combat scenarios modeled equivalently with DTS and DES approaches to examine combat modeling elements such as movement, sensing and detections, engagements and interactions. Our results show that the DTS and DES model results do not converge to the same answer for the majority of cases. This is due to the influence of time-step sizes rather than any inherent property of the model or the simulation tool. In many cases, reducing the time-step size resulted in small total error, but the complexity of combat models will have limitations to benefit from this tactic, particularly as there is no well defined methodology for choosing the appropriate time-step size in complex simulations.

Critical problems such as waypoint inaccuracies, missed detections and engagements, and detection inaccuracies can be invisible to the modelers. These problems and a number of emergent anomalies in the behavior of combat models such as skipping, state transition delays and ordering can be caused by time-step size in DTS models. Conversely, these issues generally do not arise in the corresponding discrete

event simulations. A model selecting DTS as the TAM mechanism may therefore cause potential issues and the potential for misleading recommendations in combat models.

We also observed that TAM had a huge impact on simulation models that are currently used in irregular warfare software packages such as Pythagoras and PSOM. With the DTS approach, questions about belief or behavior change sensitivity, including the various political, military, economic, social, infrastructure and information (PMESII) aspects can be difficult to answer. Conversely, in a DES environment such as CG where the TAM does not introduce anomalous conditions, the modeler can look to the main issues and controllable factors that influence the simulation outcomes. Social simulation with the DES approach provides the highest flexibility and the ability to examine potential futures that impact individuals and the society. Complex human behaviors that are expressed in a similar fashion as continuous systems were shown to sometimes behave like stiff equations. Discrete time simulation tools such Vensim use traditional integration methods and proved to sometimes provide inaccurate answers even under small time-step sizes. The implementation of DEVS quantization method demonstrated the capability of DES models to produce stable and reliable results.

Simulation models have grown considerably more complex in recent years, reflecting the demand to use M&S approaches to gain a better understanding of the challenging problems facing government, industry and society that continue to grow in size, complexity and uncertainty. Along with the steadily growing demand for the use of these tools, modelers are faced with a relentless push to design simulation with higher fidelity and resolution in order to better represent the realities of the complex systems. In today's world, the impact of time advance mechanisms (TAM) becomes even more important. Many systems directly concerning the military and government M&S communities have proven to be very sensitive to time advance mechanism, and to changes in time-step sizes. In many ways, these systems behave similarly to the unstable systems, including systems represented by "stiff equations." This study investigated the impact of TAM in M&S and perform a profound comparison analysis between discrete

time and discrete event simulation approaches by exploring various military related scenarios in applications including queueing systems, combat models, and human representation

From the most complex to the simplest generalized queueing systems, simulations are influenced by the chosen representation of time. Combat simulations, in particular, are sensitive to changes in the TAM and time-step size, affecting such central simulation elements as movement, sensing, communications, and detection. Human behavior representations are similarly sensitive, and have become a much larger concern to the M&S communities, in part, given their growing operational concern and complexity resulting from very large numbers of agents.

The time domain presents many challenges in the representation of human behavior within models and simulation. The choice of DES or DTS implementations depends on several factors such as the use case and available data, and challenges exist for either approach. In this regard, DES's main advantage over DTS model implementations from an analytic point of view is in the ability to understand causal relations as represented in the model through the power of the event list. This traceability facilitates the evaluation of model results based on the appropriate social or cognitive science theoretical underpinnings. The use of event graphs as a means of facilitating model development also lowers the bar required to gain insight from subject matter experts in the domain under study. Given the complex nature of societies and the complex adaptive systems they form, the need to understand the implementation of processes through event graphs and to be able to trace model results via the event list makes the use of DES attractive.

The two major TAMs, discrete time simulation and discrete event simulation, are extremely different in terms of the concepts that underlie the foundations of their action. Likewise, as we have shown here, the TAMs can produce vastly different results and interpretations for models that are identical in every aspect except for the time factors. Moreover, DES models are analytic, in the literal sense that they break a system down into its constituent parts. DTS models are also capable of doing that but with some difficulties than DES models. Notably, the DTS approach may be beneficial compared

with the DES approach in applications involving large amounts of visualization and human-in-the-loop training simulations that involve real-time interactions. Also, the DES future event list can often become overloaded with events in cases where there is compounding intensity of interactions between system components. In these cases, DTS and DES tend toward the same performance and DTS may be favored in these instances for its design simplicity.

The complexity as well as the usefulness of DES models rests in their representation of detail. The discrete time simulation approach takes a comparatively more general view in most cases, and where complexity does exist, it can often be found solely in the dynamic interactions between the elements of the system. Currently, there are certain elements of high resolution and fidelity in which the DES approach has not been implemented (can be considered as shortfalls) such as terrain representations, 3 dimensional obstacles (e.g., buildings and vehicles), and complex environment line-of-sight (LOS) representations. These elements are implemented in many simulation tools that use the DTS approach, but can be a challenging issue with the DES approach. A small number of simulation environments use the DES approach because it requires particular skills and knowledge to understand the fundamentals of the approach.

Although the qualitative error in discrete time simulations generally decreases as the time-step size decreases, it is often infeasible (and sometimes impossible) to foresee the precise consequences of such errors on the overall results of the simulation. Furthermore, since there is no accepted methodology for quantitatively determining the best or smallest acceptable time-step size, DTS approaches should be used with the utmost caution in constructive simulations. Since DTS results converge to DES results in most cases, we have found that DES can often provide a benchmark for M&S evaluation efforts. It is hoped that this work will benefit DES and DTS practitioners by promoting communication with a common language and set of ideas that will help facilitate the exploration and operational evaluation of M&S capabilities.

B. RESEARCH IMPLICATIONS AND FUTURE WORK

The contribution of this work lies in that it provides an empirical study in the comparison of DES and DTS model time advancement process and its impact on simulation results. Our study suggests that modelers should consider different approaches at the conceptual level if possible while building the simulation. This allows developers to choose the most appropriate approach at the early stages, and to focus development of the M&S on systems that more accurately mimic the behavior of the real system instead of proceeding with a familiar approach that may prove to be problematic later on. For instance, by the time a model gets to the verification, validation and accreditation (VV&A) process, it is generally too late to implement meaningful changes to the time advance approach.

All M&S developers need to truly understand the implications of the time advance mechanisms that they chose to use. DTS modelers must understand the importance of both the mechanism and time-step size determination, as each will have a large impact on their model construction efforts and the results eventually obtained. In particular, Δt can be used as a factor for sensitivity analyses during the early stages of development. Performing TAM sensitivity analyses with different time-step sizes can be an extremely beneficial and important part of the development process, but will be infeasible in some situations due to time restrictions. Nonetheless, initial analysis should be undertaken to determine the implications of the chosen time-step size prior to data collection.

Another extension of this work is to provide a fully developed set of benchmark tests for movements, sensing, detections, and so on, to be applied to many combat DTS models in order to ensure greater accuracy of the results. Our work shows that DES approach can provide a better alternative to the DTS approach and is recommended to be used if possible. However, if such replacement is a difficult task we recommend the modelers to use DES models as a benchmarks to validate DTS results.

The implementation of event graphs (EG) and LEGO methodology to construct DES models can be very helpful for development and debugging purposes. The

methodology can be time consuming at the initial phases but time saving at later stages of simulation. Improved model visualization helps developers understand the behavior of the system. DES modelers should make use of the EG and LEGO methodology in their implementation of the DES approach whenever possible. Both developers and decision makers need to know that the M&S tools they create and use are trustworthy. This trustworthiness must include all aspects of the representation of time in the M&S, including the chosen TAM. In the case of DTS models, they also must know that the chosen time-step sizes are trustworthy. As we have seen, this can be extremely hard to determine analytically.

We took small steps in exploring simple cases in combat models in which the effect of TAM was shown to influence the results in quantitative and qualitative ways. More complex cases also ought to be investigated to further explore and characterize the impact of time-step sizes and assess whether there are reasonable alternative DES models. We suggest that decision makers need to be informed of the results accuracy limitations as well as the selected time-step size at which recommendations are made.

Additionally, model assumptions need to be clear up front and the effects of TAMs must be reviewed and understood. Much work remains to be done in this area to fully achieve these goals. It is of central importance to clearly identify methodologies for determining appropriate time-step sizes for DTS approaches. This may be the first step toward creating trustworthy discrete time simulations. If we are to develop general theories of time advance in M&S, we must also take additional steps toward exploring alternative TAMs and comparing their effects both quantitatively and qualitatively. For instance, we should attempt to extend the Event Graph methodology to express and describe multidimensional continuous systems in DES, and compare these systems using a System Dynamic (SD) approach. This helps to develop the groundwork for more general theories of time advance, and will inevitably contribute to the body of scientific knowledge across many fields of study, not just M&S.

For Human Behavior Representations (HBR), DTS models with appropriate time-step sizes can be used to gain general understanding of the overall behavior of the society in which the society is presented at the aggregate level. The DTS approach may be more

suitable when the input and output data is in the form of events collected periodically (i.e., weekly, monthly and so on). However, current demands require HBR models to provide details from individuals to the society level as well as single event casual and effects investigation. Thus, for a deeper understanding of human systems we recommend the use of DES since it has proven to be superior in the majority of cases. It must be noted, however, that both DES and DTS approaches have advantages and challenges in the representation of human behavior. Future work will explore alternative models of human behavior and the implications of varying time scales on cognitive process representation.

Although combined TAM approaches are just beginning to emerge as trustworthy M&S, it will likely be important to continue to expand and develop these ideas, if only for scientific explorations of the effects of time advance theory. We should continue to investigate ways of combining existing DTS and DES models under a universal simulation clock, preferably a DES clock, that will increase the overall model's fidelity while not simultaneously increasing the simulation's computational complexity. The reasons are some DTS models are well built for the specific applications used and need to be modified to encounter new challenges. One method to combining TAMs would be synthesizing the future event list control mechanisms with DTS event referees.

Adaptive time step, conversely, is hard to implement in complex military simulations, but can arguably be successfully used to reduce the occurrences of some of the anomalies that are described in this work. The mathematical field is loaded with theoretical studies that provide control to the time-step size for a set of equations that represents the system. For example, in the control theory, feedback loops are used to control the behavior of the system and can play a role in adjusting the time-step size. With these systems, modelers and SME have expectations from the results and therefore can measure the accuracy and set threshold levels to control the time-step size variability across the simulation run. However, in complex combat models, the simulation results cannot be verified, hence it is difficult to compute the model accuracy and set thresholds to implement the adaptive time-step sizes. For this reason, finding the trade-off between the size of the time-step, the computational complexity and the simulation accuracy

becomes a difficult task to address. On the DES side, the current DEVS formulations use fixed increments for the state quantum sizes and it is shown that this introduces fewer errors in DES than in comparable DTS models. DEVS formulation can implement variable grid sizes. However, the future work will undoubtedly explore the dynamically variable (adaptive) quantum size approaches that could farther reduce the presence of these errors. The goal is to have small time advance (i.e. quantum sizes D) for areas where there are frequent state changes/events and relatively large time advance where there are infrequent state changes/events. Since most of agent-based models (ABM) are implemented in DTS approach (Fishwick, 2007), more work is required to introduce DES approach to the field and investigate its capability to encounter DTS limitations.

The empirical study of the effects of time advance mechanisms will likely form the future foundations for how the military and other M&S communities approach simulation construction and interpretation across a very wide variety of domains. While comparison tests among prevalent simulation architectures are becoming more frequent, the methodology for exploring and categorizing differences in useful ways is still in the early stages of development. The largest implication of these finding is perhaps a realization of the fact that modelers will need to regularly take steps back to think about the conceptual origins of their simulation approaches and time advance mechanisms to fruitfully address the issues of the future. We know that all models are wrong. But which ones will be useful, only “*time*” will tell.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. DEFINITIONS AND TERMINOLOGY

This appendix provides readers with important terms and definitions necessary to facilitate the understanding of this dissertation work. Novice readers to the modeling and simulation field are referred to (Law & Kelton, 2000; Banks et. al., 2005; Ross, 2003; Hillier & Lieberman, 2005; Sanchez, 2006; Sanchez, 2007). A **system** is a combination of components or entities that interact with each other to function. An **entity** is an object of interest in the system. The **State** of the system is described as a collection of state variables at an instant point in time. A **State variable** of a system is a variable that changes over time to describe a specific state of the system. The input, output and state variables are collectively called system variables. A **model** is a set of assumptions that are expressed in mathematical, logical and symbolic relationships between entities to represent the system. **Analytical** methods utilize the deductive logic of mathematics (close-form equations) to “solve” the model. Typically analytical methods search for optimal solutions to the model, but heuristics and approximates are the alternatives when it fails. When the system is highly complex where it is impossible to find analytical solutions, simulation becomes the solution method of choice. **Simulation** is an implementation of the model over time typically with the use of computer tools. A **Discrete** system is the one in which the state variables change only at a fix set of points in time. On the other hand, a **continuous** system is the one in which the state variables change continuously with time. **Static** simulation models generally represent systems at a particular point in time, where **dynamic** models represent systems as they change over time, these type of models are very useful to model most physical systems because they change over time. When a simulation model includes no random variables and has a known set of inputs, it is categorized as **deterministic** model. Models that have at least on random variable (input) are called **stochastic** models and the simulation results are random outputs.

Time-advanced mechanism is a technique impeded into the simulation in which the simulation clock is advance in time. Two common approaches are used to advance the simulation clock, time-driven which is the case in DTS and event-driven the case in

DES (Law and Kelton 2000). **Time-driven** simulation-mechanism allows the systems to change their states in response to a uniformly progressing physical time. The physical time is a global variable that has exactly the same value at any level of the system that is determined by the modelers in advance and is called **time-step size Δt** . **Discrete time simulation (DTS)** is a modeling paradigm in which the state variables change only at fixed time steps placed at a uniform rate. In DTS the system state variables can only be updated and changed at the fixed time steps (clock ticks). At every time-step (clock tick) at a uniform rate each state variable is checked for a possible update regardless of any event occurrence. Inputs (events) that cause state variables change are selected from the input set at every time-step. If events occur for only some but not every time-step, they need to be set to “null” as the simulation continuously checks for update at each step. Events are only permitted to occur at clock ticks but not in between ticks. The simulation clock alone is responsible for any state transition, therefore the simulation is categorized as time-driven. The simulation time is a global variable to all component of the system in which all states are synchronized by the time steps. On the other hand, **event-driven** mechanism is a simulation approach that advances the simulation clock based on events occurrence. In this type of simulation system state variables experience instantaneous state transitions in response to the occurrence of asynchronous discrete events. There are no state transitions between any event occurrences in event-driven approach. Under this approach, a sequence of events (not necessarily known in advance) is presented to the system and need not occur at a time step, in which case events occur asynchronously. Time if explicitly defined is simply used as an index to indicate event occurrence. Events are the driver of the simulation clock, hence the term event-driven. A **discrete event simulation (DES)** is modeling the system in which the state variables instantaneously change only at discrete points in time where events occur. The DES structure consist of main elements; state variables (that do change as the simulation progress), events, **event scheduling** relationships and **parameters** (input variables that are fixed though out the simulation). An **event** is the instant that causes one or more state variables (possibly none) to change (referred as state transition). Determining events in DES is not adequate to describe how the simulation processes. Events scheduling relationships demonstrate

the cause and effect between events as how events cause/schedule other events in the future. **State variables** are collection of all information needed to define what is happening within the system to a sufficient level at a given point in time. In order to fully understand the operation of DES, it is necessary to describe the **Future Event List (FEL)**. FEL is the heart of DES by which pending events are held and managed (Buss, 2011a). The FEL is responsible for keeping all pending events in chronological order and hold their information such as event occurring time, and event name. Typically, the FEL is hidden from the modeler view unless visibility options are activated but its process is acknowledged. DES operation entirely relies on the FEL as the backbone to process all events in correct occurrence order (Buss, 2011a). The next-event time advance mechanism in DES models controls the simulation time in a unique way. The simulation clock moves in increments by hopping from one event to another by the procedure shown in Figure 98.

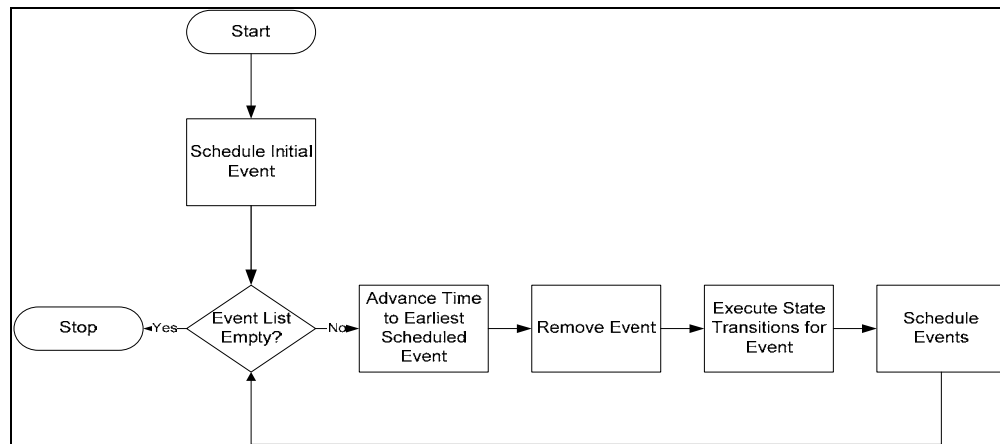


Figure 98. Next-event algorithm used by DES (Buss, 2011a)

The next-event algorithm in Figure 98 starts by scheduling an initial event put in the FEL to occur at time 0.0 (typically called the Run event responsible to initialize all state variables). The simulation clock advances the simulation time by moving it to the time of the earliest schedule event on the FEL. This event get removed from the FEL to allow the next earliest schedule event to be at the top of the list. Once removed, all state variables associated with the processed event set to change to different values (state transition) to indicate the occurrence of the event and its impact on the system.

Depending on the scheduling relationship between events, new event(s) may get schedule and put on the FEL in chronological order. The above algorithm continues as long as there are events waiting to be processed in the FEL, otherwise the simulation stops.

To enhance DES conceptual modeling, **Event Graph (EG)** is a methodology used to represent the activity of events, scheduling relationships between events and event's action on changing the corresponding state variables (Buss, 2001). This method was introduced by (Schruben, 1983; Savage, Schruben, & Yucesan, 2005), the graph consists of nodes and directed arcs linking nodes in specific relationships. Nodes represent events, each node is noted with the followings (Buss, 2011a):

- Event name (inside the node)
- State transitions caused by the event (under the node)
- Event parameter (i), used to specify a type.

On the other hand, arcs represent the scheduling relationships between events. Arcs/edges carry critical information to indicate the relationships between events and the required conditions for an event to schedule other events. Typically, arcs hold the following optional data (Buss, 1996; Buss, 2011a):

- Time delay to schedule the next event
- Boolean conditions (required to satisfy)
- Arguments to pass data to an event
- Priority parameters (used to break ties between events)

Dashed arcs are occasionally used to represent a canceling edge, that is an edge that removes scheduled events from the FEL.

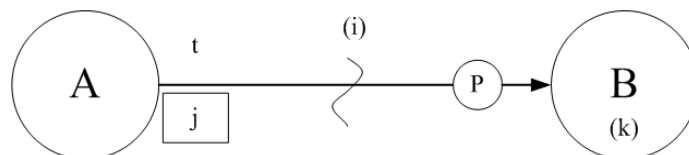


Figure 99. Basic Event Graph Construct (Schruben, 1983)

Figure 99 shows the construct for Event Graphs and is interpreted as follows: the occurrence of Event A causes Event B to be scheduled after time delay of t (or may be scheduled immediately), providing condition (i) is true, and when B occurs its parameter k is set to the value of the expression j at the time it had been scheduled with priority p . DES is an event scheduling approach, time delay between events is an essential element as it controls the event scheduling process which contributes to systematize the FEL.

For large models, event graph components are used to maintain the reliability of those large models. Each component represents a partition of the main model in the way Event Graph previously described but all components share one FEL. Each component has its own set of events, parameters and state variables. However, components interaction is provided by the use of Event Listener Pattern and Adapter Pattern (Buss, 2011a). In discrete event simulation **Event Listener Pattern** is a mechanism by which events in one component can affect the state of events in other components by using the Listener Event Graph Objects (LEGO) component framework (Buss and Sanchez 2002). In DES, model complexity is measured by the number of events, arcs, and the size of event list (Schruben & Yucesan, 1993). Since time and budget in the real world are limited, the use of LEGO is important to manage the efforts and resources in DES by simplifying the event graphs.

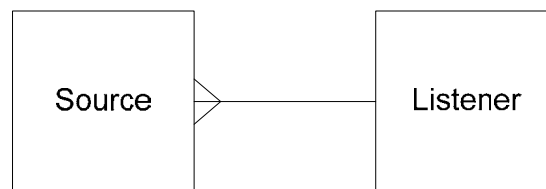


Figure 100. Listener component listening to Source component (Buss & Sanchez, 2002)

Figure 100 shows the listener relationship between two components. The listener component listens to all events in the source component and if the listener has an identical event (name and signature) to the one it listens to from the source, the event gets scheduled. **Adapter Pattern** works in a similar manner and same purpose as the listener pattern. The only difference is that it is used when one event in a component causes another event of a

different name but same signature in other component to be scheduled (Buss & Sanchez, 2002).

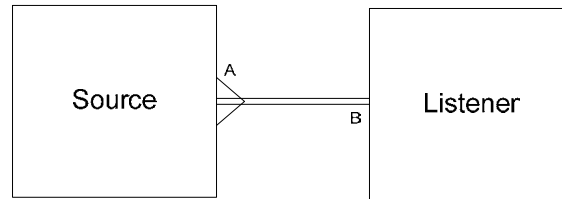


Figure 101. Adapter example: event A in Source component causes event B in Listener component (Buss & Sanchez, 2002)

Adapters are only used to listen to single specified events but not all like the case in listener pattern.

A Queueing system is described by any type of service systems in which the nature of arrivals, the service mechanism, and the queueing discipline are present in a system (Ross, 2003). These types of systems exist in our everyday life activities, ranging from simple ATM, restaurants, transportation systems, computer systems, logistic systems, maintenance systems, manufacturing systems etc. Customers in queueing systems could be people or items such data packets, physical parts etc.

We define a number of common measures of performance used in queueing systems. **Queue length** is refers to the number of items in queue, for example customers or packets waiting in some location waiting to be processed. It is often an indication of how well the queueing system is performing. **Waiting time** is a time delay and it is a measured as the duration of time between an entity's arrival and when it starts to receive service. The longer the delay, the worse this system perceived by customers. This is the most used measure of performance by customers (Alfa, 2010). **System time** is basically the waiting time in queue and the time to receive a service. It is alleged to be the same as the waiting time, however, it is useful when an entity's interruption in the system is probable. **Busy period** is a measure of the time that elapses from when the server starts service to when the server has no entities in the queue to serve. **Idle period** is a measure of the time that elapses from when the last item is processed leaving the queue empty to when a service starts again. **Queue discipline** represents the way the queue is organized.

It states the rule of inserting and removing entities to/from queue. One way is **FIFO**, first in first out or FCFS first come first serve in which entities are organized in orderly way in the queue. The second way is **LIFO**, last in first out or last come first serve in which entities are stacked in the queue.

Transient state is the start-up period in which the system starts from empty queue and builds activities as entities are added to the queue. For example, a restaurant opens in the morning without any customers, then gradually activity builds up to normal state. The beginning or start-up period is referred to as transient state. **Steady state** is when over a long period of time the system reaches normal operation conditions. Measures of performance are typically determined at the steady state level to observe the system behavior under stable operation (Ross, 2003).

Poisson process is a stochastic process in which the occurrence of a sequence of discrete events occurs with exponentially distributed time intervals between successive events. These events occur independently of one another in which the number of arrivals before time t is independent of the number of arrivals after time t (Ross, 2003).

Markov chain is random process in which the system undergoes transitions from one state to another providing the next state depends on the current state and not the past state. Markov chains are considered the basis for analytical methods of queueing systems (Ross, 2003).

Combat model is any structural activity that involves modeling weapon systems, combat situation, and military operations, including combat attrition, target acquisition and detection, weapon defense and operational availability (Washburn & Kress, 2009). Combat models classified as deterministic models in which the model states exactly what should happen, and stochastic models in which the model assumes probabilistic or uncertain inputs to the experiment and observe indefinite prediction of the outcomes. Combat models serve many purposes such as in research and development to provide theoretical investigation of combat phenomena and design of weapons. They heavily used in training as aids for preparing soldiers and units for combats. Combat models are

continuously used in operation and evaluation analysis as decision support aids for force capability, combat development and planning on-going operations.

Combat models come in three modes, descriptive (the mode used in this research), prescriptive and predictive. **Descriptive** is when the model describe the nature of combat, gain insights and describe trends. **Prescriptive** is when the model prescribe a solution to the problem or combat situations. Predictive is the most dangerous mode in which models are used to predict the battle outcomes. The degree of aggregation in combat models depends on the purpose of the model and varies from high resolution models to aggregated models. **High-resolution** combat models include detailed view of warfare by representing individual combatants a separate entities. **Aggregated** combat models involve combining a number of individual combatants into groups to represent units and decrease the simulation complexity of large-scale models (Washburn & Kress, 2009).

The basic elements of combat models are; 1) **entity/unit** is the state of how actors are defined in the model (e.g., person, vehicle, sensor) 2) **movement** is the process that defines how actors change location in the modeling environment 3) **target acquisition** is how observers use sensors to acquire targets (i.e., be aware of targets presence and positions). 4) **attrition** is the lethality of weapons and vulnerability of targets in a combat environment 5) **decision making** of an actor is all the behaviors and reactions that it undergoes while immersed in combat situation 6) **communications** between actors describe how messages are sent and received 7) **situational awareness** is how actors perceives the state of the surrounding environment such as enemy contacts.

An **Agent** as used in this study is defined as identifiable, autonomous, self-directed, and a discrete individual situated in an environment with a set of characteristics and rules governing its behaviors and decision-making capability. An agent is anything that can be viewed to perceive its environment through sensors and acting up on that by the use of rules. Agents have goals to achieve, interact with other agents in the environment and has the ability to learn and adapt its behavior over time (Russell & Norvig, 2003). Example of agents are people, groups, organizations, social insects, swarms, and robots. An **Agent-based models (ABM)** is thus “a simulation made up of agents, objects or entities that behave autonomously. These agents are aware of (and

interact with) their local environment through simple internal rules for decision-making, movement, and actions (Cioppa et al., 2002).

Map Aware Non-Uniform Automata (MANA) is a low-resolution, time-stepped simulation tool designed by New Zealand's Defence Technology Agency (DTA) and released in 1999. MANA is an agent-based model that has recently attracted significant attention in the military Operational Research community (McIntosh et al., 2007). MANA has been used in a number of studies: modeling military combat operations, modeling civil violence management, modeling modern warfare as a complex adaptive system, modeling maritime surveillance and coastal patrols and a range of agent-based model applications in the civilian and military sectors. MANA as a simulation tool is being used by a number of military colleges and defense science establishments as well as used for various Master's theses at the Naval Postgraduate School NPS at Monterey (Straver, Vincent, & Fournier, 2006).

Dynamic Allocation of Fires and Sensors (DAFS) is a low to high-resolution, discrete event simulation tool with embedded optimization designed for TRAC-Monterey in 2002 (Havens, 2002). It is built to provide maximum flexibility in the evaluation of networked fires for future combat systems. DAFS is an open source model, a constructive entity-level simulation framework programmed in JAVA and incorporates the functionality of Simkit. The initial version of this model provided a simulation tool for TRAC-Monterey to explore scenarios for the Army's Future Combat System (Buss & Anher, 2006). DAFS as a discrete event model is built on the concept of Schruben's (1983) Event Graph methodology in which this approach uniquely suited to describe and implement the types of movement, sensing and weapons effects interactions of entities. Later in 2007 the model expanded beyond the use of Army studies to include other military branches in which the model is named Joint Dynamic Allocation of Fires and Sensors JDAFS.

Simkit is a free and opened source discrete event simulation library written in JAVA by Professor Arnold Buss of the Naval Postgraduate School NPS (Buss, 2001; Buss, 2002). Simkit is used to implement event scheduling paradigm and as a foundation library for Viskit which is a visual modeling tool. Simkit interacts with the FEL by using

classes that inherit abstract class `SimEntityBase`. Simkit implements DES with LEGOs, using loosely coupled, component-based modeling with design pattern such as listener, referee, and factory (Buss, 2002). Simkit also implements random variates factories to generate any probability distribution in the model. Two listener patterns are used in Simkit; 1) “SimEventListener” which allows for dividing a large event graph in components and enables loose coupling between them, 2) “PropertyChangeListener” which allows for decoupling of statistics-data collection from events (Buss, 2001). As a simulation tool, Simkit has been reliable and capable of modeling many real world problems in both civil and military areas, and is being used intensively by numerous projects including PhD dissertations and Master’s theses at NPS.

Peace Support Operations Model (PSOM) is a human-in-loop, time stepped semiautomated wargame, campaign level model (Parkman & Hanley, 2008). The model was developed in 2004 by the Defense Science and Technology Lab (Dstl) of the UK Ministry of Defense in order to test policy guidance and provide campaign context for lower level modeling. PSOM is written in Visual Basic and built from a series of algorithms and sub-models to represent complex combat models that incorporate the Political, Economic, Social, Infrastructure, and Information (PMESII) aspects of irregular warfare (IW). PSOM focuses on the population as agents and much of its output is a cultural representation of uncertain effects military and political actions have on the population. The model is designed to first study and then understand stabilization operations in IW situations by allowing interactions between multiple political entities such as the local government, coalition forces or terrorist organizations with the populations. It is meant to show the causal effects of the players’ actions primarily through their effect on the population.

Pythagoras is a time-stepped agent based model (ABM) developed by the Marine Corps Warfighting Laboratory Project Albert team¹². The continuing work started by Project Albert is currently being managed by the Naval Postgraduate School (NPS) Simulation, Experiments, and Efficient Designs (SEED) Center in Monterey,

¹² Information on Project Albert can be accessed at <http://www.projectalbert.org/>, last accessed on

16 June 2011

California¹³. Pythagoras is a government-owned and open source ABM written in the JAVA computer programming language, making it platform independent. The maintenance and upgrades for Pythagoras are contracted to Northrop Grumman (Northrop Grumman Space & Mission Systems Corp., 2007). It allows one to create autonomous, intelligent acting agents that can act and react based on multiple decision rules. The decision rules determine the specific appearance of an agent, they can be seen as variables. The modeler can declare these variables as constants (deterministic) or can define a tolerance around a mean which will be reset every time step by a random draw (stochastic). Although Pythagoras is primarily a combat model, Version 2.1.1 is designed to offer improved capabilities to model intangibles, such as attitudinal responses and enable it to model civilian populace facing constant attempted influence from outside interests to change a believe or stance in a social network (Ferris, 2008).

Cultural Goegraphy (CG) model is a discrete-event, stochastic, agent-based model (ABM) implemented in Java that uses Simkit 1.3.7 as the simulation engine. The CG model development at TRAC-MRY is an ongoing effort, which will continue as new purposes for the model are identified. The current Cultural Geography (CG) model prototype implementation represents a potential path forward to gain insight into the response of the civilian population in IW. The CG model is a data driven multi-agent simulation (MAS) for the representation of social science theories in a coherent Java-based ‘plug and play’ module framework. It is designed to provide insight into the potential effects of blue force actions on the beliefs, values and interests of individuals in the civilian population. The CG model allows decision makers to track individual, group-level, and population-wide changes on positions related to issues that are salient to the area of operations, such as ‘Is security adequate?’ and ‘Will the outcome of upcoming elections be legitimate?’ The output of the model can be best described as equivalent to polling data (Alt et al., 2009).

¹³ Information on the NPS SEED Center can be accessed at <http://harvest.nps.edu/>, last accessed on 5 May 2011.

Continuous system (CS) normally requires that each operation is performed at every “tick” of the system clock (M&SCO, 2010). CS models typically involve differential equations that provide representation for the rates of change of the state variables over time. If these equations are simple, analytical techniques are employed to solve for the value of the state variables over all values of time. However, due to the complexity of the system, most analytical solutions are infeasible and numerical methods such as Euler, and Rung-Kutta integrations are used to provide numerical solutions to these differential equations.

System Dynamic (SD) created by Forrester in the mid-to late 1950, is a method of representing the dynamic behavior of complex systems by breaking down these systems into a number of interconnected components “blocks” connected together by links “wires” (Zacharias, MacMillan, & Van Hemel, 2008). In SD models the blocks and wires are a series of stocks and flows, in which the state changes are continuous. An SD model views “entities” as a continuous quantity, rather like a fluid, flowing through a system of reservoirs or tanks connected by pipes. The rate of flow is controlled by valves, and so the time spent in each reservoir is modelled by fixing the rates of inflow and outflow. Although the state changes are regarded as continuous, the underlying equations used to solve the model are differential or difference equations (usually solved by numerical integration) which discretise time using a time-step approach.

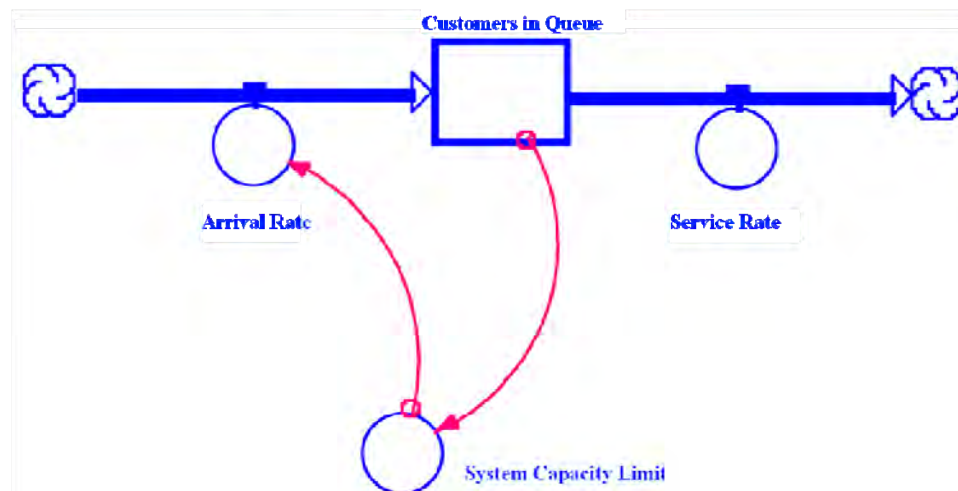


Figure 102. Example of a simple queueing service SD model

In SD state changes happen continuously at small segments of time-step sizes (Δt) and specific entities cannot be followed throughout the system. Unlike DES models, SD models are generally deterministic and variables usually represent average values.

Discrete Event System Specification (DEVS) is a formalism developed by Bernard Zeigler in the mid-seventies; it is related with modeling continuous systems and not very known approach to the numerical method and control community but widely used in computer sciences (Zeigler, Praehofer, & Kim, 2000). DEVS represents continuous systems input and output behavior by sequences of events with the condition that the state of a variable has a finite number of changes in any finite interval of time. The behavior of a DEVS model is expressed in a way which is very common in automata theory. Typically, a DEVS atomic model is defined by the following formalism:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a)$$

where,

X is the set of input event values, i.e., the set of all possible values that an input event can adopt.

Y is the set of output event values.

S is the set of state values.

δ_{int} , δ_{ext} , λ and t_a are functions which define the system dynamics.

Each state s ($s \in S$) has an associated time advance computed by the time advance function $t_a(s)$ in which $[t_a(s) : S \rightarrow \mathbb{R}^+]$. An input event enters the system with a value x_1 at time e_1 , causes the system to go from state value s_0 at time t_0 to state value s_1 at time t_1 . The new state is calculated as $s_1 = \delta_{ext}(s_0, e_1, x_1)$. In this case, it is said that the system performs an external transition and δ_{ext} is called External Transition Function. After $t_a(s_1)$ units of time (i.e. at time $t_1 + t_a(s_1)$) the system performs an internal transition to go to a new state value s_2 . The new state is calculated as $s_2 = \delta_{int}(s_1)$ where the function

δ_{int} is called the Internal transition Function. As the state goes from s_1 to s_2 an output event is produced with a value $y_1 = \lambda(s_1)$. No output should be produced during the an external transition (Zeigler, Praehofer, & Kim, 2000).

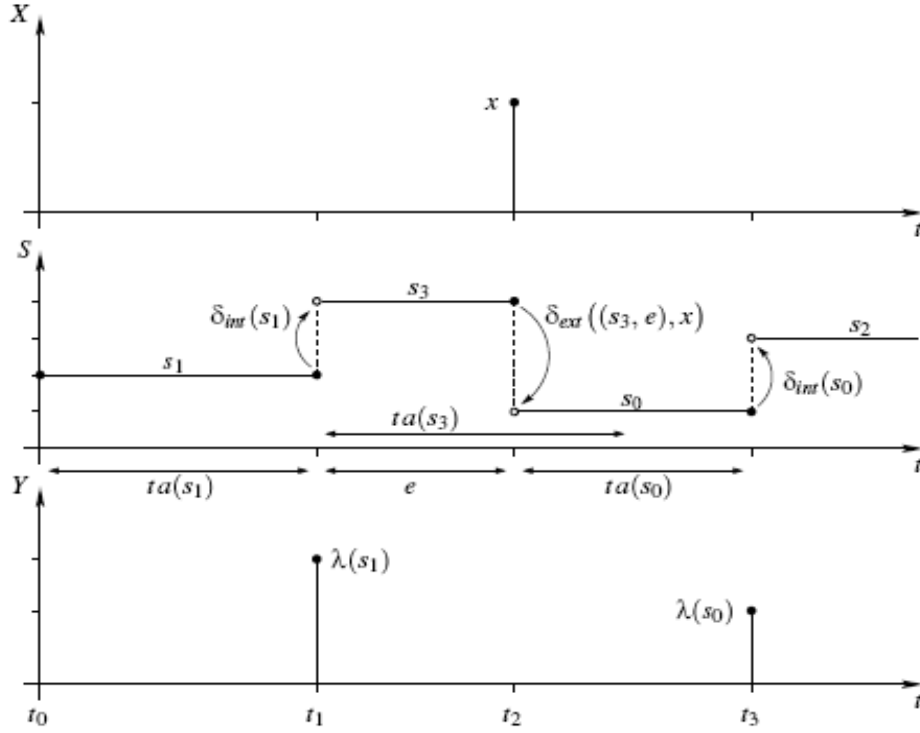


Figure 103. Illustration of atomic-DEVS Model (from Bolduc, 2002)

APPENDIX B. QUEUEING SYSTEM COMPUTER CODES

A. M/M/K & M/M/K/C DES PROGRAM IN SIMKIT

Arrival Process Program:

```
package mmk_11;
import simkit.SimEntityBase;
import simkit.random.RandomVariate;
public class ArrivalProcess extends SimEntityBase {
    private RandomVariate interarrivalTime;
    private int numberArrivals;
    public ArrivalProcess(RandomVariate interarrivalTime){
        this.setInterarrivalTime(interarrivalTime);
    }
    public void reset() {
        super.reset();
        numberArrivals = 0;
    }
    public void doRun(){
        firePropertyChange("numberArrivals","getNumberArrivals());
        waitDelay("Arrival," interarrivalTime);
    }
    public void doArrival(){
        int oldNumberArrivals = getNumberArrivals();
        numberArrivals = numberArrivals +1;
        firePropertyChange("numberArrivals","oldNumberArrivals,
            getNumberArrivals());
        waitDelay("Arrival," interarrivalTime);
    }
    public RandomVariate getInterarrivalTime() {
        return interarrivalTime;
    }
    public void setInterarrivalTime(RandomVariate interarrivalTime) {
        this.interarrivalTime = interarrivalTime;
    }
    public int getNumberArrivals() {
        return numberArrivals;
    }
}
```

Simple Server Program for M/M/k & M/M/k/c:

```
package mmk_11;
import simkit.Priority;
import simkit.SimEntityBase;
import simkit.random.RandomNumber;
import simkit.random.RandomVariate;
import simkit.random.RandomVariateFactory;
public class SimpleServer extends SimEntityBase {
    private int totalNumberServers;
    private RandomVariate serviceTimeGenerator;
    private RandomNumber rng;
    private int maxNumberInSystem;
    protected int numberInQueue;
    protected int numberAvailableServers;
    protected int numberServed;
    protected double pJoin;

    public SimpleServer(int totalNumberServers, RandomVariate serviceTimeGenerator) {
        setTotalNumberServers(totalNumberServers);
        setServiceTimeGenerator(serviceTimeGenerator);
        rng = RandomVariateFactory.getDefaultRandomNumber();
    }
}
```

```

}
public void reset() {
    super.reset();
    numberInQueue = 0;
    numberAvailableServers = totalNumberServers;
    numberServed = 0;
}
public void doRun() {
    firePropertyChange("numberInQueue," getNumberInQueue());
    firePropertyChange("numberAvailableServers," getNumberAvailableServers());
    firePropertyChange("numberServed," getNumberServed());
}
public void doArrival() {
    //Initial model when arrivals cause joining queue immed.
    waitDelay("JoinQueue," 0.0, Priority.HIGH);
    //edge conditions for NIQ balking
    //if (numberInQueue < 5) {
    //waitDelay("JoinQueue," 0.0, Priority.HIGH);
    //}
    //edge conditions for NIS dependent balking
    //if (rng.draw() < getpJoin()) {
    //waitDelay("JoinQueue," 0.0, Priority.HIGH);
    //}
}
public void doJoinQueue() {
    int oldNumberInQueue = getNumberInQueue();
    numberInQueue = numberInQueue + 1;
    firePropertyChange("numberInQueue," oldNumberInQueue, getNumberInQueue());
    if (numberAvailableServers > 0) {
        waitDelay("StartService," 0.0, Priority.HIGH);
    }
}
public void doStartService() {
    int oldNumberAvailableServers = getNumberAvailableServers();
    numberAvailableServers = numberAvailableServers - 1;
    firePropertyChange("numberAvailableServers," oldNumberAvailableServers,
getNumberAvailableServers());
    int oldNumberInQueue = getNumberInQueue();
    numberInQueue = numberInQueue - 1;
    firePropertyChange("numberInQueue," oldNumberInQueue, getNumberInQueue());

    if (true) {
        waitDelay("EndService," serviceTimeGenerator.generate(),
Priority.DEFAULT);
    }
}
public void doEndService() {
    int oldNumberAvailableServers = getNumberAvailableServers();
    numberAvailableServers = numberAvailableServers + 1;
    firePropertyChange("numberAvailableServers," oldNumberAvailableServers,
getNumberAvailableServers());

    int oldNumberServed = getNumberServed();
    numberServed = numberServed + 1;
    firePropertyChange("numberServed," oldNumberServed, getNumberServed());
    if (numberInQueue > 0) {
        waitDelay("StartService," 0.0, Priority.HIGH);
    }
}
public int getTotalNumberServers() {
    return totalNumberServers;
}
public void setTotalNumberServers(int totalNumberServers) {
    if (totalNumberServers <= 0){
        throw new IllegalArgumentException ("totalNumberServers must be" + " > 0:
"+ totalNumberServers);
    }
    this.totalNumberServers = totalNumberServers;
}
}

```

```

    public RandomVariate getServiceTimeGenerator() {
        return serviceTimeGenerator;
    }
    public void setServiceTimeGenerator(RandomVariate serviceTimeGenerator) {
        this.serviceTimeGenerator = serviceTimeGenerator;
    }
    public int getNumberInQueue() {
        return numberInQueue;
    }
    public int getNumberAvailableServers() {
        return numberAvailableServers;
    }
    public int getNumberServed() {
        return numberServed;
    }
    public int getMaxNumberInSystem() {
        return maxNumberInSystem;
    }
    public double getpJoin() {
        return 1-(getNumberInQueue()+ (getTotalNumberServers()
        getNumberAvailableServers()))/getMaxNumberInSystem();
    }
    public void setMaxNumberInSystem(int maxNumberInSystem) {
        this.maxNumberInSystem = maxNumberInSystem;
    }
}
}

```

Executable Simple Server Program for M/M/k & M/M/k/c:

```

package mmk_11;

import java.text.DecimalFormat;
import simkit.Schedule;
import simkit.random.RandomVariate;
import simkit.random.RandomVariateFactory;
import simkit.stat.SimpleStatsTally;
import simkit.stat.SimpleStatsTimeVarying;
public class RunSimpleServer {
    public static void main(String[] args) {

        RandomVariate interarrivalTime =
            RandomVariateFactory.getInstance("Exponential," 0.0645);
        ArrivalProcess arrivalProcess = new ArrivalProcess(interarrivalTime);
        RandomVariate serviceTime =
            RandomVariateFactory.getInstance("Exponential," 0.250);
        int numberServers = 1;

        int maxNumberInSystem = 5;
        SimpleServer simpleServer = new SimpleServer(numberServers, serviceTime);
        simpleServer.setMaxNumberInSystem(maxNumberInSystem);
        arrivalProcess.addSimEventListener(simpleServer);
        SimpleStatsTimeVarying niqStat = new
        SimpleStatsTimeVarying("numberInQueue");
        simpleServer.addPropertyChangeListener(niqStat);
        SimpleStatsTimeVarying nasStat = new
        SimpleStatsTimeVarying("numberAvailableServers");
        simpleServer.addPropertyChangeListener(nasStat);
        SimpleStatsTally numberInSystemOuterStat = new
        SimpleStatsTally("numberInSystem");
        SimpleStatsTally timeInSystemOuterStat = new
        SimpleStatsTally("timeInSystem");
        NinQueue & NAVlServers
        SimpleStatsTally niqReplStat = new SimpleStatsTally("niqR");
        SimpleStatsTally nasReplStat = new SimpleStatsTally("nasR");
        System.out.println();
        System.out.println(arrivalProcess);
        System.out.println(simpleServer);
        Schedule.setVerbose(false); // like doing event list by hand
        Schedule.stopAtTime(1500.0);
        int nreplications = 40;
    }
}

```

```

        long start = System.currentTimeMillis();
        for (int replication = 0; replication < nreplications; ++replication) {
            Schedule.reset();
            niqStat.reset();
            nasStat.reset();
            Schedule.startSimulation();
            niqReplStat.newObservation(niqStat.getMean());
            nasReplStat.newObservation(nasStat.getMean());
            numberInSystemOuterStat.newObservation(simpleServer.getTotalNumberServers() - nasStat.getMean() + niqStat.getMean());
            timeInSystemOuterStat.newObservation(numberInSystemOuterStat.getMean() / (arrivalProcess.getNumberArrivals() / Schedule.getSimTime()));
        }
        long end = System.currentTimeMillis();
        System.out.println("Run took " + (end - start) + " milliseconds");
        DecimalFormat df = new DecimalFormat(" 0.000;-0.000");
        System.out.println("Simulation Ended at time " + Schedule.getSimTime());
        System.out.println("There have been " + arrivalProcess.getNumberArrivals() + " arrivals.");
        System.out.println("There have been " + simpleServer.getNumberServed() + " customer served");
        System.out.println("Average Number in Queue" + df.format(niqStat.getMean()));
        System.out.println("Average Utilization" + df.format(1.0 - nasStat.getMean() / simpleServer.getTotalNumberServers()));
        System.out.println("Average Number of Customer in System" + df.format(niqStat.getMean() + (numberServers - nasStat.getMean())));
        System.out.println("Average Arrival Rate" + df.format(arrivalProcess.getNumberArrivals() / Schedule.getSimTime()));
        System.out.println("Average Time in System" + df.format((niqStat.getMean() + (numberServers - nasStat.getMean())) / (arrivalProcess.getNumberArrivals() / Schedule.getSimTime())) + " min" + "\n");
        System.out.println("Number of Replications " + (nreplications));
        System.out.println("Replications' Mean of Average Number in Queue" + df.format(niqReplStat.getMean()));
        System.out.println("Replications' Mean of Average Number of Customer in System" + df.format(niqReplStat.getMean() + (numberServers - nasReplStat.getMean())));
        System.out.println(numberInSystemOuterStat);
        System.out.println(timeInSystemOuterStat);
    }
}

```

B. M/M/K & M/M/K/C DTS PROGRAM IN SIMKIT

Geometric Distribution Arrival approach:

```

package mmk_11;
import java.text.DecimalFormat;
import simkit.random.RandomNumber;
import simkit.random.RandomNumberFactory;
import simkit.stat.SimpleStatsTally;
public class TimeStepQ {
    static int numberOfArrivals;
    static int numberInQueue;
    static double queueInteg;
    static double availableServerInteg;
    static int numberServed;
    static double pJoin;
    static double Pa = 0.0;
    static double Ps = 0.0;
    public static void clear() {
        numberOfArrivals = 0;
        numberInQueue = 0;
        queueInteg = 0.0;
        availableServerInteg = 0.0;
    }
}

```

```

        numberServed = 0;
        pJoin = 0.0;
    }
    public static void main(String[] args) {
        double meanInterarrivalTime = 0.0645;
        double meanServiceTime = 0.250;
        int numberServers = 2;
        int numberAvailableServers = numberServers;
        double dt = 0.02;
        double stopTime = 100.0;
        double nreplications = 2;
        int maxNumberInSystem = 300;
        RandomNumber rng = RandomNumberFactory.getInstance();
        SimpleStatsTally ninsystem = new SimpleStatsTally("nis");
        SimpleStatsTally tinsystem = new SimpleStatsTally("tis");
        DecimalFormat df = new DecimalFormat(" 0.000;-0.000");
        Pa = dt / meanInterarrivalTime;
        Ps = dt / meanServiceTime;
        long start = System.currentTimeMillis();
        for (int replication = 0; replication < nreplications; ++replication) {
            TimeStepQ.clear();
            for (double t = 0.0; t <= stopTime; t = t + dt) {
                queueInteg = queueInteg + numberInQueue * dt;
                availableServerInteg = availableServerInteg + numberAvailableServers*dt;
                for (int i = 0; i < numberServers - numberAvailableServers; ++i) {
                    if (rng.draw() <= Ps) {
                        numberAvailableServers = numberAvailableServers + 1;
                        numberServed = numberServed + 1;
                        if (numberInQueue > 0) {
                            numberInQueue = numberInQueue - 1;
                            numberAvailableServers = numberAvailableServers - 1;
                        }
                    }
                }
            }
            pJoin = 1-(numberInQueue + (numberServers -
            numberAvailableServers))/maxNumberInSystem;
            //1.Only if RV satisfies arrival probability then we have an arrival
            //2.Only if RV satisfies arrival probability and within queue capacity
            (for balking type 1) then we have an arrival
            //3.Only if RV satisfies arrival probability as well as CIS probability
            (for balking type 2) then we have an arrival
            //if (rng.draw() <= Pa) {
            //if (rng.draw() <= Pa && numberInQueue < 5) {
            if (rng.draw() <= Pa && rng.draw() <= pJoin) {
                numberInQueue = numberInQueue + 1;
                numberOfArrivals = numberOfArrivals + 1;
            }
            if (numberInQueue > 0 && numberAvailableServers <= numberServers &&
            numberAvailableServers != 0) {
                numberInQueue = numberInQueue - 1;
                numberAvailableServers = numberAvailableServers -1;
            }
        }
        double utilization = 1.0 - availableServerInteg / (numberServers *
        stopTime);
        double numberinsystem = (queueInteg / stopTime) + (numberServers -
        (availableServerInteg / stopTime));
        ninsystem.newObservation(numberinsystem);
        double timeinsystem = ((queueInteg / stopTime) + (numberServers -
        (availableServerInteg / stopTime))) / (numberOfArrivals /
        stopTime);
        tinsystem.newObservation(timeinsystem);
    }

    long end = System.currentTimeMillis();
    System.out.println("Run took " + (end - start) + " milliseconds");
    System.out.println("Number of Replications " + (nreplications));
    System.out.println("Replications' Mean of Average Number of Customer in
    System" + df.format(ninsystem.getMean()));

```

```

        System.out.println("Replications' StDev of Average Number of Customer in
        System" + df.format(ninsystem.getStandardDeviation()));
        System.out.println("Replications' Mean of Average Time in System" +
        df.format(tinsystem.getMean()) + " min");
        System.out.println("Replications' StDev of Average Time in System" +
        df.format(tinsystem.getStandardDeviation()));
    }
}

```

Poisson Distribution (Batch) Arrival Approach:

```

package mmk_11;
import java.text.DecimalFormat;
import simkit.random.DiscreteRandomVariate;
import simkit.random.RandomNumber;
import simkit.random.RandomNumberFactory;
import simkit.random.RandomVariateFactory;
import simkit.stat.SimpleStatsTally;
public class TimeStepQBatch {
    static int numberOfArrivals;
    static int numberInQueue;
    static int numberAvailableServers;
    static double queueInteg;
    static double availableServerInteg;
    static int numberServed;
    static double Ps;
    static int numberServers = 2;
    public static void clear() {
        numberOfArrivals = 0;
        numberInQueue = 0;
        numberAvailableServers = numberServers;
        queueInteg = 0.0;
        availableServerInteg = 0.0;
        numberServed = 0;
    }
    public static void main(String[] args) {
        double meanInterarrivalTime = (1.0 / 18.0);
        double meanServiceTime = 0.250;
        double dt = 0.002;
        double stopTime = 150.0;
        double nreplications = 40;
        double arrivalTSRate = dt * (1 / meanInterarrivalTime);
        Ps = dt / meanServiceTime;
        RandomNumber rng = RandomNumberFactory.getInstance();
        DiscreteRandomVariate poissonV =
            RandomVariateFactory.getDiscreteRandomVariateInstance("Poisson,"
                arrivalTSRate);
        SimpleStatsTally ninsystem = new SimpleStatsTally("nis");
        SimpleStatsTally tinsystem = new SimpleStatsTally("tis");
        SimpleStatsTally numberArrivalsStat = new
        SimpleStatsTally("numberArrivals");
        SimpleStatsTally numberServedStat = new SimpleStatsTally("numberServed");
        DecimalFormat df = new DecimalFormat(" 0.000;-0.000");
        long start = System.currentTimeMillis();
        for (int replication = 0; replication < nreplications; ++replication) {
            TimeStepQBatch.clear();
            for (double t = 0.0; t <= stopTime; t = t + dt) {
                queueInteg = queueInteg + numberInQueue * dt;
                availableServerInteg = availableServerInteg +
                numberAvailableServers * dt;
                int numberEndServices = 0;
                for (int i = 0; i < numberServers - numberAvailableServers; ++i) {
                    if (rng.draw() <= Ps) {
                        numberServed = numberServed + 1;
                        numberEndServices += 1;
                    }
                }
                numberAvailableServers += numberEndServices;
                int batchN = poissonV.generateInt();
            }
        }
    }
}

```

```

        numberInQueue = numberInQueue + batchN;
        numberOfArrivals = numberOfArrivals + batchN;
        numberInQueue = numberInQueue - numberAvailableServers;
        if (numberInQueue > 0) {
            numberAvailableServers = 0;
        } else {
            numberAvailableServers = -(numberInQueue);
            numberInQueue = 0;
        }
    }
    pJoin = 1-(numberInQueue + (numberServers -
    numberAvailableServers))/maxNumberInSystem;
    //1.Only if RV satisfies arrival probability then we have an arrival
    //2.Only if RV satisfies arrival probability and within queue capacity
    (for balking type 1) then we have an arrival
    //3.Only if RV satisfies arrival probability as well as CIS probability
    (for balking type 2) then we have an arrival
    //if (rng.draw() <= Pa) {
    //if (rng.draw() <= Pa && numberInQueue < 5) {
        if (rng.draw() <= Pa && rng.draw() <= pJoin) {
            numberInQueue = numberInQueue + 1;
            numberOfArrivals = numberOfArrivals + 1;
        }
    }
    double utilization = 1.0 - availableServerInteg / (numberServers *
    stopTime);
    double numberinsystem = (queueInteg / stopTime) + (numberServers -
    (availableServerInteg / stopTime));
    ninsystem.newObservation(numberinsystem);
    double timeinsystem = ((queueInteg / stopTime) + (numberServers -
    (availableServerInteg / stopTime))) / (numberOfArrivals / stopTime);
    tinsystem.newObservation(timeinsystem);
    numberArrivalsStat.newObservation(numberOfArrivals);
    numberServedStat.newObservation(numberServed);
}
long end = System.currentTimeMillis();
System.out.println("Run took " + (end - start) + " milliseconds");
System.out.println("Number of Replications " + (nreplications));
System.out.println("Replications' Mean of Average Number of Customer in System" +
    df.format(ninsystem.getMean()));
System.out.println("Replications' StDev of Average Number of Customer in System"
    + df.format(ninsystem.getStandardDeviation()));
System.out.println("Replications' Mean of Average Time in System" +
    df.format(tinsystem.getMean()) + " min");
System.out.println("Replications' StDev of Average Time in System" +
    df.format(tinsystem.getStandardDeviation()));
}
}
}

```

C. TOLL PLAZA DES PROGRAM IN SIMKIT

```

package mmk_11;
import java.util.ArrayList;
import simkit.Priority;
import simkit.SimEntityBase;
import simkit.random.RandomNumber;
import simkit.random.RandomVariate;
import simkit.random.RandomVariateFactory;
public class DesTollPlaza extends SimEntityBase {
    private int numberOfQueues;
    private RandomVariate serviceTimeGenerator;
    protected int[] numberInQueue;
    protected int[] numberAvailableServers;
    protected int numberServed;
    public double niqStatMean = 0.0;
    public double nasStatMean = 0.0;
    public DesTollPlaza(int numberOfQueues, RandomVariate serviceTimeGenerator) {
        setNumberOfQueues(numberOfQueues);
        setServiceTimeGenerator(serviceTimeGenerator);
    }
}

```

```

        numberInQueue = new int[numberOfQueues];
        numberAvailableServers = new int[numberOfQueues];
    }
    public void reset() {
        super.reset();
        numberServed = 0;
    }
    public void doRun() {
        firePropertyChange("numberServed," getNumberServed());

        waitDelay("Init," 0.0, Priority.HIGHER, 0);
    }
    public void doInit(int i) {
        numberInQueue[i] = 0;
        fireIndexedPropertyChange(i, "numberInQueue," getNumberInQueue());

        numberAvailableServers[i] = 1;
        fireIndexedPropertyChange(i, "numberAvailableServers,"
            getNumberAvailableServers());
        if (i < getNumberOfQueues() - 1) {
            waitDelay("Init," 0.0, Priority.HIGHER, i + 1);
        }
    }
    public void doArrival() {
        int value = Integer.MAX_VALUE;
        int shortest = 0;
        for (int j = 0; j < numberOfQueues ; ++j) {
            if (numberInQueue[j] + 1 - numberAvailableServers[j] < value) {
                value = numberInQueue[j] + 1 - numberAvailableServers[j];
                shortest = j;
            }
        }
        waitDelay("JoinQueue," 0.0, shortest);
    }
    public void doJoinQueue(int i) {
        int oldNumberInQueue = getNumberInQueue(i);
        numberInQueue[i] += 1;
        fireIndexedPropertyChange(i,"numberInQueue," oldNumberInQueue,
            getNumberInQueue(i));
        if (getNumberAvailableServers(i) > 0) {
            waitDelay("StartService," 0.0, Priority.HIGH,i);
        }
    }
    public void doStartService(int i) {
        int oldNumberAvailableServers = getNumberAvailableServers(i);
        numberAvailableServers[i] = 0;
        fireIndexedPropertyChange(i,"numberAvailableServers," oldNumberAvailableServers,
            getNumberAvailableServers(i));

        int oldNumberInQueue = getNumberInQueue(i);
        numberInQueue[i] -=1;
        fireIndexedPropertyChange(i,"numberInQueue," oldNumberInQueue,
            getNumberInQueue(i));
        if (true) {
            waitDelay("EndService," serviceTimeGenerator.generate(), Priority.DEFAULT,i);
        }
    }
    public void doEndService(int i) {
        int oldNumberAvailableServers = getNumberAvailableServers(i);
        numberAvailableServers[i] = 1;
        fireIndexedPropertyChange(i,"numberAvailableServers," oldNumberAvailableServers,
            getNumberAvailableServers(i));
        int oldNumberServed = getNumberServed();
        numberServed = numberServed + 1;
        firePropertyChange("numberServed," oldNumberServed, getNumberServed());
        if (numberInQueue[i] > 0) {
            waitDelay("StartService," 0.0, Priority.HIGH,i);
        }
    }
}

```



```

    public int getNumberOfQueues() {
        return numberOfQueues;
    }
    public void setNumberOfQueues(int numberOfQueues) {
        this.numberOfQueues = numberOfQueues;
    }
    public RandomVariate getServiceTimeGenerator() {
        return serviceTimeGenerator;
    }
    public void setServiceTimeGenerator(RandomVariate serviceTimeGenerator) {
        this.serviceTimeGenerator = serviceTimeGenerator;
    }
    public int[] getNumberInQueue() {
        return numberInQueue.clone();
    }
    public int getNumberInQueue(int i) {
        return numberInQueue[i];
    }
    public int[] getNumberAvailableServers() {
        return numberAvailableServers.clone();
    }
    public int getNumberAvailableServers(int i) {
        return numberAvailableServers[i];
    }
    public int getNumberServed() {
        return numberServed;
    }
    public static double sumArray(double[] p) {
        double sum = 0.0;
        for (int i=0; i<p.length; i++) {
            sum += p[i];
        }
        return sum;
    }
}
}

```

Toll Plaza DES Executable Class in Simkit

```

package mmk_11;
import java.text.DecimalFormat;
import simkit.Schedule;
import simkit.random.RandomVariate;
import simkit.random.RandomVariateFactory;
import simkit.stat.MultipleSimpleStatsTally;
import simkit.stat.MultipleSimpleStatsTimeVarying;
import simkit.stat.SimpleStatsTally;
import simkit.stat.SimpleStatsTimeVarying;

public class RunDesTollPlaza {
    public static void main(String[] args) {
        RandomVariate interarrivalTime =
            RandomVariateFactory.getInstance("Exponential," 0.025);
        ArrivalProcess arrivalProcess = new ArrivalProcess(interarrivalTime);
        RandomVariate serviceTime =
            RandomVariateFactory.getInstance("Exponential," 0.25);
        int numberOfQueues = 15;
        DecimalFormat df = new DecimalFormat(" 0.000;-0.000");
        DesTollPlaza tollPlaza = new DesTollPlaza(numberOfQueues, serviceTime);
        arrivalProcess.addSimEventListener(tollPlaza);
        MultipleSimpleStatsTimeVarying niqStat=new
        MultipleSimpleStatsTimeVarying("numberInQueue");
        tollPlaza.addPropertyChangeListener(niqStat);
        SimpleStatsTimeVarying("numberAvailableServers");
        MultipleSimpleStatsTimeVarying nasStat = new
        MultipleSimpleStatsTimeVarying("numberAvailableServers");
        tollPlaza.addPropertyChangeListener(nasStat);
        SimpleStatsTally("numberInSystem");
    }
}

```

```

MultipleSimpleStatsTally numberInSystemOuterStat = new
MultipleSimpleStatsTally("numberInSystem");
MultipleSimpleStatsTally timeInSystemOuterStat = new
MultipleSimpleStatsTally("timeInSystem");
SimpleStatsTally niqReplStat = new SimpleStatsTally("niqR");
SimpleStatsTally nasReplStat = new SimpleStatsTally("nasR");
System.out.println();
System.out.println(arrivalProcess);
System.out.println(tollPlaza);
Schedule.setVerbose(true);
Schedule.stopAtTime(300);
int nreplications = 50;
long start = System.currentTimeMillis();
for (int replication = 0; replication < nreplications; ++replication) {
    Schedule.reset();
    niqStat.reset();
    nasStat.reset();
    Schedule.startSimulation();
    tollPlaza.niqStatMean = DesTollPlaza.sumArray(niqStat.getAllMean());
    tollPlaza.nasStatMean = DesTollPlaza.sumArray(nasStat.getAllMean());
    niqReplStat.newObservation(tollPlaza.niqStatMean);
    nasReplStat.newObservation(tollPlaza.nasStatMean);
    numberInSystemOuterStat.newObservation(tollPlaza.getNumberOfQueues() -
        tollPlaza.nasStatMean + tollPlaza.niqStatMean);

    timeInSystemOuterStat.newObservation(numberInSystemOuterStat.getMean()/(arrivalPro
    cess.getNumberOfArrivals() / Schedule.getSimTime()));
}
long end = System.currentTimeMillis();
System.out.println("Run took " + (end - start) + " milliseconds");
System.out.println("Simulation Ended at time " + Schedule.getSimTime());
System.out.println("Number of Queues in the System : " + numberOfQueues);
System.out.println("There have been " + arrivalProcess.getNumberOfArrivals() + "
arrivals.");
System.out.println("There have been " + tollPlaza.getNumberOfServed() + " customer
served");
System.out.println("Average Number in Queue " +
df.format(tollPlaza.niqStatMean));
System.out.println("Average Utilization" + df.format(1.0 - tollPlaza.nasStatMean/
tollPlaza.getNumberOfQueues()));
System.out.println("Average Number of Customer in System" +
df.format(tollPlaza.niqStatMean + (tollPlaza.getNumberOfQueues() -
tollPlaza.nasStatMean)));
System.out.println("Average Arrival Rate" +
df.format(arrivalProcess.getNumberOfArrivals() / Schedule.getSimTime()));
System.out.println("Average Time in System" +
df.format((tollPlaza.niqStatMean +
(tollPlaza.getNumberOfQueues() - tollPlaza.nasStatMean)) /
(arrivalProcess.getNumberOfArrivals() / Schedule.getSimTime())) + " min" + "\n");

System.out.println("Number of Replications " + (nreplications));
System.out.println("Replications' Mean of Average Number in Queue" +
df.format(niqReplStat.getMean()));
System.out.println("Replications' Mean of Average Number of Customer in System" +
df.format(niqReplStat.getMean() + (tollPlaza.getNumberOfQueues() -
nasReplStat.getMean())));
System.out.println(numberInSystemOuterStat);
System.out.println(timeInSystemOuterStat);
}
}

```

Toll Plaza DTS Program in Simkit

```

package mmk_11;
import java.text.DecimalFormat;
import java.util.Arrays;
import simkit.SimEntityBase;
import simkit.random.DiscreteRandomVariate;
import simkit.random.RandomNumber;

```

```

import simkit.random.RandomNumberFactory;
import simkit.random.RandomVariateFactory;
import simkit.stat.MultipleSimpleStatsTally;
import simkit.stat.SimpleStatsTally;
public class DtsTollPlazaBatch extends SimEntityBase {
    static int numberOfArrivals;
    static int[] numberInQueue;
    static int[] numberAvailableServers;
    static int[] niqTemp;
    static double[] queueInteg;
    static double[] availableServerInteg;
    static double sumQueueInteg;
    static double sumAvailableServerInteg;
    static double sumAllNiqMean;
    static double sumAllNasMean;
    static double replSumAllNiqMean = 0.0;
    static double replSumAllNasMean = 0.0;
    static double numberinsystem;
    static double timeinsystem;
    static int numberServed;
    static double Ps;
    static int[] numberServers;
    static int numberOfQueues = 16;
    static int numberOfCustomersEndServices;
    public static void clear() {
        numberOfArrivals = 0;
        Arrays.fill(numberInQueue, 0);
        Arrays.fill(numberAvailableServers, 1);
        Arrays.fill(niqTemp, 0);
        Arrays.fill(queueInteg, 0.0);
        Arrays.fill(availableServerInteg, 0.0);
        sumQueueInteg = 0.0;
        sumAvailableServerInteg = 0.0;
        numberServed = 0;
        sumAllNiqMean = 0.0;
        sumAllNasMean = 0.0;
        numberOfCustomersEndServices = 0;
    }
    public static void main(String[] args) {
        double meanInterarrivalTime = 0.025;
        double meanServiceTime = 0.25;
        double dt = 0.01;
        double stopTime = 300;
        double nreplications = 50;
        double arrivalTSRate = dt * (1 / meanInterarrivalTime);
        Ps = dt / meanServiceTime;
        if (Ps > 1.0){
            System.out.println("**WARNING*:Probability of service (Ps) is greater than 1");
        }
        numberInQueue = new int[numberOfQueues];
        numberAvailableServers = new int[numberOfQueues];
        niqTemp = new int[numberOfQueues];
        queueInteg = new double[numberOfQueues];
        availableServerInteg = new double[numberOfQueues];
        RandomNumber rng = RandomNumberFactory.getInstance();
        DiscreteRandomVariate poissonV =
            RandomVariateFactory.getDiscreteRandomVariateInstance("Poisson," arrivalTSRate);
        SimpleStatsTally ninsystem = new SimpleStatsTally("nis");
        SimpleStatsTally tinsystem = new SimpleStatsTally("tis");
        MultipleSimpleStatsTally niq = new MultipleSimpleStatsTally("numberInQueue");
        MultipleSimpleStatsTally nas = new MultipleSimpleStatsTally("numberAvailableServers");
        SimpleStatsTally numberArrivalsStat = new SimpleStatsTally("numberArrivals");
        SimpleStatsTally numberServedStat = new SimpleStatsTally("numberServed");
        DecimalFormat df = new DecimalFormat(" 0.000;-0.000");
        long start = System.currentTimeMillis();
        for (int replication = 0; replication < nreplications; ++replication) {
            DtsTollPlazaBatch.clear();
            for (double t = 0.0; t <= stopTime; t = t + dt) {
                for (int i = 0; i < numberOfQueues; ++i) {

```

```

        niq.newObservation(numberInQueue[i], i);
        nas.newObservation(numberAvailableServers[i], i);
        queueInteg[i] += numberInQueue[i] * dt;
        availableServerInteg[i] += numberAvailableServers[i] * dt;
    }
    int numberEndServices = 0;
    for (int i = 0; i < numberOfQueues; ++i) {
        if (numberAvailableServers[i] == 0 && rng.draw() <= Ps) {
            numberAvailableServers[i] = 1;
            numberServed = numberServed + 1;
            numberEndServices += 1;
        }
    }
    numberOfCustomersEndServices += numberEndServices;
    int batchN = poissonV.generateInt();
    for (int r = 0; r < batchN; ++r) {
        int value = Integer.MAX_VALUE;
        int shortest = 0;
        for (int j = 0; j < numberOfQueues; ++j) {
            if (numberInQueue[j] + 1 - numberAvailableServers[j] < value){
                value = numberInQueue[j] + 1 - numberAvailableServers[j];
                shortest = j;
            }
        }
        numberInQueue[shortest] = numberInQueue[shortest] + 1;
    }
    numberOfArrivals = numberOfArrivals + batchN;
    for (int i = 0; i < numberOfQueues; ++i) {
        niqTemp[i] = numberInQueue[i] - numberAvailableServers[i];
        numberInQueue[i] = Math.max(niqTemp[i], 0);
        numberAvailableServers[i] = Math.max(-niqTemp[i], 0);
    }
}
for (int i = 0; i < numberOfQueues; ++i) {
    sumQueueInteg += queueInteg[i] / stopTime;
    sumAvailableServerInteg += availableServerInteg[i] / stopTime;
    numberArrivalsStat.newObservation(numberOfArrivals);
    numberServedStat.newObservation(numberOfQueues);
    numberinsystem = (sumQueueInteg) + (numberOfQueues - (sumAvailableServerInteg));
    ninsystem.newObservation(numberinsystem);
    timeinsystem = ((sumQueueInteg) + (numberOfQueues -
        (sumAvailableServerInteg))) / (numberArrivalsStat.getMean() / stopTime);
    tinsystem.newObservation(timeinsystem);
    double [] queueMeanArray = niq.getAllMean(); //this gets the mean of each queue,
    puts it into array
    double [] nasMeanArray = nas.getAllMean();
    for (int i = 0; i < numberOfQueues; ++i) {
        sumAllNiqMean += queueMeanArray[i];
        sumAllNasMean += nasMeanArray[i];
    }
    replSumAllNiqMean += sumAllNiqMean;
    replSumAllNasMean += sumAllNasMean;
    double meanReplSumAllNiqMean = 0.0;
    double meanReplSumAllNasMean = 0.0;
    meanReplSumAllNiqMean = replSumAllNiqMean/nreplikations;
    meanReplSumAllNasMean = replSumAllNasMean/nreplikations;
    long end = System.currentTimeMillis();
}
}

```

APPENDIX C. CASE II-SCENARIO 8 DESIGN POINTS DATA

This appendix contains 80 design points obtained from the original study 512 design points that are believed to produced highly inaccurate results (Jacobson, 2010). In the original study, these design points were obtained from conducting the Nearly orthogonal Latin Hypercube (NOLH) as part of the design of experiment (Jacobson, 2010). In our study, only six design points (shaded rows) were explored to demonstrate the effects of TAM approach on the results.

LIST OF REFERENCES

- Abdelwahab, H., & Brown, T. (2006). A traffic simulation for a drive-thru pharmacy. *Institute of Transportation Engineering, ITE Journal on the web*. Retrieved from <http://www.ite.org/itejournal/webarticles.asp>
- Abdul Majid, M., Aickelin, U., & Siebers, P. (2009). Comparing simulation output accuracy of discrete event and agent based models: a quantitative approach. *Proceedings of the 2009 Summer Computer Simulation Conference*. Retrieved from <http://arxiv.org/pdf/1001.2170>.
- Abou El-Ata, M., & Hariri, A. (1992). The M/M/C/N queue with balking and reneging. *Computers and Operations Research*, 19, 713–716.
- Ahner, D., Buss, A., & Ruck, J. (2006). *ASC-U User/Analyst manual* (Technical report). Unpublished manuscript. Retrieved from <http://diana.nps.edu>.
- Ahner, D., Buss, A., & Ruck, J. (2006). Assignment scheduling capability for unmanned aerial vehicles- a discrete event simulation with optimization in the loop approach to solving a scheduling problem. *Proceedings of the 2006 Winter Simulation Conference*, 1349–1356. Retrieved from <http://dodreports.com/pdf/ada520432.pdf>
- Alexander, R. (2011). Models, Gaming, and Simulation, SYST–683 and OR–649 Class Notes. Retrieved from <http://classweb.gmu.edu/relaxan3/SYST683/archive.htm>.
- Alexopoulos, C., & Seila. (1998). Output data analysis. In *Chapter 7 in handbook of simulation*. New York: John Wiley and sons Inc.
- Alfa, A. (2010). *Queueing theory for telecommunications: discrete time modeling of a single node system*. New York: Springer.
- Al-Hashimi, B. (1995). *The art of simulation using Pspice: analog and digital*. Florida: CRC Press, Inc.
- Alrowaei, A., Buss, A., & Lieberman, S. (2011). The effect of time advance mechanism on simple agent behaviors in combat simulation. *Proceedings of the 2011 Winter Simulation Conference*. (In Press)
- Alt, J., & Lieberman, S. (2009). Agent frameworks for discrete event social simulations. *Proceedings of the 2009 Behavior Representation in Modeling Simulation*. 134—139. Retrieved from <http://brimsconference.org/archives/2010/papers/10-BRIMS-127%20Alt.pdf>

- Alt, J., Jackson, L., & Lieberman, S. (2009). The cultural geography model: an agent based modeling framework for analysis of the impact of culture in irregular warfare. *Proceedings of the 2009 International Conference on Computer and Communication Devices*, 11–17.
- Alt, J., Jakson, L., Hudak, D., & Lieberman, S. (2009). The cultural geography model: evaluating the impact of tactical operational outcomes on a civilian population in an irregular warfare environment. *Journal of Defense Modeling and Simulation*, 6(4), 185–199.
- Alt, J., Lieberman, S., & Alrowaei, A. (2010). Exploring the implications of time in discrete event social simulation. *Proceedings of the 2010 Association for the Advancement of Artificial Intelligence, Spring Symposium*. 2–6. Retrieved from www.aaai.org/ocs/index.php/SSS/SSS10/paper/download/1193/1421
- Amos, J., Galstad, J., & Higgins, M. (2005). *Model of number of tollbooths needed to optimize traffic flow through toll plazas* (technical report). Department of Mathematics, Kansas State University, Kansas.
- Bakeman, R., Quera, V., & Gnisci, A. (2009). Observer agreement for timed-event sequential data: A comparison of time-based and event-based algorithms. *Behavior Research Methods*, 41(1), 137–147.
- Banks, J., Carson, J., Nelson, B., & Nicol, D. (2005). *Discrete-event system simulation*. (4th ed.). New Jersey: Pearson Prentice Hall.
- Barton, P., & Tobias, A. (2000). Discrete quantity approach to continuous simulation modeling. *Journal of the Operational Research Society*, 51, 485–489.
- Blais, C., Stork, K., Upton, S., Eaton, J., Hoffman, M., & Rollins, S. (2010). *Team 7: data farming to support model validation of the BTRA-BC Battle Engine (BBE)*. Monterey: SEED Center - IDFW 20, Retrieved from <http://harvest.nps.edu/>
- Bletscher, J. (2008). *Ground warfare and troop moral: A system dynamics approach*. Retrieved from <http://ese.wustl.edu/>
- Bolduc, J. (2002). *Mapping continuous-time formalisms onto the devs formalism: Analysis and implementation of strategies* (Thesis proposal). McGill University, Montreal, Canada.
- Brailsford, S., & Hilton, N. (2001). A comparison of discrete event simulation and system dynamics for modeling healthcare systems. *Proceeding of the 2000 Operational Research Applied to Health Services*, 40(2), 18–39. Glasgow, United Kingdom.

- Buss, A. (1996). Modeling with event graph. *Proceedings of the 1996 Winter Simulation Conference*, 153–160. Retrieved from <http://www.informs-sim.org/wsc96papers/020.pdf>
- Buss, A. (2000). Component-based simulation modeling. *Proceeding of the 2000 Winter Simulation Conference*, 964–971. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.807&rep=rep1&type=pdf>
- Buss, A. (2001). Discrete event programming with Simkit. *Simulation News Europe*, 32(33), 15–25.
- Buss, A. (2002). Component based simulation modeling with Simkit. *Proceeding of the 2002 Winter Simulation Conference*, 243–249.
- Buss, A. (2011a). *Discrete event simulation modeling, OA 3302* (Class notes). MOVES Institute, Naval Postgraduate School
- Buss, A. (2011b). *Simkit simulation software package*. Retrieved from <http://diana.nps.edu/Simkit>
- Buss, A., & Sanchez, P. J. (2002). Modeling very large systems: building complex models with LEGOs (Listener Event Graph Objects). *Proceeding of the 34th Winter Simulation Conference*, 732–737.
- Buss, A., & Sanchez, P. (2005). Simple movement and detection in discrete event simulation. *2005 Winter Simulation Conference*, 992–1000.
- Buss, A., & Ahner, D. (2006). Dynamic allocation of fires and sensors (DAFS): a low-resolution simulation for rapid modeling. *Proceeding of the 2006 Winter Simulation Conference*, 1357–1364.
- Buss, A., & Alrowaei, A. (2010). A comparison of the accuracy of discrete event and discrete time. *2010 Winter Simulation Conference*, 1468–1477.
- Caldwell, B., Hartman, J., Parry, S., Washburn, A., & Youngren, M. (2000). *Aggregated combat model* (Technical report). Naval Postgraduate School.
- Carson, J. (2004). Introduction to modeling and simulation. *Proceedings of the 36th Winter Simulation Conference*, 9–16.
- Cassandras, C., & Lafortune, S. (2008). *Introduction to discrete event systems*. New York: Springer Science.

- Ceballos, G., & Curtis, O. (2008). *Queue analysis at toll and parking exit plazas: a comparison between multi-server queueing models and traffic simulation*. Retrieved from <http://cgi.ptv.de/download/traffic/library/VISSIMQueueAnalysis.pdf>
- Cellier, F., & Kofman, E. (2006). *Continuous system simulation*. New York: Springer Science.
- Sanchez, S., & Lucas, T. (2002). Exploring the world of agent-based simulation: simple models, complex analyses. *Proceeding of the 2002 Winter Simulation Conference*. 116–126.
- Clarke, J. (2005). *Simulation modeling & analysis with Arena®: A business student's primer*. Wilfrid Laurier University: Creative Commons License.
- Corwin, I., Ganatra, S., & Rozenblyum, N. (2005). A single-car interaction model of traffic for highway toll plaza. *The UMAP Journal*, 26(223), 299–315.
- Davidsson, P. (2000). *Multi-agent based simulation: beyond social simulation*. Retrieved from <http://jasss.soc.surrey.ac.uk/5/1/7.html>
- Davis, P. (1995). Distributed interactive simulation in the evolution of dod warefare modeling and simulation. *Proceeding of the Institute of Electrical and Electronics Engineers*, 83(1), 1138–1155.
- Delaney, W., & Vaccari, E. (1989). *Dynamic models and discrete event simulation*. New York and Basel: Marcel Dekker, Inc.
- Deo, N. (2006). *System simulation with digital computer* (Eastern Economy Edition ed.). New Jersey: Prentice-Hall.
- Devenish, B., & Thomson, D. (2006). *The effect of time-step size on particle pair seperation in kinematic simulation*. Retrieved from http://www.metoffice.gov.uk/publications/HCTN/HCTN_65.pdf
- Dewar, J., Gill Gillogly, J., & Juncosa, M. (1996). Non-monotonicity, chaos and combat models. *Military Operations Research*, 2(2), 37–49.
- Dubiel, B., & Tsimhoni, O. (2005). Integrating agent based modeling into a discrete event simulation. *Proceedings of the 2005 Winter Simulation Conference*, 1029–1037.
- Eberly, D. (2008). *Stability analysis for systems of differential equations*, (Technical notes), Retrieved from <http://www.geometrictools.com/Documentation/StabilityAnalysis.pdf>

- Elkosantini, S., & Gien, D. (2007). *Human behavior and social network simulation: Fuzzy sets/logic and agents-based approach*. Norfolk, USA: Agent-Directed Simulation. Retrieved from <http://www.eng.auburn.edu/SCS-TM/P12.pdf>
- Fakheri, A., & Fazel, F. (2007). Determination of state probabilities for M/M/s queueing models. *Proceedings of the Decision Sciences Institute Conference*, 3681–3686.
- Fazel, F., & Fakheri, A. (2008). An approximate method for staffing in M/M/S queueing systems. *Proceedings of the Southeast Decision Sciences Institute Conference*, 868–875.
- Ferris, T. (2008). *Modeling methodologies for representing urban cultural geographies in stability operations* (master's thesis). Naval Postgraduate School.
- Fishman, G. (2001). *Discrete-event simulation: Modeling, programming and analysis*. New York: Springer-Verlag.
- Fishwick, P. (2007). *Handbook of dynamic system modeling*. Boca Raton, FL: Chapman & Hall/CRC.
- Galluscio, A., Douglass, J., Malloy, B., & Turner, A. (1995). A comparison of two methods for advancing time in parallel discrete event simulation. *Proceedings of the 1995 Winter Simulation Conference*, 650–657. Retrieved from <http://dl.acm.org/citation.cfm?id=224401.224703>
- Giambiasi, N., & Carmona, J. (2006). Generalized discrete event abstraction of continuous systems: GDEVs formalism. *Simulation Modelling Practice and Theory*, 14, 47–70.
- Golub, G., & Ortega, J. (1992). *Scientific computing and differential equations*. San Diego, CA: Academic Press.
- Grimes, J. (2005). *Modeling sound as a non-lethal weapon in the combatxxi simulation model* (master's thesis). Naval Postgraduate School.
- Gross, D., & Harris, C. (1998). *Fundamentals of queueing theory* (3rd ed.). New York: John Wiley & Sons.
- Haight, F. A. (1957). Queueing with balking. *Biometrika*, 44(1), 360–369. Retrieved from <http://www.jstor.org/stable/2333300>
- Hairer, E., & Wanner, G. (2010). *Solving ordinary differential equations ii, stiff and differential-algebraic problems*. New York: Springer.
- Hall, R. (1991). *Queueing methods for service and manufacturing*. New Jersey: Prentice-Hall.

- Hartman, J., Parry, S., & Caldwell, W. (1992). *Airland combat models i: high resolution combat modeling* (master's thesis). Naval Postgraduate School.
- Hattaway, S. (2008). *Adapting the dynamic allocation of fires and sensors (DAFS) model for use in maritime combat analysis* (master's thesis). Naval Postgraduate School.
- Havens, M. (2002). *Dynamic allocation of fires and sensors* (master's thesis). Naval Postgraduate School.
- Hill, R., Miller, J., & McIntyre, G. (2001). Applications of discrete event simulation modeling to military problems. In J. S. B. A. Peters (Ed.), *Proceedings of the 2001 Winter Simulation Conference*, 780–787. Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00977367>
- Hillestad, R., Owen, J., & Blumenthal, D. (1995). Experiment in variable resolution combat modeling. *Naval Research Logistics*, 42(2), 209–232.
- Hillier, F., & Lieberman, G. (2005). *Introduction to operations research* (8th ed.). New York: McGraw–Hill.
- Hinkson, D. (2010). *Supporting marine corps enhanced company operations: A quantitative analysis* (master's thesis). Naval Postgraduate School.
- Hoffoss, D. (2005). The booth tolls for thee. *2005 MCM*. University of San Diego, Retrieved from <http://www.math.duke.edu/news/student/mcm2005.pdf>
- Huang, T., Duo, L., & Li, L. (2010). Adaptive time-stepping particle fluid motion simulation. *Proceedings of the International Conference on Computer Application and System Modeling*, 12(1), 117–121. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5622141
- Jacobson, K. (2010). *The littoral combat ship (LCS) surface warfare (SUW) module: Determining the best mix of surface-to-surface and air-to-surface missiles* (master's thesis). Naval Postgraduate School.
- Joukhadar, A., Laugier, C., & Alpes, I. (1996). Adaptive time step for fast converging dynamic simulation system. *Proceeding of the IEEE-RSJ International Conference on Intelligent Robots and Systems*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.7259&rep=rep1&type=pdf>
- Kaminski, P. (1996). *Defense modeling and simulation: Speech*, Vol. 11 Number 64. Retrieved from <http://www.defense.gov/speeches/speech.aspx?speechid=997>
- Kiesling, T. (2005). *Approximate time-parallel simulation*. (Ph.D. dissertation), Universität der Bundeswehr, München. Retrieved from <http://ub.unibw-muenchen.de/dissertationen/ediss/kiesling-tobias/inhalt.pdf>

- Kiesling, T., & Krieger, T. (2006). Efficient distributed queuing system simulation. Retrieved from http://webdoc.sub.gwdg.de/ebook/serien/ah/unibw_informatik/2006-01.pdf
- Kleinrock, L. (1975). *Queueing systems-volume 1: Theory*. New York: John Wiley and Sons.
- Klodzinski, J., & Al-Deek, H. (2002a). New methodology for evaluating a toll Plaza's level of service. *ITE*, 72(2), 34–41
- Klodzinski, J., & Al-Deek, H. (2002b). Transferability of a stochastic toll plaza computer model. *81st Transportation Research Board Annual Meeting*. Washington D.C: National Research Council.
- Kofman, E. (2006). A third order discrete event method for continuous system simulation. *Latin American Applied Research*, 36, 101–108. Retrieved from <http://www.scielo.org.ar/pdf/laar/v36n2/v36n2a07.pdf>
- Kokar, M., & Baclawski, K. (2000). Modeling combined time-driven and event-driven dynamic systems. In *Ninth OOPSLA Workshop on Behavioral Semantics*, 112–129. Retrieved from <http://www.ccs.neu.edu/home/kenb/pub/2000/04/public.pdf>
- Kolker, A. (2008). *Queueing analytic theory and discrete events simulation for healthcare: Right application for the right problem*. Children's Hospital of Wisconsin. Retrieved from http://www.iienet.org/uploadedFiles/SHS_Community/Resources/Queueing%20Analytic%20Theory%20and%20Discrete%20Events%20Simulation.pdf
- Kos, S., Hess, & Hess, S. (2006). A simulation method in modeling exploitation factors of Seaport Queueing System. *Pomorstvo*, 20(1), 67–85.
- Krull, C., & Horton, G. (2007). Application of proxels to queueing simulation, in simulation and visualization. Magdeburg, Germany, 299–310. Retrieved from http://www.sim.ovgu.de/sim_media/downloads/publications/krull/simvis07_krull.pdf
- Krull, C., & Horton, G. (2009). Proxel-based simulation: Theory and applications. *Proceedings of the 6th St.Petersburg Workshop on Simulation*. 1–6. Retrieved from http://www.sim.ovgu.de/sim_media/downloads/publications/krull/pws09_krull.pdf
- Lain, H., & Wan, Z. (2007). The computer simulation for queueing system. *Proceedings of the World Academy of Science, Engineering and Technology*, 34. Retrieved from <http://www.waset.org/journals/waset/v34/v34-33.pdf>

- Lalis, V. (2007). *Exploring naval tactics with UAVs in an island complex using agent-based simulation* ,(master's thesis). Naval Postgraduate School.
- Law, A. M., & Kelton, W. D. (2000). *Simulation Modeling and Analysis* (3rd ed.). New York: McGraw-Hill.
- Ledin, J. (2001). *Simulation engineering: Build better embedded system faster*. Gilroy, CA: CMP books.
- Lee, H., & Strawderman, L. (2009). An approximation for the system size of M/G/c queueing systems. *Proceeding for the 2009 Industrial Engineering Research Conference*. Miami, Florida.
- Lin, T., Chang, K., Tian, B., Chu, H., & Huang, P. (2008). Modeling and simulation comparison of two time synchronization protocols. *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*,12, 117–123.
- Lozano, E., Laita, L., & Macias, E. (2004). An accelerated-time simulation of departing passengers' flow in airport terminals. *Mathematics and Computers in Simulation*, 67, 163–172.
- Lucas, T. (2009). *Introduction to joint combat modeling - OA4655* (Class notes). Naval Postgraduate School.
- M&SCO. (2010). *Modeling and simulation coordination office*. Strategic Vision for DoD Modeling and Simulation. Retrieved from <http://www.msco.mil>
- Macal, C. (2010). To agent-based simulation from system dynamics. *Proceeding of the 2010 Winter Simulation Conference*. 371–382. Retrieved from <http://www.informs-sim.org/wsc10papers/034.pdf>
- Marlin, B. (2009). *Ascertaining validity in the abstract realm of PMESII simulation models: an analysis of the peace support operations model (PSOM)* (master's thesis). Naval Postgraduate School.
- Marquez, A. (2010). *Dynamic modeling for supply chain management: Dealing with front-end, back-end and integration issues*. New York: Springer.
- Matsopoulos, A. (2007). *Little by little does the trick: Design and construction of a discrete event agent-based simulation framework* (master's thesis). Naval Postgraduate School.
- McClanahan, T., Feinberg, J., Goalwin, P., & Blemberg, P. (2001, June). *Human and organizational behavior modeling (HOBM)*, Retrieved from <http://www.au.af.mil/au/awc/awcgate/dms/hobm-tech-assessment.pdf>

- McIntosh, G. C. (2009). *MANA-V (Map Aware Non-Uniform Automata-Vector)*. Supplementary Manual, Provided with MANA 5.0.1 Software Package.
- McIntosh, G., Galligan, D., Anderson, M., & Lauren, M. (2007). *MANA (Map Aware Non-Uniform Automata) version 4 user manual*. Operations Analysis Section, Defence Technology Agency.
- Mielke, R. (1999). Applications for enterprise simulation. *Proceedings of the 1999 Winter Simulation Conference*, 1490–1495. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.2445&rep=rep1&type=pdf>
- Neuts, M. (1973). The single server queue in discrete time-numerical analysis I. *Naval Research Logistics*, 20(2), 297–304.
- Northrop. (2007). *Pythagoras user manual version 2.0*. Northrop Grumman Space & Mission Systems Corp.
- Nutaro, J. (2005). Constructing multi-point discrete event integration schemes. *Proceeding of the 2005 Winter Simulation Conference*, New Jersey, 276–273.
- Odoni, A. R. (2004). *Queueing systems: lecture 1*, Retrieved from <http://ocw.mit.edu/courses/civil-and-environmental-engineering>
- Ossimitz, G., & Mroczek, M. (2008). The basic of system dynamics: Discrete vs. continuous modeling of time. *Proceeding of the International System Dynamics Conference*. Athens, Greece. Retrieved from http://www.uni-klu.ac.at/gossimit/pap/dc_time.pdf
- Overeinder, B. (2000). Distributed event-driven simulation: Scheduling strategies and resource management, *ASCI* (Ph.D. dissertation), University of Amsterdam, Amsterdam, Netherlands. Retrieved from <http://www.science.uva.nl/research/scs/papers/archive/Overeinder2000a.pdf>
- Ozgun, O., & Barlas, Y. (2009). Discrete vs. continuous simulation: When does it matter? *Proceeding of the 27th International Conference of The System Dynamic Society*. NM. Retrieved from <http://www.systemdynamics.org/conferences/2009/proceed/papers/P1199.pdf>
- Paoli, F., & Tisato, F. (1996). On the complementary nature of event-driven and time-driven models. *Pergamon, Control Eng. Practice*, 4(6), 847–854.
- Parkman, J., & Hanley, N. (2008). *Peace support operations model V2 functional specifications*. (Technical manual), Farmborough, UK.

- Peitso, L. (2011). Communications latency hiding for distributed SoS. *Proceeding of the 2011 6th International Conference on System of System Engineering (SoSE)*, Albuquerque, New Mexico: IEEE, 113–118.
- Pew, R., & Mavor, A. (1998). *Modeling human and organizational behavior: application to military simulations*. New York: National Academy Press.
- Qiu, F., & Hu, X. (2010). Exploiting spatial-temporal heterogeneity for agent-based simulation of pedestrian crowd behavior. In H. D. B., *Activity-Based Modeling and Simulation*. CNRS Interdisciplinary Seminar. Retrieved from http://www.msh-clermont.fr/IMG/pdf/06-QIU-HU_107-127_.pdf
- Reed, J. (1980). Computer Simulation: A tool to teach queueing theory. *Experiential Learning Enters the Eighties*, 7, 63–66. Retrieved from <http://sbaweb.wayne.edu/~absel/bkl/vol07/07at.pdf>
- Ross, S. (1994). *A first course in probability* (4th ed.). New Jersey: Prentice Hall.
- Ross, S. (2003). *Introduction to probability models* (8th ed.). San Diego: Academic Press.
- RTO. (2009). *Human behavior representation in constructive simulation*. Research and Technology Organisation (RTO) & North Atlantic Treaty Organisation (NATO). Retrieved from <http://www.rto.nato.int/>
- Russell, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach*, New Jersey: Prentice Hall.
- Rusu, P. (1988). *Two-dimensional combat modeling with partial differential equations*. Oak Ridge National Laboratory–Martin Marietta. Retrieved from <http://www.ornl.gov/info/reports/1990/3445603290418.pdf>
- Sanchez, P. (2006). As simple as possible, but not simpler: A gentle introduction to simulation modeling. *Proceeding of the 2006 Winter Simulation Conference*, 2–10.
- Sanchez, P. (2007). Fundamentals of simulation modeling. *Proceeding of the 2007 Winter Simulation Conference*, 54–62.
- Savage, E., Schruben, L., & Yucsan, E. (2005). On the generality of event-graph models. *INFORMS Journal on Computing*, 17(1), 3–9.
- Schruben, L. (1983). Simulation modeling with event graphs. *Communications of the ACM*, 26, 957–963.
- Schruben, L., & Yucsan, E. (1993). Complexity of simulation models: A graph theoretic approach. *Proceedings of the 1993 Winter Simulation Conference*, 641–649.

- Schruben, L., & Yucesan, E. (1993). Modeling paradigms for discrete event simulation. *Operations Research Letters*, 13(5), 265–275.
- Seck, M., Frydman, C., Giambiasi, N., Oren, T., & Yilmaz, L. (2005). Use of a dynamic personality filter in discrete event simulation of human behavior under stress and fatigue. *Proceeding of the 1st International Conference on Augmented Cognition*. Retrieved from <http://www.site.uottawa.ca/~oren/pubs-pres/2005/pub-0506-augCog-filter.pdf>
- Seck, M., Giambiasi, N., Frydman, C., & Baati, L. (2005). *Devs for human behavior modeling in CGFs*, Retrieved from <http://www.scs.org/pubs/jdms/vol4num3/Seck1.pdf>
- SEED Center for Data Farming. (2011), Retrieved from <http://harvest.nps.edu/>
- Seitz, T. (2008). *Representing Urban Cultural Geography in Stabilization Operation: Analysis of a Social Network Representation in Pythagoras* (master's thesis), Naval Postgraduate School.
- Sloot, P. (2003). *Lectures of Prof. Dr. Sloot: introduction to simulation and modeling, 1.4 model execution: Event driven versus time driven*. Retrieved from <http://www.wszib.edu.pl/>
- Smith, R. (1998). Essential techniques for military modeling & simulation. *Proceeding to 1998 Winter Simulation Conference*, 805–812.
- Sokolowski, J. A., & Banks, C. M. (2009). *Principles of modeling and simulation: A multidisciplinary approach*. New Jersey: Wiley.
- Stewart, R. (2004). *Simulation: The practice of model development and use*. New Jersey: John Wiley & Sons, Ltd.
- Straver, M., Vincent, E., & Fournier, P. (2006). *Experiences with the MANA simulation tool*. Valcartier: DRDC Valcartier Operationa; Research Team R&D, Valcartier, Canada. Retrieved from <http://pubs.drdc.gc.ca/PDFS/unc51/p525762.pdf>
- Surmit, G. (2004). *Algorithm design for networked information technology systems*. New York: Springer-Verlag.
- Sweetser, A. (1999). A comparison of system dynamics and discrete event simulation. *17th International Conference of the System Dynamics Society*. Retrieved from <http://www.systemdynamics.org/conferences/1999/PAPERS/PARA78.PDF>
- Tako, A., & Robinson, S. (2008). *Model building in system dynamics and discrete-event simulation: A quantitative comparison*. UK: Univesity of Warwick. Retrieved from <http://www.systemdynamics.org/conferences/2008/proceed/papers/TAKO286.pdf>

- Tako, A., & Robinson, S. (2009). Comparing discrete-event simulation and system dynamics: users' perceptions. *Journal of the Operational Research Society*, 60, 296–312.
- Tan, B. (2007). *A study to model human behavior in discrete event simulation using*, (master's thesis). Naval Postgraduate School.
- Todd, J. (2005). *Duke student math aims to alleviate tollbooth lines*. Retrieved from <http://www.dukenews.duke.edu/2005/06/tollbooths.html>
- TRAC-MRY. (2011). *Cultural Geography CG model scenario development*. Monterey: TRAC-MRY, Retrieved from <http://www.nps.edu/research/TRAC/about.html>
- Turing, A. (1950). Computing machinery and intelligence. *Mind*, 59(1), 433–460.
- U.S.Army. (2008). *FM 3-7 stability operations*. Washington D.C: Department of the Army. Retrieved from <http://usacac.army.mil/cac2/Repository/FM307/FM3-07.pdf>
- Wang, Q. (2005). For whom the booth tolls. *2005 MCM contest*, University of San Diego, 35, 1–21.
- Warhola, P. (1997). *An analysis of alternative methods to conduct high-resolution activities in a variable-resolution simulation* (master's thesis). Naval Postgraduate School.
- Washburn, A. (2002). *Search and detection* (4th ed.). Military Applications Society of Institute for Operations Research and the Management Sciences. Hanover, Maryland: INFORM.
- Washburn, A., & Kress, M. (2009). *Combat modeling, international series in operations research & management science*. New York: Springer.
- Wedemann, R., Barbosa, V., & Donangelo, R. (1999). Defeasible time-stepping. *Parallel Computing*, 25(4), 461–489.
- Willig, A. (1999). *WS0203 Curricula*. Technical University Berlin Retrieved from <http://www.tkn.tu-berlin.de/curricula/ws0203/ue-kn/qt.pdf>
- Wu, Y., & Gong, W. (2001). Time Stepped Simulation of Queueing Systems. *SPIE*, 4367, 262–269.
- Yi, Y., & Shakkottai, S. (2007). *FluNet: A hybrid Internet simulation/emulation environment for a fast queue regime*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.5330&rep>

- Yildiz, B. (2009). *Exploration of the use of unmanned aerial vehicles along with other assets to enhance border protection*, (master's thesis). Naval Postgraduate School.
- Zacharias, G., MacMillan, J., & Van Hemel, S. (2008). *Behavioral modeling and simulation: From individuals to societies*. Washington, DC: National Academy Press.
- Zaft, G., & Zeigler, B. (2002). Discrete event simulation and social science: The xeriscape artificial society. *Proceeding of the 8th World Multiconference on Systemics, Cybernetic and Informatics*, 6, 1–6. Retrieved from <http://www.zaft.org/gordon/XeriScape/SCI2002.pdf>
- Zeigler, B., & Lee, J. (1998). Theory of quantized systems: Formal basis for DEVS/HLA distributed simulation environment. *Proceedings of the Society of Photographic Instrumentation Engineers*, 3369(1), 49–58.
- Zeigler, B., Praehofer, H., & Kim, T. (2000). *Theory of modeling and simulation: Integration discrete event and continuous complex dynamic system* (2nd ed.). San Diego, CA: Academic Press.
- Zhang, Y., Yue, D., & Yue, W. (2005). Analysis of an M/M/1/N queue with balking, reneging and server vacations. *International Symposium on OR and Its Applications 2005*. Retrieved from <http://www.aporc.org/LNOR/5/ISORA2005F04.pdf>

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Dr. Arnold H. Buss
MOVES Institute
Naval Postgraduate School
Monterey, California
4. Dr. Rudolph P. Darken
Department of Computer Science
Naval Postgraduate School
Monterey, California
5. Dr. Christian J. Darken
Department of Computer Science
Naval Postgraduate School
Monterey, California
6. Dr. Paul Sanchez
Department of Operations Research
Naval Postgraduate School
Monterey, California
7. Mr. Arijit Das
Department of Computer Science
Naval Postgraduate School
Monterey, California
8. Mr. Kirk Strok
20512 NE 258th Ave.
Battle Ground
Washington, WA 98604
kastork@mac.com
9. Dr. Thomas Lucas
Department of Operations Research
Naval Postgraduate School
Monterey, California

10. Dr. Imre Balogh
MOVES Institute
Naval Postgraduate School
Monterey, California
11. Mr. Curtis Blais
MOVES Institute
Naval Postgraduate School
Monterey, California