

SIMULATING MULTIPLE SYSTEMS OF SYSTEMS USING THE HIGH LEVEL ARCHITECTURE

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SCHOOL OF COMPUTER SCIENCE

UNIVERSITY OF ADELAIDE

By

Anthony Cramp

October 26, 2009

Contents

Abstract	xiii
Declaration	xvii
Acknowledgment	xix
1 Introduction	1
1.1 Computer Simulation	4
1.2 Problem Simulation Multiple System of Systems	11
1.3 Local Data Filtering	15
1.4 Data Distribution Management	16
1.5 Federation Community	18
1.6 Background Research	20
1.6.1 High Level Architecture	20
1.6.2 Problem	23
1.6.3 Local Data Filtering	25
1.6.4 Data Distribution Management	25
1.6.5 Federation Community	27
1.6.6 Summary	28
1.7 Thesis Overview	29
2 Methods	33
2.1 Federation Contrived	37
2.2 Class Per System	46
2.3 Local Data Filtering	51
2.4 Data Distribution Management	59
2.5 Federation Community	68
2.6 Summary	77
3 Metrics	81
3.1 Experiment Scenario	85
3.2 Federation Metrics	88
3.3 Federation Community Architecture	92
3.4 Experiment Resources	97
3.4.1 Metric Federate	98
3.4.2 Observer Federate	99
3.4.3 Distributed Federate Proxy	99
3.4.4 Execution Manager	100
3.4.5 RTI	101
3.4.6 Ethereal	102
3.4.7 Python/Software Testing Automation Framework	102
3.5 Summary of Experimentation Setup	102
3.6 Metric: Latency	105
3.6.1 UML 2	106
3.6.2 Description of the Latency Sequence Diagram	107
3.6.3 Latency results for intra-system data	109

3.6.4	Latency results for inter-system data	113
3.6.5	Conclusions from latency results	117
3.7	Metric: Throughput	118
3.7.1	Throughput results for intra-system data	121
3.7.2	Throughput results for inter-system data	123
3.7.3	Conclusions from throughput results	126
3.8	Metric: Intra-system data communication effectiveness	126
3.8.1	Results	128
3.8.2	Conclusions for intra-system data communication effectiveness	132
3.9	Conclusions	133
4	Case Study: Virtual Election	137
4.1	Overview	139
4.2	Federation Virtual Election	142
4.2.1	Federation Virtual Election Federation Object Model	144
4.2.2	Congruence with Federation Contrived	148
4.3	Simulation	149
4.3.1	Overview	150
4.3.2	Initialisation	152
4.3.3	Utility Classes	156
4.3.4	Vote Generation	158
4.3.5	BallotStation Registration	160
4.3.6	Vote Tallying	164
4.3.7	Electorate Registration	171
4.3.8	Summary	172
4.4	Local Data Filtering Federation Virtual Election	173
4.4.1	Initialisation	175
4.4.2	Vote Generation	178
4.4.3	BallotStation Registration	179
4.4.4	Vote Tallying	182
4.4.5	Electorate registration	189
4.4.6	Summary	189
4.5	Data Distribution Management Federation Virtual Election	192
4.5.1	Initialisation	194
4.5.2	Vote Generation	199
4.5.3	BallotStation Registration	199
4.5.4	Vote Tallying	200
4.5.5	Electorate Registration	203
4.5.6	Summary	203
4.6	Federation Community Federation Virtual Election	206
4.6.1	Simulation	208
4.6.2	Summary	210
4.7	Conclusion	211
5	Future Work	215
5.1	Quantitative Analysis	215
5.2	Other HLA services	216
5.3	Different inter-federation communication architecture	217
5.4	Testing on other network architectures	218
5.5	Automatic selection of method used	218
5.6	Middleware abstraction	219
5.7	Object composition/aggregation for the HLA	219
5.8	Multiple systems of systems in other simulation frameworks	220
5.9	Use of formal methods to describe the solutions	220
5.10	Summary	221

6	Conclusions	223
6.1	Summary of Contributions	228
A	Overview of the High Level Architecture	231
A.1	High Level Architecture	231
A.2	HLA Object Model	233
A.3	HLA Interface Specification	235
A.3.1	Declaration Management	237
A.3.2	Object Management	239
A.3.3	Data Distribution Management	240
A.3.4	Summary	241
A.4	Summary	242
B	RID File Configuration	243
B.1	Latency and Throughput	244
B.1.1	Latency	245
B.1.2	Throughput	247
B.1.3	Conclusion	250
B.2	DDM	250
B.2.1	Performance	250
B.2.2	Intra-system communication effectiveness	255
B.2.3	Conclusion	256
B.3	Conclusion	257
C	Instant Runoff Algorithm	259
C.1	Setup	260
C.2	Algorithm	261
C.3	Analysis	262
C.4	Run-time performance	263
C.5	Distributed Instant Runoff Algorithm	264
C.6	Distributed Run-time performance	265
C.7	Comparison	267
C.8	Summary	268
D	Election Simulation Configuration	269
D.1	Example Election Configuration	272
D.2	Summary	274
	Bibliography	279

List of Figures

1.1	A ship provides an example of a system of systems architecture.	2
1.2	Inter- and intra-ship data flows in a multiple ship environment.	3
1.3	UML collaboration diagram for simulating a ship.	5
1.4	UML collaboration diagram for simulating multiple ships.	5
1.5	Component based ship simulation.	6
1.6	Upgraded component based ship simulation.	6
1.7	Distributed simulation components for the ship simulation	8
1.8	HLA federation structure for simulating a ship.	9
1.9	HLA federation structure for simulating multiple ships.	13
1.10	Local Data Filtering solution to simulating multiple ships.	16
1.11	Data Distribution Management solution to simulating multiple ships.	17
1.12	Federation Community solution to simulating multiple ships.	19
2.1	Single system of systems.	34
2.2	Federation simulating a single system of systems.	34
2.3	Federation simulating two systems of systems.	36
2.4	The effective multi-system simulated by the federation in Figure 2.3.	37
2.5	The correct multi-system.	38
2.6	Single system of systems communicating data defined in Federation Contrived's FOM.	40
2.7	Sequence diagram of publication and subscription process for a federation simulating a single system of systems.	41
2.8	Sequence diagram for intra-system object class communication in a federation with a single system of systems.	42
2.9	Sequence diagram for intra-system interaction class communication in a federation with a single system of systems.	42
2.10	Sequence diagram of publication and subscription process for federation simulating a two systems of systems.	43
2.11	Sequence diagram for intra-system object class communication in a federation with multiple systems of systems.	45
2.12	Sequence diagram for intra-system interaction class communication in a federation with multiple systems of systems.	45
2.13	Simulating two systems of systems using the Class Per System version of Federation Contrived.	47
2.14	Publication and subscription of intra-system data in the LDF version of Federation Contrived.	53
2.15	Sequence diagram for intra-system object class communication between federates implementing LDF: initialisation and object instance registration/discovery.	55
2.16	Sequence diagram for intra-system object class communication between federates implementing LDF: object instance update/reflection.	56
2.17	Sequence diagram for intra-system object class communication between federates implementing LDF: object instance remove/delete.	57
2.18	Sequence diagram for intra-system interaction class communication between federates implementing LDF.	58
2.19	Routing Space example.	60
2.20	Sequence diagram for intra-system data class communication between federates implementing DDM: region creation.	63
2.21	Region overlap.	63

2.22	Sequence diagram for intra-system data class communication between federates implementing DDM: publication and subscription.	64
2.23	Sequence diagram for intra-system object class communication between federates implementing DDM: object instance registration and discovery.	66
2.24	Sequence diagram for intra-system object class communication between federates implementing DDM: object instance update and reflection.	66
2.25	Sequence diagram for intra-system object class communication between federates implementing DDM: object instance deletion and removal.	67
2.26	Sequence diagram for intra-system interaction class communication between federates implementing DDM: send and receive.	68
2.27	Federation Community simulating two systems of systems.	69
2.28	Publication/Subscription sequence diagram for F^1 in the Federation Community version of Federation Contrived.	70
2.29	Publication/Subscription sequence diagram for F^2 in the Federation Community version of Federation Contrived.	70
2.30	Inter-federation communication.	72
2.31	Sequence diagram of publication/subscription interests for inter-system data classes in Federation Community version of Federation Contrived.	73
2.32	Sequence diagram of registration and discovery of an inter-system object instance in Federation Community version of Federation Contrived.	74
2.33	Sequence diagram of update and reflection of inter-system object instance's attributes in Federation Community version of Federation Contrived.	75
2.34	Sequence diagram of deletion and removal of inter-system object instance in Federation Community version of Federation Contrived.	76
2.35	Sequence diagram of sending and receiving an inter-system interaction class in Federation Community version of Federation Contrived.	76
3.1	Single system of systems.	86
3.2	Federation simulating two systems of systems.	87
3.3	Two systems of systems.	88
3.4	General Federation Community architecture.	93
3.5	Federate Proxy joined to two federations.	93
3.6	Federate Proxy joined to two federations in a WAN environment.	94
3.7	Distributed Federate Proxy.	95
3.8	Hardware resources.	97
3.9	Sequence diagram for calculating interaction latency.	106
3.10	Hardware setup and software assignment for testing Base, LDF, and DDM intra-system latency.	109
3.11	Hardware setup and software assignment for testing Federation Community intra-system latency.	110
3.12	Round Trip Latency vs Payload Size for LocalObject.	110
3.13	Round Trip Latency vs Payload Size for LocalInteraction.	112
3.14	Hardware setup and software assignment for testing Base, LDF, and DDM inter-system latency.	113
3.15	Hardware setup and software assignment for testing Federation Community inter-system latency.	114
3.16	Round Trip Latency vs Payload Size for GlobalObject.	115
3.17	Round Trip Latency vs Payload Size for GlobalInteraction.	116
3.18	Sequence diagram for calculating interaction throughput.	119
3.19	Throughput vs Payload Size for LocalObject.	121
3.20	Throughput vs Payload Size for LocalInteraction.	122
3.21	Throughput vs Payload Size for GlobalObject.	124
3.22	Throughput vs Payload Size for GlobalInteraction.	125
3.23	Setup for DDM/LDF network utilisation tests.	128
3.24	Setup for FC network utilisation tests.	128
3.25	Data received at f_1 via $MP_{f_2 \rightarrow f_1}^{LO}$ vs payload size for LocalObject.	129
3.26	Data received at f_3 via $MP_{f_2 \rightarrow f_3}^{LO}$ vs payload size for LocalObject.	130
3.27	Data received at federate f_1 via $MP_{f_2 \rightarrow f_1}^{LI}$ vs payload size for LocalInteraction.	132
3.28	Data received at federate f_3 via $MP_{f_2 \rightarrow f_3}^{LI}$ vs payload size for LocalInteraction.	133
4.1	Electorate system.	138

4.2	Data flows in an election simulation containing two electorates.	144
4.3	Federation Virtual Election electorate system.	148
A.1	Typical federation diagram.	233
A.2	Communication between federates in a federation.	236
B.1	RID Test: Round Trip Latency vs Payload Size for GlobalObject.	246
B.2	RID Test: Round Trip Latency vs Payload Size for GlobalInteraction.	247
B.3	RID Test: Throughput vs Payload Size for GlobalObject.	248
B.4	RID Test: Throughput vs Payload Size for GlobalInteraction.	249
B.5	RID Test: Round Trip Latency vs Payload Size for LocalObject varying DDM strategy.	251
B.6	RID Test: Round Trip Latency vs Payload Size for LocalInteraction varying DDM strategy.	252
B.7	RID Test: Throughput vs Payload Size for LocalObject varying DDM strategy.	253
B.8	RID Test: Throughput vs Payload Size for LocalInteraction varying DDM strategy.	254
B.9	Amount of intra-system data received at federate f_1	256
B.10	Amount of intra-system data received at federate f_3	257
C.1	Run-time results for single process implementation of the instant runoff algorithm.	263
C.2	Run-time performance of distributed instant runoff algorithm for ten candidates.	267
C.3	Time to run single and distributed versions of the instant runoff algorithm for ten candidates.	268

List of Tables

1.1	Object Class Structure Table for a system of systems ship federation.	9
1.2	Attribute Table for a system of systems ship federation.	10
1.3	Interaction Class Structure Table for a system of systems ship federation.	10
1.4	Parameter Table for a system of systems ship federation.	11
2.1	Object Class Structure Table for Federation Contrived.	39
2.2	Attribute Table for Federation Contrived.	39
2.3	Interaction Class Structure Table for Federation Contrived.	39
2.4	Parameter Table for Federation Contrived.	40
2.5	Object Class Structure Table for Federation Contrived Supporting two systems.	47
2.6	Interaction Class Structure Table for Federation Contrived supporting two systems.	47
2.7	Attribute Table for Federation Contrived supporting two systems.	48
2.8	Parameter Table for Federation Contrived supporting two systems.	48
2.9	Object Class Structure Table for Federation Contrived supporting two systems using inheritance.	49
2.10	Interaction Class Structure Table for Federation Contrived supporting two systems using inheritance.	49
2.11	Attribute Table for Federation Contrived supporting two systems using inheritance.	49
2.12	Parameter Table for Federation Contrived supporting two systems using inheritance.	50
2.13	Example Object Class Structure Table containing a hierarchical intra-system object class.	50
2.14	Example Object Class Structure Table for Federation Contrived supporting two systems using inheritance.	50
2.15	Object Class Structure Table for LDF version of Federation Contrived FOM.	51
2.16	Interaction Class Structure Table for LDF version of Federation Contrived FOM.	52
2.17	Attribute Table for LDF version of Federation Contrived FOM.	52
2.18	Parameter Table for LDF version of Federation Contrived FOM.	52
2.19	Routing Space Table for the DDM version of Federation Contrived.	61
2.20	Object Class Structure Table for the DDM version of Federation Contrived.	61
2.21	Attribute Table for the DDM version of the Federation Contrived FOM.	61
2.22	Interaction Class Structure Table for the DDM version of the Federation Contrived FOM.	62
2.23	Parameter Table for the DDM version of Federation Contrived.	62
3.1	Object Class Structure Table for Federation Metrics.	89
3.2	Attribute Table for Federation Metrics.	89
3.3	Interaction Class Structure Table for Federation Metrics.	90
3.4	Parameter Table for Federation Metrics.	90
3.5	Attribute Table for LDF version of Federation Metrics.	90
3.6	Parameter Table for LDF version of Federation Metrics.	91
3.7	Routing Space Table for DDM version of Federation Metrics.	91
3.8	Attribute Table for DDM version of Federation Metrics.	91
3.9	Interaction Class Structure Table for DDM version of Federation Metrics.	91
3.10	Round Trip Latency data for LocalObject.	111
3.11	Round Trip Latency data for LocalInteraction.	112
3.12	Round Trip Latency data for GlobalObject.	115
3.13	Round Trip Latency data for GlobalInteraction.	117
3.14	Throughput data for LocalObject.	122
3.15	Throughput data for LocalInteraction.	123

3.16	Throughput data for GlobalObject.	124
3.17	Throughput data for GlobalInteraction.	125
3.18	Data received at f_1 via $MP_{f_2 \rightarrow f_1}^{LO}$ vs payload size for LocalObject.	130
3.19	Data received at f_3 via $MP_{f_2 \rightarrow f_3}^{LO}$ vs payload size for LocalObject.	131
3.20	Data received at federate f_1 via $MP_{f_2 \rightarrow f_1}^{LI}$ vs payload size for LocalInteraction.	132
3.21	Data received at federate f_3 via $MP_{f_2 \rightarrow f_3}^{LI}$ vs payload size for LocalInteraction.	132
4.1	Ballot paper.	141
4.2	First round totals.	141
4.3	Second round totals.	141
4.4	Third round totals.	141
4.5	Complex Datatype Table of the Federation Virtual Election FOM.	145
4.6	Object Class Structure Table of the Federation Virtual Election FOM.	145
4.7	Attribute Table of the Federation Virtual Election FOM.	145
4.8	Interaction Class Structure Table of the Federation Virtual Election FOM.	146
4.9	Parameter Table of the Federation Virtual Election FOM.	147
4.10	Attribute Table for the LDF Federation Virtual Election FOM.	173
4.11	Parameter Table for the LDF Federation Virtual Election FOM.	174
4.12	Routing Space Table for the DDM Federation Virtual Election FOM.	193
4.13	Attribute Table for the DDM Federation Virtual Election FOM.	193
4.14	Interaction Class Structure Table for the DDM Federation Virtual Election FOM.	193
4.15	Object Class Structure Table of the DFPC SOM in the FC version of Federation Virtual Election.	207
4.16	Interaction Class Structure Table of the DFPC SOM in the FC version of Federation Virtual Election.	207
A.1	Example class structure tables.	234
A.2	Example attribute and parameter tables.	235
B.1	RID Test: Round Trip Latency vs Payload Size for GlobalObject.	246
B.2	RID Test: Round Trip Latency vs Payload Size for GlobalInteraction.	247
B.3	RID Test: Throughput vs Payload Size for GlobalObject.	248
B.4	RID Test: Throughput vs Payload Size for GlobalInteraction.	249
B.5	RID Test: Round Trip Latency data for DDM testing with LocalObject.	252
B.6	RID Test: Round Trip Latency vs Payload Size for LocalInteraction varying DDM strategy.	253
B.7	RID Test: Throughput data for LocalObject varying DDM strategy.	254
B.8	RID Test: Throughput data for LocalInteraction varying DDM strategy.	255
C.1	Interaction classes used in the distributed instant runoff algorithm.	265

Abstract

Simulation provides the ability to obtain results from, and analyse, a system without physically building the system. These results can be used to inform the actual construction of the physical system, how best to use a system, how best to integrate a system with another system, and so on. A simulation can also be used to train and educate the end-users of a system either before the system is actually produced or when the availability of the actual system is limited.

Most end systems are in some way composed of subsystems. The subsystems themselves may be composed of subsystems. This type of architecture is generically referred to as a system of systems. For example, a ship is composed of a hull, engines, sensors, etc. The engine system may be composed of the fuel and cooling subsystems, for example. Systems constructed this way have numerous benefits including allowing subsystems to be built independently of each other (after creating well defined interfaces), and allowing for subsystems to be replaced without affecting other subsystems. These same benefits are desirable in the construction of a simulation of a system. One simulation framework that supports these ideals is the High Level Architecture (HLA).

The HLA is an international modelling and simulation framework that specifically provides for distributed simulation. The HLA uses the term federate for component simulations that are then brought together in a distributed computing environment to form a federation. The HLA defines a data model for documenting the data interfaces of the federates and the application programming interface used by the federates to communicate data. A simulation of a systems of systems architecture can be implemented in the HLA by creating federates for each subsystem and defining the data communicated between subsystems in terms of HLA's data model.

HLA's default communication model defines publishers and subscribers of data classes. The HLA provides class based filtering, i.e., a federate only receives data for a data class to which it has subscribed. However, HLA's default communication model has no notion of direct 'wiring' between federates. Thus, it is not possible to have data sent to a specific federate. This creates a problem if multiple instances of a system of systems are simulated concurrently, which may be desirable so as to observe the interactions between systems. In this case, the data sent within one system is exposed to all other systems in the simulation.

This thesis explores this problem of simulating multiple systems of systems using the HLA. The problem is stated formally by introducing the concept of a message path and showing that a federation containing multiple systems of systems contains incorrect message paths which communicate intra-system data between systems. Three methods are presented and shown to solve the problem by either eliminating the incorrect message paths or allowing a receiving federate to determine whether intra-system data was delivered via an incorrect message path.

The three solutions are Local Data Filtering (LDF), Data Distribution Management (DDM), and Federation Communities (FC). The LDF solution marks all intra-system data with a system identifier, allowing receivers to distinguish whether they should process it. The DDM method uses a service defined by the HLA that essentially provides an automated version of the LDF solution. The FC method restricts one federation to simulating one system and requires a multiple system simulation to enable inter-federation communication, something that is not defined in the HLA.

These three methods are analysed both quantitatively and qualitatively. The quantitative analysis looks at performance overhead imposed by each method and how well each method reduces the number of incorrect intra-system messages communicated. The qualitative analysis is presented in terms of identifying the complexity of implementing each method for a specific systems of systems federation: the election process for the Australian federal government.

The thesis concludes that the LDF method is simple to understand but potentially finicky to implement and is wasteful of network resources. The DDM method is advantageous in that it is a service defined by the HLA standard. However, the implementation of the DDM services

by a Runtime Infrastructure (RTI) is not defined by the HLA. Thus, the performance of the DDM method is coupled to a specific RTI and its configurability. The FC method achieves an ideal of replicating the simulation of a single system without modification to achieve a multi-system simulation. However, it requires an inter-federation communication mechanism that is not defined by the HLA. The FC method introduces extra latency and reduced throughput to inter-system messages in a Local Area Network (LAN) environment.

Declaration

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library catalogue, and the Australasian Digital Theses Program (ADTP) and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Anthony Cramp
October 26, 2009.

Acknowledgment

I first have to thank my original supervisors John Best at the Defence Science and Technology Organisation (DSTO) and Michael Oudshoorn at the University of Adelaide for supporting me in starting this research. When life took them in other directions, two new supervisors stepped in to assist me: Shane Canney at the DSTO and Andrew Wendelborn at the University of Adelaide. I thank them for their ongoing guidance and (especially) patience during the writing of this thesis.

I thank the management of Maritime Operations Division (MOD), DSTO for allowing me to pursue this higher research. This period of study has spanned four Chiefs of MOD: Roger Creaser, Nanda Nandagopal, David Heilbronn, and John Riley.

Extra thanks are extended to my colleagues within MOD who had to take up some of my work during my absences while studying. In no particular order I thank the following people for their patience: Garry Brown, Greg Denehy, Rod Cheater, David Munro-Ford, Doug Scott, Peter Trenorden, Anh Tu, Karen Nugent, Jason Pennock, Paul Solomon, Mark Coombs, Chris Martin, Russell Anderson, Heath James, and Shane Canney.

I thank my family, especially my mother, for supporting my education through my life and for supporting me during the writing of this thesis when it appeared it would never be complete.

Finally, I thank my wife for her support and patience while I finished writing this thesis

This thesis is dedicated to the memory of my sister Kellie. I am sorry I did not complete it in time.

Document Details

This document was typeset with \LaTeX . All figures were created with PSTricks. Graphs were created with GNUPlot.