



**A DISTRIBUTED SIMULATION METHODOLOGY  
FOR LARGE-SCALE HYBRID MODELLING AND  
SIMULATION OF EMERGENCY MEDICAL  
SERVICES**

**A thesis submitted for the degree of Doctor of Philosophy**

**By**

**Anastasia Anagnostou**

**Department of Information Systems and Computing**

**January 2014**

---

## Abstract

Healthcare systems are traditionally characterised by complexity and heterogeneity. With the continuous increase in size and shrinkage of available resources, the healthcare sector faces the challenge of delivering high quality services with fewer resources. Healthcare organisations cannot be seen in isolation since the services of one such affects the performance of other healthcare organisations. Efficient management and forward planning, not only locally but rather across the whole system, could support healthcare sector to overcome the challenges.

An example of closely interwoven organisations within the healthcare sector is the emergency medical services (EMS). EMS operate in a region and usually consist of one ambulance service and the available accident and emergency (A&E) departments within the coverage area. EMS provide, mainly, pre-hospital treatment and transport to the appropriate A&E units. The life-critical nature of EMS demands continuous systems improvement practices. Modelling and Simulation (M&S) has been used to analyse either the ambulance services or the A&E departments. However, the size and complexity of EMS systems constitute the conventional M&S techniques inadequate to model the system as a whole.

This research adopts the approach of distributed simulation to model all the EMS components as individual and composable simulations that are able to run as standalone simulation, as well as federates in a distributed simulation (DS) model. Moreover, the hybrid approach connects agent-based simulation (ABS) and discrete event simulation (DES) models in order to accommodate the heterogeneity of the EMS components. The proposed FIELDS Framework for Integrated EMS Large-scale Distributed Simulation supports the re-use of existing, heterogeneous models that can be linked with the High Level Architecture (HLA) protocol for distributed simulation in order to compose large-scale simulation models.

Based on FIELDS, a prototype ABS-DES distributed simulation EMS model was developed based on the London EMS. Experiments were conducted with the model and the system was tested in terms of performance and scalability measures to assess the feasibility of the proposed approach. The yielded results indicate that it is feasible to develop hybrid DS models of EMS that enables holistic analysis of the system and support model re-use.

The main contributions of this thesis is a distributed simulation methodology that derived along the process of conducting this project, the FIELDS framework for hybrid EMS distributed simulation studies that support re-use of existing simulation models, and a prototype distributed simulation model that can be potentially used as a tool for EMS analysis and improvement.

## Acknowledgements

First and foremost, I would like to thank my supervisor, Dr Simon Taylor, for his continuous guidance, support and positive criticism during the long way of this thesis. It would not have been possible to complete this work without his support, motivation and inspiration. I would also like to show my appreciation to my second supervisor, Dr David Bell, for his precious advice.

I owe a great thanks to Mr Athar Nouman, whose assistance in the programming part of the RTI implementation was invaluable.

I am also obliged to Professor Terry Young for securing the funding through the MATCH Programme and has made the realisation of this research possible. Also, I would like to thank the MATCH team in Brunel, who made the process smooth with their continuous support and advice.

I would like also to thank the most important people in my life, my parents. Without their on-going support, I would never have been able to complete my studies. Also, I would like to thank my sister, my brother and their families for encouraging me all the way long. Thank you all for believing in me...

Last, but not least, I would like to thank my friends and colleagues. Their existence and their support were of great help to me.

Thank you all...

## Declaration

I hereby declare that the research presented in this thesis is my own work except where otherwise stated, and has not been submitted for any other degree.

Anastasia Anagnostou

Part of this thesis has been disseminated as below.

Anagnostou, A. and Taylor, S.J.E. (submitted) “Towards A Methodology For Building Large-Scale Distributed Hybrid Agent-Based And Discrete-Event Simulations: The Case Of Emergency Medical Services.” In *Proceedings of the Operational Research Society 7<sup>th</sup> Simulation Workshop (SW14)*, Worcestershire, England, April 1-2.

Anagnostou, A., Nouman, A. and Taylor, S.J.E. (2013) “Distributed Hybrid Agent-Based Discrete Event Emergency Medical Services Simulation.” In *Proceedings of the 45<sup>th</sup> Winter Simulation Conference (WSC14)*, Edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill and M. E. Kuhl, pp. 1625-1636, Washington, DC, December 8-11.

Nouman, A., Anagnostou, A. and Taylor, S.J.E. (2013) “Developing a Distributed Agent-Based and DES Simulation Using poRTico and Repast.” In *Proceedings of the 17<sup>th</sup> IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2013)*, pp. 97-104, Delft, NL, October 30-November 1.

Lord, J., Willis, S., Eatock, J., Tappenden, P., Trapero-Bertran, M., Miners, A., Crossan, C., Westby, M., Anagnostou, A., Taylor, S., Mavranzouli, I., Wonderling, D., Alderson, P., Ruiz, F. (2013) “Economic modelling of diagnostic and treatment pathways in

National Institute for Health and Care Excellence clinical guidelines: the Modelling Algorithm Pathways in Guidelines (MAPGuide) project.” *Health Technology Assessment*, Vol. 17, No. 58.

Anagnostou, A., Eatock, J. and Taylor S.J.E. (2012) “Nosopolis: Towards A Hybrid Agent-Based Discrete Event Simulation Tool For Emergency Medical Services Improvement.” In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference (WSC12)*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, Extended Abstract, Berlin, DE, December 9-12.

Anagnostou, A. and Taylor S.J.E. (2012) “An Agent-Based Discrete Event Simulation Tool for Emergency Medical Services Analysis.” In *Proceedings of the Operational Research Society 6<sup>th</sup> Simulation Workshop (SW12)*, Poster, Worcestershire, England, March 27-28.

---

## Abbreviations

A&E	Accident and Emergency
ABS	Agent-Based Simulation
ALS	Advanced Life Support
ALSP	Aggregated Level Simulation Protocol
ANSI	American National Standards Institute
BLS	Basic Life Support
Bluesss	Blues Simulation System
BOM	Base Object Model
BOSS	Burroughs Operational Systems Simulator
CA	Cellular Automata
CAS	Complex Adaptive Systems
CATOCs	Causal and Total Order communication support
CF	Conceptual Framework
CO	Causal Order
COTS	Commercial-Off-The-Shelf
CSL	Control and Simulation Language
CSPI PDG	COTS Simulation Package Interoperability Product Development Group
DES	Discrete Event Simulation
DIF	Data Interchange Format
DIS	Distributed Interactive Simulation
DMSO	Defense Modelling and Simulation Office
DoD	Department of Defense
DoH	Department of Health

---

DS	Distributed Simulation
DSEEP	Distributed Simulation Engineering and Execution Process
EMS	Emergency Medical Services
FIELDS	Framework for Integrated EMS Large-scale Distributed Simulation
FIFO	First-In-First-Out
FOM	Federation Object Model
GASP	General Activity Simulation Program
GIS	Geographic Information System
GPSS	General Purpose Simulation System
GSP	General Simulation Program
HLA	High Level Architecture
HSCIC	Health and Social Care Information Centre
IRM	Interoperability Reference Model
ITU-T	Telecommunication Standardisation Sector of the International Telecommunication Union
KPIs	Key Performance Indicators
LAN	Local Area Network
LAS	London Ambulance Service
LCIM	Levels of Conceptual Interoperability Model
LISI	Levels of Information Systems Interoperability
M&S	Modelling and Simulation
MASON	Multi-Agent Simulator Of Neighbourhoods (or Networks)
MASS	Multi-Agent Simulation Suit
MOM	Management Object Model
MSCO	Modelling and Simulation Coordination Office
NCC	Norwegian Computing Center



---

NER	Next Event Request
NHS	National Health Service
OMT	Object Model Templates
OOP	Object Oriented Programming
OSS	Open-Source Software
PASION	PAScal simulatION
PDU	Protocol Data Units
PO	Priority Order
PSM	Pascal Simulation and Modelling
RNG	Random Number Generator
RO	Received Order
RTI	Run Time Infrastructure
SD	Standard Deviation
SDL	Specification and Description Language
SIMAN	Simulation Analysis
SIMULA	Simulation Language
SISO	Simulation Interoperability Standards Organisation
SLAM II	Simulation Language for Alternative Modelling
SOL	Simulation Oriented Language
SOM	Simulation Object Model
SPL	Simulation Programming Language
TAG	Time Advance Grant
TAR	Time Advance Request
TSO	Timestamp Order
VV&A	Verification, Validation and Accreditation

---

## Table of contents

<b>Abstract</b> .....	ii
<b>Acknowledgements</b> .....	iv
<b>Declaration</b> .....	v
<b>Abbreviations</b> .....	vii
<b>Table of contents</b> .....	x
<b>List of Tables</b> .....	xvi
<b>List of Figures</b> .....	xvii
<b>CHAPTER 1</b> Introduction .....	1
1.1 Overview .....	2
1.2 Context and rationale .....	2
1.3 Research hypothesis .....	6
1.4 Research motivation.....	6
1.5 Research aims and objectives.....	7
1.6 Research methodology .....	8
1.7 Thesis outline and roadmap .....	10
1.8 Summary .....	14
<b>CHAPTER 2</b> Research context.....	15
2.1 Overview .....	16
2.2 Literature Review .....	17
2.3 Computer simulation.....	23
2.4 Simulation methodology .....	24

---

2.5 Simulation world views.....	32
2.5.1 Event-based approach.....	33
2.5.2 Activity-scanning approach.....	35
2.5.3 Process-based approach.....	36
2.5.4 Three-phase approach.....	37
2.5.5 Object-oriented world view .....	39
2.6 Discrete event simulation.....	41
2.6.1 Concepts of DES .....	42
2.6.1.1 DES objects .....	42
2.6.1.2 DES entities characteristics .....	43
2.6.1.3 DES entities actions.....	43
2.6.1.4 DES components .....	45
2.7 Agent-based simulation.....	47
2.7.1 ABS and object orientation .....	53
2.8 Hybrid simulation .....	56
2.8.1 Hybrid ABS-DES .....	56
2.9 Model reusability .....	57
2.10 Distributed simulation.....	60
2.10.1 High Level Architecture.....	63
2.10.1.1 Time management .....	66
2.10.2 Interoperability reference models.....	71
2.11 Software and tools.....	73
2.12 Summary .....	76

---

<b>CHAPTER 3</b> Towards a distributed simulation methodology for large-scale hybrid modelling .....	79
3.1 Overview .....	80
3.2 Issues on modelling the EMS.....	81
3.3 Hybrid simulation or single approach? .....	83
3.4 Semantic relationships between ABS and DES .....	84
3.5 DES and ABS modelling and simulation for EMS.....	86
3.5.1 Level of abstraction.....	87
3.5.2 Simulation studies in EMS .....	88
3.6 Why DS is advantageous over single composite simulation .....	90
3.7 Developing the proposed methodology for large-scale hybrid ABS-DES DS models..	92
3.7.1 Background and rationale.....	92
3.7.2 Development phases.....	95
3.7.2.1 Development phase one.....	96
3.7.2.2 Development phase two .....	99
3.7.2.3 Development phase three .....	100
3.8 Summary .....	103
<b>CHAPTER 4</b> The FIELDS Framework for Integrated EMS Large-scale Distributed Simulation .....	105
4.1 Overview .....	106
4.2 EMS components and interactions.....	107
4.2.1 Software tools.....	108
4.3 FIELDS interoperability reference model.....	110
4.4 Data communication and time synchronisation .....	112
4.5 Individual models conceptualisation.....	113

---

4.5.1 Ambulance service conceptual model.....	113
4.5.2 A&E department conceptual model .....	116
4.5.3 EMS model events.....	120
4.5.4 Data collection.....	122
4.6 Summary .....	122
<b>CHAPTER 5 EMS prototype model.....</b>	<b>124</b>
5.1 Overview .....	125
5.2 Realisation phase of the prototype model .....	125
5.2.1 Ambulance service ABS federate.....	127
5.2.1.1 Ambulance service model pseudocode .....	128
5.2.1.2 Data collection.....	131
5.2.1.3 Ambulance model verification and validation .....	133
5.2.2 A&E department DES federate .....	134
5.2.2.1 A&E department model pseudocode.....	136
5.2.2.2 Data collection.....	139
5.2.2.3 A&E model verification and validation .....	140
5.2.3 Time management strategy .....	142
5.2.4 Middleware implementation.....	143
5.2.4.1 Middleware verification and validation .....	147
5.3 Experimental design.....	148
5.3.1 Network settings.....	149
5.3.2 Results .....	149
5.3.2.1 Performance testing.....	149
5.3.2.2 Scalability testing .....	152

---

5.3.3 Conclusions .....	154
5.4 Summary .....	155
<b>CHAPTER 6</b> Evaluating the proposed DS methodology and the FIELDS framework ...	156
6.1 Overview .....	157
6.2 Distributed system performance and scalability testing.....	158
6.2.1 Network settings.....	158
6.2.2 Experimental results .....	158
6.2.2.1 Performance testing .....	159
6.2.2.2 Scalability testing .....	163
6.3 Revisiting the proposed DS methodology and FIELDS framework.....	165
6.3.1 Distributed simulation methodology revisited .....	165
6.3.1.1 Planning phase revisited.....	166
6.3.1.2 Development phase revisited.....	166
6.3.1.3 Experimentation phase revisited .....	167
6.3.1.4 Refined distributed simulation methodology .....	167
6.3.2 FIELDS framework.....	168
6.4 Summary .....	172
<b>CHAPTER 7</b> Conclusions and future work.....	174
7.1 Overview .....	175
7.2 Research summary .....	175
7.3 Research aims and objectives.....	178
7.4 Research contributions.....	180
7.5 Research limitations and future work .....	181
7.5.1 Research limitations .....	182

---

7.5.2 Future work .....	183
7.5.3 Reflections .....	184
7.6 Summary .....	185
<b>References</b> .....	186
<b>Appendix</b> .....	203
A1. Ambulance Service Model: Context Class.....	203

---

## List of Tables

Table 1: Ulgen’s et al. simulation methodology (Ulgen et al., 1994) .....	25
Table 2: Event routines example pseudocode.....	35
Table 3: Entity/object process routine .....	37
Table 4: Relationships between ABS and OOP .....	55
Table 5: Semantic relationships between ABS and DES .....	87
Table 6: ABS and DES characteristics .....	89
Table 7: Ambulance and A&E models events.....	120
Table 8: Ambulance model specifications.....	134
Table 9: A&E model specifications.....	141
Table 10: HLA 1.3 and 1516e simulation execution time when executed on a single-node and on a homogeneous network .....	150
Table 11: Scalability results.....	153
Table 12: Performance results .....	160
Table 13: Scalability results.....	164



---

## List of Figures

Figure 1: Research methodology .....	11
Figure 2: Thesis roadmap .....	13
Figure 3: a) Law and Kelton’s simulation methodology (Law and Kelton, 2000)      b) Banks et al.’s simulation methodology (Banks et al., 2000) .....	28
Figure 4: Robinson’s simulation methodology (Robinson, 2004).....	29
Figure 5: Simulation world views.....	34
Figure 6: Relationship between actions of the entities (source: Pidd, 2004).....	44
Figure 7: Components of DES models and control flow (source: Law and Kelton, 2000).....	46
Figure 8: Agents’ structure (source: Macal and North, 2010).....	50
Figure 9: ABS topologies (source: Macal and North, 2010) .....	51
Figure 10: Active and passive agents concept of this thesis.....	53
Figure 11: Re-use of simulation models .....	59
Figure 12: Levels of re-use of simulation models .....	61
Figure 13: High Level Architecture.....	63
Figure 14: Time management example .....	70
Figure 15: Simulation software evolution .....	73
Figure 16: Emergency medical services .....	82
Figure 17: Levels of Conceptual Interoperability Model (source: Wang, Tolk, and Wang, 2009) .....	93
Figure 18: Distributed simulation methodology development approach .....	96
Figure 19: Steps in a standalone simulation project .....	97
Figure 20: Mapping the phases in a standalone and a DS project .....	99

---

Figure 21: Distributed simulation methodology .....	101
Figure 22: FIELDS conceptual framework .....	108
Figure 23: FIELDS interoperability reference model.....	110
Figure 24: FIELDS HLA concept.....	112
Figure 25: Ambulance service timeline (source: Fitzsimmons, 1973).....	113
Figure 26: Ambulance service model abstraction .....	114
Figure 27: Ambulance service use case diagram.....	116
Figure 28: Driving distance corrective coefficient .....	117
Figure 29: A&E departments model abstraction .....	118
Figure 30: A&E department flowchart diagram.....	119
Figure 31: Screenshot of published online ambulance service performance data spreadsheet (source: NHS England(a)).....	132
Figure 32: Ambulance service area .....	133
Figure 33: Ambulance service model response time a) simulation, b) real system.....	135
Figure 34: Screenshot of published online A&E attendance data spreadsheet (source: NHS England(b)) .....	140
Figure 35: Total time spent in the six A&E departments .....	142
Figure 36: Lookahead .....	143
Figure 37: DS implementation.....	145
Figure 38: Sequence diagram showing the interactions via the RTI.....	147
Figure 39: Comparison of execution time on a single node and a distributed environment with different number of federates .....	151
Figure 40: Time increase difference .....	152
Figure 41: Scalability results graph .....	154

---

Figure 42: Execution time on a single-node versus distributed environment for a) 1 week, b) 2 weeks, c) 3 weeks and d) 4 weeks simulation time .....	161
Figure 43: Execution time difference on single-node versus distributed environment ...	162
Figure 44: FIELDS speedup results.....	163
Figure 45: Execution time in relation to workload increase .....	165
Figure 46: Refined distributed simulation methodology .....	169
Figure 47: Simulation techniques combinations.....	170

# CHAPTER

# 1

---

## Introduction

*This chapter presents an outline of the research context and states the problem area under study. Moreover, it discusses the aims and objectives of this research, as well as the methodology applied to reach the solution. Finally, it introduces descriptively and diagrammatically the rest of the thesis.*

## **1.1 Overview**

The research presented in this thesis is an investigation of simulation approaches that can be beneficial for modelling large-scale healthcare systems. The emphasis of this thesis is on the emergency medical services (EMS) and, more specifically, on the issues around modelling large and complex systems. The proposed framework addresses the challenges faced by the healthcare stakeholders to provide quality and cost effective EMS. In line with this, the main focus is to understand and address the managerial and clinical challenges, as well as support re-use of existing resources.

Before diving further into the particularities of this study, it would be helpful for the reader to be introduced to the context of the research together with the research approach taken to conduct this research. Therefore, next in this first introductory chapter, the context of the research is established. This thesis investigates novel modelling and simulation (M&S) technologies that can be utilised as managerial tools to assist healthcare policy makers in improving EMS performance with minimal cost. To support this, this research develops the foundation on which, large-scale models can be developed by reusing existing simulation models of the subsystems that compose the complex model. Building on this, the research aims and objectives were set. A discussion on the adopted research approach to fulfil the aim and to meet the objectives of the project is presented. The chapter closes with an overview of the remaining document, as well as a diagrammatic roadmap where the reader can go through the whole structure of the thesis.

## **1.2 Context and rationale**

As the technology advancements in computing are moving in a rapid pace, more and more computing resources are becoming available in affordable costs. Desktop personal computers, nowadays, are powerful machines with large storage and memory capacity. Additional-

ly, more recent advancements, such as cloud computing, constitute a robust computing power resource that can be used as a utility. Consequently, a user can take advantage of the offered services without investing in expensive assets.

These advances in computing technology provide a new perspective in the area of M&S. Simulation modelling is an important decision-making and system analysing tool. It is being used in a range of scientific disciplines to study new systems or changes to existing ones and answer “what-if” questions when the physical implementation of a system is difficult or even impossible to be achieved. Naturally, simulation uses historical data as input and outputs an estimation of the system behaviour.

However, simulation experiments require massive computing capacity. As the model complexity increases, the required computing resources increase, too, and the execution time of a single experiment increases in a non-linear way. This research proposes the employment of distributed simulation (DS) techniques in order to efficiently model a traditionally complex system, such as the EMS, and to support experimental execution over several processors.

Discrete event simulation (DES) is being used to analyse problems in the healthcare sector for several decades (Mustafee *et al.*, 2010; Brailsford *et al.*, 2009a). During which time, a growing number of healthcare providers worldwide have deployed simulation modelling in order to improve healthcare services. However, according to Professor Brailsford in (Taylor *et al.*, 2013), this is not reflected in the academic literature. More recently, agent-based simulation (ABS) has been adopted by operational research (OR) and is being used to model complex systems that consist of interacting agents (Heath *et al.*, 2011). A significant proportion of this research has focused on EMS. For example, Ramirez *et al.* (2011) and Su and Shih (2003) use DES to model EMS; Aringhieri (2010) employs combined ABS-DES

approach for EMS analysis. EMS are the units of the healthcare system that respond to medical emergencies and involve pre-hospital and/or in-hospital care.

Often, large and complex systems can be divided into smaller subsystems that have their own characteristics. The fact that all subsystem may be modelled by the same simulation technique is a possibility but most frequently the individual systems are heterogeneous entities and, therefore, it is possible that different simulation techniques are more appropriate for modelling the component subsystems. Consequently, hybrid DS is adopted in this study.

DS initially meant the execution of DES models on parallel and distributed platforms (Chen *et al.*, 2008) and can be defined as the distributed execution of a simulation program across multiple processors (Fujimoto 2000). That is, if there are several simulations running separately on several computers, then a DS model is one in which these simulations cooperate (or interoperate) together, linked by a communications network and specialist software (middleware), as if they were a single simulation. Typically, a popular middleware used to link simulations together is an implementation of the IEEE 1516 High Level Architecture (HLA) standard (IEEE-1516, 2010). HLA is a set of standard rules that specify information sharing and coordination during the interactions of simulation models. HLA is realised programmatically via its run-time infrastructure (RTI) implementation. It has been developed by the Defense Modelling and Simulation Office (DMSO), now re-designated to Modelling and Simulation Coordination Office (MSCO), for the US Department of Defense (DoD). Up until now, HLA/RTI is mainly utilised in military simulation applications.

Various standards, based on the HLA, have been developed to either aid in the development of DS or to address specific domain M&S requirements (such as, common representation of activities, or maintaining data objects). One such standard is the Simulation Interoperability Standards Organisation's (SISO) SISO-STD-006-2010 "SISO Standard for Commercial-off-the-shelf (COTS) Simulation Package Interoperability Reference Models

(IRM)” (Taylor *et al.*, 2012a) which was developed to facilitate the development of DS with COTS simulation packages.

Hybrid simulation, as inferred from the name, is when more than one simulation techniques are used to model a system. The utilisation of the hybrid approach largely increases the realism of the simulation model when it is in fact difficult to emulate the real system by a single simulation approach. In this thesis, the hybrid model combines ABS with DES techniques.

DES is a technique that models a system behaviour as discrete events. That is an instant of time at which, for example, an entity enters or leaves an activity. An activity changes the state of an entity. For example, if the entity is a customer awaiting some service the state of the customer could be awaiting to be served, being served and is served. DES is being used in various industries to analyse process behaviours within a system. ABS simulates the individual behaviour of agents. Agents are individuals that have certain properties. The agents interact with other agents and the environment of the system. These interactions have as a result the change of agents’ properties which define the agents’ behaviour. ABS is being used mainly to model individual behaviours within a system. Arguably, ABS is continuously gaining in popularity within the simulation community (Taylor *et al.*, 2013). One of the reasons is its similarities with the object oriented paradigm (North and Macal, 2007). In both DES and ABS, simulation time progresses in discrete time steps and their modelling view can be a bottom-up approach of a system behaviour. However, the boundaries between the two techniques are not yet clearly defined (Brailsford, 2014). In this thesis, hybrid DS is a combination of DES and ABS.

Nevertheless, as the scope of a system being studied grows in terms of size and/or complexity, it becomes increasingly more difficult to use M&S. DS has been proposed as a technique that can be used to ease the difficulty of large model development. Healthcare



systems are, traditionally, considered to be complex with high levels of uncertainty. Moreover, healthcare organisations worldwide are seeing continuous shrinkage of resources and growth of structural complications. M&S has been used to study some of these problems. For example, simulation models are expensive to build in terms of development time and technical skills. The re-use of existing models, facilitated by DS, to save this cost of simulation development has attracted attention in recent years. Additionally, it may be more convenient to use different M&S techniques that might not be available in a single simulation package. Again, DS makes the use of multiple models, bind on multiple techniques, possible.

On this basis, the research hypothesis is now presented.

### **1.3 Research hypothesis**

The hypothesis that will be tested in this research is that it is feasible to develop a hybrid DS model for analysing EMS as a holistic system that consists of independent components. The underlying importance of this hypothesis is that by achieving efficient integration and collaboration of individually independent simulation model components, it enables model re-use of existing simulations with minimal modifications.

### **1.4 Research motivation**

This research was motivated by the fact that although a lot of effort has been done to model and develop computer simulation artefacts in healthcare, and especially in emergency medicine, these models are typically used for the purpose they are developed only. Being able to re-use individual models and build different ecosystems of independent but interoperating simulations in order to study the behaviour of larger systems, would be beneficial to both the healthcare management and the M&S community. Furthermore, capturing the interactions between all components of emergency services in runtime can provide better in-

formed decisions for EMS improvements. Another aspect of large-scale M&S is its requirements in computational power. Often, large simulations take vast amount of time to execute on a single CPU, and sometimes this is impossible due to memory consumption. Therefore, distributed simulation was selected as a technique that enables interoperability and reusability of heterogeneous simulation ecosystems. Moreover, distributed simulation enables the computational distribution of large-scale simulation execution over many CPU in a networked environment.

## 1.5 Research aims and objectives

The aim of this research is:

“To explore new approaches to emergency medical services simulation modelling by using distributed simulation techniques to support model reusability and to develop a novel simulation methodology for developing hybrid distributed simulation models.”

To achieve the aim of this thesis, the following six objectives will be met:

- Objective 1

To formulate the research hypothesis and establish the aim of this thesis. Furthermore, to identify the appropriate research methodology that will be the vehicle to reach the aim and test the hypothesis.

- Objective 2

To review the normative literature and investigate issues in the EMS modelling field that can be studied efficiently by applying M&S.

- Objective 3

To develop a distributed simulation methodology for constructing hybrid ABS and DES distributed simulation models.

- Objective 4

To develop a framework for developing hybrid ABS-DES DS of EMS systems based on the distributed simulation methodology as stated in objective three.

- Objective 5

Based on the framework, to develop a prototype model of the London EMS and test the feasibility of the framework developed as specified in objective four.

- Objective 6

To evaluate the distributed simulation methodology, stated in objective three, and the framework for ABS-DES DS for EMS, stated in objective four, by applying it to a case study. This will be achieved by developing a realistic large-scale EMS model and experiment with different scenarios.

## **1.6 Research methodology**

This section discusses the research methodology that is adopted in this work and provides the justification behind this selection. The appropriate selection of the research strategy plays a vital role in achieving the goals of a research project. The research questions should act as a guide for the use of suitable research tools. A combination of methods is often used in order to be complementary and to form a better framework to achieving a research goal.

The purpose of this research is to explore new approaches to EMS simulation modelling by using DS techniques to support model reusability and to develop a novel simulation methodology for developing hybrid DS models. In doing so the following research questions will be addressed:

- Which simulation techniques are more appropriate for EMS modelling?
- Is DS better than standalone simulation for holistic EMS modelling?
- Is it feasible to support model re-use with DS?

In this study, empirical research methods are adopted in order to address the research questions and test the hypothesis that stated earlier in this Chapter. Based on the empirical research cycle the steps that will be followed are:

1. Review the literature and form the hypothesis;
2. Build scenario case studies based on the hypothesis by using simulation techniques;
3. Validate the derived simulation model by comparing the findings with those of the literature;
4. Conduct a case study and test the hypothesis using real-world data; and
5. Evaluate the findings of the case study.

Generally, empirical research starts with a given theory and the researcher continuously tests this theory or develops it by practice in the real-life arena or by testing a hypothesis (Davis *et al.*, 2007).

Empiricism has been widely used in computer science (Hoefler and Tichy, 2007). “A case study is an empirical inquiry that investigates a contemporary phenomenon within its

real-life context, especially when the boundaries between the phenomenon and context are not clearly evident” (Yin 2009, p. 13). There is a wide spectrum of case study designs each of which is fit to answer specific questions (Demeyer, 2011). Driven by the hypothesis that is tested here, namely, if it is feasible to develop a hybrid DS model for analysing EMS as a holistic system that consists of independent components, a feasibility study is selected for this thesis.

Furthermore, computer simulation is a software artefact. The derived model of the real system is a product of the design science (March and Smith, 1995). According to Carlsson (2006), design research attempts to create artefacts that have as a purpose to address a specific problem. The process of a design science research consists of four steps, i.e., problem identification, design, implementation and evaluation (Bilandzic and Venable, 2011). To address the simulation artefact design and evaluation, design science research approach is adopted as well.

In brief, a combination of empirical and design research methodologies is applied in this thesis (see Figure 1). Empiricism is a well-established research methodology in computer science. In the same way, design research frames the process of a software artefact development that a computer simulation model can be seen as such. One empirical research tool for testing the hypothesis is the case study. From the various types of cases studies, a feasibility case study is conducted in this thesis, in line with the hypothesis that is under testing.

## **1.7 Thesis outline and roadmap**

To help the reader follow the structure of this project, a brief description of each chapter and a diagrammatic roadmap (see Figure 2) of the thesis are provided below. The thesis is divided into seven chapters, each of which serves a particular purpose in the documentation of a research project.

In Chapter 1, the research context and rationale are presented. Furthermore, a brief background is provided to support the following subsections where the problem area and the research hypothesis of this thesis are presented. Further on, the research aim is stated and the identified objectives in order to achieve the aim are explained. A brief analysis of the methodology followed in order to guide this work is mentioned. Finally, an overview of the following chapters is provided together with a diagrammatic roadmap of the document.

The background theory and the review of the relevant literature are provided in the subsequent Chapter 2. This chapter analyses in depth the underlying theoretical topics of simulation and modelling discipline that are relevant to this thesis. There is an introductory section of the general concepts of computer simulation, and a thorough analysis of the simula-

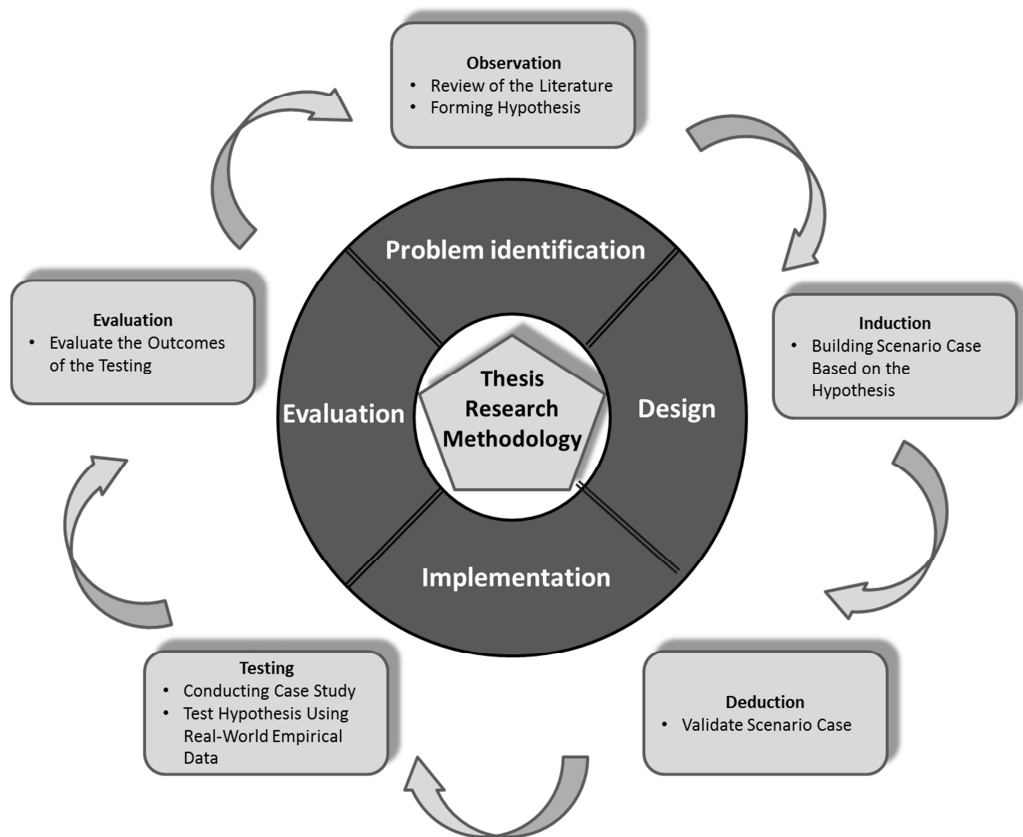


Figure 1: Research methodology

tions methodologies that are relevant to this project and support the proposed DS methodology. Furthermore, there is an analysis of the different simulation world views that underpin the implementation of various simulation packages and languages. The theoretical background of the simulation paradigms that are used in this thesis, i.e., ABS and DES, is provided. Subsequently, the hybrid simulation concept is discussed briefly and the issues of reusability of simulation models are discussed in more details. Finally, the theoretical background of DS is presented and the software tools that are relevant to the presented project are discussed.

Chapter 3 clarifies and justifies the statement that ABS and DES are the appropriate simulation paradigms for modelling EMS. Furthermore, an analysis of the EMS is provided in order to help the reader to understand the challenging and complex nature of the system. Additionally, a discussion on the relevant literature is provided together with the identified gap that helped in the planning and design phases of this research. Moreover, the semantic relationship between ABS and DES is presented and the selected interface approach, i.e., DS, is compared with the alternative option of standalone modelling. To end with, the proposed methodology development process and rationale is described. Built upon existing strategies for simulation projects construction, the proposed distributed simulation methodology expands the scope in order to incorporate the main concepts of this research, namely hybrid ABS-DES modelling and DS simulation.

In Chapter 4, there is a step-by-step walk-through of the proposed Framework for Integrated EMS Large-scale Distributed Simulation (FIELDS) for designing the hybrid ABS-DES distributed EMS M&S. The proposed framework derived after analysing the EMS issues and complexities, integration issues of ABS and DES techniques, the interoperability reference models (IRM) design, and issues on DS and standards.

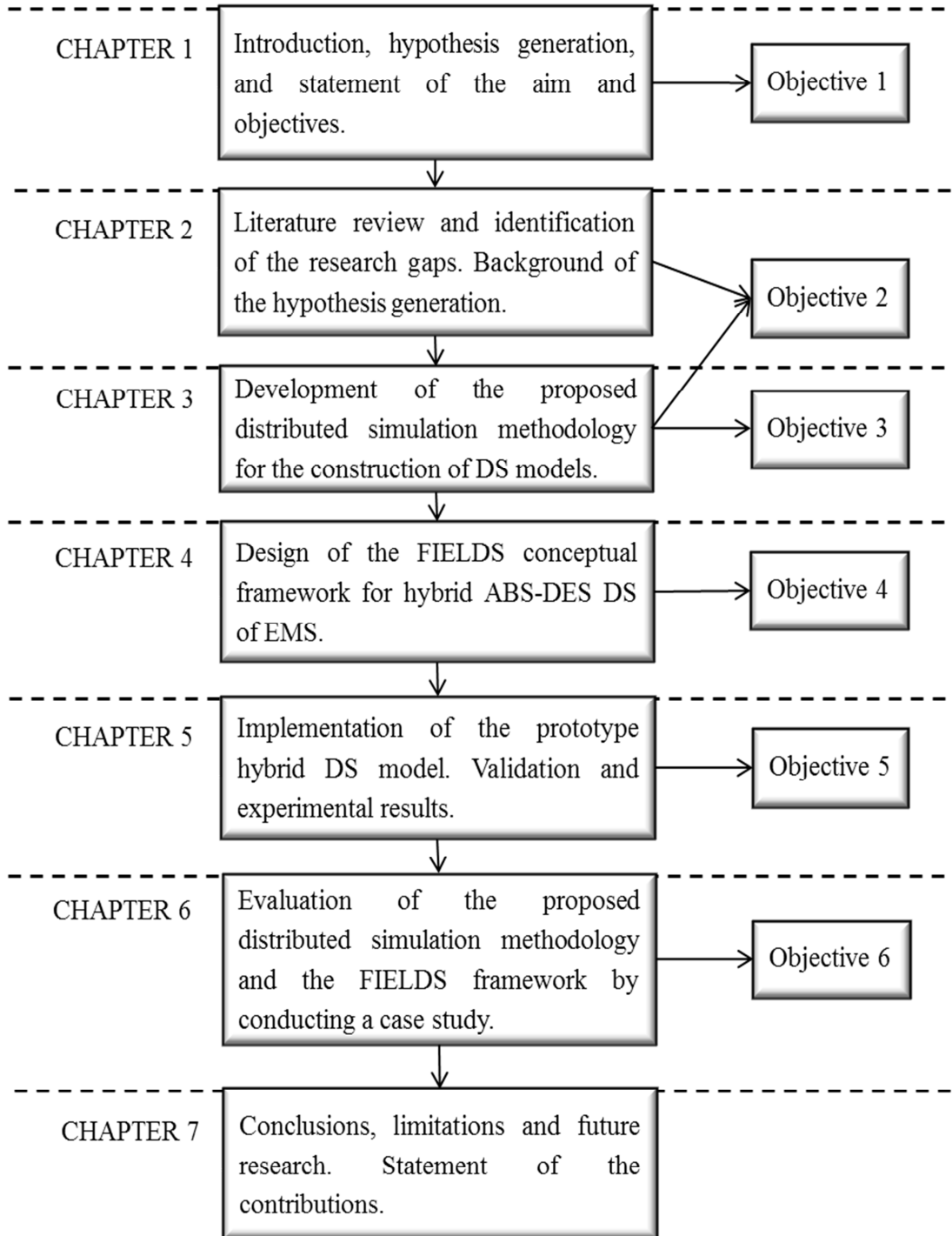


Figure 2: Thesis roadmap



The planning of the case study and the prototype design and development are discussed in Chapter 5. Here, a prototype model of a 1:5 scaled-down model of the London EMS is developed. The data that populate the model is data published online by the UK's Department of Health (DoH). Several experiments conducted in order to validate the prototype model and to demonstrate the feasibility of the proposed framework. The results are encouraging and a further evaluation of the concept is provided in Chapter 6.

Chapter 6 presents the evaluation of the proposed approach and supports the hypothesis testing. Several scenarios are run in a large-scale hybrid DS model of the EMS system. The produced results were analysed and interesting conclusions are drawn. Furthermore, the proposed DS methodology is revisited and refined. Finally, the proposed FIELDS conceptual framework for hybrid distributed EMS modelling is discussed further.

Finally, Chapter 7 provides the summary and the conclusions of the thesis. A reflection on the identified objectives to reach the research aim is provided. The contributions of this thesis are analysed. Finally, the limitations and the potential future work are stated.

## **1.8 Summary**

In this chapter, an overview of the research context has been presented with the problem area and the motivation behind this study. Further, the thesis aims, objectives and methodology are discussed.

# CHAPTER

# 2

---

## Research context

*This Chapter presents a review of the recently published articles in the area of EMS modelling and simulation. Also, it presents the research theoretical context of this thesis. There is an introduction to the simulation methodologies that frame a simulation project. Also, the different simulation world views are discussed and explained. Finally, the simulation technologies, relevant to the current thesis, are analysed. That is, DES, ABS, hybrid technique, DS concepts and standards, as well as software tools that implement the aforementioned technologies. Furthermore, simulation model reusability issues are discussed.*

## 2.1 Overview

Chapter 1 introduced the topic of this thesis. The problem area behind this research that motivated this study was stated. Furthermore, the hypothesis that it is feasible to develop a hybrid ABS-DES DS model to study EMS as a whole system that consists of composable component models was established. The motivation, and the research aim and objectives were discussed and the methodology to achieve the objectives was presented. Finally, an outline of the whole thesis was presented.

This Chapter presents a review of the recently published articles in the area of EMS modelling and simulation and identifies the gap in the published research in this area. Also, the theoretical context of the thesis is presented. There is a discussion on every theoretical aspect that supports the hypothesis of this research. First, this chapter introduces the reader to the general concepts of computer simulation. Then, there is an analysis of the simulations methodologies that are relevant to this project and support the completion of simulation projects. Subsequently, it provides the background on the different simulation world views (or conceptual frameworks) that support the logic of the simulators' implementation. As stated in Chapter 1, within the context of EMS, the suitable simulation techniques for studying the system are ABS and DES. Therefore, the theoretical backgrounds of both M&S paradigms are explained in detail in the following sections. Henceforth, hybrid simulation concept is discussed briefly, in terms of DES and ABS hybrid models, and the issues of re-usability of simulation models are discussed in more detail. Another aspect of the presented approach is the DS utilisation for the integration of the component models. Hence, the theoretical background of DS and the time management implications are presented in this Chapter, too. Finally, software tools that are relevant to the presented project are discussed.

## 2.2 Literature Review

In the recently published literature, i.e., after 2010, many articles present simulation studies of the ambulance services. The studies focus on a variety of problems that ambulance services policy-makers face, such as dispatching strategies, scheduling, covering, etc. For example, improvement scenarios of the French ambulance service were studied by Abouljinane *et al.* (2012) using DES. Response time and resource utilisation were used as the benchmarking KPIs for the system performance. They used the ARENA ([www.arenasimulation.com](http://www.arenasimulation.com)) commercial simulation software to develop the simulation and the ARENA input analyser to fit the input data to distributions. The travel times were calculated according to data provided by the National Geographic Institute of France. The model was verified by stakeholders and developers and was validated by comparing the model output to the real system performance. The authors experimented with three improvement scenarios, i.e., an increase of resources, ambulance redeployment, and decreasing the call triage time, and concluded that the ambulance redeployment strategy can achieve the targeted performance.

Ibri *et al.* (2012) modelled the emergency medical services in Switzerland using multi-agent systems (MAS). They studied two decentralised approaches and compared them with the current centralised approach in terms of response time and covering area. The MAS was developed in the JADE ([jade.tilab.com](http://jade.tilab.com)) platform and the simulator was programmed as discrete event simulation using java. The ambulance vehicles are considered busy only when attending an incident and pre-emption is allowed both en route to an emergency, if the decision module finds that this improves the performance, and en route to the ambulance station. They found out that a decentralized approach, i.e., the dispatching and coverage optimum is decided locally, where the agents are coordinated implicitly can improve the system's quality of service, i.e., response time, while the system's performance, i.e., coverage, is not affected.

Air-ambulance services were modelled by Lee *et al.* (2012) using Korean trauma cases data. The authors used DES with mathematical optimisation modeling. They studied the problem of trauma centres and helicopter ambulance location for designing an efficient and effective a trauma care system in Korea, i.e., maximising the covering area with the minimum resources. They use recursive optimization and simulation to find the optimum in stochastic environments. The DES development was realised in C# programming language. After experimentation, they concluded that the feature of simulation to analyse temporal data can give more realistic results.

Van Buuren *et al.* (2012) developed an EMS simulator for the Netherland's ambulance service in the Amsterdam region. For portability purposes, they used C++ programming language to realise the simulator. Their tool, called TIFAR, has a GUI for visualisation of the simulation and the output. The authors use a postal code method for emergencies location. The distance and travel times are calculated using route planning services. The dispatch of an emergency vehicle is decided according to the fastest approach to the incident which is calculated taking into account the distance and the travel time dependent on road conditions. TIFAR supports relocation of ambulance vehicles for covering purposes. The simulation can run in a visual mode where the visualisation is more realistic. However, it can run in a speed mode as DES, where an ordered list of current and future events is maintained.

The ambulance redeployment problem was also studied by Naoum-Sawaya and Elhedhli (2013). Their work was motivated by the fact that 27% of the ambulance activities, in the Region of Waterloo, Canada ambulance service, were relocations in the year 2006-07. The authors used stochastic optimisation that minimizes the number of ambulance relocations while keeping an acceptable quality of service. Their model optimises ambulance relocation in two stages and takes into account the time of day demand. They tested redeployment scenarios against the new introduced target of the ambulance service which is to reach the

incident site in less than 8 min. The results show, that this target is not achievable without redesigning the ambulance stations, even if the ambulance fleet is increased. Also, they experimented using Euclidean and Manhattan distance calculation and they concluded that the performance is better with the Euclidean calculation, as expected since the distances are smaller than the Manhattan calculation. However, they do not comment whether the difference is statistically significant.

At the same time, a lot of simulation studies focus on the operations of the A&E departments. For instance, Wang (2009) presented a prototype ABS model of the emergency department at the University hospital, Virginia. The author argues that ABS may be adopted easier than complex mathematical models by healthcare managers. However, the model does not capture the decision making and the interactions among the agents but rather models the processes as workflow.

DES using the commercial simulation software FlexSim Healthcare ([www.flexsim.com/flexsim-healthcare](http://www.flexsim.com/flexsim-healthcare)) was utilised by Holm and Dahl (2010) for modelling the Akershus University Hospital emergency department Norway. The model was developed to study the effect of an expected increase in patient volume. The authors report on the model validation process and present the results of different resource levels scenarios. The effect of the patient volume increase was measured against the current system using waiting times and resource utilisation as KPIs.

A DES model of a busy A&E department in West London, UK was developed by Eatock *et al.* (2011) using the commercial package Simul8 ([www.simul8.com](http://www.simul8.com)). The authors studied the four-hour length of stay target for the A&E departments in the UK. The model was able to capture the complexity and the details of the A&E behaviour towards this target. The authors comment on the usefulness of simulation in gaining insight and under-

standing the factors that affect the A&E performance. Also, they point out that developing a complex simulation model can be a very lengthy process.

Paul and Lin (2012) used DES to investigate the reasons for patient overcrowding in emergency departments and identify strategies for resolving them. They developed the model using ProModel ([www.promodel.com](http://www.promodel.com)) simulation software. They followed a four-phase methodology to conceptualise, develop, validate and experiment with the simulation. Parametric regression models were developed, using the simulation results, and the authors provide the regression coefficients for the length of stay in the emergency department for both the admitted and discharged patients. They found that the addition of a doctor during the peak hours would improve the patient throughput. After following up the implementation of the suggestions to the participating hospital, the authors comment on the usefulness of DES in analysing complex systems such as emergency departments.

A decision support framework based on simulation was developed by Abo-Hamad and Arisha (2013). Balanced scorecard tool was incorporated in the tool in order to improve the communication within the organisation and assist in making better informed decision taking into account the vision and objectives of the emergency department management and operations. The simulation model was developed using an object-oriented programming language and the scenarios that were tested were suggestions by the senior management team of the emergency department. They applied different strategies to the emergency department of a University Hospital in Dublin and concluded that the management of the out-flow of admitted patients improves the performance of the emergency department more than the increase of capacity and workforce of the department alone. The weight of the KPIs in each scenario was assessed by a preference model developed using preference ratios in multi-attribute evaluation.

The overcrowding of the emergency departments was studied by Ashour and Okudan-Kremer (2013), too. The authors developed a DES model using the commercial package Simio ([www.simio.com](http://www.simio.com)). The authors point out the importance of the triage process in the department's crowding and compare two triage algorithms against the major performance measures of an emergency department. According the results, the proposed algorithm, i.e., Fuzzy Analytic Hierarchy Process (FAHP) and Multi-Attribute Utility Theory (MAUT), does not perform better, when reviewing the averages of the KPIs, than the popular Emergency Severity Index (ESI) triage algorithm. However, FAHP-MAUT outperformed ESI when comparing the severity level specific statistics. For this reason, and due to the nurse judgment involvement in ESI, the authors propose the use of the FAHP-MAUT triage algorithm.

The split-flow process of the emergency department of Saint Vincent Hospital in Worcester, USA was studied by Konrad *et al.* (2013) using DES. The authors developed the model in the ARENA package ([www.arenasimulation.com](http://www.arenasimulation.com)) and experimented with 17 scenarios in order to study the impact of the split-flow process redesign. The scenarios decided after consultation with the stakeholders and involved different staffing levels and different patient arrival patterns. Furthermore, the authors point out that an organisational culture with minimum resistance to changes and the quantitative evidence of the DES study contributed in the implementation of the suggested changes.

Lim *et al.* (2013) modelled the clinical staff of an emergency department as pseudo-agents (i.e., entities with embedded decision logic) in a DES model. Instead of the conventional resources of the DES technique, they introduced interactions among the clinical staff. The difference of the true agents, as defined in agent-based modelling, is that the pseudo-agents are not autonomous. The pseudo-agents interact with each other and make decision about the treatment of a patient. The computer simulation was realised using the ARENA simulation package ([www.arenasimulation.com](http://www.arenasimulation.com)) and the experiments involved comparison



between a pseudo-agent and a traditional DES approach. Data from the emergency department of a University Hospital in Ontario, Canada was used for the experiments. The authors conclude that modelling the interactions among the emergency department clinical staff can contribute to more realistic representation of the system and better resource scheduling analysis. However, they point out that the pseudo-agent approach using DES packages can become overcomplicated and hinder the scalability of the simulation.

Rahmat *et al.* (2013) used ABS to model the re-triage process within emergency departments. ABS was selected in order to model the interactions among the emergency department objects. The authors investigated the introduction of re-triage process in the emergency department of a Malaysian hospital according to the Canadian and Australian triage models. The experiments suggested improvements in the waiting times of the patients with severe conditions.

An ABS model was developed by Wang *et al.* (2012) to study preparedness and response of an emergency service to a mass casualty event in urban area. The topology of the ABS model is a Geographic Information System (GIS) that provides data related to road networks and location of the ambulance vehicles and hospitals. The model can react to the temporal variable amount of information about the scene of the disaster. Evacuation ambulances travel between the scene and the regional hospitals in order to transport the casualties to medical facilities. The authors modelled both aspects of an EMS, as presented in this thesis, too. For example, they include in the simulation both pre-hospital and hospital operations. They modelled the whole disaster management simulation logic using three sub-models, i.e., the incident scene, the pre-hospital responders, and the in-hospital processes. The medical facilities are modelled with high level of abstraction. For example, there are no diagnostic facilities, and there is no mention on clinical staff capacity. The model takes into account only the bed capacity of these facilities. Also, there is no information about patient arrivals by other means. The experiments involve 12 different dispatch policies, related to

the amount of the available information based on which the decisions are taken, grouped in two categories. The first category does not reserve specialised care while the second group of policies include reservation of specialised care, according to the number of casualties that need this type of care, when still on scene. The authors comment on the service improvement when resource reservation for specialised treatment is implemented. However, they mention the potential high cost of this policy. Furthermore, the authors point out that the more informed decisions result in better outcomes, however, this comes at a cost to the model execution time since the level of detail becomes higher.

As it is evident from the above, there is a gap in the literature regarding studies that incorporate all the components of an EMS. Although many simulation models have been developed to analyse the operations of ambulance services and emergency facilities in hospitals, little evidence from the literature indicates that the whole EMS system and the interactions among the different subsystems have been adequately studied. This is due to multiple reasons. First, developing all the involved models requires massive effort from modellers, second, the required data may not be available, third, the execution of such large and complex simulations are very computationally expensive.

This research proposes the re-use of existing simulation models that run on different nodes of a computer network and can be composed to form a large distributed simulation model.

### **2.3 Computer simulation**

In systems analysis, the commonly practised modelling paradigm is mathematical modelling, which can be used to give analytical or simulation solutions (Gruene-Yanoff and Weirich, 2010). However, complex systems, such as EMS, are very difficult to study using analytical models. With the advances in computer science and the increase in computational power, computer simulation has become a widely available tool for systems analysis and,

therefore, continuously gains in popularity. For example, in an academic literature survey on M&S in healthcare, Brailsford *et al.* (2009b) found out that after year 2000, there has been an increase in the use of simulation more than 50 per cent. The range of publications dates was from 1952 to 2007, where the majority of articles (82 per cent) were published after 1990.

Computer simulation can be characterised regarding the randomness, or the lack of it, in the system as stochastic or deterministic, and the time progression as discrete or continuous. Also, a simulation model is either *terminating* or *non-terminating*. In terminated simulations, there is a natural cause for the simulation to stop, while non-terminating simulations do not have a natural point where the simulation stops. Such a model will run for a long period of time, ideally until the output reaches a steady-state.

## 2.4 Simulation methodology

Conducting a simulation study is more than often a laborious task. It requires a lot of effort, time and technical expertise by researchers and practitioners to plan, develop and conduct experiments with computer simulation models. A simulation project is not just the development of a computer simulation artefact but rather involves multiple tasks, all of which are equally important to successfully complete the project. Having a defined methodology with clear steps to follow during a simulation project can save unnecessary work and reduce the possibility of errors.

Several efforts can be found in literature that attempted to frame the process of conducting a simulation study. Ulgen *et al.* (1994) tackled this issue from a practitioner's viewpoint. They analysed the phases of a simulation project as a set of guidelines mainly for supporting future modellers and concluded in an eight-phases methodology, as shown in Table 1, with the steps defined for each phase.

Table 1: Ulgen’s et al. simulation methodology (Ulgen et al., 1994)

<b>Phase 1.</b>	<p style="text-align: center;"><b>Define-the problem</b></p> <p>Step 1. Define the objectives of the study.  Step 2. List the specific issues to be addressed.  Step 3. Determine the boundary or domain of the study.  Step 4. Determine the level of detail or proper abstraction level.  Step 5. Determine if a simulation model is actually needed; will an analytical method work?  Step 6. Estimate the required resources needed to do the study.  Step 7. Perform a cost-benefit analysis.  Step 8. Create a planning chart of the proposed project.  Step 9. Write a formal proposal.</p>
<b>Phase 2.</b>	<p style="text-align: center;"><b>Design the study</b></p> <p>Step 1. Estimate the life cycle of the model.  Step 2. List broad assumptions.  Step 3. Estimate the number of models required.  Step 4. Determine the animation requirements.  Step 5. Select the tool.  Step 6. Determine the level of data available and what data is needed.  Step 7. Determine the human requirements and skill levels.  Step 8. Determine the audience (usually more than one level of management).  Step 9. Identify the deliverables.  Step 10. Determine the priority of this study in relationship to other studies.  Step 11. Set milestone dates.  Step 12. Write the Project Functional Specifications.</p>
<b>Phase 3.</b>	<p style="text-align: center;"><b>Design the conceptual model</b></p> <p>Step 1. Decide on continuous, discrete, or combined modelling.  Step 2. Determine the elements that drive the system.  Step 3. Determine the entities that should represent the system elements.  Step 4. Determine the level of detail needed to describe the system components.  Step 5. Determine the graphics requirements of the model.  Step 6. Identify the areas that utilize special control logic.  Step 7. Determine how to collect statistics in the model and communicate results to the customer.</p>
<b>Phase 4.</b>	<p style="text-align: center;"><b>Formulate inputs, assumptions, and process definition</b></p> <p>Step 1. Specify to operating philosophy of the system.  Step 2. Describe the physical constraints of the system.  Step 3. Describe the creation and termination of dynamic elements.  Step 4. Describe the process in detail.  Step 5. Obtain the operation specifications.  Step 6. Obtain the material handling specifications.  Step 7. List all the assumptions.  Step 8. Analyse the input data.  Step 9. Specify the runtime parameters.  Step 10. Write the detailed Project Functional Specifications.</p>

	Step 11. Validate the conceptual model.
<b>Phase 5.</b>	<p style="text-align: center;"><b>Build, verify, and validate the simulation model</b></p> <p>Guideline 1. Beware of the tool limitations.  Guideline 2. Construct flow diagrams as needed.  Guideline 3. Use modular techniques of model building, verification, and validation.  Guideline 4. Reuse existing code as much as possible.  Guideline 5. Make verification runs using deterministic data and trace as needed.  Guideline 6. Use proper naming conventions.  Guideline 7. Use macros as much as possible.  Guideline 8. Use structured programming techniques.  Guideline 9. Document the model code as model is built.  Guideline 10. Walk through the logic or code with the client.  Guideline 11. Set up official model validation meetings.  Guideline 12. Perform input-output validation.  Guideline 13. Calibrate the model, if necessary.</p>
<b>Phase 6.</b>	<p style="text-align: center;"><b>Experiment with the model and look for opportunities for design of experiments</b></p> <p>Step 1. Make a pilot run to determine warm-up and steady-state periods.  Step 2. Identify the major variables by changing one variable at a time for several scenarios.  Step 3. Perform design of experiments if needed.  Step 4. Build confidence intervals for output data.  Step 5. Apply variance reduction techniques whenever possible.  Step 6. Build confidence intervals when comparing alternatives.  Step 7. Analyse the results and identify cause and effect relations among input and output variables.</p>
<b>Phase 7.</b>	<p style="text-align: center;"><b>Document and present the results</b></p> <p>1. Project Book.  2. Documentation of model input, code, and output.  3. Project Functional Specifications.  4. User Manual.  5. Maintenance Manual.  6. Discussion and explanation of model results.  7. Recommendations for further areas of study.  8. Final Project Report and presentation.</p>
<b>Phase 8.</b>	<p style="text-align: center;"><b>Define the model life cycle</b></p> <p>Step 1. Construct user-friendly model input and output interfaces.  Step 2. Determine model and training responsibility.  Step 3. Establish data integrity and collection procedures.  Step 4. Perform field data validation tests.</p>

The above methodology is a very detailed process description and guidelines for good practice that mostly applies to consultants and practitioners. However, M&S, apart from a

powerful systems analysis tool for practitioners, is a technology that is widely applied in academia as a tool and methodology for scientific research.

In Figure 3a, the 10-step methodology of Law and Kelton (2000) is depicted. The simulation process appears as a sequence of distinguished tasks that are performed with some iteration during the conceptual and coding validation processes.

Figure 3b illustrates the simulation methodology that was developed by Banks *et al.* (2000). They introduced the parallel activities of *data collection* and model *conceptualisation* as separated activities, in addition to the iterative processes at the experimentation phase. Further, Law and Kelton (2000) include the conceptual design validation process before commencing the programming activity while Banks *et al.*'s methodology (2000) revisits the model conceptualisation step only if the model validation failed.

Another view of simulation projects development is the cyclic methodology of Robinson (2004). Robinson (2004) describes the key stages in a simulation study and with the double-headed arrows forming a cycle indicates that there is movement between the key stages during the simulation project.

From Figure 4, there can be seen that Robinson's (2004) cyclic simulation methodology describes the simulation study with four key stages (rectangles) and four processes (arrows). The key stages represent the deliverables of the project and the processes enable movement between the stages.

All four methodologies that are presented in this section include the fundamental procedures and the key steps that a modeller should follow for completing a simulation project. These procedures are explained next.

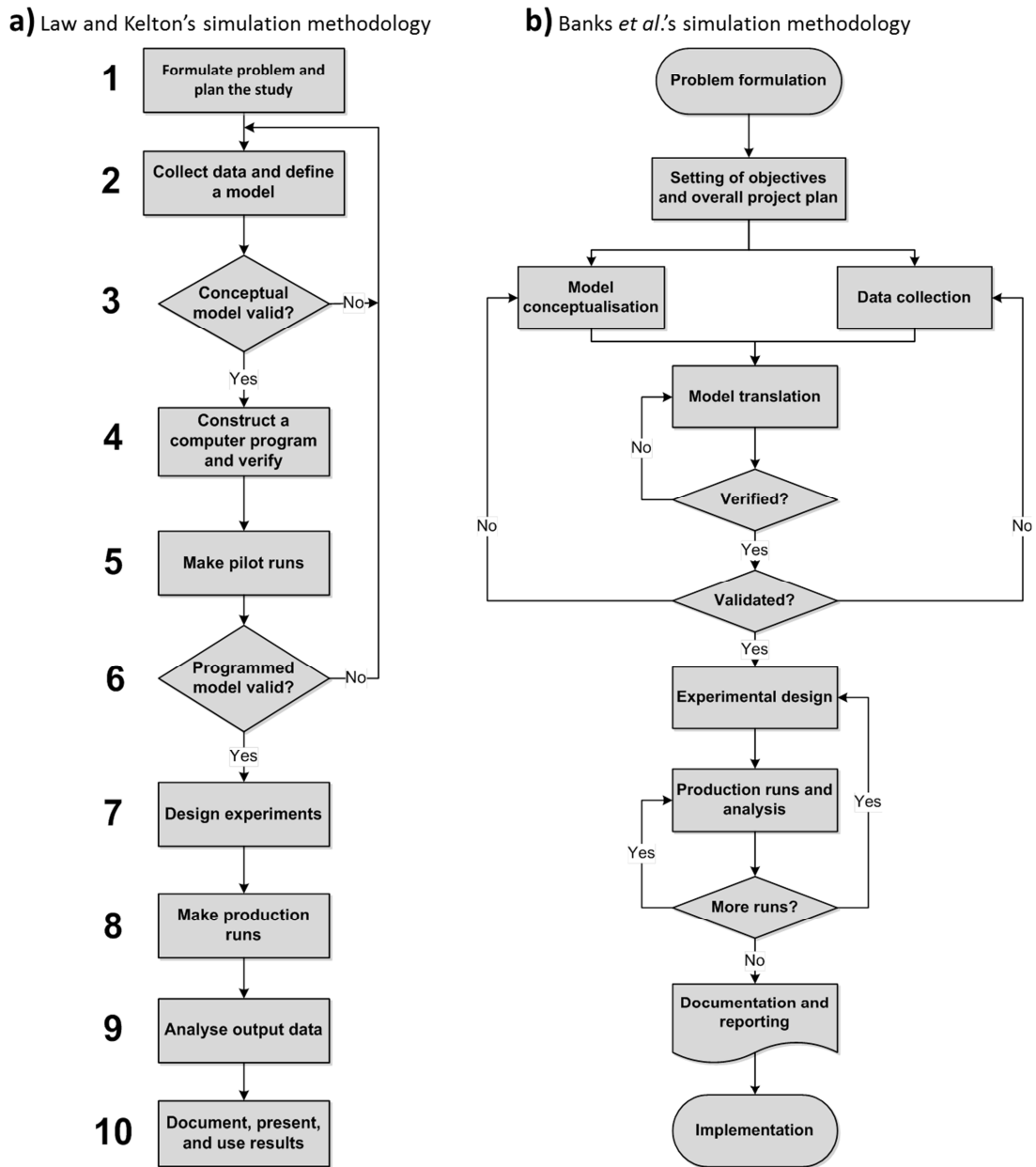


Figure 3: a) Law and Kelton’s simulation methodology (Law and Kelton, 2000)      b) Banks et al.’s simulation methodology (Banks et al., 2000)

*Problem formulation/definition* is the first stage of a simulation study and defines the problem to be analysed. Usually the involved parties, e.g., managers/policy-makers and modellers, recognise that there is a problem but the nature of the problem is identified while

the analysis is evolving. Banks *et al.* (2000) raised awareness that managers/policy-makers and modellers should understand and agree of the problem to be simulated.

Banks's *et al.* (2000) second step is the *Setting of objectives and overall project plan*, during which, the questions to be answered by the simulation study are specified. The software tools to be used should be decided by the involved parties and, also, the time and resources requirements. Law and Kelton (2000) and Ulgen *et al.* (1994) position the identification of objectives and scope of the study in the first phase, namely problem formulation/definition. Their first stage also includes determining the questions to be answered and the issues to be addressed, as well as the key performance indicators (KPIs), i.e., the performance measures to be output by the simulation project. Lastly, the required resources and the timeframe and the plan of actions and milestones are identified at this starting stage.

The *Model conceptualisation and data collection* phase is considered as one step in Law and Kelton's (2000) methodology, while Banks *et al.* (2000) described then as two different parallel processes. Also, Robinson (2004) points out that data collection and analysis starts with the conceptual modelling process since contextual data is necessary in order to con-

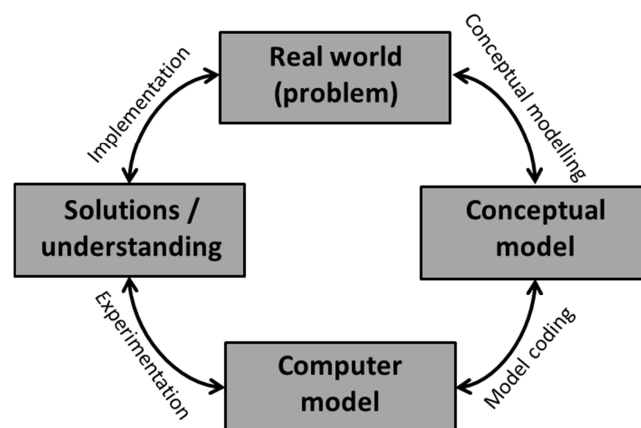


Figure 4: Robinson's simulation methodology (Robinson, 2004)



struct the conceptual model. However, detailed data is needed for the model coding stage. Ulgen *et al.* (1994), taking the practitioner’s point of view, divide the activities that are involved in this stage into three phases, i.e., the second, third and fourth of their methodology. At the conceptualisation stage, there is a selection of the underlying assumptions of the model and the level of abstraction of the simulation model is decided in order to reach the desired approximation of the reality. Banks *et al.* (2000) do not mention a conceptual model validation step, however, they have clarified that the involvement of the user is essential during conceptualisation which implies validation of the conceptual design. Data collection begins at this early stage, too. Data collection usually is a demanding procedure and frequently the data requirements change while the programming of the simulation evolves.

The next stage is the *Model translation/programming* and refers to the coding of the model. In order the system to be simulated in a computer, the model should be transformed to the appropriate format, e.g., a computer program. There are various tools available to assisting in model coding. The modeller can use a simulation language to program the model or can use simulation software packages which can considerably reduce the developing time.

The *Verification and validation* phase involves the process of debugging the simulation program and testing that it represents the simulated system. The term *verification* refers to the former and the term *validation* to the latter. The modeller verifies whether the computer program performs as it is supposed to and there are no inconsistencies in the code. Validation is the process of determining whether the model is a true representation of the real system. In doing so, the model behaviour is compared with the real system behaviour against some performance measures. The procedures of verification of the code and validation of the model are iterative processes and stop when acceptable accuracy is achieved. Robinson (2004) does not include verification and validation as separate stages but rather are incorporated in the four key stages.

In the step of *Experimental design and productions runs and analysis*, the modeller has to decide whether initialisation or warm-up period is needed and if so, how long this period will be. Also, the length and the number of simulation runs using different random numbers are decided. The analysis of the runs will determine whether more runs are needed and what design the experiments will have.

It is essential, in the phase of *Documentation and reporting*, that the analyst produces proper documentation for the program and the progress of the simulation study. The program documentation is important for future modifications by the same or different analysts. Also, it helps users to better understand and trust the model. The progress report can be considered as a documented chronological roadmap of the simulation project that can give insight to the model. Furthermore, reporting to the team during the process helps all the concerned parties to be updated of the progress even if they are not involved in the day-to-day model construction, and most importantly can give credibility to the model (Law and Kelton, 2000). Ulgen *et al.* (1994) defined different levels of documentation, i.e., project book, user manual, maintenance manual, etc., in their simulation methodology. According to Ulgen *et al.* (1994), project book is a diary of the development process, including minutes of the project team meetings, project scope changes, verification and validation details, etc., and is usually kept by the development team.

The last phase of the simulation methodologies that are presented here is the *Implementation* phase, as mentioned in Banks *et al.* (2000), or the *use of the results*, as mentioned in Law and Kelton (2000). Ulgen *et al.* (1994) refers to the model lifecycle. The successful implementation of the simulation project heavily depends on the involvement of the user of the model during the whole process. This stage is actually the use of the simulation model as a tool for planning and decision making. Often, simulation projects do not reach the implementation phase (Taylor *et al.*, 2009a), a fact that is more commonly seen in simulation projects conducted for academic research purposes.

The common characteristic of the published simulation methodologies is that a simulation methodology cannot be seen as a sequential process but rather as an iterative one with clearly defined steps that are repeatable. However, regardless of the repeatability and iterative nature, a simulation project should be conducted following ordered steps that can be revisited and revised. For example, the problem definition always precedes the conceptual design and the model programming always precedes the experimentation phase. Essentially, a rigorous planning and conceptualisation can lead to a successful and credible simulation project.

## 2.5 Simulation world views

In the theory of DES, from early days, there was a discussion on the different world views of simulation modelling (Overstreet and Nance, 2004). The term “*world views*” is used to describe the different conceptual frameworks for scheduling the next events and the different perspectives of the simulation objects. As defined by Derrick *et al.* (1989):

*“A conceptual framework (CF) is an underlying structure and organisation of ideas which constitute the outline and basic frame that guide a modeller in representing a system in the form of a model. ‘Simulation Strategy’, ‘world view’ and ‘formalism’ are other terms used in lieu of CF.”*

Another definition is given by Pegden (2010):

*“A simulation modelling world view provides the practitioner with a framework for defining a system in sufficient detail that it can be executed to simulate the behaviour of the system. Unlike simple static descriptive tools such as Visio, IDEF, UML, etc., a simulation modelling world view must precisely define the dynamic state transitions that occur over time. The world view must provide a definitive set of rules for advancing time and changing the state of the model.”*

Essentially, the simulation world views describe how the simulation software operates and vice versa. According to Overstreet and Nance (2004), there are three classic DES world views but in the literature one can find many more. For example Derrick *et al.* (1989) attempted to compare thirteen conceptual frameworks or world views that applied to DES.

The three classic world views are the *event-based approach*, the *activity scanning approach*, and the *process-based approach*. Another approach that is based on the activity scanning approach is the *three-phase approach* and is considered to be preferable than the other three (Pidd 2004, p.85).

Overstreet and Nance (2004) proclaimed the opinion that each world view captures a different type of simulation locality. That is, for the three classic world views, the event-based approach provides locality of time, the activity-scanning approach provides locality of state, and the process-based approach provides locality of object. In their justification of this view they mentioned: “event-based: each event routine in a model specification describes related actions that should always all occur in one instance; activity-scanning: each activity routine in a model specification describes all actions that should occur due to the model assuming a particular state (that is, due to a particular condition becoming true); and, process-based: each process routine in a model specification describes the action sequence of a particular model object”.

The four approaches are described in the following subsections and their program flowcharts are shown in Figure 5.

### **2.5.1 Event-based approach**

In the event-based approach the sequence of the simulation program is controlled by the event routines. The event routines cover all possible logical consequences of the state changes that occur due to the events. The events occur at an instant of time. The modeller

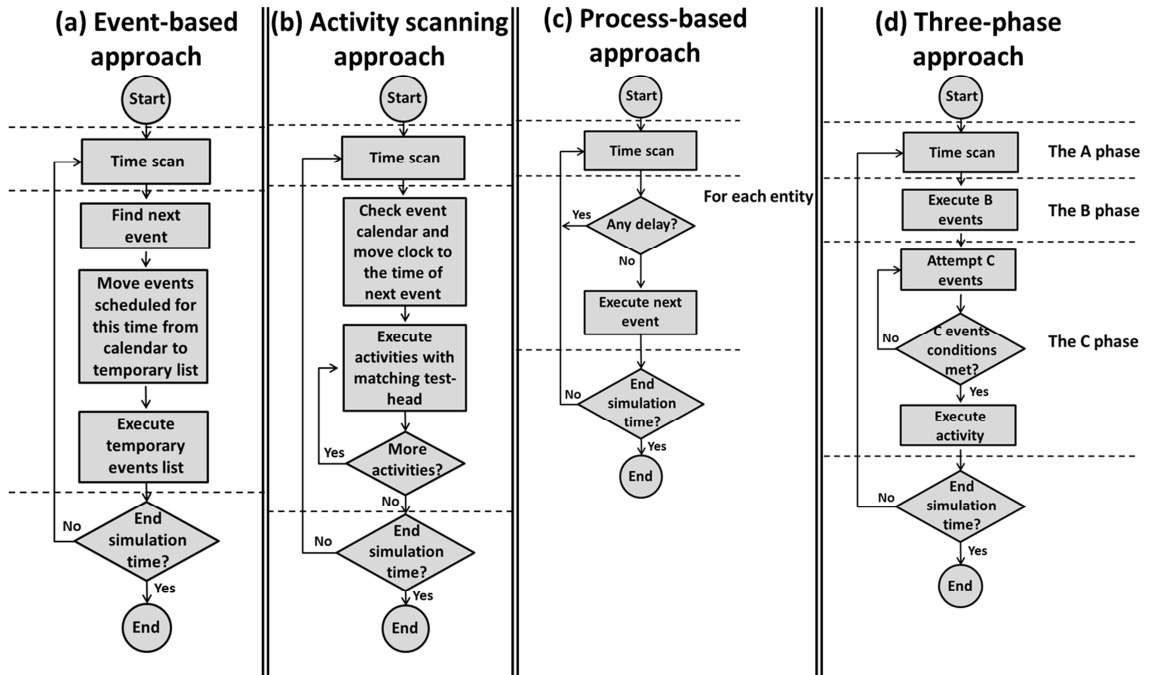


Figure 5: Simulation world views

needs to schedule the event routines, within which the following tasks are known. The flowchart of an event-based simulation program is shown in Figure 5a. That is, the event-based simulation executive keeps a list of all known future events; this list is enquired for any events that are due to occur at the current simulation time; all the events that are scheduled for the current simulation time are moved to a temporary events list; while the simulation clock is held to the current simulation time, all the event routines that are noted in the temporary events list are executed; the temporary events list is emptied; and, finally, the simulation clock can be advanced. The above process is repeated until the simulation run is over.

Since the logical sequences of the events are kept in the event routines if a change happens to the logic, the event routines should be modified, in the programming level. For ex-

ample, when different scenarios are tested, the event routines program should be modified for each scenario.

An example of the required event routines for a single service queuing model can be: 1) arrival routine, 2) start service routine, and 3) end service routine. A simplistic pseudocode of a version of the three event routines can be seen in Table 2.

### 2.5.2 Activity-scanning approach

The flowchart of an activity-scanning approach simulation program can be seen in Figure 5b. In the activity-scanning all events are turn into activities. Each activity carries checking mechanism, the 'test-head', that is used to test whether this activity is due to happen at the current simulation time. The 'test head' could be, for example, `IF simulation clock = start activity time DO the actions`. The simulation executive follows the sequence: check the time for when the next activity is to happen (this can be an event calendar); scan repeatedly for all the matching 'test-head' checks to the current simulation time; execute the activities with matching 'test head'; and, finally, the simulation clock can be advanced.

Table 2: Event routines example pseudocode

Arrival routine	Start service routine	End service routine
1 IF server is available 2 Call start service routine 3 ELSE 4 Add to service queue 5 Schedule next arrival routine 6 END IF	1 Make server busy 2 Calculate service time 3 Schedule end service routine	1 Make server idle 2 IF service queue not empty 3 Take the next item 4 Call start service routine 5 END IF

The above process is repeated until the simulation run is over.

### 2.5.3 Process-based approach

In the process-based approach the modeller should identify the whole process of each different type of entity/object of the simulation. The complete sequence of tasks and, most importantly, the delays that an entity might come across should be defined in a form of entity/object process template. Each new object that enter the simulation will inherit the process template of the same type of entities/objects. As shown in Figure 5c, the flow of a process-based approach simulation program is generally consists of the following steps: for each entity the simulation executive checks whether there is a delay due at the current time; if there is no delay, executes the next event; if there is delay, the simulation clock advances and check again; when the simulation clock reaches the simulation end time, the simulation run stops.

In Table 3, there is a simple pseudocode example of a process routine for a single server queuing model. In this example there are three delays; the blue-highlighted delay in line 13 is a conditional delay while the other two in lines 4 and 9 are unconditional delays. In the unconditional delays, the entity/object just waits for the specified delay time to pass and then progresses in the process. The conditional delays hold the entity/object until a specific condition is satisfied. In the given example, the entity/object waits until it reaches the head of the service queue.

As stated in Pidd (2004, p.104), the process-based simulation program should maintain two lists. The first list keeps the future reactivation events of all entities/objects processes that have unconditional delays. The second list keeps the current event list, first for the entities/objects that are conditionally delayed and the current simulation time equals the reactivation time, and second for the entities/objects that are unconditionally delayed and the current simulation time reaches their reactivation time.

Table 3: Entity/object process routine

Process routine
1 Entity arrives
2 Schedule next arrival
3 Instantiate process routine for next arrival
4 DELAY next arrival until arrival time is reached
5 IF service queue is empty AND server is available
6     Seize server
7     Calculate service time
9     DELAY entity/object until end of service time is reached
10    Release server
11 ELSE
12    Join service queue
13    DELAY until first in the service queue
14    IF server is available
15        GO TO line 6
16    END IF
17 END IF

### 2.5.4 Three-phase approach

The three-phase approach is mainly interested in the conditions that cause events to occur, rather than the sequence of events as in the process-based approach.

Similar to the conditional and unconditional delays, in the three-phase approach there is the concept of conditional and unconditional events and activities (Pidd, 1995). The unconditional events are called B events from the initial of the word ‘Bound’ or ‘Bookkeeping’, because these event are bound to happen when the simulation clock reaches a specific time or because they may be used for keeping records of the system, respectively. The conditional events are called C events from the initial of the word ‘Conditional’ or ‘Cooperative’,



because these events will occur only if some conditions are met or because some other entity is ready to cooperate in the activity they are involved, respectively (Pidd 2004, p.85-87).

The B events are known to the system and therefore can be scheduled directly. On the other hand, the system does not know in advance when a C event will occur. This depends on whether the related conditions are met. These conditions are kept within the C event in a test-head. Usually the end of an activity represents a B event while the start of an activity represents a C event. The conditions of the “start an activity” C event may be, first the existence of an entity in the respective queue and second, the availability of the required resources (Tocher, 1965).

In the current approach, the simulation engine keeps records about three information units of the entities. Through this record, the simulator manages the entities and their connection to B and C events. The units of information are:

- The *time cell* – If the entity is involved in a future B event, the time cell is the time when the entity is due to change state.
- The *availability* – The availability is a Boolean variable and indicates whether an entity is committed to a future B event. If the availability field is ‘false’, the entity is committed to a B event and the time cell will indicate the time that this is due to occur. If the availability field is ‘true’, the entity is not involved to any future B event and therefore its time cell is not taken into account.
- The *next activity* – The next activity defines which B event is due to occur next. In the same way as the time cell, the next activity is taken into account only when the availability variable is ‘false’.

The complete three-phase approach logic is executed in three phases, as the name infers. In the flowchart of Figure 5d, the three phases are illustrated. The **phase A** is usually called ‘time scan’ and involves the checking of the event calendar for the time of the next event

and the advance of the simulation clock to that time. Also, in this phase, there is a check on the availability and time cell of the entities' records. The **phase B** indicates the execution of all the unconditional events that are due to happen. Once the B events are executed, they are deleted from the current events list. The **phase C** involves the execution of the conditional events. The simulation executive scans the list with the C events and tests the condition in their 'test-head'. The ones that their conditions are satisfied are executed. During the execution of B events and the testing and execution of C events, the simulation clock remains still. After the execution of all C events that their conditions are satisfied, if the simulation has reached the end run time, the simulation ends, otherwise the simulation clock advances to the time of the next event (phase A).

### 2.5.5 Object-oriented world view

In the case of the ABS paradigm, the aforementioned world views cannot be applied. The agents, which are a fundamental component of ABS, are intelligent entities that can take decisions according to some conditions. Thus, there is no predetermined process that the agents follow, but rather this depends on their interaction with the other agents and their environment. However, agents are governed by rules which can be considered as a process to be followed by each individual agent.

The object-oriented world view can naturally represent a physical system, which can be viewed as a collection of independent and interacting objects (Fujimoto, 2000). A class of objects includes the properties of the objects, namely, their characteristics and their behaviours. The characteristics are described by fields, i.e., variables and attributes, and the behaviours are described by methods (Pegden, 2010). Instances of objects can be created dynamically during the simulation runtime. Objects can interact with other objects and invoke their methods. Significant concepts of the object-oriented world view are the inheritance, encapsulation and polymorphism.

Inheritance allows the creation of classes in a hierarchical structure, where the children classes can inherit properties from the parent class. The parent class represents a general representation of an object type while children classes are more specific type of objects. For example, in the context of ambulance services, a parent class can represent the vehicle objects with the general properties of vehicles, e.g., `speed`, `move()`. From the `Vehicle` class, children classes can be created, such as `Ambulance` and `TwoWheelVehicle` classes that inherit the general properties of the parent class.

Encapsulation allows the objects to control the modification of their fields. That is, their variables and attributes can be private (can be modified only by the class), protected (can be modified by the class and its subclasses) or public (can be modified globally).

Polymorphism allows the existence of different methods under the same name. For example, the classes `Nurse` and `Surgeon` can both have a method named `treatPatient()`. However, in the first case it can be to administer medication while in the latter case to perform an operation.

The object-oriented world view in simulation programming languages first appeared as an idea in the 1960s in SIMULA language (Pidd, 1995). SIMULA introduced the notion of classes, behaviours and instances. SIMULA I was developed by the Norwegian Computing Center (NCC) in the spring of 1961 and was a predecessor of the SIMULA 67, the basic concept of which was classes and objects (Nygaard and Ole-Johan, 1978). Subclasses can be created with the hierarchical inheritance capabilities of the language.

Object-oriented world view seems a natural choice for ABS languages. Agents can be seen as instances of objects with their individual behaviours governed by rules and their attributes that distinguish agents among each other. Also, interactions among the agents can be supported by the remote access of objects and message passing feature of object-oriented paradigm, usually supported by the “method call” capability. However, the con-

nection of DES with object-oriented world view is not so straight forward. The process-oriented nature of DES indicates that there is a centralised view of the system where individual entities do not take part in decision making. Usually, a type of hybrid approach is taken for object-oriented DES languages.

Pollacia (1989), in a survey of DES languages, mentions, among others, the PAsION and the SDL object-oriented DES languages. PAsION (PAscal simulatION) combines process-based approach with object-oriented world view. PAsION, and its later version PSM++ (Pascal Simulation and Modelling), generates Delphi Pascal code, while the new version, Bluesss (Blues Simulation System) generates C++ code. SDL (Specification and Description Language) combines activity-scanning approach with object-oriented world view. SDL defined in the Z.100 recommendation of the Telecommunication Standardisation Sector of the International Telecommunication Union (ITU-T) (Fonseca-i-Casas, 2008). Pidd (1995) discusses the combination of the object-oriented world view with the three-phase approach. This approach was taken to implement an evacuation planner system using C++ language (Pidd *et al.*, 1993 ). He pointed out that for the sake of encapsulation a large amount of source code was produced without actually having computational functionalities. Also, he highlighted that the B and C activities (or events) should be kept in a separate class and cannot be part of the entities class in order, first to avoid deadlocks and, second to make it efficient (and perhaps possible) to update the event lists that belong to the system.

In general, in object-oriented world view, when adopted by DES, a system class should exist that keeps track of events and the system state changes.

## 2.6 Discrete event simulation

DES is a dynamic simulation technique. That is, the processes of the system evolve over time. For example, a treatment for a chronic illness takes place for the whole lifetime of a

patient. Such a system can be simulated using DES. Basically, DES involves queuing systems. In the previous example of a chronic illness, a queue may be a waiting list for a hospital treatment. The mechanism for advancing the time in a DES model is usually next-event advance. That is, the simulation clock will “jump” to the time instance of the next scheduled event in uneven amounts of simulation time units. However, the simulation time in a DES model can advance in a fixed-time increment, too. That is the progress of simulation time will always be in the same amount of simulation time units. Within these steps, there may be events to occur or not.

In the next subsection, there is an analysis of the DES fundamental concepts.

## 2.6.1 Concepts of DES

### 2.6.1.1 DES objects

A DES model is characterised by its objects. The objects of a DES model are the *entities* and the *resources* (Pidd, 2004).

*Entities* are individual items of the system under study that their behaviour is fully followed. The simulation program preserves information about each entity so each one can be individually identified. As an entity changes state, the program keeps track of these state changes. The number of entities in a model indicates its complexity and possibly indicates the speed of its run. Examples of entities in a healthcare model could be the patients.

*Resources* are again individual elements of the system but without being modelled individually. A resource consists of countable identical items that their states are not being tracked by the simulation program. Instead, the program keeps a count of how many resource items are available. The number of resource types in a model is another indicator of its complexity. Here, a distinction should be made between the resource types and resource items. Resource types are the different categories of resources, whereas resource items are

the identical objects within a resource type. Examples of resource types in a healthcare model could be hospital beds and clinical staff. While resource items are the individual beds and the individual clinical staff.

### **2.6.1.2 DES entities characteristics**

When an element is classified as entity, further distinction should be made to help in organising the entities. According to Pidd (2004) the organisation of entities consists of three groups that can be permanent or temporary, i.e., classes, sets and attributes.

*Classes* are permanent groups of similar entities. A class is a suitable way to represent similar entities with similar behaviour. Sometimes it is possible to subdivide classes to subclasses for more detailed distinction. An example of a class in a healthcare model could be the patients, while a subclass could be the patients with chronic illnesses.

*Sets* are the temporary groups that entities consist when in a state or in a queue. Entities move from set to set as the simulation progresses and they change states. The order in which the entities are kept within a set can be particular, for example first-in-first-out (FIFO) or last-in-first-out (LIFO), or according to some priority rules, or even random. An example of a set in a healthcare model could be the patients that wait for a CT scan.

*Attributes* are items of information that belong to each entity. Their purpose is the distinction between members of the same class and/or to control the behaviour of an entity. If the latter is happening, attributes can substitute sets. Example of attributes in a healthcare model could be the blood glucose level or the age of patients.

### **2.6.1.3 DES entities actions**

The entities in a simulation model are participating in various actions during the course of simulation while the flow of time is progressing. The flow of time is measured by the *simu-*

*simulation clock* and represents the current value of simulated time. The time units in the simulation clock are decided by the modeller and can vary depending on the system that is simulated. If the simulation clock reaches the point of time that an event is scheduled, then the appropriate activity will be initiated.

*Events* are the instances of time when a change of state occurs in the simulation. Event can be the time when an entity enters or leaves a set. For example, the point of the simulation clock that an entity enters a queue is an event that changes the state of the entity.

*Activities* are the operations and procedures that commence at each event. These activities are the actions that transform the state of the entities. For example, when the entity enters a queue its state changes to “waiting for a service”. Accordingly, when the service begins, the state of the entity changes to “being served”.

*Process* is a group of sequential events in the chronological order in which they will oc-

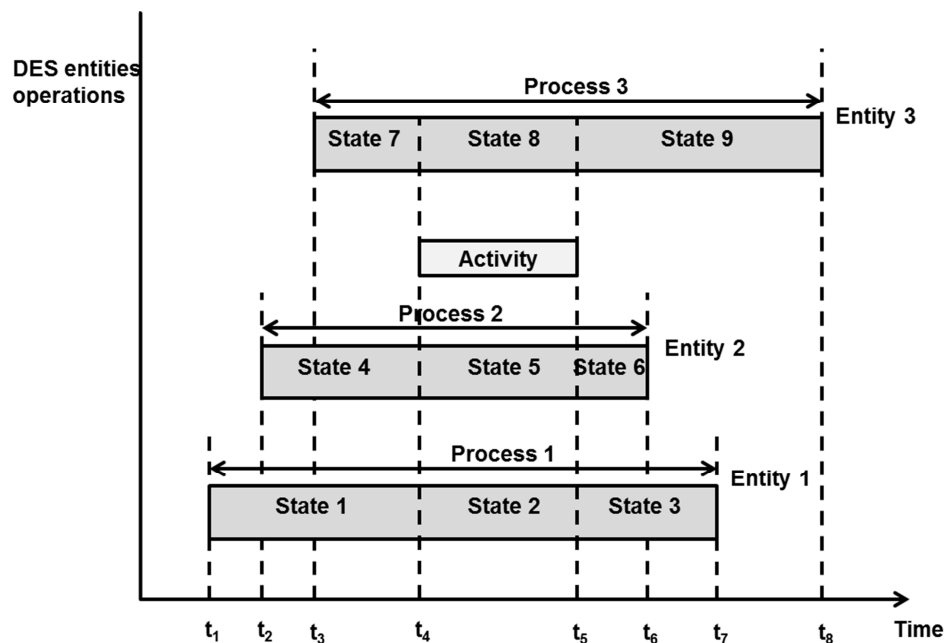


Figure 6: Relationship between actions of the entities (source: Pidd, 2004)

cur. A process is used to represent a part or all of the life of an entity. An example of a process could be: 1. A call arrives at a call centre, 2. The call is answered, 3. The call is dismissed.

An illustration of entities' actions in a DES model is depicted in Figure 6. The time instances represent events that start or end an activity and change the state of the entities (Nance, 1981). For example, at time  $t_1$  the first activity for Entity 1 starts and Entity 1's state is State 1, at time  $t_4$  the second activity for Entity 1 starts and its state changes to State 2. The whole process that Entity 1 is going through finishes at time  $t_7$ . Similarly, Entity 2's process starts at time  $t_2$  and ends at time  $t_6$ , after going through activities that changes its state. Entity's 3 process starts at time  $t_3$  and ends at time  $t_8$ .

#### 2.6.1.4 DES components

The flowchart in Figure 7 illustrates the control flow of the DES fundamental components when the time advance mechanism is next-event, which is almost always the case in DES.

Before explaining the control flow of the DES components, it is wise to briefly mention what these are. Listed below are the main components of a DES simulation program as mentioned in Banks *et al.* (2000) and Law and Kelton (2000).

- System is a set of entities that interact and act according to specific targets over time.
- Model is an abstract representation of a real system. A model is usually described by logical, structural, or mathematical relationships.
- System state is the collection of state variables at a certain time that can describe the system.
- Simulation clock is a variable that gives the current value of simulation time.
- Event list is a list that contains the time when the next event is scheduled to happen.



- Statistical counters are variables that are used for storing statistical information during the simulation run. Statistical counters record historical data and enable statistical analysis of the results.
- Initialisation routine is a subroutine that resets the simulation clock to 0. This routine is useful for experimental run of the simulation model.
- Timing routine is a subroutine that identifies the next event from the event list and then sets the simulation clock to the time when this event is to happen.
- Event routine is a subroutine that updates the system state when a specific type of event occurs.

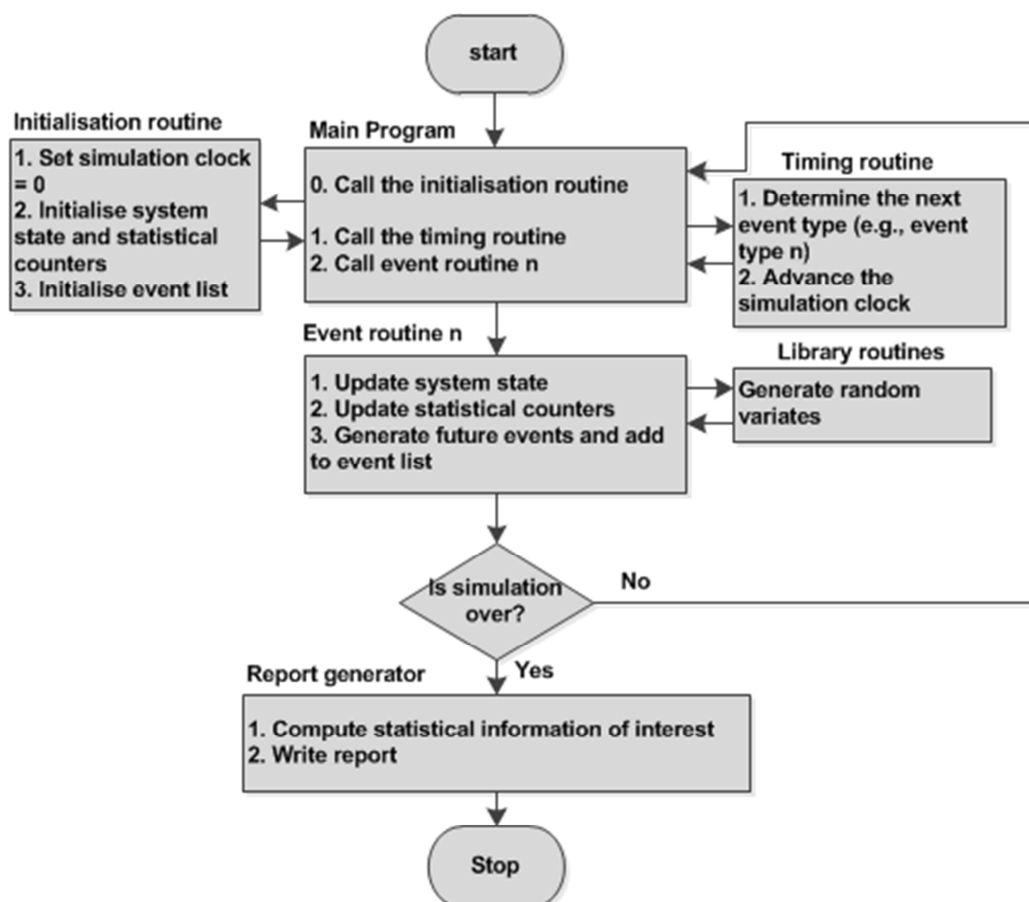


Figure 7: Components of DES models and control flow (source: Law and Kelton, 2000)

- Library routines are a set of subroutines that generate random numbers from probability distributions that were determined from the modeller.
- Report generator is a subprogram that produces reports from the aggregated results of the statistical counter.
- Main program is a subprogram that coordinates the various subroutines of the simulation model. More specific, the main program triggers the timing routine to determine the next event and then gives control to the corresponding event routine to update the system state accordingly. It may as well check for termination and trigger the report generator.

When the execution of a simulation program starts, the main program calls the initialisation routine, as shown in Figure 7. The initialisation routine will set the simulation clock to zero, initialise the variables and the statistical counters, and initialise the event list. Next, the main program calls the timing routine, which will determine the event that is due to happen next and advance the simulation clock to the time that this event is scheduled. Then, the main program calls the specific event routine. The event routine updates the system state, the statistical counters and the event list. If the simulation end time is reached, the statistical calculations are happened in the report generator and the results are exported. If the simulation end time is not reached yet, the main program will call the timing routine to determine the next event and progress the time. This flow will continue until the simulation end time is reached and the results are produced from the report generator.

## **2.7 Agent-based simulation**

ABS historically originated from the complex adaptive systems (CAS), where the principal area of study is the complex behaviours among individual and autonomous agents. CAS have the ability to self-organise and dynamically restructure their components in order to adapt and stand out in their environment. Their initial interest was to investigate into adap-

tation and emergence of biological systems (Macal and North, 2006). CAS can be characterised by properties and mechanisms that demonstrate a valuable framework for ABS design. As identified by Holland (1995), the properties and mechanisms of CAS are:

- Properties:
  - *Aggregation* that allows groups to form, i.e., individuals can be classified into general categories.
  - *Nonlinearity* that invalidates simple extrapolation, i.e., simple changes can cause large effects, not easily predictable.
  - *Flows* that allow the transformation and transfer of information and resources between the nodes of a network. Two main concepts can describe the flows in CAS: the multiplier effect and the recycling effect. The multiplier effect denotes the changes in the system when a node is added, and the recycling effect denotes the changes in the system when information and resources are reused.
  - *Diversity* that allows agents to behave differently from one another.
- Mechanisms:
  - *Tagging* that allows agents to be named and identified. Tagging may refer to a simple name or ID of an item of the aggregated group or it may refer to more complex behaviours that characterise an item.
  - *Internal models* that allow agents to reason about their micro-worlds, for example, an agent is able to anticipate the outcome of an input if this input reoccurs.
  - *Building blocks* that allow components and whole systems to be composed of simpler components. For example, a bicycle can be a combination of a frame, wheels, etc. These components can have different characteristics, i.e., colour, size, etc. and can be reused and recombined as building blocks to compose different bicycles.

ABS is used mainly to model decentralised, complex systems that consist of many interdependencies. Comparing with other modelling techniques, ABS provides a more realistic view of the system (Hawe *et al.*, 2012; Sumari *et al.*, 2013). The main components of ABS are the agents. Agents are autonomous components that have a sort of intelligence, for example, they can recognise their environment and other agents and interact with them. Also, they can be heterogeneous, adaptive and goal-directed components. That is, the agent characteristics and behaviours may vary, agents may learn from their environment and change their behaviour accordingly, and they may have a goal to reach and therefore compare their status with their goals and adjust accordingly. Agents can contain a basic level set of rules that determine their behaviour and a higher level set of rules that can change these rules (Loefstrand *et al.*, 2003; Macal and North, 2010).

In Figure 8, the structure of the agents and their characteristics is depicted as identified in Macal and North (2010). The agents have attributes and methods (Hawe *et al.*, 2012). According to Macal and North (2010), the agents' attributes can be static, that is they do not change, i.e., name, ID, or dynamic, that change during the simulation run. Dynamic attributes can be the agent's memory that holds instances of past events, the resources that the agents may have (i.e., food), the knowledge of their neighbours, etc. The methods of an agent are, among others, its behaviours, its ability to modify these behaviours, and the ability to update its rules and its dynamic attributes.

However, the essential characteristics of the agents are four, according to Macal and North (2010), and analysed below:

- Agents are distinguished, independent individuals with rules that administer their behaviour and decision-making capability. Their nature is discrete, which means that they have clear boundaries and it can be easily determined whether a characteristic belongs to a specific agent or not, or it is shared among agents.

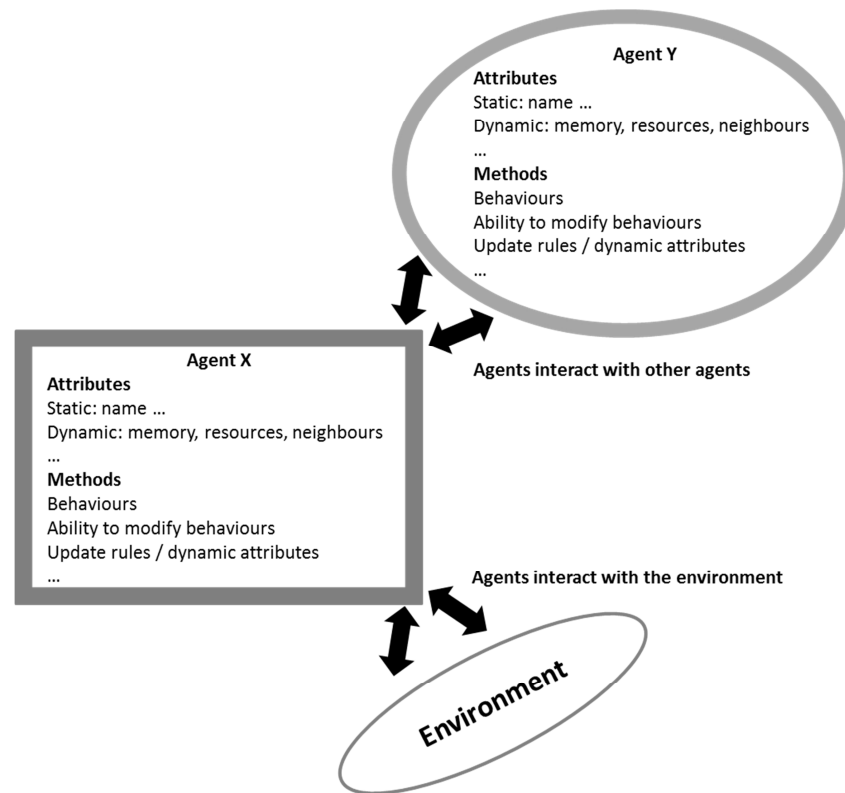


Figure 8: Agents' structure (source: Macal and North, 2010)

- Agents are active components of an environment and coexist with other agents, and, therefore, can be characterised as social components. Usually, communication protocols enable agents to interact with one another and their environment. Agents can recognise the behaviour of other agents.
- Agents are autonomous and self-directed. They have their own set of behavioural rules that dictate their decisions and actions. The degree of sophistication of these behavioural rules indicates the intelligence of the agent which is decided according to the scope of the model.
- Agents have a state that varies over the simulation time. The state of an agent is dictated by its state variables and can be a set or a subset of its attributes.

As mentioned earlier, agents can be heterogeneous entities that are characterised by their behavioural rules. The level of the behavioural rules sophistication depends on the agents' cognition, the agents' internal model of the external environment and other agents, the extent of memory of past events that agents use as experience for decision making. Also, the diversity of agents consists of different attributes and accumulated resources (Macal and North, 2010).

The way that agents are connected to each other constitutes the topology of the ABS model. Agents usually do not communicate with all the other agents in the space. A common concept of agents is the neighbourhood. Each agent can hold information about its local neighbourhood and the neighbouring agents and communicate with them. The agents can move in a number of different spatial topologies as shown in Figure 9.

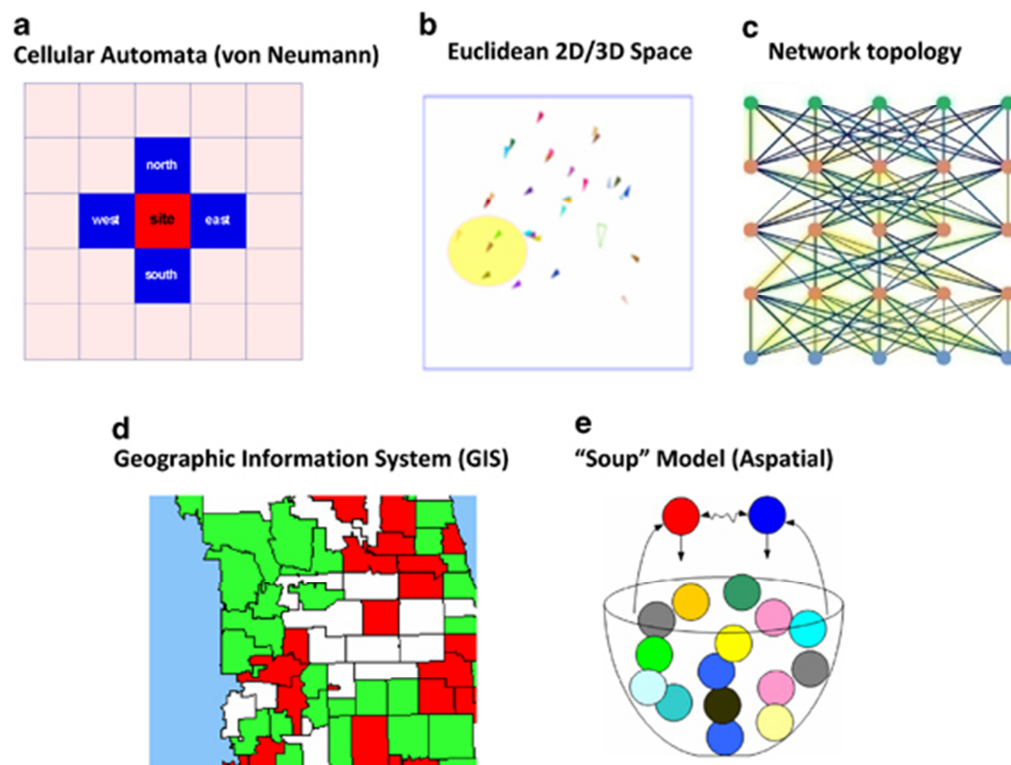


Figure 9: ABS topologies (source: Macal and North, 2010)

A very common spatial topology for agents is a form of cellular automata (CA). Agent move on a grid and their neighbourhood consists of the adjacent grid cells. A common grid neighbourhood is the von Neumann neighbourhood that consists of five cells, i.e., the central cell and the four adjacent cells that represent right angle directionality, as shown in Figure 9a. Another commonly used neighbourhood is the Moore neighbourhood that is formed of nine grid cells and includes the 45° angle directionality. In the Euclidean topology, agents can travel in two- or three-dimensional space, as depicted in Figure 9b. The radius of the agents' neighbourhoods is to be decided by the modeller. Figure 9c shows a network topology. The nodes of the network are the interacting agents and the links indicate the communication between the nodes. When the links are predefined, the network is called static. However, a network topology can be dynamic too. In a dynamic network, the communication links are changed during the course of the simulation. Sometimes the nodes of the dynamic network are changed, as well. Another popular topology for an ABS is the geographic information system (GIS), as shown in Figure 9d. Agents are moving on a realistic geospatial environment. GIS deployment gives a more realistic view of the model. Finally, agents can have no locality. This type of topology is called aspatial or “soup” model. The interactive agents are randomly selected and they return in the aspatial model for further selections. This topology is depicted in Figure 9e. It is possible one ABS model to comprise more than one topology.

In this thesis, the agents are moving on an Euclidean two-dimensional topology and are distinguished in two types. First, there are the active agents that have attributes, rules and behaviours, and generally the characteristics that are described earlier. Second, there are the passive agents that do not interact with other agents or the environment. They do not have the ability to learn and adapt their rules of behaviour. Therefore, from the agents' characteristics, they possess only attributes but not methods. Consequently, the passive agents are defined as agents that are part of the environment, they have attributes, and these attributes

contribute to active agents' behaviours, but they do not hold any form of intelligence. In Figure 10, the structure of this thesis ABS model is depicted, where between the environment and the typical active agents, there is a layer of the passive agents that support the decisions of the active agents.

### 2.7.1 ABS and object orientation

ABS programming can be directly related to object-oriented programming (OOP). Thus, OOP languages are usually used for developing ABS models. The main concepts of OOP can relate in some forms with the main characteristics of the agents and the underlying theory of CAS. The main concepts of OOP are described briefly here.

A *class* is a structure of certain attributes and methods of a system. It can be considered as the mould for creating objects. An example of a class can be “Trees” that describes the general characteristics of trees that are common to every kind of tree, i.e., height, age, leaves\_colour, etc. The “Trees” class has some methods too, i.e., grows(), dies(), etc. From the trees class, “tree” *objects* can be created. Every tree object is called an *instance* of a tree

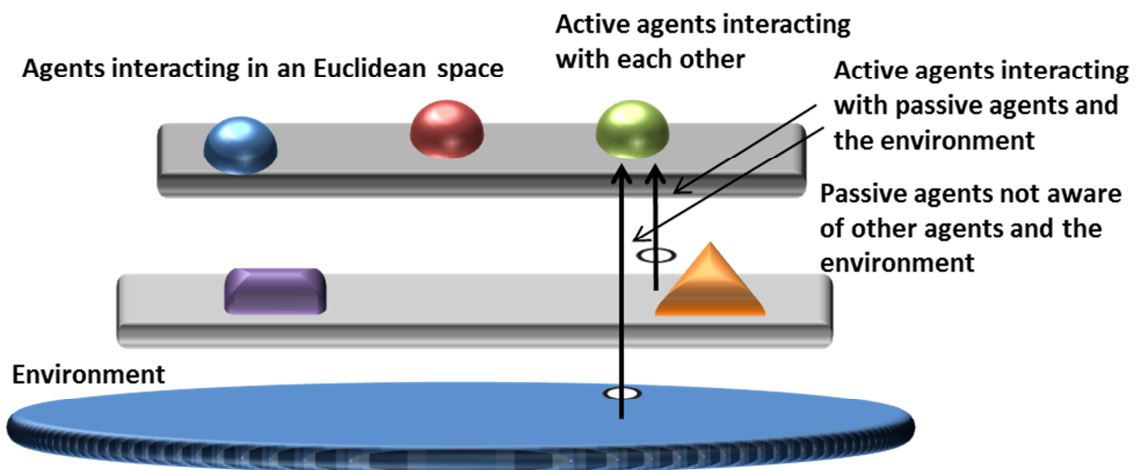


Figure 10: Active and passive agents concept of this thesis



and has all the *attributes* and the *methods* of the “Trees” class. From the “Trees” class, subclasses can be developed. These subclasses may be a more specified kind of tree, i.e., “ForestTrees” class, “FruitTrees” class, etc. These subclasses *inherit* all the characteristics, attributes and methods, of the parent “Trees” class and also they may have some additional attributes and methods that characterised the specific type. Each object can hide their internal components from other objects. However, the hidden components information can be accessed by other objects through methods. For example, the tree attribute “age” can be held in a private variable for a tree object, however it can be accessed by a public `getAge()` method. This concept is called encapsulation. Another concept of OOP is *polymorphism*. Polymorphism is the ability of the subclasses to differentiate from the parent class and yet share some common characteristics and functionalities. For example, the FruitTrees class share all the attributes and methods of the parent Trees class, i.e., height, `grows()`, etc., and also has some unique behaviours, i.e., `fruitProduction`, `fruitColour`, `pruning()`, `fertilise()`, etc. Also, in OOP, the concept of *message passing* between objects exists. That is, for example, an object can instruct another object to do something or invoke a method to be implemented. In the example that is used here, there may be another class which is called “Gardeners”. An object of this class, i.e., gardener object, can send a message to a fruitTree object to `fertilise()`.

From the characteristics of CAS and agents that were reviewed in the previous subsection and the concepts of OOP that are mentioned above, some relationships can be drawn, as shown in Table 4.

The concept of a class and the ability to create subclasses that inherit the elements of the parent class is related with the property of aggregation of ABS and the underlying CAS theory. Individual instances of objects are similar with the agents in ABS in a way that both have attributes and behaviours. In both ABS and OOP, agents and objects have attributes and methods. Attributes are variables that characterise the agents/objects and methods are

the functions that allow them to perform actions. The flows property of ABS is associated with the message passing concept of OOP. By the means of flows/message passing, agents/objects can communicate with other agents/objects and invoke some actions, i.e., an agent/object may ask another agent/object to perform a task. That is, for example, a customer agent/object asks the mechanic agent/object to repair a car. The agents are autonomous entities and, therefore, can hold private information and can decide what to share with other agents. This property is associated with the encapsulation concept of OOP. Similarly, objects can permit access to specific elements.

Table 4: Relationships between ABS and OOP

<b>Agent-based simulation</b>	<b>Object-oriented programming</b>
Aggregation	Class/Subclass/Inheritance
Agent	Object
Attributes	Attributes
Methods	Methods
Flows	Message passing
Autonomy	Encapsulation
Heterogeneity/Diversity	Polymorphism
Building blocks	Inheritance/Polymorphism

Agents are heterogeneous and diverse entities. Each agent can have different characteristics, even if it is generated from other agents. Likewise, in OOP, objects of subclasses can have different characteristics even if they are created from the same parent class. Therefore, polymorphism can be linked to the heterogeneity/diversity property of ABS. Finally, an ABS mechanism mentioned above is the ability to compose an agent from other agents which are conceived as building blocks. This can be related with the concepts of inheritance and polymorphism in OOP, i.e., a class can be composed by other classes, inherit what is required and introduce new characteristics, too.

## 2.8 Hybrid simulation

Hybrid simulation is when two or more techniques are combined in the simulation. The term applies in a broad range of simulation applications. For example, hybrid simulation may refer to the combination of physical and numerical models, of analytical and simulation models, of continuous and discrete time models, or, as it is in this thesis, of different paradigms of M&S.

In a review, contacted by Jahangirian *et al.* (2010), it was found that the hybrid simulation techniques have increased in popularity. The review included 281 articles that were published in peer-reviewed literature between 1997 and 2006. Complex and large systems that consist of heterogeneous subsystems are generally difficult to analyse using only one simulation paradigm (Zulkepli *et al.*, 2012). Therefore, hybrid simulation is considered beneficial for dealing with complexity rather than utilising a single simulation technique.

### 2.8.1 Hybrid ABS-DES

In this project, ABS and DES are used to form a hybrid simulation model of a large-scale healthcare system. Healthcare systems, in general, are characterised by high level of complexity, and more so, if the systems are large and consist of heterogeneous subsystems.

ABS is mainly used to study complex social phenomena or the behaviour of a system at an individual level. Agents have their own behaviours that are governed by rules and these behaviours can change during the course of the simulation. Cognition elements can be added to agents and, therefore, are intelligent objects that adjust their internal processes according to their goals. Agents have local view of the system and have the control of their actions. Therefore, ABS models are characterised as decentralised models.

DES is mainly used to analyse the processes of a system. Typically, systems that are modelled by DES are the queuing systems. DES is a centralised model, where there are system-level rules that decide the global state of the simulation. Some of the systems analysis areas that DES is commonly used for, are: queue management, “what-if” scenario analysis and process re-engineering.

Consequently, when the system under study consists of process level concern subsystems and individual behaviour concern subsystems, a combination of those two paradigms can be considered the most appropriate approach for simulating such a system. Hybrid simulation presents the difficulty that the modeller should have knowledge of more than one simulation technique, and possibly, tools (Brailsford *et al.*, 2013). Usually models are built independently and communicate in real- or logical-time.

## 2.9 Model reusability

The idea of simulation model re-use is not new. Re-use of simulation models has been debated among simulation experts and advantages and disadvantages of this practice have been discussed in the literature. As early as in the 1980s, Reese and Wyatt (1987) discussed simulation software re-use in terms of software re-use. They discussed mainly component re-use in a simulation software support level, such as time management routines, statistics collection routines, etc. However, re-use in the simulation model support level is discussed, too, in respect to domain specific applications. Pidd (2002) raised concerns about the cost

of simulation model re-use. Using a simple cost-benefit model for model re-use, he reached the obvious conclusion that re-use is worthwhile only if the average cost per use is less than the initial model development cost. Also, the vital issues of validity and credibility were discussed and it was pointed out that when the model is intended to be reused in a different domain that it was designed for, new validation and credibility assessment strategies should be followed.

Paul and Taylor (2002) viewed the re-use of models from the commercial packages modeller's point of view. They classified the re-use of simulation models into three categories: basic modelling component re-use, re-use of subsystem models, and re-use of similar models. They, particularly, emphasise the role that trust plays in reusing simulation models. Building trust to the existing model can be a costly and timely process, many times, more than building the model anew. However, modern technologies, such as web-enabled simulations, can support re-use of models, in a way of assisting modellers in better understanding the problem rather than assisting in model building.

In line with the above, Robinson *et al.* (2004) discuss the benefits and drawbacks of the different levels of model re-use. They mention, among others, the analogies with the object-oriented paradigm and the different artefacts of a simulation project that can be considered for re-use. One of the conclusions is that when modellers build simulation models with taking reusability into consideration from the early stages of the project, it is more likely to build re-use enabled simulations. However, there is limited motivation for modellers to do so. Nevertheless, careful recycling of simulation models has the potential to reduce modelling time and cost.

The analysis above can be summarised in the diagram depicted in Figure 11, where the different levels of reusability are depicted in a pyramid scheme. The base of the pyramid represents the lowest level of reusability, i.e., software support level, and has the least de-

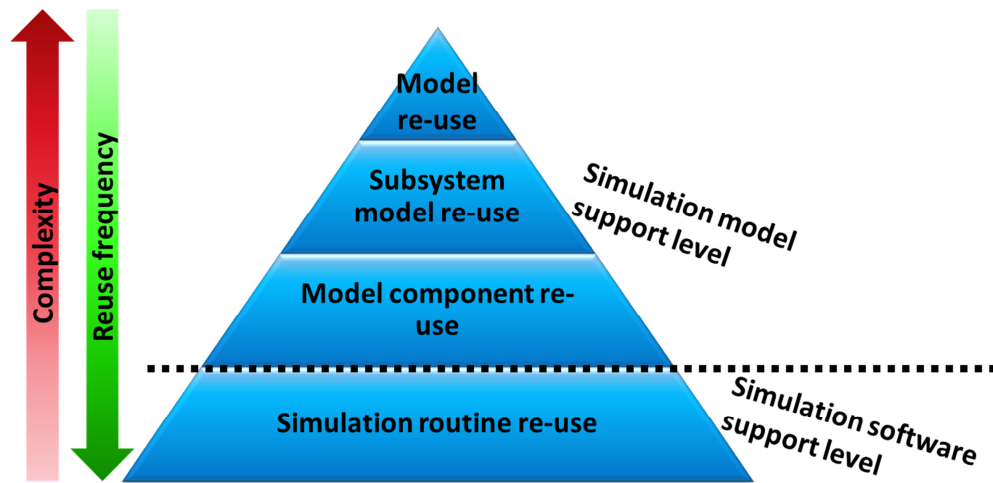


Figure 11: Re-use of simulation models

gree of complexity and the highest frequency of re-use. On the other end, the top of the pyramid represents the simulation model reusability that has the greatest degree of complexity and, consequently, the lowest frequency of re-use. The middle layers of the pyramid represent the less complex re-use types of the simulation model support level re-use, i.e., subsystem model re-use and model component re-use.

Balci *et al.* (2008) studied the simulation model re-use domain in terms of conceptual modelling. They argue that a high level conceptual model of an interest-specific domain has the highest level of applicability regarding the re-use of models. Further, they debate that a conceptual model with a high level of abstraction can be reused regardless of the simulation technique that will be selected for implementation. However, they discuss the re-use of static models rather than the re-use of dynamic simulation models.

Another view in M&S re-use is that of ontologies deployment. Ontologies, as the term is used in the information systems context, use well-defined languages to explicitly conceptualise and describe the relationships between the disciplines within a domain (Silver *et al.*, 2011). Turnitsa *et al.* (2010) analysed the implementation of two modelling ontologies for

DES systems. They noted that the ontologies present a semiotic representation of M&S and address the philosophical questions “who, what and why”. For example, “who” this model is in respect of its reality, “what” is known about this reality, and “why” this model should exist, i.e., the purpose of implementing the model. Both ontologies have been used for implementing DES models in different domains. In the level of ontology-based simulation re-use, Bell *et al.* (2008) developed an ontology for DES component re-use based on commercial simulation packages using semantic web services. It is a generic ontology for supply chain simulations that is structured in industry-specific classes.

After the latter additions in the modelling reusability analysis, another layer should be added in the re-use pyramid of Figure 11. Consequently, in Figure 12, the conceptual model support level of model re-use can be seen. Here, the reverse pyramid indicates that, in the ontology layer, the reusability frequency is potentially higher. The adverb “potentially” is added because, as yet, there has not been a significant number of ontologies developed for M&S. Therefore, the field is still immature. However, it is expected to present a lesser degree of re-use complexity than that of the conceptual model re-use. That is because it can be used in a wide area of applications, comparing with conceptual models that should be developed for a domain-specific application.

The focus of this thesis is on the subsystem model reusability. A key technology in achieving re-use in that level is DS. The issues that the modellers face when trying to reuse subsystem models, i.e., composability and data sharing problems, are discussed in Chapter 3.

## 2.10 Distributed simulation

DS, typically, can be defined as the distributed execution of a simulation program across multiple processors (Fujimoto, 2000). However, in areas outside computer science and operational research (i.e., medical training), it has been used to describe accessible and porta-

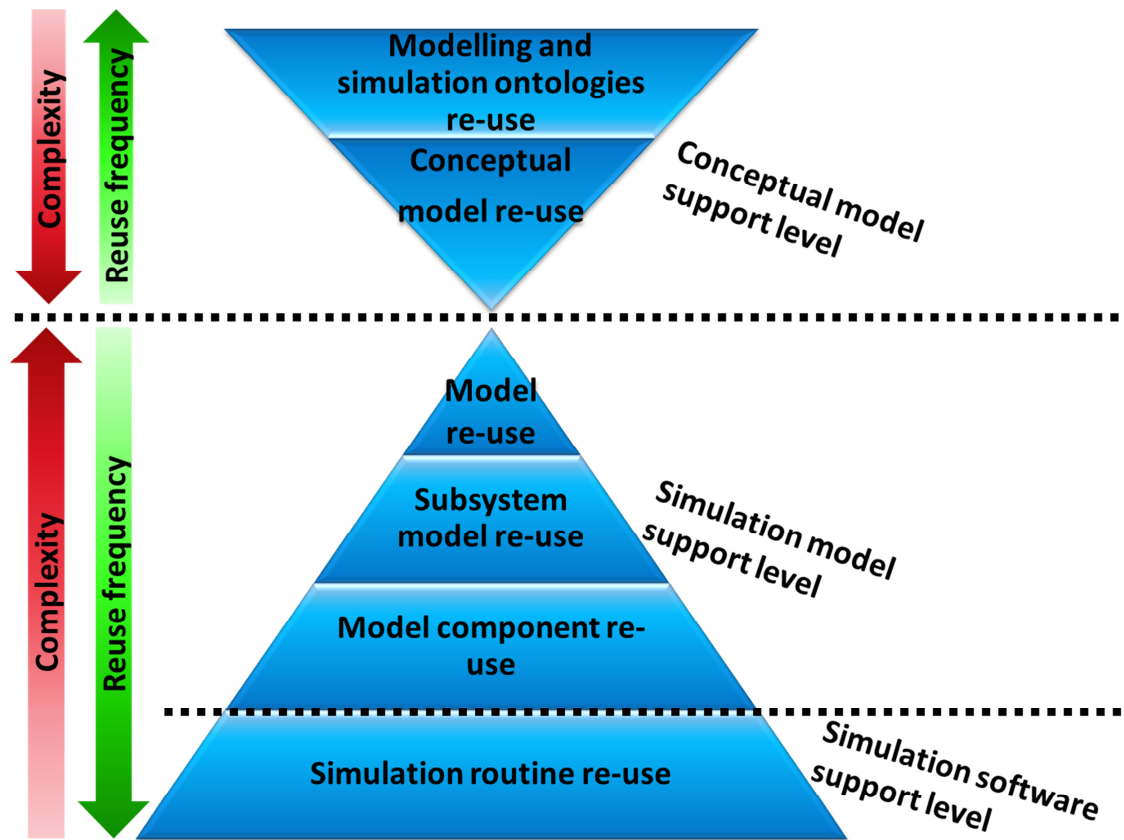


Figure 12: Levels of re-use of simulation models

ble physical simulation environments (Kneebone *et al.*, 2010). In the context of this thesis, DS refers to the execution of computer simulation models over multiple nodes in a networked environment that are linked by specialist software (frequently referred to as interface).

In a DS system, the participating simulation models are able to interoperate with each another. That is, the simulation models can send information to and receive information from other simulations and be able to operate effectively together, i.e., sending the right information to the right destination and at the right time, also without adding prohibitive communication time overhead. Hence, data and time synchronisation is essential in DS systems.



DS adoption is widespread in military applications. However other sectors are still lagging behind (Taylor *et al.*, 2002a). One of the main reasons for reluctance in DS adoption is the intense technical expertise that is required for implementing communication interfaces. Standardisation of the DS practices could contribute largely in overcoming this barrier (Taylor *et al.*, 2012a).

Several efforts have been made towards this direction. The most commonly used standard for DS is the IEEE-1516 High Level Architecture (HLA) which has replaced its predecessors Distributed Interactive Simulation (DIS) and Aggregated Level Simulation Protocol (ALSP) (Baker, 1999).

DIS is an IEEE standard (IEEE-1278, 1993) and was developed within the US DoD. The foundation for DIS was the SIMNet project which implemented one of the first large-scale real-time simulator networking environment. SIMNet was a battlefield simulation and used for training purposes. In DIS, each node broadcasts protocol data units (PDU), with data about the entity state, when an event occurs. DIS uses “dead reckoning” algorithms, a term borrowed from navigation, for self-corrections and compensation for lost datagrams. For example, if the receiving node does miss a PDU, it continues with the behaviour anticipated by the dead reckoning prediction. When the next PDU arrives, the behaviour will be corrected and a new extrapolation will be initiated (Miller and Thorpe, 1995).

ALSP was developed by The MITRE Corporation ([www.mitre.org](http://www.mitre.org)) within the US DoD. It was an initiative motivated by the limitations of DIS, i.e., supports only real-time simulations, does not provide time and data sharing management, and performs well only in local area network (LAN) environments. ALSP is a confederated protocol and supports communication between confederate DES models. Its most important additions to DIS were the data and time management services and the object ownership. The ownership of an object can change dynamically during the course of the DS execution.

Both DIS and ALSP were replaced by the HLA standard for DS. HLA is a set of standard rules that specify information sharing and coordination during the interactions of simulation models. It has been developed by the DMSO, now re-designated to MSCO, for the US DoD.

In this study, the DS project implements the HLA standard and thus, HLA will be analysed in more detail in the following subsection.

### 2.10.1 High Level Architecture

The HLA is an IEEE standard for DS developed by the US DoD, as mentioned above. HLA is a federated architecture; all participating individual simulations are called federates and the complete DS is called federation. The coordination of data exchange and time management occurs in the RTI component of the HLA which is connected with the federation through an interface (see Figure 13). The ultimate goal of HLA is to support interoperability and reusability of simulation models.

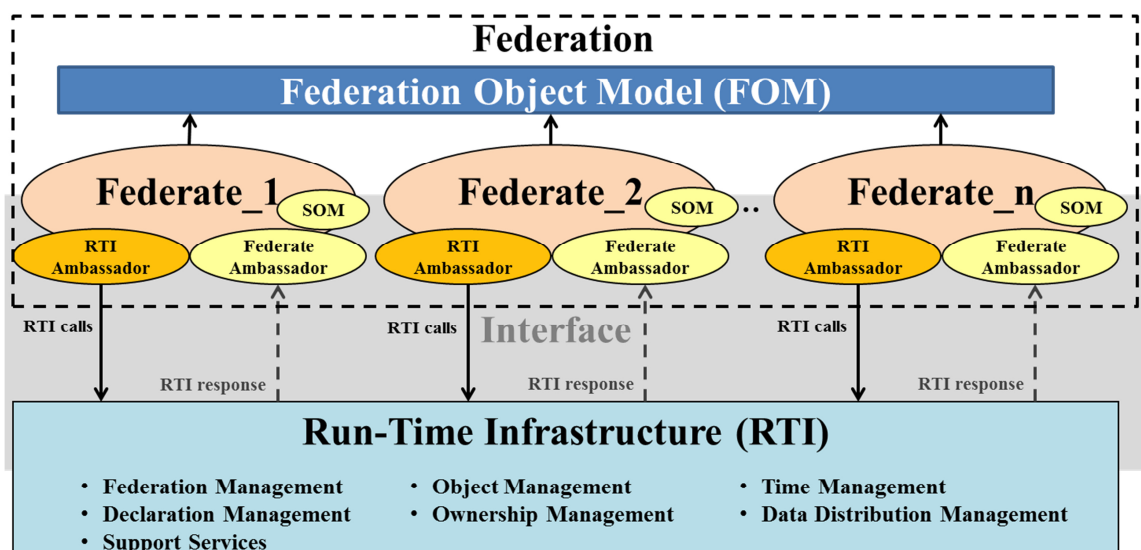


Figure 13: High Level Architecture

The main three parts of the HLA, i.e., federation, interface and RTI, as can be seen in Figure 13, are described by the three different components of the HLA, listed below:

- HLA Framework and Rules (IEEE-1516)
- Federate Interface Specification (IEEE-1516.1)
- Object Model Template Specification (IEEE-1516.2)

*HLA Framework and Rules* provides the ten HLA rules that every federation and federate must obey. There are five rules defined for the federations (1-5) and five rules defined for the federates (6-10) (IEEE-1516, 2010). In summary, the HLA rules dictate that there should be formalisation of information exchange within a federation. The supporting infrastructure should not have any information about each individual simulation but rather all federate attributes should be owned by the federate and maintained by it, not by the RTI. All communication of the federation object model (FOM) data among the participating federates in a federation execution should happen only through the RTI services. Federates should comply with the HLA interface specification. The ownership of an attribute can be dynamically changed during the simulation execution but it can never be owned by more than one federate, simultaneously; however, an attribute can be owned by no federate. There should be a clear description of each simulation federate (object classes, attributes, interactions) to support federate's reusability (Moeller and Loeffstrand, 2009). All internal representations and interactions that are made public in a federation execution should be stated in the simulation object model (SOM) of the respective federate. The attributes ownership requirements should be documented in the SOM of the respective federate. Each federate can update the owned attribute according to the local conditions, and these conditions should be documented. Lastly, federates should manage their local time so that it complies with the time management approach of the federation.

*Federate Interface Specification* documents the services provided by the interface during a federation execution, as well as programming languages mapping (Java and C++). There are seven service groups that provided by the RTI implementation, that is, Federation Management; Declaration Management; Object Management; Ownership Management; Time Management; Data Distribution Management; and, Support Services (IEEE-1516.1, 2010).

*Object Model Template Specification* provides a template, or a common structural framework, for describing the objects of a HLA DS. Its purpose is to provide a common understanding of the DS components and to assist in designing and developing potential federation members. Object Model Templates (OMT) define the individual federates (SOM) or the federation (FOM), or subsets of these, i.e., SOM modules and FOM modules. The OMT can be presented either in tabular format or in data interchange format (DIF), depending on the purpose of the document (IEEE-1516.2, 2010).

It should be noted that the terminology of HLA does not imply correspondence with OOP.

Apart from the main three parts for the DS implementation, HLA defines three more standards. The IEEE-1516.3 recommends a methodology for federation development and execution. In the “IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)”, procedures for implementing and experimenting with HLA simulations, and analysing the produced results are defined (IEEE-1516.3, 2003). The IEEE-1516.4 standard discusses verification, validation and accreditation (VV&A) processes for HLA simulations (IEEE-1516.4, 2007). The “IEEE Recommended Practice for Verification, Validation, and Accreditation of a Federation – An Overlay to the High Level Architecture Federation Development and Execution Process” provides guidelines for VV&A in the implementation level of HLA DS. Building upon

IEEE-1516.3, IEEE with the support of the SISO approved the IEEE-1730 in September 2010, which was approved by the American National Standards Institute (ANSI) in June 2011. The “IEEE-1730 Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP)” is a high level framework that describes recommended processes and procedures for DS development (IEEE-1730, 2010). SISO has developed two more standards relevant with the HLA DS systems: the Base Object Model (BOM) Template Specification (SISO-STD-003-2006) and the Guide for BOM Use and Implementation (SISO-STD-003.1-2006). The first document describes the BOM standard and the latter document provides guidance for developing BOMs. BOM specifies how to map conceptual model designs to FOMs.

#### **2.10.1.1 Time management**

Simulations are dynamic models. That is, the state of the model or its entities changes over time in a similar way to the state of the system that the model represents or imitates. Events that occur during the course of the simulation have a cause and an effect. It is of utmost importance for the correct functionality and output of a simulation program to maintain the causality of the events. For example, in a real system, a departure event can never precede the arrival event for the same entity. Therefore, this relationship should always be true in the simulation system, too.

In standalone simulations that run in a single node the causality of the events is managed by an event list or the simulation scheduler. The simulation engine ensures that the event with the smallest timestamp will be executed first. In a DS model, however, there are multiple simulation programs, that run in different nodes in a network and they communicate. An event that occurs in one federate can affect one or more federates in a federation. Therefore, it is critical to synchronise the federation execution, such as to maintain the local “cause and effect” relationships in each federate.

Time can have three dimensions in a simulation project: physical time, simulation (or logical) time and wallclock time (Fujimoto, 2000). The physical time refers to the actual time in the physical system. For example, a working day starts at 9.00am and ends at 5.00pm. The simulation time refers to the concept of time used by the simulation program to model the physical time. For example, a working day starts at  $T_l$  and has duration of eight time units, assuming that the simulation time unit corresponds to one hour of physical time. Finally, wallclock time refers to the time of the execution of the simulation program and depends on the hardware characteristics and the operating system. For example, the simulation of one working day may run from 7pm to 7.10pm in wallclock time.

The federates in a HLA federation execution can have either different simulation and wallclock times or synchronised. However, the federation must ensure wallclock synchronisation in order to deliver the messages in the right order and maintain the causality of the events. In the case that the simulation time and wallclock time are synchronised, the relationship between the simulation time and the wallclock time is described by the following equation:

$$\Delta T = S * \Delta W \quad (2)$$

where,  $\Delta T$  is the simulation time lapse,  $S$  is a scale factor  $\{S \in \mathbb{R} : S > 0\}$ , and  $\Delta W$  is the wallclock time lapse (Fujimoto, 1998).

When  $S = 1$  then the simulation is called *real-time simulation*. The virtualisation effect is realistic since the duration of the activities is the same as in real-life. When  $S \neq 1$  then the simulation is called *scaled real-time simulation* and can run faster or slower in relation to wallclock time. For example, if

$$\Delta W = \text{current wallclock time} - \text{wallclock time at start of simulation run}$$

$\Delta T = \text{current simulation time} - \text{simulation time at start of simulation run}$

and  $S = 2$  then  $\Delta T = 2 * \Delta W$

which means that the simulation runs twice the speed of the wallclock time. If  $\Delta W = 1$  hour, then in 1 hour wallclock time the simulation executed 2 hours of simulation time.

Real-time simulation is commonly used in training systems and gaming, where the virtualisation should appear realistic.

However, when there is no direct relationship between the simulation time and the wallclock time, the simulation is called *as-fast-as-possible simulation* and attempt to complete the execution the quickest possible. In as-fast-as-possible simulation, the simulation time and wallclock time relationship varies during the course of the simulation execution. Such simulations are usually used for the purpose of analysing a system.

In a HLA federation, simulation time can be envisioned as a global *federation time axis*. During the execution of the simulation and at any instance of time, each federate's simulation time is a point along the federation time axis and is called *federate time* or sometimes *logical time* (Fujimoto, 1998).

Time management services, in a HLA federation execution, are implemented programmatically in the RTI, which is responsible for the time advancement of the federation. These services are responsible for synchronising federation and federates time across the DS. Another functionality of the RTI, regarding the time management, is to ensure that the transportation of the messages is performed efficiently. Efficiency is defined by terms of reliability, latency (network delay) and bandwidth consumption (Fujimoto, 1996).

Transportation services employ different techniques for message passing. RTI can transmit messages in Received Order (RO), Priority Order (PO), Causal Order (CO), Caus-

al and Total Order communication support (CATOCS) and Timestamp Order (TSO). RO refers to the mechanism that puts all receiving messages in the end of a FIFO queue and then sends them in order from the head of the queue. PO refers to the technique that the RTI sets the messages in a priority queue, with the timestamp being the priority criterion, and sends the message with the smallest timestamp first. CO technique ensures that a cause event will be transmitted before the effect event. CATOCS extends the CO technique and guarantees that all messages, not only the cause-effect related, will be delivered to all interested federates in exactly the same order. Finally, TSO mechanism delivers messages in the order of their timestamp. At the same time, it ensures that the federates will not receive any message with timestamp less than the federate's current logical time, as opposed to PO, that cannot guarantee that the federate will not receive messages with timestamp that refer to their logical past (Fujimoto, 1996).

There are two services for simulation time advancement: Time Advance Request (TAR) and Next Event Request (NER). Usually, TAR is used in time-stepped federates and NER in event-driven federates. RTI will not send any messages unless the federates request time advancement, then, after the RTI ensures the correct messages transmission, will permit the logical time advancement using the Time Advance Grant (TAG) service. In TAR, the federate asks time advance usually to the next time step. The federate and RTI send all messages with timestamp less or equal to the time of the next time step. When the RTI can guarantee the transmission of all messages, then it will grant time advancement and the federate is allowed to advance its local logical time to the time of the next time step. In NER, the federate asks time advance to the time  $T$  that the next event is scheduled to occur, locally. If there are no requests from other federates for time advance to time  $T'$ , where  $T' < T$ , then the RTI will send and receive all the timestamp messages with timestamp less than  $T$  and grant the logical time advance to  $T$  (Fujimoto, 1998).



Time management is of utmost importance in DS. Due to communication network delays, federates may not receive messages in a temporal order. This, likely, will cause disturbances in the causality order, i.e., the effect will occur before the cause. An example is portrayed in Figure 14, where there are four federates in the federation, an ambulance service federate, an A&E federate and two federates A and B that receive information for system analysis purposes. In the depicted scenario, the ambulance service federate arrives at the A&E to handover the patient, the message that a patient has arrived is received by the A&E federate that models the arrival of the patient. Both federates send messages to federates A and B. Federate A receives the messages in temporal order, therefore, maintains the cause-effect relationship. But, federate B first receives the message from the A&E federate, hence, this federate sees first the patient to be received at the A&E and then the ambulance arrival at the A&E location. As a result the analytical information that federate B collects does not represent the system. Time management would ensure that federate B receives the messages correctly.

Another important concept in HLA time advance service is the *lookahead*. Lookahead is

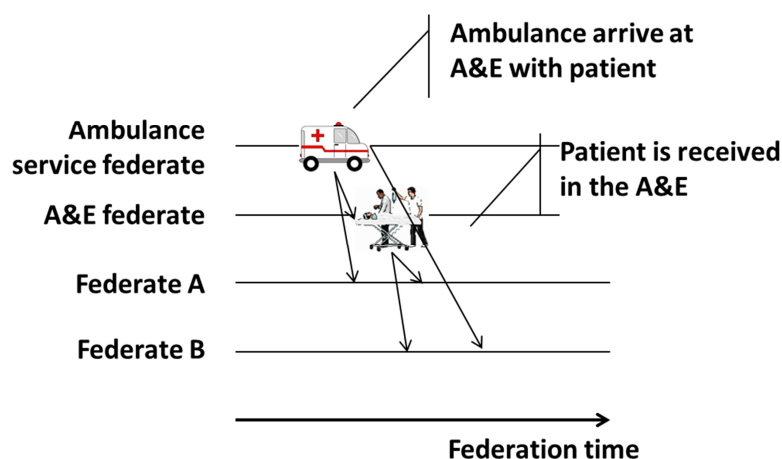


Figure 14: Time management example

the minimum distance in the federate's logical future time that an event will be scheduled. All transmitted events must have timestamp more or equal to the current logical time plus the lookahead. Lookahead is federate-specific and can change dynamically during the federation execution (Fujimoto, 1996).

Finally, the RTI is responsible for time synchronisation within the federation. The two approaches that can be implemented in RTI are the conservative and the optimistic synchronisation techniques. The conservative synchronisation ensures that the RTI will send messages with timestamp less than the current simulation time of a federate. This comes with a cost on synchronisation time overhead. The optimistic synchronisation performs better in terms of latency. The optimistic time management federation allows violation of the local causality constraints, however, the optimistic algorithms provide error recovery mechanisms in the case that a causality error has occurred. Therefore, optimistic synchronisation supports rolling back in simulation time and recovering system states prior to the error event.

### **2.10.2 Interoperability reference models**

As mentioned earlier, in a DS system the participating individual simulations must be able to interoperate (Mustafee and Taylor, 2006). However, there are numerous problems that modellers face when developing interoperable models for DS. For example, when an entity is passing from one simulation to the other, it is not received in the same way as a resource, for instance. The entity will be placed usually in a queue in the receiving model, however, a resource will be utilised to complete an activity. Therefore, it is of crucial importance, for understanding the interoperations between interacting simulations to define these issues (Taylor *et al.*, 2012b).

The SISO standardisation group CSPI PDG (commercial-off-the-shelf (COTS) simulation package interoperability product development group) developed and standardised a series of IRM that address interoperability issues between DES models.

As defined by (SISO-STD-006-2010), an IRM is the simplest representation of an interoperability problem type and can be divided into different subcategories. IRMs are relevant to the boundaries between two or more interoperating models. The models are represented in the simplest possible way to capture the interoperability problem and to avoid possible confusion. Time synchronisation requirements are specified only where appropriate. According to Taylor *et al.* (2012a) there are four types of IRMs:

- Type A: Entity transfer – a model may pass an entity (or agent) to another model. For example, a patient agent from the ambulance ABS model can be passed as a patient entity to a hospital DES model. There are three sub-types for entity transfer between models:
  - Type A.1: General entity transfer – describes the direct pass of an entity from one model to the other when no feedback is needed.
  - Type A.2: Bounded receiving element – describes the case when the receiving model has a limited receiving queue. When the queue is full the receiving model has the ability to block the entity passage until the queue is free to receive this entity. The rest of the sending model continues to run.
  - Type A.3: Multiple input prioritisation – this is the case when there are multiple sending models and the receiving model needs to prioritise entity entries when these entities arrive at the same time.
- Type B: Shared resources – deals with models that have shared resources. In a hospital setting this could be the case, for example, when two hospitals share specialised surgeons.

- Type C: Shared event – deals with models that have shared events. For example, when a variable in a model reaches a threshold value, this model signals this fact to all involved models for an event to occur.
- Type D: Shared data structures – deals with models that share data structures. For example, when two or more models share a global variable or a data set.

Defining the IRM is the first step in building a DS system. IRM belongs to the conceptual phase rather than the implementation phase of a DS project.

## 2.11 Software and tools

The first Simulation Programming Language (SPL), namely General Simulation Program (GSP), was developed by Tocher around 1958 (Tocher and Owen, 1960). From the early 1960s until now there was an evolution in simulation languages (Goldsman *et al.*, 2010) and in the more recent years integrated software packages, also known as simulation environments, with graphical user interfaces, animation and other visualisation tools emerged in the market (see Figure 15).

Period of Search	Advent	Formative Period	Expansive Period	Period of Consolidation & Regeneration	Period of Integrated Environments
1955-1960	1961-1965	1966-1970	1971-1978	1979-1986	1987-present
Simulation languages					Simulation environments
FORTTRAN	GPSS	GPSS variations	GPSS-H	SLAM II	ARENA
GSP	SIMSCRIPT	SIMSCRIPT II	GASP-IV	SIMAN	AnyLogic
	GASP	ECSL	SIMULA		Simul8
	SIMULA	SIMULA with classes & inheritance			NetLogo
	CSL				VenSim
					MicroSaint
					WITNESS
					etc

Figure 15: Simulation software evolution

Nance (1995) divided the history of SPLs into five periods:

- 1) 1955-1960: Period of Search
- 2) 1961-1965: Advent
- 3) 1966-1970: Formative Period
- 4) 1971-1978: Expansion Period
- 5) 1979-1986: Period of Consolidation and Regeneration

In Figure 15, one more time period was added, the period of Integrated Environments from 1987 to present, in order to add the recent developments in SPLs (*Banks et al.*, 2000). In the first period, efforts were made to conceptualise simulation and to find ways of representing the models. This is when GSP was developed. During the advent period, the General Purpose Simulation System (GPSS) was developed on several IBM computers. In 1963, SIMULA, SIMSCRIPT and the Control and Simulation Language (CSL) were appeared. SIMULA was most popular in Europe and considered to be descendant of ALGOL, while SIMSCRIPT and CSL were heavily influenced by FORTRAN. In the same period, the General Activity Simulation Program (GASP) was developed based on both ALGOL and FORTRAN. Other SPLs that emerged in the advent period were, DYNAMO, SIMPAC, Simulation Oriented Language (SOL) and MILITRAN. During the third period, there was further development and clarifications in SPLs. New languages of this period are the Burroughs operational systems simulator (BOSS) and Q-GERT, among others. In the expansion period, visualisation and databases were introduced in SPLs. During the fifth period, two important SPLs emerged: simulation language for alternative modelling (SLAM II) and simulation analysis (SIMAN), both considered GASP descendants (Nance, 1995; Nance and Sargent, 2002). From 1986 onwards, with the increase of computer capabilities, several sophisticated simulation packages appeared in the market. These packages are complete simulation environments with graphic interfaces that enable analysts to perform simulation studies with limited or no coding.

According to Banks et al. (2000), the simulation software is organised into three categories:

- General-purpose programming languages, such as C, C++, and Java,
- Simulation programming language, such as GPSS; and
- Simulation environments, such as the Visual Interactive Modelling Systems (VIMSS) that support all (or most) aspects of a simulation study.

DES software tools are, in their majority, commercial packages with sophisticated 2D and 3D visualisation capabilities. Some of these tools are: SIMUL8, AnyLogic, Arena, Flexsim, WITNESS, Simio, etc.

As mentioned in Pidd (2004), most packages implement a process-based approach simulation engine, although, this is not obvious to the user. DES software tools progress time in irregular time intervals. The simulation time “jump” depends on the time that the next event is due to occur. The software looks up for the next event in an event list that is maintained for the future events.

ABS simulators are mainly open-source free software tools, such as NetLogo, Repast, StarLogo, MASON (Multi-Agent Simulator Of Neighbourhoods (or Networks)), etc. However, there are commercial packages too, such as AnyLogic. The world view that is implemented is usually the object-oriented world view that looks at the system as individual components that can interact with each other. For example, an agent object can invoke a method that belongs to another agent. The simulation time advances in fixed-time steps. In every time step, the simulation engine scans through the system’s agents and executes the methods where their conditions are met. Usually, the time step is equal with one simulation time unit. For example, in a population ABS model, the simulation engine is scanning all agents every simulation time unit, once an agent reaches the reproduction age (age can be an attribute of the agent), a reproduce() method will be executed, and an offspring object

instance will be created. As mentioned above, many ABS simulators exist, most of them open-source free software. Perhaps the most well-known commercial package is AnyLogic that provides ABS, DES and system dynamics modelling paradigms environments, with the ability to create hybrid simulations. Other commercial packages, some of which are application-oriented, include: AgentBuilder, AgentSheets, MASS (Multi-Agent Simulation Suit), etc.

For the DS interface and the implementation of the HLA standards, there are several tools presently available. There are commercial RTI implementations, such as Pitch pRTI, MAK High Performance RTI, OpenSkies RTI, SimWare RTI, RTI NG Pro, etc., and open-source free implementations, such as Open HLA, OpenRTI, poRTIco, jaRT, etc.

Open-source software (OSS) packages, as opposed to commercial, are cost-free platforms with accessible source code. This makes OSS flexible and sometimes customisable. Furthermore, as the open-source community supports the dissemination of knowledge, there are several networks from enthusiastic users that provide speedy help and support. OSS is often a valuable tool for researchers. The main advantages of OSS are the cost-free usage and distribution, the accessibility of the source code and that it is independent of commercial companies. The main disadvantage is that open-source applications are not straight forward to use and generally it requires high technical skilled and trained users.

## **2.12 Summary**

This chapter presented a review of the related literature and framed the theoretical context of the thesis. The relevant concepts were analysed based on the reviewed literature.

First, there was a review of the literature on research articles that study aspects of EMS using modelling and simulation.

Subsequently, an analysis of the simulation methodologies that are relevant to this project and support the completion of simulation projects was provided. Then, the different simulation world views (or conceptual frameworks) that support the logic of simulators' implementation were analysed and explained.

For this thesis, a hybrid simulation approach is adopted. Therefore, the two simulation paradigms that constitute the hybrid model were discussed in detail, as well as the concept of hybrid modelling. The first simulation technique that was discussed was the DES that is an event-driven technique and studies a system as a set of processes. The second simulation technique that was discussed was the ABS that is a time-stepped technique and studies the system in terms of its individual objects and their interactions.

The proposed Framework for Integrated Emergency Medical Services Large-scale Distributed Simulation (FIELDS) supports model re-use. Thus, the issues on simulation models reusability were discussed in more detail. The different levels of model reusability, i.e., simulation software, simulation model and conceptual model re-use issues, were analysed.

Another aspect of the presented approach is the DS utilisation for the integration of the component models. Therefore, the theoretical background of DS is presented in this chapter, too. The standards for DS were discussed, with the IEEE-1516 HLA, which was implemented in this thesis, to be analysed in depth. Also, the SISO-STD-006-2010 standard for IRMs was analysed, which was also implemented in the current thesis.

Finally, software tools that are relevant to the presented project are discussed.

In the following Chapter, the distributed simulation methodology, that derived of the current project is presented, together with the underlying rationale of its development. In the subsequent Chapters, the conceptual design of the model developed in this project, following the proposed methodology, is discussed. The implementation process and the exper-



imental design are analysed. Further on, the evaluation of the proposed distributed simulation methodology and the FIELDS framework is presented and the concluding remarks of this thesis are discussed.

# CHAPTER

# 3

---

## **Towards a distributed simulation methodology for large-scale hybrid modelling**

*This chapter presents the development of the proposed distributed simulation methodology for developing large-scale hybrid ABS-DES DS models. It analyses the issues and challenges of the selected system, which is the EMS, and the rationale behind the chosen technologies.*

### 3.1 Overview

In the previous Chapter 2, the gap in the literature was identified. Further, there was a discussion on the theoretical context to support the hypothesis of this research, which is that it is feasible to develop a hybrid DS model for analysing EMS as a holistic system that consists of independent components. As stated earlier, within the context of EMS and for the purpose of this research, the suitable simulation techniques for studying the system are ABS and DES M&S. The theoretical background of both paradigms was explained in detail in the previous Chapter. Another aspect of the presented approach is the DS utilisation for the integration of the component models. Again, the theoretical background of DS was presented in Chapter 2.

This Chapter clarifies and justifies further the statement of the previous chapter that ABS and DES are arguably the appropriate simulation paradigms for modelling EMS. The semantic relationship between ABS and DES is presented. Moreover, the selected interface approach, namely DS, is compared with the alternative option of standalone modelling. After reviewing the above, the proposed distributed simulation methodology development process and rationale is described. Built upon existing strategies for simulation projects construction, the present methodology expands the scope in order to incorporate the main concepts of this research, namely hybrid ABS-DES DS M&S. In general, the presented methodology can be seen as a set of rules or steps for building hybrid ABS-DES DS models. The concept has been conceived from the healthcare sector, and specifically the EMS, but its applicability can be expanded to any system that can be modelled in similar way. That is, systems that can be modelled as a combination of individual ABS and DES models.

Chapter 3 is organised in five sections. First, there is a discussion on potential issues that the modellers are facing when attempting to develop large-scale simulation models of EMS. Then, there is an argument over single technique and hybrid simulation approach and

which of the two is more appropriate for EMS modelling. Further on, the semantic relationship between the modelling techniques that constitute the hybrid model is discussed. In the next section, the appropriateness of the selected simulation techniques and their implications are conferred. The following section discusses the issues of standalone modelling as opposed to DS and justifies the DS selection as an integration technology. Finally, the last section presents the proposed methodology for constructing hybrid DS models.

### **3.2 Issues on modelling the EMS**

EMS are complex, heterogeneous and multidimensional systems that provide immediate care to patients with acute illnesses or serious injuries. In the past, the role of EMS was to offer transport to those patients that were unable to transport themselves to the hospital. Nowadays, at least in the developed countries, EMS offers pre-hospital care on the site of incident and during transport to the hospital.

EMS is accessible to the public by an emergency telephone number that put them through a control centre. The control centre personnel initially assess the incidents and find and dispatch the appropriate emergency vehicle and crew. EMS can be public, private or voluntary organisations. Usually, the offered services are classified into two categories, the basic life support (BLS) and the advanced life support (ALS). BLS deals with less serious illnesses and injuries and the crew does not have medical training, whereas ALS deals with serious illnesses and injuries, where the flashing blue lights are on, and the crew has medical skills, i.e., paramedics and emergency medical technicians.

The structure and functionality of EMS vary considerably worldwide. However, this project focuses on the UK's EMS and thus when the term "emergency medical services", either in full or in the abbreviated form, is mentioned the reference is for the UK's EMS.

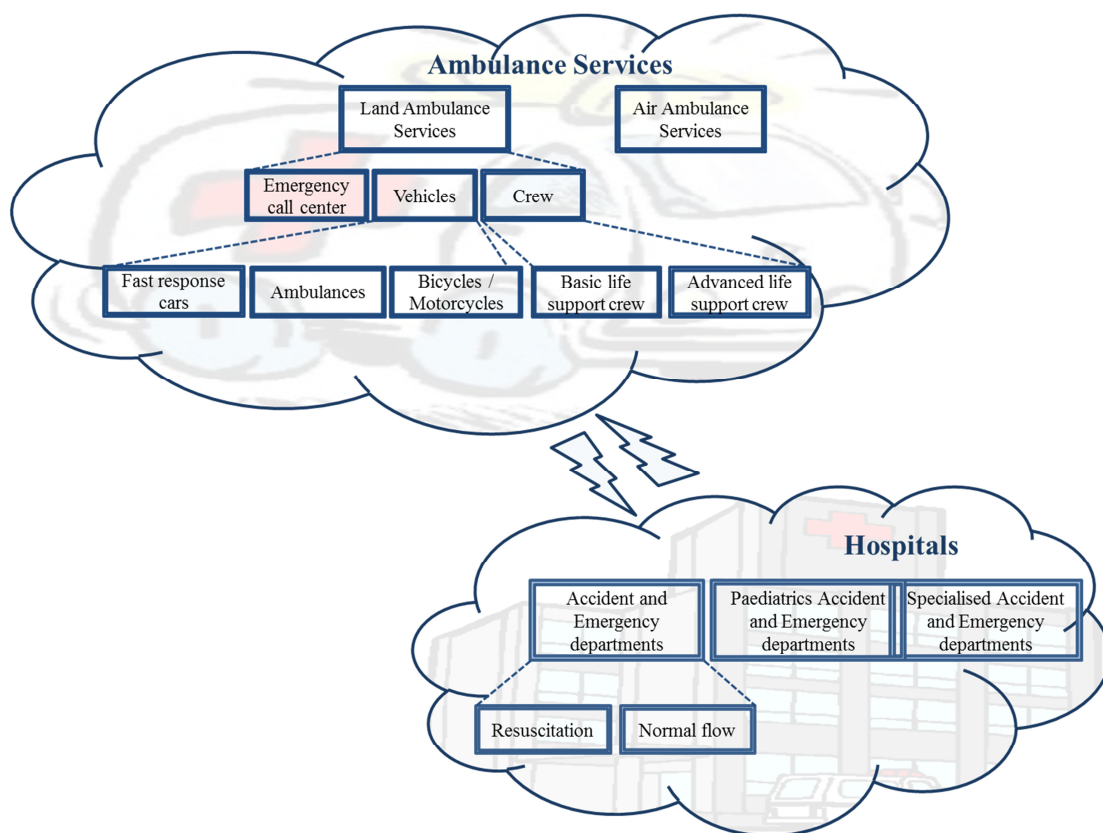


Figure 16: Emergency medical services

As stated earlier, EMS offers pre-hospital care. However, its role does not end there. Another important and critical service offered is to transport patients, unable to travel by their own means, to hospitals. A timely response and transfer to the regional hospitals' A&E departments, more than often, has saved the life of the patient. It is clear that there should be close collaboration between the ambulance service and A&E departments. As it is evident from the published literature (see subsection 3.5.2), the majority of EMS simulation projects consider only the ambulance service up to the point of patient handover (Aboueljjanane *et al.*, 2013). In this thesis, a comprehensive perspective of EMS is taken (see Figure 16) in order to perceive a global view of the emergency systems. Thus, in order

to model the whole EMS, the A&E departments of the hospitals should be included in a realistic simulation model and operate in collaboration with the ambulance service.

### **3.3 Hybrid simulation or single approach?**

As discussed in the previous section, EMS consists of the ambulance service and the A&E departments, usually situated in the hospitals of the coverage area. The fact that two different organisations are involved raises the question, whether a single simulation technique is capable to accommodate the functionalities of those two organisations.

The ambulance service model, in a rough outline, includes the emergency call centre, or centres, the vehicles and the crews. The call operators have to respond to the emergency call, assess the incident severity in order to send the appropriate vehicle and crew, find the closest available vehicle to the site of the incident and send the vehicle to the patient. The crew, in turn, apart from the medical treatment on site, has to decide whether the patient needs to be transferred to a hospital or released after the on-scene treatment. If the patient has to be transferred, the closest available hospital and the fastest route should be found. All the above indicate a high degree of interaction among the system's objects. From the background theory on M&S, presented in Chapter 2, the most promising simulation technique to realistically capture and represent all the interactions of an ambulance service model is the ABS. In the following Chapter 4, the particular issues on implementing the ambulance ABS models will be analysed in detail. For example, what agents will populate the system, in what environment they will act, and how they interact with each other and their environment?

On the other hand, the A&E departments in a hospital are highly process oriented organisations. When a patient arrives at the A&E, generally and regardless of the means of arrival, a treatment decision is made according to the patient's condition. If the required resources are available, i.e., cubicle or bed, nurse or doctor etc., the patient progresses

through the system to the next activity. Otherwise, the patient enters a queue until the formerly mentioned resources became available. Similarly, in DES, an entity passes through the system processes, being processed when there are available resources or waiting in a queue, otherwise. The entity's state can be changed in accordance with the model's activities. But, the entity itself is not able to take decisions or interact with the objects of the simulation, but rather it is driven by the system's processes until, eventually, exits the simulation.

Another motivation factor for selecting hybrid approach is the issue of model reusability. From literature, it is obvious that the most popular approach for A&E departments modelling, so far, is DES. For example, a search in the Web of Science database with keywords “discrete event simulation” and “emergency departments” yields 96 publications from 2000 to 2014, while in the same database and for the same period using the keywords “agent based simulation” and “emergency departments” yields 13 results. Having this in mind, the proposed FIELDS framework's ambition is to enable re-use of the existing models in a comprehensive EMS simulation system. So being able to use both is an advantage.

### **3.4 Semantic relationships between ABS and DES**

To achieve interoperability between systems modelled with different simulation techniques, the semantic relationships between the fundamental notions of these techniques should be defined.

The FIELDS framework, which is presented in Chapter 4, involves interoperability between ABS and DES models. Both paradigms can be described as discrete time-stepped simulation techniques. As opposed to continuous time models, the simulation progresses by leaping from one event to the next (the concept of event in M&S was analysed in Chapter 2). However, ABS is time-driven, that is, the simulation progresses according to a pre-set time step, while DES is event-driven, that is, the simulation progresses according to a

schedule of events. In the latter case, if there is no event scheduled for the next simulation time unit, the software will “jump” to the simulation time unit that an event is scheduled to happen.

Moreover, in DES, three main world views are noted (Overstreet and Nance, 2004), or else known as conceptual frameworks (Derrick *et al.*, 1989). These are *event scheduling*, *activity scanning* and *process interaction*. In the event scheduling world view, the focus is on the events of the model and the simulation looks at all the events that may occur at a specific simulation time. The activity scanning focuses on the activities of the system that may occur due to a specific parameter. Finally, the process interaction emphasises the objects and describes the process of the particular entity. The different world views of the DES are fully described in Chapter 2. In ABS there is no differentiation in the conceptualisation stage. The agents are located in an environment, which usually consists of passive agents, and follow some rules that define their routines.

ABS and DES are both microscopic simulations. This is because they model a system at individual level. Therefore, in this research a horizontal composition is implemented. As it is defined by Davis and Anderson (2003), horizontal composition is the composition of models with the same resolution as opposed to vertical composition which involves models with different resolutions or levels of detail.

Nonetheless, the individual objects in ABS systems are active objects that possess behavioural rules, while in DES systems are passive objects that are driven by the processes in the simulation. In ABS, the resources, which are fundamental components of DES, are agents. In DES, there is a distinctive meaning of a queue that agrees with the queuing theory, while in ABS, the agents just wait for an activity and there is no concept of a queue (Siebers *et al.*, 2010). Property and attribute hold the same underlying meaning in ABS and DES, respectively. Event and activity are two terms that are met in both techniques and



share the same meaning. In ABS, a set of rules defines the steps that the agent will follow, and can be associated with the system-level rules of DES that define the process that an entity will flow through. The environment in DES serves visualisation purposes mainly and the entities are not aware of it. However, the system can hold information about the entities' environment, e.g., location. In ABS modelling, the agents interact with the environment and learn from it. Finally, the term *state* holds similar semantics (meaning) in both techniques. All the above are summarised in Table 5 where the semantic relationships of the basic notions between the two simulation paradigms are listed.

### 3.5 DES and ABS modelling and simulation for EMS

DES is a technique that models a system's behaviour as discrete events. That is an instant in time at which an entity enters or leaves an activity. An activity changes the state of an entity (Pidd, 2004), as explained in Section 2.5.1.3. DES is being used in various industries to analyse process behaviours within a system (Brailsford *et al.*, 2009b; Robinson, 2005).

A more recently adopted simulation technique is ABS. Agents are individuals that have certain properties. The agents interact with other agents and the environment of the system. As a result, these interactions change the agents' properties, which define the agents' behaviour. ABS is being used mainly to analyse individual behaviours within a system. Arguably, ABS is continuously gaining in popularity within the simulation community. One of the reasons is its similarities with the object-oriented paradigm (North and Macal, 2007).

Regardless of the fundamentally different philosophies of the two simulation paradigms, there are significant similarities, too. For example, in both DES and ABS, the simulation time progresses in discrete time steps (Pawlaszczyk and Strassburger, 2009) and both are detailed level simulations.

Table 5: Semantic relationships between ABS and DES

Agent-based simulation		Discrete event simulation	
Term	Semasiology	Term	Semasiology
Agent	Active object that make decisions and change its behaviour.	Entity	Passive object that flows through the processes.
		Resource	
Waiting	Agents just waiting to perform an activity.	Queue	A building block that stores the entities while waiting for a service. It agrees with the queuing theory.
Property	Defines the agent's characteristics.	Attribute	Defines the entity's characteristics.
Event	A specific time that an activity begins or ends.	Event	A specific time that an activity begins or ends.
Activity	Action that starts and ends with events.	Activity	Action that starts and ends with events.
Rule	Defines the routine that an agent will follow.	System-level rules	Defines the routine that an entity will flow through.
Environment	Agents can interact with the environment and learn from it.	Environment	Entities are not aware of it.
State	The set of agent's properties at a specific time. The state of an agent can be changed by an activity.	State	The set of entity's attributes at a specific time. The state of an entity can be changed by an activity.
Time-driven	The simulation progresses according to a pre-set time step.	Event-driven	The simulation progresses according to a schedule of events.

### 3.5.1 Level of abstraction

By definition, a model is a representation of a real system. When modelling for computer simulation, the analyst has to decide the amount of data to be included in the model. Level of abstraction is the term that describes the amount of information used to constitute the model. The level of abstraction and the quantity of information in a model are inverse pro-

portional quantities. For example, a high level of abstraction means that the model contains fewer details.

A good practice for modelling a system is to keep it as simple as possible. As mentioned in Pidd (2004), following the principle of parsimony can lead to a more understandable model, since the parsimony paradox states that, as a model of a complex system becomes more complete, it becomes less understandable. There need not be a one-to-one correspondence between the model and the real system. Especially when modelling large and complex systems, the simulation model can only be an approximation of it.

However, some simulation techniques require lower level of abstraction. This is because they require individual modelling of the system's objects. Both ABS and DES are such approaches and can be classified as microscopic simulations.

In Table 6, the main characteristics of ABD and DES, at a conceptual level, are summarised.

### **3.5.2 Simulation studies in EMS**

Many simulation studies in the area of EMS exist in the literature. Ramirez *et al.* (2011) present a DES model of emergency care delivery systems. They use simulation optimisation techniques to analyse ambulance diversion policies. They implement the model in a modular approach where emergency incidents generation, ambulance decisions, and hospitals are separate modules. Su and Shih (2003) use simulation to evaluate and suggest improvements of EMSs. They use DES technique to model the Taipei EMS. The coordination and synchronisation of the BLS and ALS units is done by an event controller object implemented in eM-Plant software. Their study focuses on EMS performance by testing different dispatch scenarios taking into account personnel utilisation and waiting times.

Table 6: ABS and DES characteristics

<b>Simulation Approach</b>	<b>Level of Abstraction</b>	<b>System Analysis Approach</b>
Discrete Event Simulation	Microscopic	Top Down & Bottom up
Characterised by states and events. State variables are probabilistic distributions and the next event is triggered by the state variables.		
Agent-based Simulation	Microscopic	Bottom up
Characterised by its agents that interact with each another and with their environment. Agents are flexible entities with the ability to learn from their experience and make decisions.		

Henderson and Mason (2004) use DES to model the Auckland, New Zealand ambulance service and GIS for geospatial visualization. Their tool, BartSim, includes a travel model which produces deterministic computations of travel time when this is time dependent. That is, the travel time for the same distance varies during the day. The authors pointed out the usefulness of simulation as a decision support tool to interested parties such as managers and front line personnel.

The above studies used DES to analyse different aspects of EMSs. However, using different simulation techniques can possibly accommodate better the complexity of EMSs. The Milan EMS was modelled by Aringhieri (2010) using ABS-DES approach. The model is built using AnyLogic simulation software which can combine the two simulation techniques. The service and travel times were modelled by introducing delays in a logistics-based DES model. By modelling the ambulance movements as delays, it was possible to

assign a new mission to an ambulance on transit rather than when it is stationed. This employs the ABS approach where ambulances and operators are modelled as agents. Emergency calls and service times are retrieved directly from the Milan EMS database. Aringhieri does not include the emergency departments in his hybrid model. Furthermore, the model is a composite simulation of the ambulance services that deploys DES for modelling the workflow of the ambulance service managements and ABS for modelling the interactions and movements of the ambulance vehicles. In the proposed approach of this thesis, the ABS and DES models are individual models of different subsystems of a holistic EMS. The ABS model of the ambulance service and the DES models of the A&E departments are independent simulations that can stand alone as well as be combined in a hybrid EMS by DS technologies. By doing this, the models can be reused further as independent individual models that can form a different DS environment.

Ambulance diversion policies analysis at emergency departments with the support of different modelling methodologies is discussed by Hagtvedt *et al.* (2009). They pointed out that ambulance diversion is a patient-safety issue as well as hospital revenue concern. To evaluate strategic ambulance diversion policies, they use multiple methods, namely they use continuous time Markov process modelling, DES, ABS and game theory approaches. However, they did not model the ambulance services but rather the emergency departments' ambulatory demand. In this study, the EMS systems are viewed comprehensively. Therefore, the ambulance service and the A&E departments are included.

### **3.6 Why DS is advantageous over single composite simulation**

In the previous section, there was a brief reference to some of the existing simulation studies on EMS. However, to the best of the author's knowledge, there are not any studies in the field of EMS using DS with standards implementation. Large-scale simulation models that contain more than one organisational subsystem can be modelled either as a single

composite simulation or as separate, independent models that are linked in some form of distributed system.

There are various motivations for using DS against attempting to reuse models in a standalone model and using a single simulation package. Explored in full in Taylor *et al.* (2012a), these include:

#### *Data transfer/access problems*

A simulation often draws data from local data sources. As soon as a simulation leaves its domain of use the data must go with it. Data sources can be very large, multiple and/or connected to real-time sources. Moving these and ensuring that they are up to date may not be particularly convenient. This can lead to inaccuracies in results due to inconsistent data.

#### *Privacy and data sharing issues*

Models and data may contain secrets that owners may not want to share openly. Privacy issues may prevent model sharing across different organisations or even across different departments in the same organisation.

#### *Model composability issues*

Composing several models together in the same simulation software might not be as convenient as it sounds. Even if each of the stakeholders had previously developed models using the same simulation package, these models cannot simply be ‘cut and pasted’ into the same single model. Variable name clashes, global variables and different validation assumptions are three examples of the many problems of this approach. How do updates occur? How is version control maintained?

### *Model execution time*

As models grow in size, their process demands may increase predominately due to large event lists. Storage (RAM in particular) may also be strained by the larger model's demands.

Nevertheless, DS itself is not without its problems and can be extremely complex and often difficult to implement. However, it is attractive to think that the creation of large, distributed models that are private, access local up-to-date data, implement local changes efficiently and share the processing load of the model across the computers of the organisations is possible.

## **3.7 Developing the proposed methodology for large-scale hybrid ABS-DES DS models**

In this section the background and rationale of the proposed methodology is stated. Moreover, the development process is explained.

### **3.7.1 Background and rationale**

In the field of M&S, a lot of discussions have been made about formalising the area as scientific discipline. A common outcome of these discussions is that M&S needs standard terminology and procedures. More specifically, in the area of DS, that can be considered as a specialty of M&S, there is a further requirement. That is, to define the interoperability level among the participating models.

Tolk and Muguira (2003) introduced the idea of a layered view of the stages in DS projects. Inspired by the area of systems engineering, and the levels of information systems interoperability (LISI) model, Tolk and Muguira established the levels of conceptual interoperability model (LCIM) as a framework for simulation models composability. From

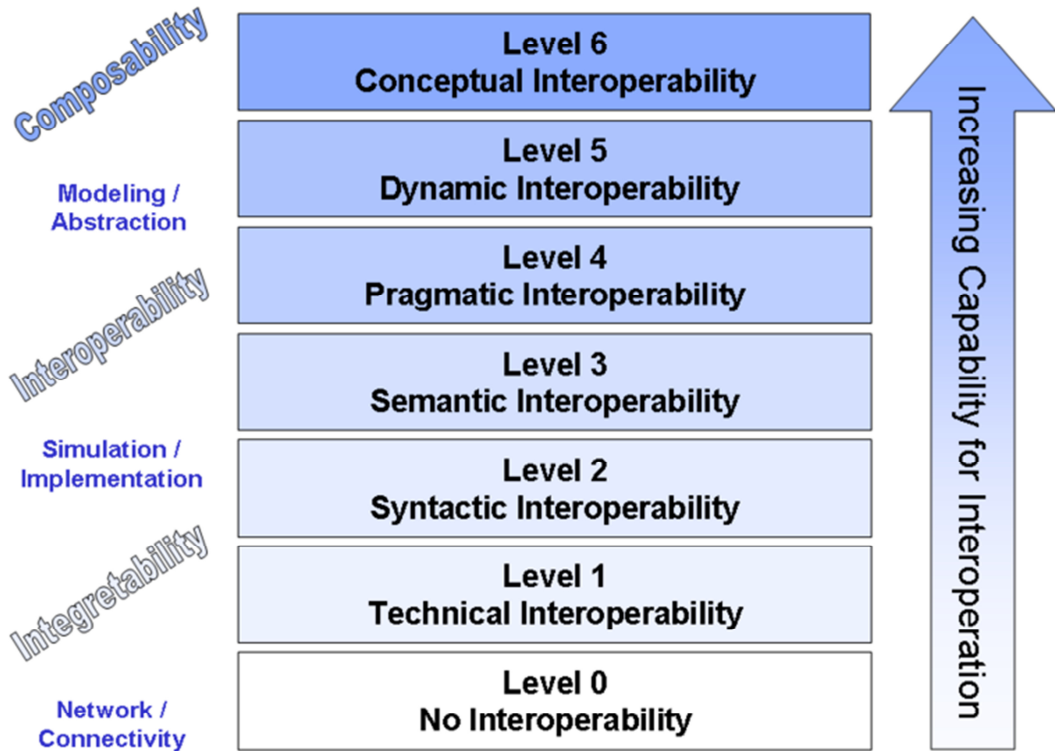


Figure 17: Levels of Conceptual Interoperability Model (source: Wang, Tolk, and Wang, 2009)

the idea conception until now, LCIM has been evolved and the current form is shown in Figure 17.

Wang *et al.* (2009) discuss the underlying notions of the model. Seven levels are defined in LCIM. Ranging from level zero (L0), where there is no interoperability at all, to level six (L6), where conceptual interoperability is achieved. Level one (L1) refers to the technical interoperability and describes the communication protocol between the interoperating models. Level two (L2) refers to syntactic interoperability. That is, the common data structures of the shared variables. Directly higher in the hierarchy stands level three (L3), the semantic interoperability, where there is a semantic mapping of the terms that are used in the in-



teroperating models. For example, in ABS and DES interoperating models, *agents* and *entities*, respectively, refer to the same object of the DS, namely the object that its state is changed when certain events are happening. Level four (L4) is the pragmatic interoperability, that is the common workflow of the interoperating models and defines the context of the exchanged information. The next level five (L5) refers to the dynamic interoperability of the distributed model. Here, the effect of the flow of information during execution is defined. Finally, the highest level in LCIM is level six (L6), where conceptual interoperability is achieved.

The above levels are categorised in three distinctive layers (Page *et al.*, 2004). The higher levels are included in the *composability* layer. Composability belongs to the model territory and is the objective of the lower layers. The immediate lower layer is the *interoperability* layer and describes the implementation issues, such as software details, data exchange, etc. Lastly, the lowest layer is the integratability category and deals with the physical connections of the DS.

Composability is the highest level in the LCIM. To achieve composability all the underlying levels must be achieved. As very clearly stated by Petty and Weisel (2003), interoperable models are not necessary composable. Interoperability is necessary but not sufficient to achieve composability. As defined by Page *et al.* (2004):

“The defining characteristic of composability is that different simulation systems can be composed at configuration time in a variety of ways, each suited to some distinct purpose, and the different possible compositions will be usefully valid simulation systems. Composability is more than just the ability to put simulations together from parts; it is the ability to combine and recombine, to configure and reconfigure, sets of parts from those available into different simulation systems to meet different needs.”

From the definition, it can be concluded that by achieving composability, the component models of a DS can be recombined with each other or with different interoperable models and form different DS systems. Therefore, simulation model reusability is supported by composable models.

Another distinction of composability is made by Davis and Anderson (2003). They differentiate the composition of models “of same resolution” and “of different resolution”, with the term resolution, here, to suggest the abstraction level of the models. The former is described as *horizontal* composability and involves composing models of the same level of abstraction. For example, horizontal composability is when all the interoperating simulations are modelled in the individual entity level, namely microscopic simulation. The latter term is described as *vertical* composability and involves models with different abstraction level. For example, vertical composability is when some of the component simulations are modelled in the individual entity level and some others in the organisation level. The horizontal composability is relatively easier to be achieved than the vertical composability. However, as the number of horizontal objects, i.e., component models or entities within the models, increases so does the complexity of the distributed system. Furthermore, difficulties lay in the realm of semantics of the different domains the component models may belong.

### 3.7.2 Development phases

As derived from the background analysis in the previous subsection, a simulation project can be conceptualised in a layered structure. The methodology proposed in this thesis can be considered as a guideline methodology for developing large-scale distributed ABS-DES simulation models and is based on the steps methodology for simulation project formed by Banks *et al.* (2000). Details of this methodology were discussed in Chapter 2.

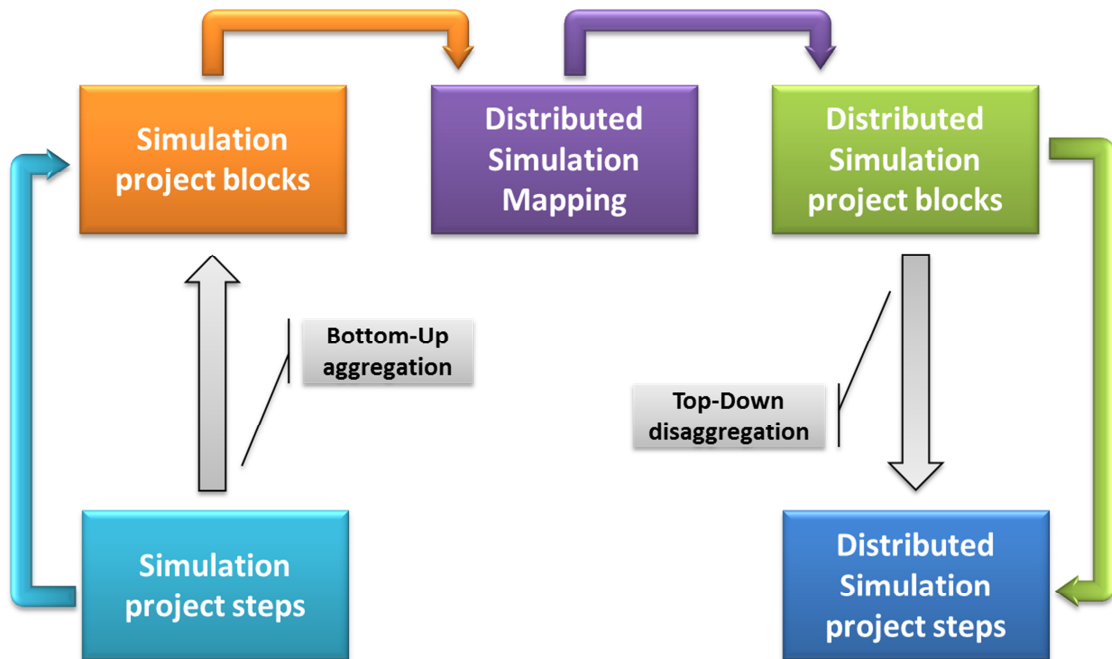


Figure 18: Distributed simulation methodology development approach

The proposed methodology was developed in a three-phase approach. In Figure 18, the complete process of the methodology development is depicted. The first phase is a bottom-up approach and involves the aggregation of the simulation steps into blocks. The second phase is a horizontal approach and involves the mapping of the standalone simulation project blocks with the DS project blocks. Finally, the third phase is a top-down approach and involves the disaggregation of the DS project blocks into distinctive steps. The three phases are analysed in the following subsections.

### 3.7.2.1 Development phase one

The first approach taken in developing the proposed methodology is a bottom-up view of the steps in a standalone simulation project. The slightly altered stepped-process of simulation studies that is used in this research is shown in Figure 19.

The first step of the process involves the problem formulation. In this step, a modeller should state clearly the real-world problem that the particular simulation project will attempt to solve or analyse.

The next step is a parallel activity of starting the collection of data and conceptualising the model, as well as the selection of the simulation software or language to be used for the

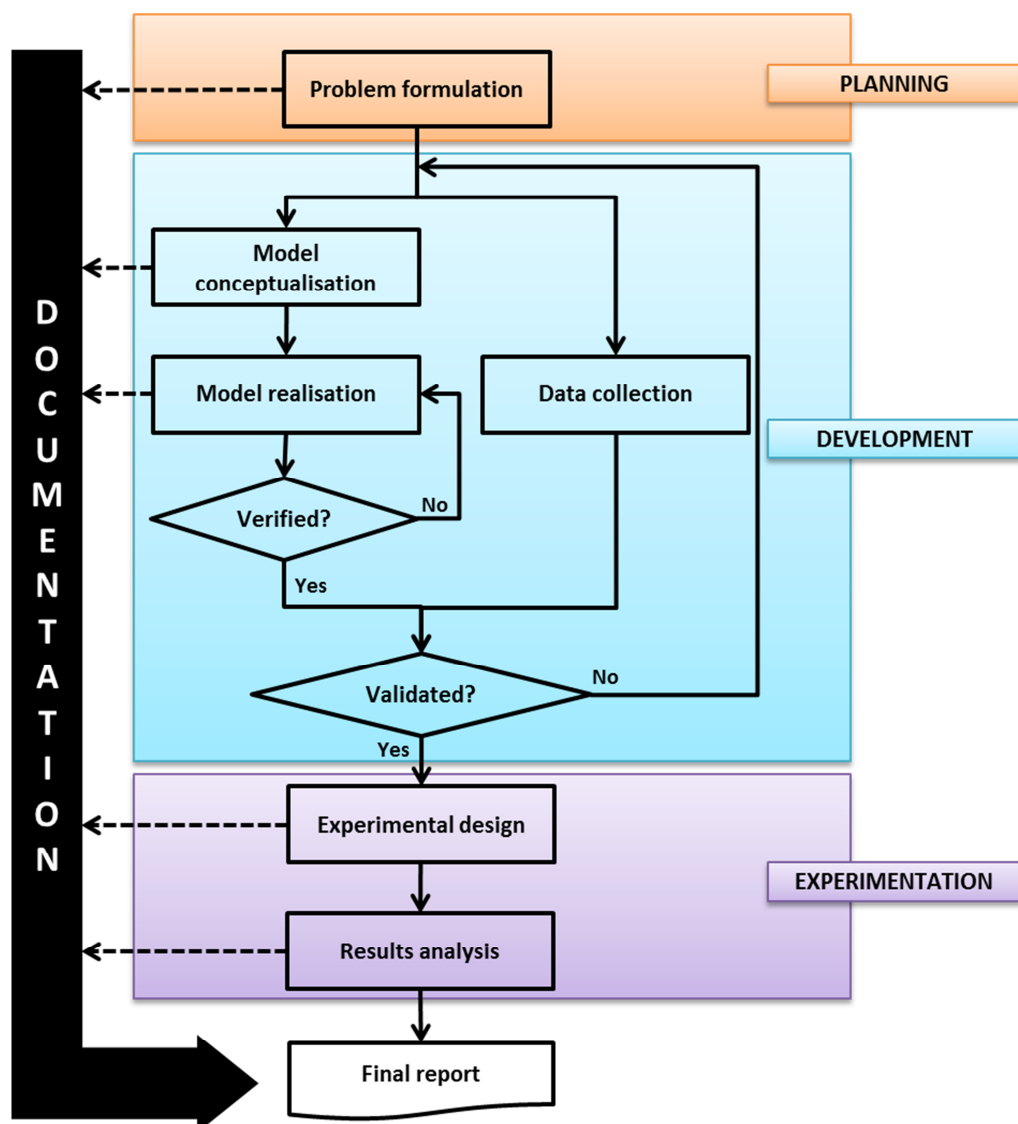


Figure 19: Steps in a standalone simulation project

realisation step. Sequentially, but in parallel with data collection, the model realisation step starts. The model realisation is actually the coding of the simulation software program. Model verification refers to debugging the computer simulation program. Debugging is an iterative process and is happening throughout the coding process. The parallel processes of data collection and model realisation should finish together in order to populate the model with the actual data and be able to start the validation process. The simulation program is valid when it does what was intended to do by the design. Usually the validation process involves pilot runs and testing the results against some performance measures of the real-world system under study. If the model is not valid, the conceptual model and the available data should be revisited. When the model is valid, the process of designing the experiments can commence. Finally, the experiments can be conducted and the produced results can be analysed.

From the first step, the documentation process commences and continues throughout the simulation project. This documentation leads to the final report of the project. The small alterations of Banks *et al.*'s (2000) methodology are the two parallel activities. Namely, the data collection and the documentation processes. From experience, the computer simulation program can be verified even when the model is not populated with the actual data. The model coding, as well as the data collection, processes can be very time-consuming. By performing the two activities in parallel fashion, valuable time can be saved. Similarly, the parallel documentation of each step can help to avoid omissions in the final report.

As mentioned earlier, the conceptualisation of a simulation project can be seen in a layered layout. Therefore, the bottom-up approach that commences the proposed methodology development process involves the aggregation of the aforementioned steps into layers, or blocks as mentioned in this study. The complete process of building a simulation study can be divided into three blocks. That is, the *planning block* stands in the higher level and concerns the pre-modelling activities, the *development block* forms the middle level and con-

cerns the actual modelling activities, and the *experimentation block* is the lowest level and concerns the post-modelling activities (see Figure 19).

### 3.7.2.2 Development phase two

The second phase in the proposed methodology development rationale is a horizontal approach. This phase involves the mapping of the processes of the standalone simulation project with the DS project.

The first block of a DS project is similar to any single simulation project. That is, the problem formulation, where the objectives of the study should be clarified. However, careful considerations should be made in the development and experimentation blocks (see Figure 20).

In the development block, the conceptualisation and realisation procedures exist. The conceptual design for the distributed system, mainly, involves the interactions between the

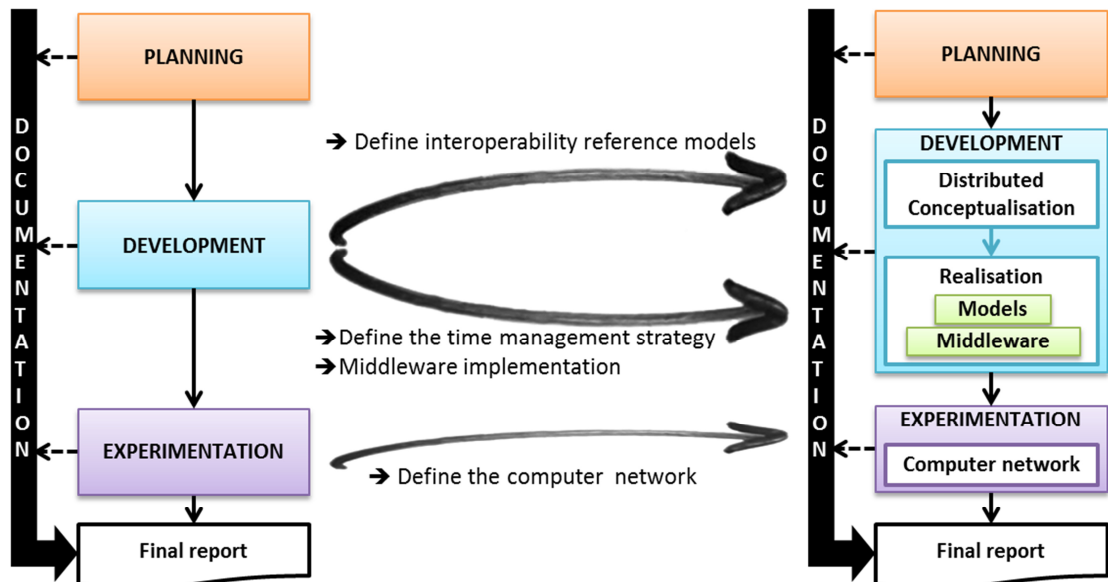


Figure 20: Mapping the phases in a standalone and a DS project

participating simulation models. At this point the individual models can be considered as highly abstract entities (or black boxes) and the objective is to identify the boundaries between the interacting models. In other words, the first, and very important, activity is to define the IRMs. In Chapter 2, there is a detailed reference to IRMs and the efforts that have been made, so far, to standardise all the possible ways that simulation models can interact with each other.

The second procedure in the development block is the realisation process, and that involves the simulation software development. The individual participating models either exist already or have to be built anew. In the former case, the models should be modified. In the latter case, the models will be built according to the distributed system conceptual design. The middleware implementation will be responsible for the data and time synchronisation.

In a standalone simulation, the only option is to execute the experiments in a single computer processor. Conversely, DS systems, as the name implies, run in different machines. These can be processors connected via a local network, the internet, or the cloud, or even different cores in the same processor.

### **3.7.2.3 Development phase three**

The third phase of the development of the proposed methodology involves the disaggregation of the DS project blocks into detailed steps (see Figure 21). For this phase, a top-down approach is adopted as mentioned earlier.

The most important differences between a standalone and a DS project lay in the development layer of the building process. The first activity in the development block is to conceptualise the distributed system. During this process, the interactions between the simula-

tion models should be defined. This involves the development of the IRM of the distributed system.

The IRM defines the exchanged information but does not necessarily include time syn-

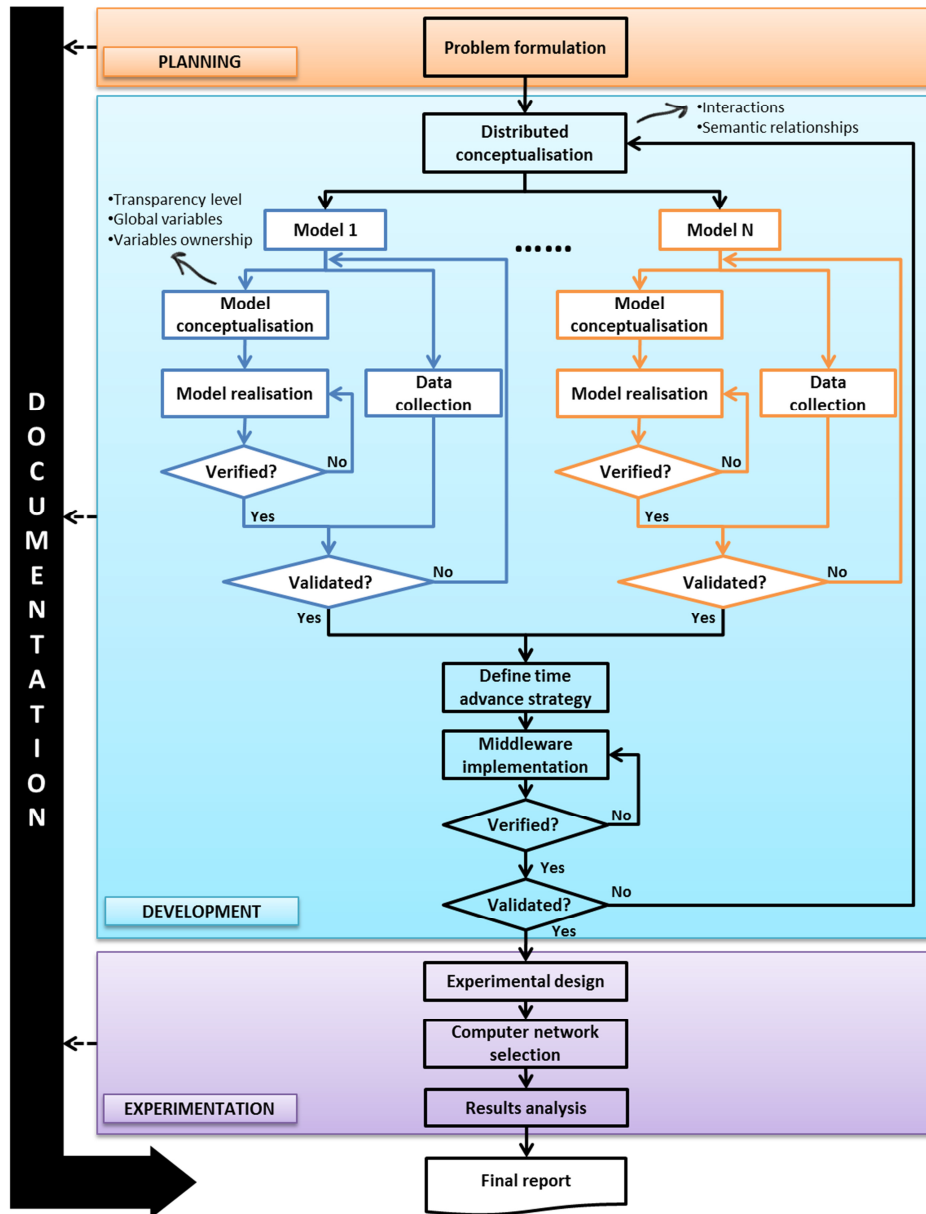


Figure 21: Distributed simulation methodology



chronisation information. The time dimension is only mentioned when there may be a conflict and the order of events must be in the specified relationship. Further, in the conceptualisation step, the semantic relationships between the interacting models should be clarified. If the system is homogeneous, i.e., all participating models are modelled with the same simulation paradigm, this is not necessary. However, if the system is hybrid, as is in this study, there should be a semantic coupling between the different simulation paradigms. At this point, the software tools should be decided.

Then, the next activity in the same block is the model realisation step. The realisation step includes, first, the participating models building and, second, the middleware implementation. Regarding the model building, there are three possible scenarios: a) none of the individual simulation models exist, b) all of the individual simulation models exist, or c) some of the individual simulation models exist. In case (a), the models should be built from the beginning, following the usual procedure. However, when conceptualising the component models, certain considerations should be taken into account for the sake of interoperability. This includes, the transparency level of each component model (what information each model allows to be visible), the global variables of the distributed system and the ownership of these variables (which model can update the global variables). In case (b) the local models already exist, therefore, they should be modified in order to become interoperable. Lastly, in case (c), there will be a combination of both the above procedures.

The second part of the realisation step includes the DS middleware. The main role of the middleware is the synchronisation of data and logical time management. The synchronisation scheme can be either conservative or optimistic. As mentioned in Chapter 2, a conservative scheme does not allow processing of local events in a future logical time of the system, while in the optimistic approach there can be violation of this rule but there is a rollback mechanism to recover from errors and process always the event with the lower timestamp. Furthermore, the logical time advance service should be decided before com-

mencing the actual coding of the middleware. For example, in the HLA standards for DS that is used in this project, there are two such services, Time Advance Request (TAR) and Next Event Request (NER). Each federate asks the HLA RTI to advance its logical time. The RTI will then send all the messages (i.e., attribute updates, etc.) to the interested federates and then grants the advance of the logical time (Fujimoto and Weatherly, 1996). Usually, time-driven simulations are facilitated by TAR while event-driven simulations are facilitated by NER.

Once the time advance service is defined, the actual coding of the middleware can begin. Debugging of the software happens iteratively until the middleware is verified. Afterwards, the complete distributed system should be validated against specified performance measures. If the system is not valid, the distributed model should be checked, starting from the distributed conceptualisation step.

The next layer that differs between the standalone and the DS projects is the experimental design. As mentioned above, the computer network for the DS projects' experiments execution should be defined, i.e., private network, public network, etc.

### **3.8 Summary**

This Chapter presented the development process and rationale of the proposed methodology for building large-scale hybrid DS models.

Beginning with the issues a modeller may face when analysing the selected exemplar case study, it was shown that the EMS is a complex system that consists of more than one organisation. Next, there was a justification of the selection between hybrid and single simulation technique. It is established that a single simulation paradigm cannot adequately represent a complex, heterogeneous system. Therefore, a hybrid simulation technique is adopted in this thesis. The selection of the different simulation techniques was stated next. ABS

was found the most appropriate technique for modelling the ambulance service, while DES was found better suited for modelling the A&E departments. A brief analysis and the characteristics of the two simulation paradigms were presented. The semantic relationship between ABS and DES modelling was presented, as well as some indicative examples of existing models from the literature. Thereafter, the advantages and drawbacks of DS over standalone modelling were listed and there was a justification for selecting the former for large-scale systems' simulation.

The last section discusses the development of the proposed distributed simulation methodology for building large-scale hybrid ABS-DES DS models. Initially, there was an analysis of LCIM, which presents a layered view of interoperable systems. LCIM consists of seven distinguished levels. The lower levels refer to integrability and describe the technical layer of a distributed system, i.e., hardware, firmware, etc. The middle levels refer to interoperability and deals with the exchange of information between simulation models. At the top layer stand the composability levels. Composability is a highly abstracted conceptual design, that when it is achieved the models can be reused in various distributed systems. The definition of composability and the different types, i.e., horizontal and vertical, are provided next.

Based on the layered structure, the way for developing the DS methodology was unfolded. The first act was a bottom-up aggregation of the blocks/layers in the standalone simulation project step building process. Then, a horizontal approach was adopted for the mapping to a DS project. And, finally, the top-down disaggregation of the blocks was analysed, which resulted in the final distributed simulation methodology.

# CHAPTER

# 4

---

## **The FIELDS Framework for Integrated EMS Large-scale Distributed Simulation**

*This chapter presents the conceptual framework design for the developed simulation software. Following the steps of the distributed simulation methodology, as described in the previous chapter, the conceptual design of the London EMS is analysed in this chapter.*

## 4.1 Overview

In this Chapter, the conceptual framework design for the distributed hybrid ABS-DES EMS simulation model is analysed. The proposed framework, which is named FIELDS and stands for Framework for Integrated Emergency Medical Services Large-scale Distributed Simulation, is designed in accordance to the proposed methodology for building large-scale DS projects.

In the previous Chapter, there was an analysis of the encountered issues when attempting to construct large and complex simulation models of systems that consist of more than one organisation. Further, the issues of experimenting with these simulations were discussed. Lastly, the rationale behind the development of the proposed distributed simulation methodology was discussed thoroughly.

The FIELDS framework that is analysed in this Chapter constitutes the highest level of the *development* layer in the simulation building process. The layer above the *development* one, namely *planning*, was established in Chapter 1, where the problem formulation was analysed.

Following the steps in the conceptualisation stage, Chapter 4 is organised as follows. Commencing with the distributed system conceptualisation, the analysis of the interactions between the ambulance service and an emergency department is presented. A discussion about the semantic relationships between the chosen simulation techniques was obtained in Chapter 3. Further, the underlying IRM is formulated and the selection of the standards for the distributed system is justified. In the individual models level, the conceptualisation of each participating simulation is demonstrated, together with the strategy for the data collection process.

## 4.2 EMS components and interactions

EMS systems consist of an ambulance service and several A&E departments. The A&E departments are located in the regional hospitals of the ambulance coverage area. Therefore, there is a need for several heterogeneous models to communicate with each other, namely, the ambulance service model and a number of A&E models in the area of coverage. The ambulance service objects interact enormously with each other and their environment and have to make decisions depending on some parameters, such as, to allocate the appropriate ambulance for an incident, to decide whether there is a need for transfer to an A&E department and to find the most suitable hospital. On the other hand, A&E departments are mainly process-oriented. The objects that are processed by the system, e.g., patients, do not make decisions but rather are driven by the hospital processes. For the purpose of this research, the objects that can make decisions in an A&E, e.g., clinical staff, are perceived as resources. The above are the main reasons for the selection of the simulation paradigms. That is, the ambulance service is modelled using ABS and the A&E departments are modelled using DES techniques.

The central and leading component of an EMS system is the ambulance service. Therefore, the ABS model should be able to identify and communicate with all the DES models. The hospital DES models, now, should be able to communicate with the central ambulance service component but there is no need to communicate between each other.

In Figure 22, the highly abstracted conceptual design of the FIELDS framework can be seen, where the arrows indicate the interactions between the components.

The ambulance service, when there is a need to transfer a patient to a hospital, should be able to find the nearest available A&E department. Therefore, at any point of time the ambulance service should be able to access the most up-to-date availability of the A&E departments. When a hospital is found, the ambulance model should be able to reserve re-

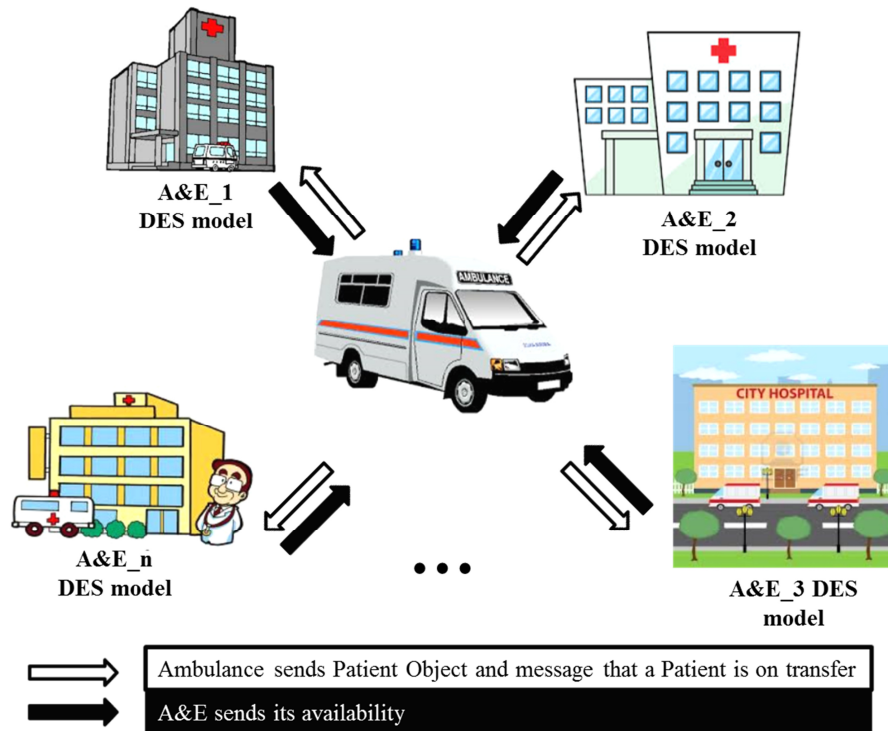


Figure 22: FIELDS conceptual framework

sources in the specific A&E department. The point of reserving resources is to avoid conflict in the availability due to the elapsed travel time from the scene of the incident to the hospital. When the patient arrives at the hospital, patient object, with all its attributes, pass from the ambulance model to the A&E model. All the information sharing described in this paragraph is depicted in Figure 22.

#### 4.2.1 Software tools

Since the rationale behind the simulation techniques has been analysed and the semantic relationships between those techniques have been defined, the simulation software and the distributed protocol can be decided at this point, as was indicated in the step building process that was analysed in the previous Chapter 3.

For the implementation of the simulation models of the case study and the DS software artefact, it was decided to utilise Open Source Software (OSS) for accessibility and flexibility purposes. OSS, as opposed to commercial, are cost-free platforms with accessible source code. This makes OSS flexible and sometimes customisable. Furthermore, as the open source community supports the dissemination of knowledge, there are several networks from enthusiastic users that provide speedy help and support. OSS often appears to be an invaluable tool for the researchers' community.

As described in Chapter 2, there are several protocols for DS. However, the dominant, at present, is the HLA IEEE-1516 standard. A fundamental component of the HLA is the RTI implementation. RTI implements a set of standard rules that specify information sharing and coordination during the interactions of simulation models. Up until now, the HLA standard is mainly utilised in military simulation applications and it is selected as the protocol for data and time synchronisation between the ambulance service and the several A&E department models that constitute the DS system of this project.

There are many software packages available, both open-source and commercial, for creating simulation models and the RTI. Therefore, the criteria for the selection of the software tools were mainly two: the code accessibility and the available documentation. After considering the offered options, it was decided to utilise the Repast Symphony toolkit ([repast.sourceforge.net](http://repast.sourceforge.net)) for the individual models implementation and the poRTico ([www.porticoproject.org](http://www.porticoproject.org)) RTI implementation for the interface development. Both packages have a Java application programming interface, which results in a more efficient implementation of the synchronisation interface. Repast Symphony is originally considered to be an ABS tool but can be easily converted to a DES simulator. Furthermore, these were utilised in several research projects, published in scientific journals (Kuhn *et al.*, 2010; Malik *et al.*, 2010; Minson and Theodoropoulos, 2008; Tatara *et al.*, 2007; Tu *et al.*, 2011). However, a lack of research projects that combined the two software toolkits was identified.



### 4.3 FIELDS interoperability reference model

As outlined in Section 4.2, there are three types of interchanged information between the component models of the FIELDS framework. First, all A&E department models should be able to communicate their availability to the ambulance model. It is essential for the functionality of the model, that when the ambulance model searches for the most appropriate hospital to transfer a patient, the most up-to-date information about all A&E departments' availability is known to the ambulance model. Second, the patient object/agent should be transferred from the ambulance service model to the ambulance patient object/entity entry point of the A&E department model. Third, the ambulance service model should be able to notify the A&E department model that a patient is on transfer in order to reserve resources

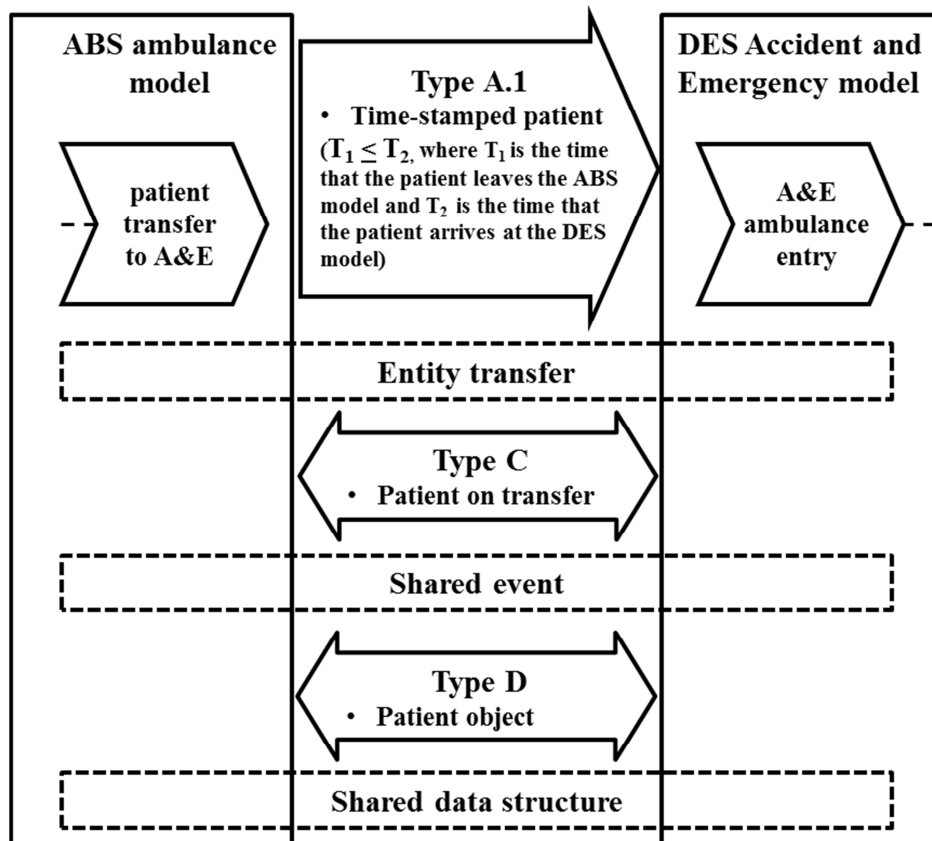


Figure 23: FIELDS interoperability reference model

and avoid conflict in the hospital availability level.

A way to define conceptually the boundaries between interoperating models and denote the interaction points is the IRM. A series of IRM have been developed and standardised to address model level interoperability issues between DES (SISO-006-2010). However these do not fully address model heterogeneity, when, for example, there are different simulation paradigms involved (Taylor et al., 2009b). To illustrate the interactions between the ambulance ABS and the A&E department DES models, the standardisation of IRM can be adopted and expanded. In this project, the interactions between the two parts of the hybrid model can be represented by a tuple  $\text{Type}(A.1, C, D)$  or  $\text{Type}(\text{general entity transfer, shared event, shared data structure})$ . The different types of IRM are explained in Chapter 2.

Figure 23 shows the IRM of the FIELDS framework. The interface between the ambulance service ABS model and the A&E DES model is between the ambulance exit point to a hospital and the hospital's ambulance entry point. As mentioned above, a combination of the standard IRM type is required in order to capture the interactions between the two models. The Type A.1 aspect of the IRM, namely *general entity transfer*, indicates that when there is a need for hospital transfer then a patient object passes from the ambulance model to the A&E model and, consequently, a former *agent* becomes an *entity*. The time relationship must always be  $T_1 \leq T_2$ , where  $T_1$  is the time when the patient object leaves the ABS model and  $T_2$  is the time when the patient object arrives at the DES model. The Type C aspect of the IRM, namely *shared event*, indicates that when the travel to hospital begins in the ABS model, it will trigger (by sending a message) the DES model to reserve resources. Both events must happen at the same time. The Type D aspect of the IRM, namely *shared data structure*, indicates that there are shared data structures between the two interoperating models. The data structure could be an array, a table, an object, etc. In this project, it is the object that represents the patient agent.

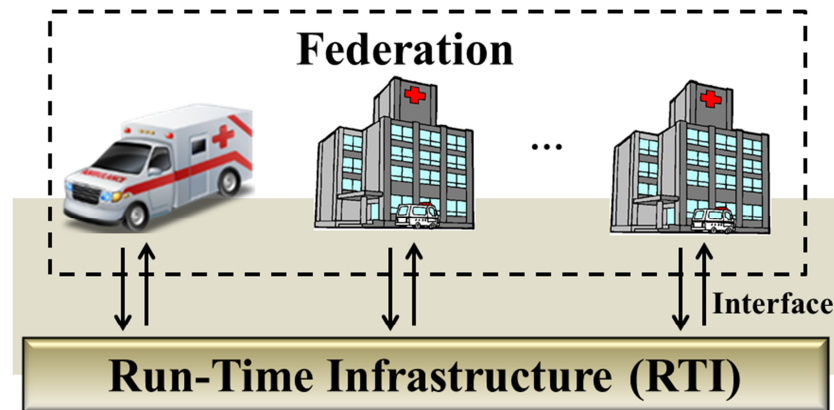


Figure 24: FIELDS HLA concept

#### 4.4 Data communication and time synchronisation

The data communication and time synchronisation between the system component models is implemented using DS standards. Several standards for DS have been developed, such as DIS, ALSP and HLA, details of which are described in Chapter 2.

HLA is the most mature protocol. As such, it is the selected standard for this project. A typical DS is composed of a number of simulation models which are known as federates and all these federates run under one federation connected through a typical RTI. In Figure 24, the HLA conceptualisation for the FIELDS framework is depicted. In the presented framework, there is an ambulance service ABS federate that communicates with the regional hospitals DES federates. All hospital federate models run as independent federates and each one exchange information with the ambulance federate. All models execute within the same federation. The RTI provides information, synchronisation, and coordination services between each federate.

## 4.5 Individual models conceptualisation

In this section the conceptualisation of the individual participant models, i.e., federates, is presented.

### 4.5.1 Ambulance service conceptual model

The first element of EMS is the ambulance service. As mention in Chapter 3, the ambulance services can be either air- or land-based. Air ambulance services respond to serious injuries when timing is extremely critical or the landscape is inaccessible by road. The land ambulance services coordinate the emergency calls and decide which vehicle and crew to be sent to an incident. Generally, the ambulance service fleet consists of fast response cars, ambulances and two-wheel vehicles. Furthermore, there are two types of crew: the BLS crew that deal with non-life threatening incidents, and the ALS crew that are able to provide medical care and deal with life-threatening injuries. Depending on the incident, ambulances transfer patients to hospitals or just treat them at the scene.

The timeline of the ambulance service is characterised by three distinguished periods, as shown in Figure 25. The *waiting* period is the time from the emergency call receiving to the time that an ambulance is found. The *service* period spans from the starting of a journey to the scene to the ending of the journey to the hospital; in the case that no hospitalisation is needed, the service time ends at the completion of on-scene treatment. The last period is the

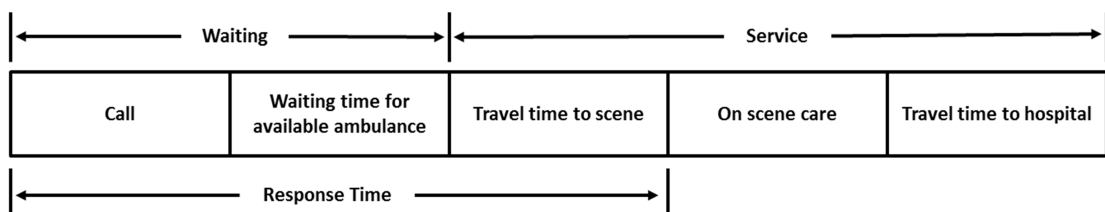


Figure 25: Ambulance service timeline (source: Fitzsimmons, 1973)

*response time* period that starts when receiving an emergency call and ends when the ambulance reaches the scene of the incident. The above three time periods are considered to be some of the performance measures of the ambulance service organisations.

At the conceptualisation stage, a decision should be taken about the level of detail that the simulation models will include. One of the objectives of this thesis is to test the feasibility of the proposed framework, which is mainly focused on the interface between the interacting models of the DS system. Therefore, the particularities inside the individual models are not a priority, at present time. Thus, it was decided to model the participating simulations with just enough details to validate the models. Consequently, high level of abstraction is adopted for the individual models of this project. As shown in Figure 26, for the ambulance service model, only the highlighted components are included in the FIELDS framework. This includes the land ambulance service and, from the available vehicles, only the ambulances are considered. The emergency call centre is a simplified form of the real system. Further, the ambulance crew is not modelled.

As mentioned before, ABS technique is selected for the ambulance service model due to the high interaction level of the simulation entities with each other and the environment. Simulation agents are of two types, passive agents, which are part of the environment, and

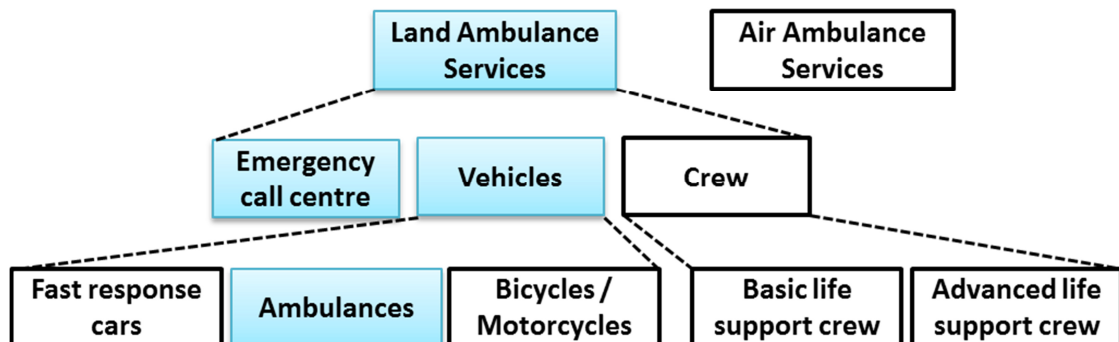


Figure 26: Ambulance service model abstraction

active agents, which interact with each other and the environment. Passive agents are the hospitals and the ambulance stations which have predetermined attributes such as location coordinates and capacity. The active agents are the emergency calls, the patients and the ambulances. Emergency calls are generated according to an arrival probability distribution. A generated emergency call carries location and incident information. Simultaneously with a call generation, a patient is generated that adopts the location and incident information from the call. Incident information is the patient condition and whether this patient needs transfer to an A&E. In real ambulance services, when a call arrives at the dispatch centre, an assessment is carried out and then the appropriate ambulance crew is sent to the incident scene. An assumption of this model is that all calls will be attended, thus the emergency call is searching for the nearest available ambulance. Once an ambulance is found the call is removed from the simulation. When an ambulance is found, it is flagged as unavailable and starts the journey towards the incident scene. At the scene, there is a delay for on-site treatment. If the patient is flagged as one that needs transfer to an A&E, the ambulance searches for the nearest available A&E and starts the journey to the hospital. After the patient's handover, the ambulance goes back to station and is flagged as available (see Figure 27).

Travel times are calculated assuming an average ambulance speed and the Euclidean distance between the current location and the destination location. According to a study by Jones *et al.* (2010), Euclidean distance with a corrective factor can be used in research with a high degree of confidence to represent real driving distance in an urban setting. Silva and Pinto (2010) used a corrective coefficient to realistically represent the relationship between the actual distance and the Euclidean distance of two points in an urban environment. Each location point has an X and a Y Cartesian coordinate, so the Euclidean distance  $|s|$  between two points is calculated by the following equation:

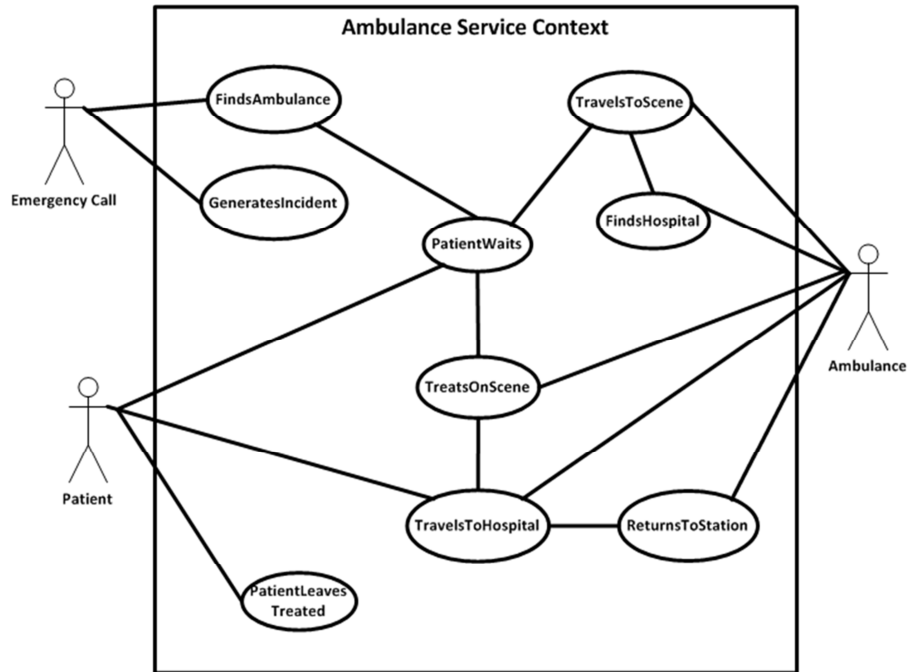


Figure 27: Ambulance service use case diagram

$$|s| = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2}$$

and the travel time is:  $t = c * s/v$ , where  $t$  is the travel time in hours,  $s$  is the distance in miles,  $v$  is the average speed in miles per hour and  $c$  is the corrective coefficient. The corrective coefficient was calculated using Google Maps ([maps.google.com](https://maps.google.com)) in order to find the driving distance between two points within London. For 40 samples, ranging from 0.5 to 20.5 miles across Greater London, a simple linear regression indicated a corrective coefficient for London  $c = 1.32$ , as shown in Figure 28.

#### 4.5.2 A&E department conceptual model

The second element of EMS is the A&E departments in the region. As explained in the previous section, since the focus of this thesis is the feasibility of a hybrid DS system model of EMS, the individual models are kept as simple and parsimonious as possible. There-

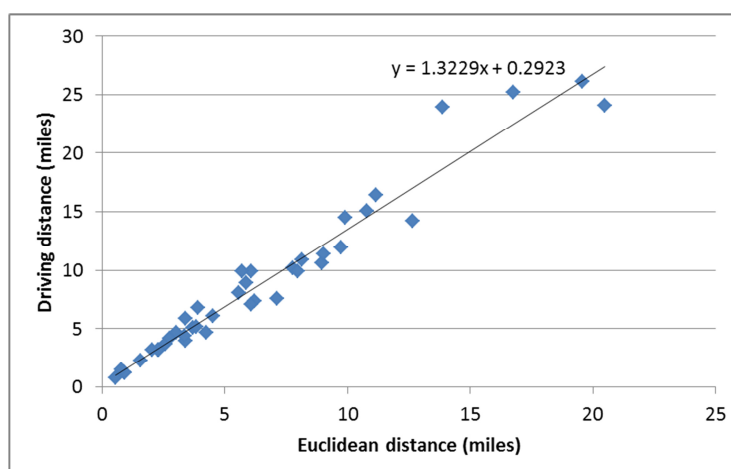


Figure 28: Driving distance corrective coefficient

fore, the level of abstraction in the A&E department model is high, as it is for the ambulance model, too. Furthermore, all the models in the region represent general A&E departments without considering the resuscitation units in them, as highlighted in Figure 29. Apart from the general departments, there are A&E hospitals for specific medical specialties. These specialised A&E departments can be for children, i.e., paediatrics departments or for a specific condition, i.e., ophthalmology. The normal flow includes an initial triage, the minors units and the majors units. The minors units treat minor injuries and illnesses while the majors units deals with seriously ill patients. One more assumption of the model is that all hospitals are similar. It is planned, in future research, to encompass more details in both models and hence, to analyse the operations and performance of EMS.

In a general A&E department, there are two streams of patient input: the ambulance arrivals and the walk-in arrivals. Also, there are observation cubicles and ward beds for inpatients. In the current prototype model, the A&E service ends after the patients' observation. The inpatient ward of the A&E is not modelled for reasons that are explained earlier, i.e., the project is focused on the interface and therefore the processes beyond that are kept as simple as possible.



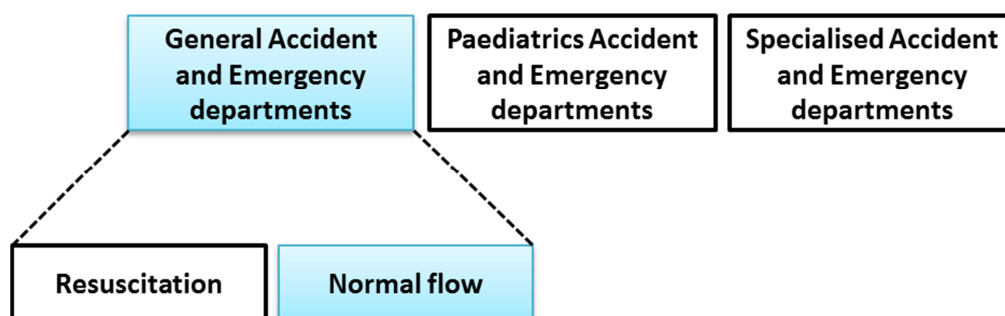


Figure 29: A&E departments model abstraction

In Figure 30, there is a flowchart of the processes that are included in the modelled A&E departments. Patients arrive at the hospital, either by ambulance or by their own means (walk-in). Ambulance arrivals are directed to the appropriate section (i.e., minors or majors) according to their condition. Walk-in arrivals are directed to the above sections or leave the hospital after an initial assessment (i.e., triage service).

Walk-in patients enter a queue for triage. The triage staff performs the initial assessment and directs patients accordingly. Then the patients join a queue for the minors or majors department according to their condition. When the required resources become available, patients receive treatment in the respective unit. If no treatment is required, they leave the system after the triage service. Ambulance patients go directly to the minors or majors department, based on the previous communication with the ambulance model, and do not enter the subsequent queues. This is due to the fact that the hospital has been notified of the ambulance patient arrival and has reserved the appropriate resources. After the end of the service, patients exit the A&E and the resources are released. Clinical staff is modelled as one type of resource. Therefore, there is no differentiation among nursing, medical or technical personnel. Also, another simplification of the FIELDS framework is that there is no imaging or lab tests process.

DES is selected as the technique for the A&E department model due to the process-oriented nature of A&E departments. However it can be argued that emergency department entities are making decisions. But, the aim of this study is not to model entities' behaviors but rather the processes within an A&E.

The hospital availability depends on the clinical staff resources and the particular type of A&E observation beds/cubicles (from now on are mentioned as beds), namely minors and majors. This can be calculated for each particular bed type as:

*Hospital availability* =  $\min(\text{available clinical staff}, \text{available beds})$ , where

*Available clinical staff* =  $\text{clinical staff capacity} - \text{clinical staff occupancy}$

*Available beds<sub>type</sub>* =  $\text{beds capacity}_{type} - \text{beds occupancy}_{type}$ :  $type = \{\text{minors}, \text{majors}\}$

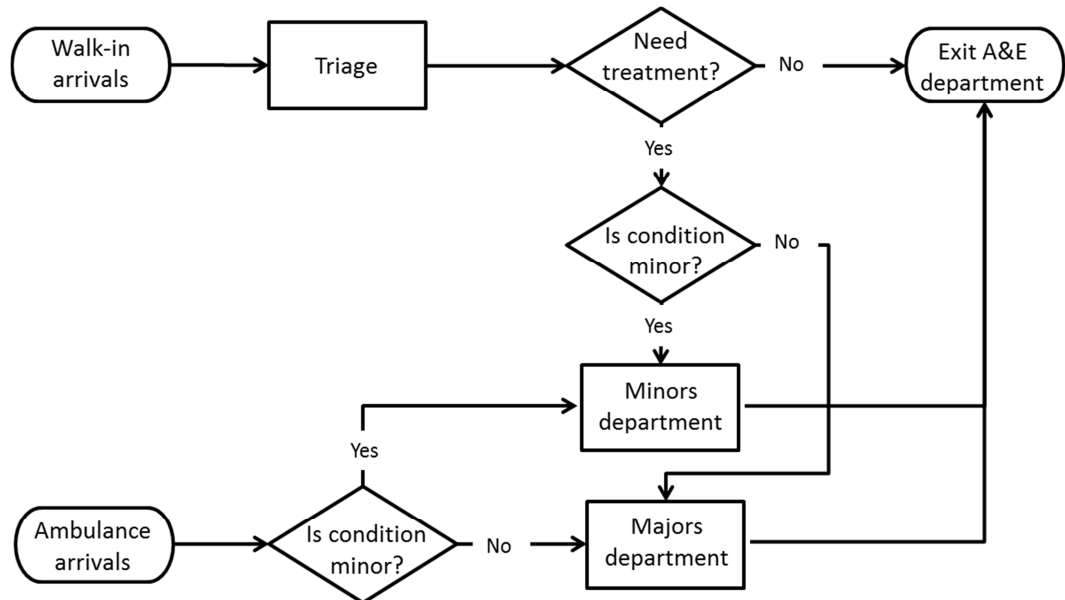


Figure 30: A&E department flowchart diagram

The hospital availability is updated in the DES A&E department model. This information is passed to the ABS ambulance service model in order to locate the suitable hospital when a patient transfer is needed.

#### 4.5.3 EMS model events

From the analysis of the ambulance service and the A&E department models, the events of both models can be formulated. In Table 7, there is a list of the events that will be recorded in the results sheets.

From the times that these events happen, important system metrics will be extracted. For example, the distance between the *emergency call generation* and the *selected ambulance arrived at the incident scene* equals the *ambulance service response time* measure.

$$\text{response time} = t_{a3} - t_{a1} \quad (\text{A1})$$

Table 7: Ambulance and A&E models events

Events of the ambulance model		Events of the A&E model	
$t_{a1}$	Emergency call generation	$t_{h1}$	Patient arrival
$t_{a2}$	An ambulance is found	$t_{h2}$	Triage queue entry
$t_{a3}$	The selected ambulance arrived at the incident scene	$t_{h3}$	Triage service entry
$t_{a4}$	An A&E is found	$t_{h4}$	Minors queue entry
$t_{a5}$	The ambulance departs from the incident scene	$t_{h5}$	Minors service entry
$t_{a6}$	The ambulance, carrying the patient, arrive at the A&E	$t_{h6}$	Majors queue entry
$t_{a7}$	The ambulance arrive back to station	$t_{h7}$	Majors service entry
		$t_{h8}$	A&E exit

Similarly, the distance between the *triage queue entry* and the *triage service entry* events equals with the *time to initial assessment* measure.

$$\textit{time to initial assessment} = t_{h3} - t_{h2} \quad (\text{H1})$$

The important performance metrics for the ambulance service system, apart from the ambulance service response time, are: the waiting time and the service time (see Figure 25), and can be calculated by the following equations.

$$\textit{waiting time} = t_{a2} - t_{a1} \quad (\text{A2})$$

$$\textit{service time} = t_{a6} - t_{a2} \quad (\text{A3i})$$

$$\text{or } \textit{service time} = t_{a5} - t_{a2} \quad (\text{A3ii})$$

In line with the above, the A&E system performance metrics, apart from time to initial assessment, are: total time in A&E, waiting for minors service, and waiting for majors service, and can be calculated by the following equations.

$$\textit{total time in A\&E} = t_{h8} - t_{h1} \quad (\text{H2})$$

$$\textit{waiting for minors service} = t_{h5} - t_{h4} \quad (\text{H3})$$

$$\textit{waiting for majors service} = t_{h7} - t_{h6} \quad (\text{H4})$$

For the distributed system the relationship  $t_{a6} = t_{h1}$  should always be true for the ambulance arrivals in the A&E. This indicates that the distributed system has the desired behaviour, that is, the time that the ambulance patient arrives to an A&E in the ambulance federate, the same time arrives to the selected A&E federate model. One of the limitations of FIELDS is its inability to study the patient handover delay, and therefore the following time relationships are also true:  $t_{h1} = t_{h5}$ , for ambulance arrivals with minor conditions and

$t_{h1} = t_{h7}$ , for ambulance arrivals with major conditions. The handover delays present a serious problem to the EMS in the UK, and it is planned to be incorporated in future research.

#### 4.5.4 Data collection

As mentioned in Chapter 3, the data collection activity commences in parallel with the conceptualisation phase and can be continued before the model validation phase. This project tests the feasibility of the proposed framework in a case study of the London EMS. The UK's DoH publishes regularly performance data of the National Health Service (NHS). Therefore, the required data to populate the London EMS distributed model is collected from online published data on the London ambulance service (LAS) and the A&E departments in the area of the LAS coverage.

#### 4.6 Summary

In the current Chapter, there was a discussion about the process followed to develop the conceptual design for conducting the case study of the London EMS. The proposed FIELDS framework regardless of the focus on a specific system can be used to conceptualise most of the EMS, at least in the UK.

The Chapter commences with the analysis of the interaction points between the participating organisational components of the distributed system. Afterward, the rationale behind the software tools selection was stated.

The following section demonstrated the definition of the IRM for the FIELDS framework and the boundaries between the interoperating component models. Moreover, there was a discussion about the protocol used in this DS project for data communication and

---

time synchronisation. The HLA RTI standard was used to conceptualised the DS federation.

In sequence, there was an analysis on the conceptualisation phase for each individual federate model. The main functionalities and the level of abstraction for the ambulance service and the A&E department models were determined. From this analysis, the list of the more important events for the results collection was defined. Finally, the data collection process was determined.

In the following Chapter 5, the realisation phase of the hybrid system will be demonstrated, together with the experimental design stage. Furthermore, the first results of the prototype DS model will be analysed in the same Chapter.

---

## EMS prototype model

*This chapter presents the implementation of the prototype hybrid ABS DES DS model of EMS based on the proposed FIELDS framework. The technical implications of developing the individual models are discussed. Further, the validation of both the federates and the federation of the distributed system is analysed. Finally, experimentation for testing the DS model's performance was conducted and the results are discussed.*

## 5.1 Overview

In this Chapter, the realisation phase of the proposed FIELDS framework is presented. The development of the hybrid ABS-DES DS model of EMS is demonstrated. Initial runs of the model are performed for validation purposes (Anagnostou *et al.*, 2013) and further experimentation on performance testing is conducted (Nouman *et al.*, 2013).

The previous Chapter 4 analysed the FIELDS framework. That is the conceptual design of this simulation project and constitutes the highest level of the *development* layer in the distributed simulation methodology. The layer above the *development* one, namely *planning*, was established in Chapter 1, where the problem formulation was analysed. This Chapter deals with the realisation step that constitutes the lower level of the development block in the DS creation process, which was discussed in Chapter 3. Furthermore, the experimentation block is discussed by running experiments with the prototype model on performance testing.

Following the steps in the realisation and experimentation stages, Chapter 5 is organised as follows. First, the explanation of the development of the ambulance service and the A&E department models is discussed, respectively, where the technical issues are analysed. Furthermore, the data collection process is discussed. In sequence, the validation of the individual models, as well as the DS is demonstrated. Lastly, the results of running experiments on performance testing of the prototype distributed system are discussed.

## 5.2 Realisation phase of the prototype model

As pointed out in Chapter 3, there are three possible scenarios for the individual models, that is, when none of the models exist, all models exist, or some of the models exist. This project falls in the first category where none of the federate models exist. Therefore, the



---

federate models are designed to be interoperable. That is, the transparency level of each component model (what information each model allows to be visible), the global variables of the distributed system and the ownership of these variables (which model can update the global variables) were defined from the initial design. Additionally, the federate models can run independently as standalone simulations with some modifications in the ambulance entry point at the A&E DES models.

As mentioned in Chapter 4, the simulation model of this project is developed entirely with open-source software. Repast simulator ([repast.sourceforge.net](http://repast.sourceforge.net)) and poRTIco RTI implementation ([www.porticoproject.org](http://www.porticoproject.org)) open-source software packages are used for the development of the simulation models and the middleware interface, respectively. Repast suite is an ABS toolkit that comes with a Java application programming interface. Its basic simulation engine is the `Schedule` class, which is a discrete event engine. Similarly, poRTIco comes with a Java application programming interface and supports the RTI implementation of the HLA standards. At the time that this project was under development poRTIco v2.0 was not yet released. Therefore, the first implementation was done with poRTIco v1.0 release that implements HLA 1.3. However, by the time that the project completed, poRTIco v2.0 has been released and, therefore, it was adopted for the latest version of the presented DS EMS. poRTIco v2.0 implements HLA-1516-2010. The earlier version, HLA 1.3, was first implemented in 2001 and is compatible with industrial standard XML format for object modelling (IEEE-1516, 2000). The latest version, HLA Evolved, is built on the previous HLA 1.3 version and was first published in 2010. The new added features, such as web service support that enables HLA-based DS models to run over the internet, and improved fault tolerance mechanism that enables failing federates to be removed without affecting the whole federation, are expected to improve the HLA experience (IEEE-1516, 2010).

---

The following subsections describe in details the implementation of each federate model and the RTI.

### 5.2.1 Ambulance service ABS federate

In the ambulance service conceptual design, the interactions among the objects of the model were identified. The use case diagram (see Figure 27) depicted the active agents and their activities. The active agents live and act in an environment, and interact with it. Therefore, apart from the active agents, the passive agents and the topology of the living space should be defined.

For this project, a grid topology was implemented. However, admittedly a GIS topology would have added value to the current model, i.e., more accurate distance calculation, more realistic visualisation, etc. The different topologies for ABS are detailed in Chapter 2. To add the feature of GIS topology is considered for the future work of this thesis.

The grid topology constitutes the area of the ambulance service coverage. On this surface, there are some passive agent objects, namely, the ambulance stations, where the ambulance vehicles are stationed, and the hospitals, where the A&E departments are located. The ambulance stations have location and capacity properties, similarly with the A&E departments that have location and capacity, too. The locations of both are defined with X and Y Cartesian coordinates in the grid. The capacity of the ambulance stations defines how many ambulances are stationed in the specific location. However, the hospital capacity is used to calculate the availability of the A&E departments, locally.

$$Capacity = Availability + Occupancy$$

If the model executes as a standalone simulation, the capacity value is defined in the initialisation of the model. In the case of a distributed system, each regional hospital federate

sends the capacity value at the initialisation phase of the DS, that is, the registration of the model instance to federation.

### 5.2.1.1 Ambulance service model pseudocode

The main logic of the ambulance service model is described in the following pseudocode. As in every simulation program, the program is running until the simulation time reaches the predetermined end of simulation time. Before the while loop begins, the simulation's initialisation routine, which is not included in the pseudocode, is run. At the initialisation, the creation of the environment takes place.

The simulation routine starts with the generation of an emergency call. This event triggers the scheduling of the next emergency call arrival. The time to next arrival is taken from the next value of a normal distribution. Normal distribution was selected, whereas one might expect an exponential one to represent randomness in emergency calls generation, because the data that populated the model was normalised, as will be explained in the next subsection. It is acknowledged that using normal distribution is not realistic; however, at this stage of developing the prototype, it helps to get practical results. The probability density function (pdf) of a normal distribution is described by Gaussian functions as:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

where,  $x$  is the variate,  $\mu$  is the mean, and  $\sigma^2$  is the variance.

The emergency call comes with a location property that is later adopted by the patient of the specific incident. That is, the X and Y coordinates of the incident and are read from a data file. Additionally, the emergency call has a unique ID which is passed to the patient and it is used for identification of the incident and verification and validation purposes.

Once the emergency call is located in the grid, a patient agent/object is created at the same location. The patient agent also has some properties, i.e., condition (whether it is major or minor incident), on-scene treatment time that the patient needs, and hospitalisation (whether the patient needs transfer to an A&E department or not). The emergency call agent has the method of finding an ambulance and sends it to the scene of the incident (lines five to 10 of the pseudocode).

```
1. WHILE current time < simulation end time
2.   Generate emergency call
3.   Schedule next emergency call generation
4.   Generate patient
5.   Find the closest available ambulance
6.   SET ambulance availability to FALSE
7.   IF ambulance is found
8.     Calculate the distance between emergency and ambulance locations
9.     Calculate the travel time from ambulance station to emergency scene
10.    Move ambulance to emergency
11.    IF patient needs hospitalisation
12.      Find closest available hospital
13.      SET on-scene treatment time
14.      Calculate distance between scene and hospital
15.      Calculate travel time from scene to hospital
16.      Ask chosen hospital to reserve resources
17.      Move ambulance and patient at current time + treatment time + travel time to chosen hospital
18.      Calculate distance between chosen hospital and ambulance station
19.      Calculate travel time from hospital to ambulance station
20.      Move ambulance from hospital to station
21.      SET ambulance availability to TRUE
22.    ELSE
```

```
23.          SET on-scene treatment time
24.          Move ambulance at current time + treatment time to am-
            bulance station
25.          SET ambulance availability to TRUE
26.      END IF
27.  END IF
```

Because the emergency call is cleared from the environment after completing its job, that is, to find an ambulance and send it to the scene, a condition of running the routine only if the ambulance is found is necessary for continuing the attempts in every simulation time unit. Otherwise, after the first attempt the emergency call object would be deleted. As a consequence, if at this first attempt there is no ambulance available, the patient would stay at the incident location forever, or until the end of simulation run. The methods of calculating the distance between the ambulance initial position and the incident, and moving the ambulance vehicle to the scene also belong to the emergency call agent. As explained in Chapter 4, the calculated distance is the Euclidean distance between the two points. The travel time is extracted, from the formula given in Chapter 3, in hours and then converted to minutes, which is a more convenient time unit for the ambulance service simulation.

The remaining routine (from line 11) of the pseudocode is run by the ambulance agents. The ambulance object checks whether the patient needs transfer to a hospital and if it is the case, the ambulance searches for the closest available A&E department. Additionally, it sets the on-scene treatment, property that comes with the patient and is relevant to the condition, calculates the distance to the chosen hospital and then calculates the travel time in a similar way that the emergency call performed the same activities. Line 16 is highlighted and here is when the ambulance service model communicates with the chosen hospital model and asks to reserve resources for the specific incident. A note should be made at this point that the ambulance will continue to search for the appropriate hospital if not found at the first search. In the actual model, there are conditions to ensure that this is happening but

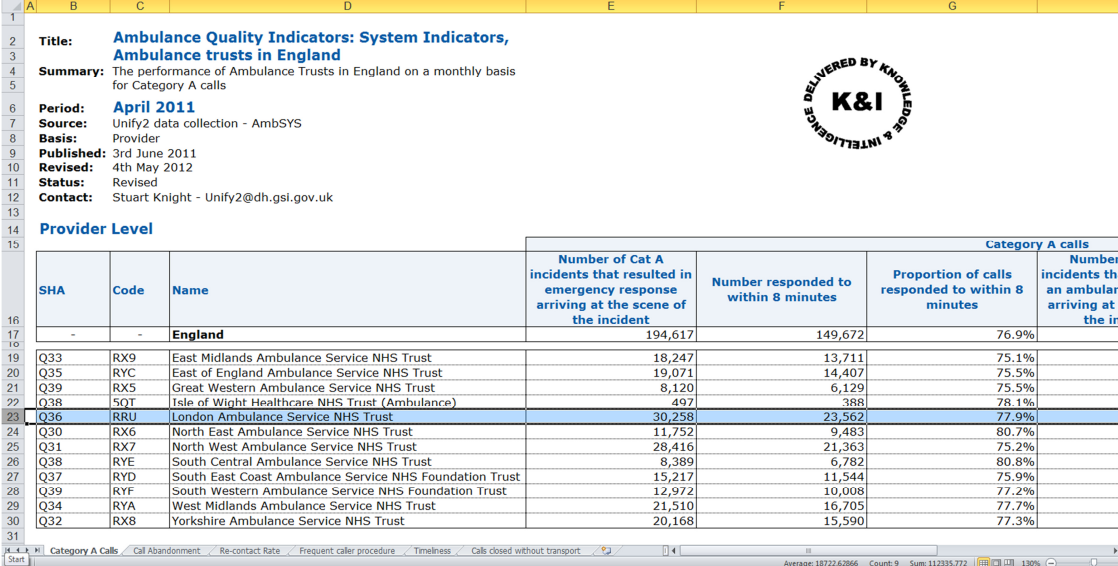
---

it is omitted in the pseudocode in order to make the process better understood. The next step for the ambulance is to schedule the travel. The ambulance with the patient will move to the hospital at time equal to current time plus on-scene treatment time plus travel time from the scene to the hospital. The same procedure takes place in order to move the ambulance from the hospital back to the ambulance station. If the patient does not need transfer to a hospital the ambulance will perform the on-scene treatment and then travel back from the scene to the ambulance station. Another note should be made here to clarify that there is a unique ID for the specific emergency that is generated with the emergency call generation, then it is passed to the patient and after that it is passed to the ambulance. Also, the hospitals, ambulance stations and ambulance vehicles can be identified by unique IDs. This is happening for validation and verification purposes. That is, to be able to track the whole route of the incident. Furthermore, it should be noted that the `moveTo()` method in `Repast` is happening instantaneously and, therefore, the ambulance stays at the starting point of each journey, but of course is unavailable, until the arrival to destination time is reached in the simulation execution and then moves directly to the destination grid cell. This process does not offer the best of the visualisation effects, but it does not affect the functionality of the ambulance model.

The results are exported in comma separated values (csv file). The values that are recorded are the properties of the agents, i.e., IDs, condition etc., and the timestamps of the notable events, as presented in Chapter 4 (see Table 7).

### **5.2.1.2 Data collection**

The ambulance service prototype model of this project is a 1:5 scaled-down EMS system based on the London EMS. London ambulance service (LAS) is a NHS trust that ultimately is responsible to the DoH. Healthcare statistics are published on a regular basis by NHS England, the DoH and the health and social care information centre (HSCIC). The current



**Title:** Ambulance Quality Indicators: System Indicators, Ambulance trusts in England

**Summary:** The performance of Ambulance Trusts in England on a monthly basis for Category A calls

**Period:** April 2011

**Source:** Unify2 data collection - AmbSYS

**Basis:** Provider

**Published:** 3rd June 2011

**Revised:** 4th May 2012

**Status:** Revised

**Contact:** Stuart Knight - Unify2@dh.gsi.gov.uk

**Provider Level**

SHA	Code	Name	Category A calls		
			Number of Cat A incidents that resulted in emergency response arriving at the scene of the incident	Number responded to within 8 minutes	Proportion of calls responded to within 8 minutes
-	-	England	194,617	149,672	76.9%
Q33	RX9	East Midlands Ambulance Service NHS Trust	18,247	13,711	75.1%
Q35	RYC	East of England Ambulance Service NHS Trust	19,071	14,407	75.5%
Q39	RX5	Great Western Ambulance Service NHS Trust	8,120	6,129	75.5%
Q38	50T	Isle of Wight Healthcare NHS Trust (Ambulance)	497	388	78.1%
Q36	RRU	London Ambulance Service NHS Trust	30,258	23,562	77.9%
Q30	RX6	North East Ambulance Service NHS Trust	11,752	9,483	80.7%
Q31	RX7	North West Ambulance Service NHS Trust	28,416	21,363	75.2%
Q38	RYE	South Central Ambulance Service NHS Trust	8,389	6,782	80.8%
Q37	RYD	South East Coast Ambulance Service NHS Foundation Trust	15,217	11,544	75.9%
Q39	RYF	South Western Ambulance Service NHS Foundation Trust	12,972	10,008	77.2%
Q34	RYA	West Midlands Ambulance Service NHS Trust	21,510	16,705	77.7%
Q32	RX8	Yorkshire Ambulance Service NHS Trust	20,168	15,590	77.3%

Figure 31: Screenshot of published online ambulance service performance data spreadsheet (source: NHS England(a))

ambulance model is populated with data that was published by the DoH (at the time of this thesis writing the data is available in the NHS England website). DoH publishes monthly performance measures for ambulance services against some quality indicators. The data that is used in the ambulance service prototype involves the period from April 2011 to March 2012. Figure 31 shows an example of the data spreadsheets that are available online.

The ambulance service model was constructed based on a simplified version of the LAS. LAS has a fleet of 998 vehicles, 375 of which are ambulances. It covers an area of 620mi<sup>2</sup> and has 70 ambulance stations and five headquarters across Greater London ([www.londonambulance.nhs.uk](http://www.londonambulance.nhs.uk)). In the area of coverage there are 32 general A&E departments ([www.nhs.uk](http://www.nhs.uk)). In Figure 32 the blue landmarks indicate their location on the map and the green landmarks indicate the ambulance station locations.

It can be seen from Figure 32 that in highly populated areas, i.e., central London, the density of available services is higher. In this prototype the call generation and the capacity of the A&E departments are equally distributed in the available space. Hence, the ambulance stations and hospitals are located in equally distributed distances. Moreover, the average travel speed in London is generally low, ranging from less than 10mph in central London and up to 22mph in outer London (Steinbach *et al.*, 2012). In this project the average speed of the ambulance vehicles is considered to be 15mph. From the above analysis the specifications of the ABS ambulance service model are: cover area = 150mi<sup>2</sup>, number of A&Es = 6, number of ambulance stations = 14, number of ambulances = 75 and ambulance average speed = 15mph, emergency calls arrivals = 23.80 per hour. A summary of the ambulance model specifications is shown in Table 8.

### 5.2.1.3 Ambulance model verification and validation

The term verification in a simulation project describes the process taken to ensure that the

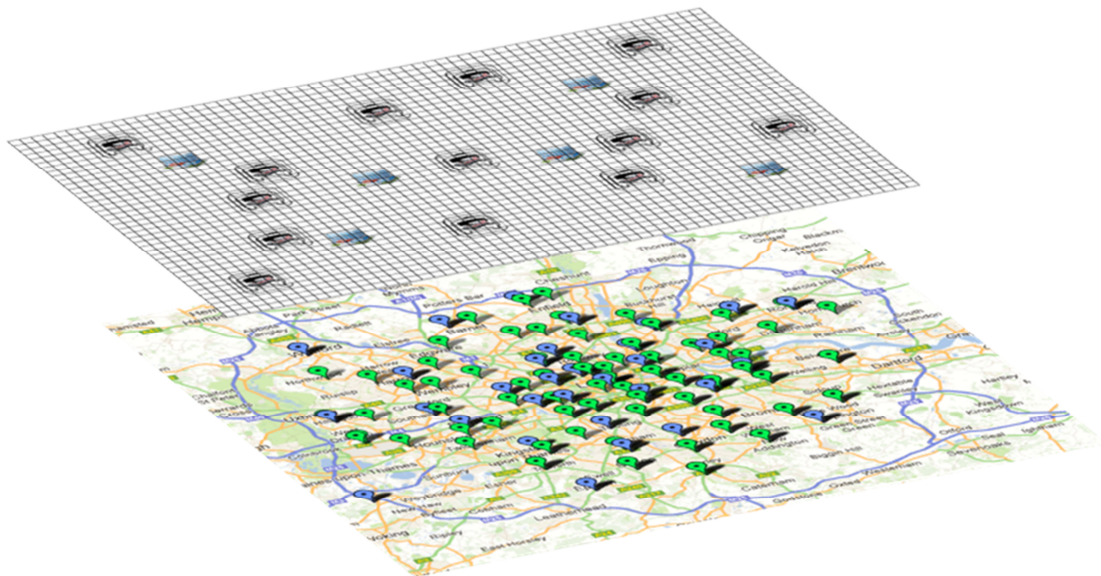


Figure 32: Ambulance service area



Table 8: Ambulance model specifications

<b>Ambulance model</b>					
<b>inter-arrival time normal distribution</b>		<b>patient condition</b>		<b>average speed</b>	15mph
mean	2.52	minors	26%	<b>correction factor</b>	1.32
SD	0.09	majors	74%	<b>coverage area</b>	150sqmi
<b>time on scene (min) normal distribution</b>		<b>need transfer to A&amp;E</b>		<b>ambulance stations</b>	14
mean	22.52	yes	62%	<b>ambulances per station</b>	9*5+5*6
SD	10.54	no	38%	<b>hospitals</b>	6

model is programmatically correct. Throughout the model development, the code was being checked by the author for errors and inconsistencies. Upon the completion of the programming stage, the model was checked by an expert Java programmer with experience in M&S, too.

For the validation of the ambulance service model, i.e., to ensure that the model is doing what is intended to do by design, pilot runs were performed and the results were compared with the real system performance. The ambulance services in the UK operate against some targets. One crucial performance measurement for the ambulance service is the response time. That is, the time from receiving the emergency call until the time that the ambulance arrives at the scene of incident. The target is that 75 per cent of category A calls (life threatening emergencies) are reached within eight minutes and 95 per cent within 19 minutes. From published data in the financial year 2011-12, LAS response time for 99 per cent of category A calls was 19 min (LAS Annual Review 2011-12). The results from the model for one month simulation time are shown in Figure 33. As can be seen from the graph, the response time of 99 per cent of incidents was less than 19 min.

### 5.2.2 A&E department DES federate

In Chapter 4, the conceptual design of the A&E department model was described. From the flowchart illustrated in Figure 30, the process and the decisions in the model can be seen.

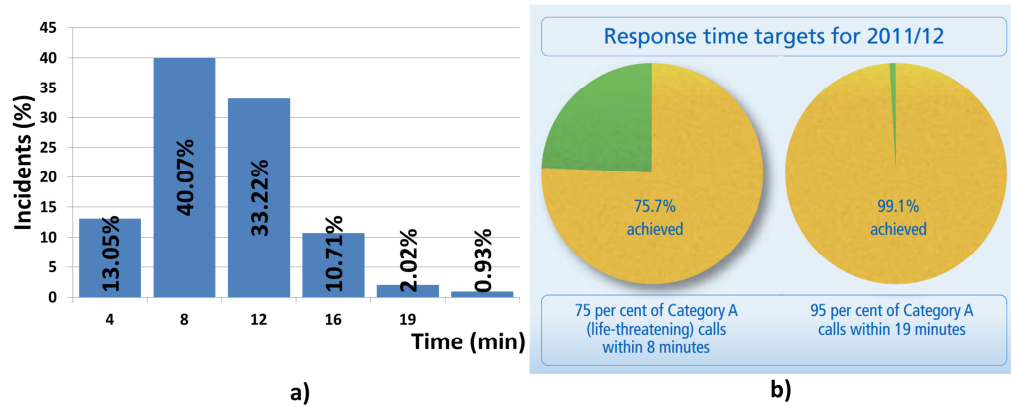


Figure 33: Ambulance service model response time a) simulation, b) real system

Due to the fact that the focus of this thesis is in the interoperability of the federate models, the simulation model is designed with high level of abstraction. For example, all the test procedures within the A&E department, i.e., lab tests, imaging, etc. are omitted. Another convention is that there is only one type of resources, and that is called clinical staff and includes nurses, doctors, etc.

As mentioned in previous Chapters, both the ambulance service and the A&E department models are built in Repast Suite. To convert the ABS simulator to a DES one, the fundamental components of a DES, namely queues, work stations and resources, were hard-coded. The discrete event nature of the `Schedule` class of the Repast simulation engine was utilised to manually add the next event in the schedule list. To avoid conflict when updating the shared variables via the RTI, the time progresses in every simulation time unit, called “tick” in Repast, even when there is no event scheduled for that time unit. This synchronisation compromise was necessary, even though it comes with a cost of a local execution time overhead for the DES model.

The A&E department models that constitute the DS emergency medical system are identical and their structure is a simplified version of a busy A&E department in London. As

mentioned above, the different processes are aggregated. Therefore, the department consists of three procedures (triage, minors and majors) in order to differentiate the main categorisation of patients. For example, walk-in patients that need only advice exit the hospital straight after the triage service. Walk-in patients with minor injuries and more serious conditions, after triage, enter the minors and majors queue, respectively. Patients that arrive by ambulance enter directly to minors or majors service without joining the queue. The total time of the services is the time that the patients spend with the clinical staff in each process. By doing this, it is expected the results to show lower total time in the system than the actual A&E departments. The model configuration is based on literature and online data published by the UK NHS and the DoH.

When the model executes in the DS system, the ambulance arrivals is the interface point with the ambulance service models. The ambulance model is responsible for sending patients at the ambulance arrivals entry point. Before the ambulance patient arrives, the ambulance model has already reserved resources. Further, whenever there is a change in the department availability, this is sent to the ambulance service model. When the A&E department model executes as a standalone simulation, the model should be modified in order to receive ambulance patients by a probability distribution.

### 5.2.2.1 A&E department model pseudocode

The main logic of the A&E department model is described in the following pseudocode.

```
1. WHILE current time < simulation end time
2.   Schedule next walk-in arrival
3.   Schedule next ambulance arrival // in the standalone execution
3.   // Ambulance patient arrival is controlled by the ambulance model
4.   IF patient is walk-in
5.     Add to triage queue
6.   ELSE IF patient came with ambulance
```

```
7.         IF patient has majors conditions
8.             Start majors service
9.             Exit at current time + majors treatment time
10.          Release majors resources
11.          Update availability
12.        ELSE IF patient has minors conditions
13.            Start minors service
14.            Exit at current time + minors treatment time
15.            Release minors resources
16.            Update availability
17.        END IF
18.    END IF
19.    IF majors queue is not empty
20.        IF majors resources are free
21.            Start majors service
22.            Seize majors resources
23.            Update availability
24.            Exit at current time + majors treatment time
25.            Release majors resources
26.            Update availability
27.        END IF
28.    END IF
29.    IF minors queue is not empty
30.        IF majors resources are free
31.            Start minors service
32.            Seize minors resources
33.            Update availability
34.            Exit at current time + minors treatment time
35.            Release minors resources
36.            Update availability
37.        END IF
38.    END IF
39.    IF triage queue is not empty
```

```
40.          IF triage resources are free
41.              Start triage service
42.              Seize triage resources
43.              Update availability
44.              IF patient has majors conditions
45.                  Add to majors queue at current time + triage
                    treatment time
46.              ELSE IF patient has minors conditions
47.                  Add to minors queue at current time + triage
                    treatment time
48.              END IF
49.              Release triage resources
50.              Update availability
51.          END IF
52.  END IF
```

In the same way with the ambulance service model, as in every simulation program, the program runs until the simulation time reaches the predetermined end of simulation time. Before the while loop begins, the simulation's initialisation routine, which is not included in the pseudocode, is run. At the initialisation, the values of the available resources are set.

The simulation routine starts with the scheduling of the patient arrivals. A&E departments have two arrival points, one for walks in patients and one for the ambulance arrivals. When the model is run as individual simulation, both entry point arrivals should be scheduled. When it runs in the distributed system, the code schedules the walk-in arrivals only. The time to the next arrival is taken from a normal distribution, similar to the arrival logic explained in the ambulance service model. The ambulance patients' arrivals are controlled by the ambulance services federate. Afterwards, there is a check; if the patient object is walk-in patient, the entity joins the triage queue. The ambulance arrivals will join either the majors or the minors department according to the entity's condition. The exit time is scheduled at current time plus the treatment time. At the exit time the resources are released. For

the ambulance arrivals, there is no seizing of resources before starting the service because the required resources have been reserved earlier by the ambulance service model. The three queues, i.e., triage, minors and majors queues, are observed at every time unit. If a queue is not empty and the respective service block and resources are available, the service starts, the resources are seized and the availability is updated and sent to the ambulance model through the RTI. The exit time is scheduled at simulation time equal to current time plus the respective treatment time. At the exit time the resources are released and the availability is updated again and sent via the RTI. If the service is the triage, then the entity does not exit the department but rather joins the queue for minors or majors according to the condition attribute.

The results are exported in comma separated values (csv file). The values that recorded are the attributes of the entities, i.e., IDs, condition etc., and the timestamps of the notable events, as presented in Chapter 4 (see Table 7).

### **5.2.2.2 Data collection**

Similar to the ambulance service model, the data that populates the A&E department models is acquired from published online sources. Healthcare statistics are published in regular basis by the NHS England, the DoH and the HSCIC. The A&E models are populated with data that was published by the DoH (at the time of this thesis writing the data is available in the NHS England website). DoH publishes weekly performance measures for A&E departments attendances. The data that is used in the A&E prototype model involves the period from April 2011 to March 2012. Figure 34 shows an example of the data spreadsheets that are available online.

As part of delivering the Government's reforms to the NHS, changes have been made in the A&E departments across Greater London. In the data collection, only the departments

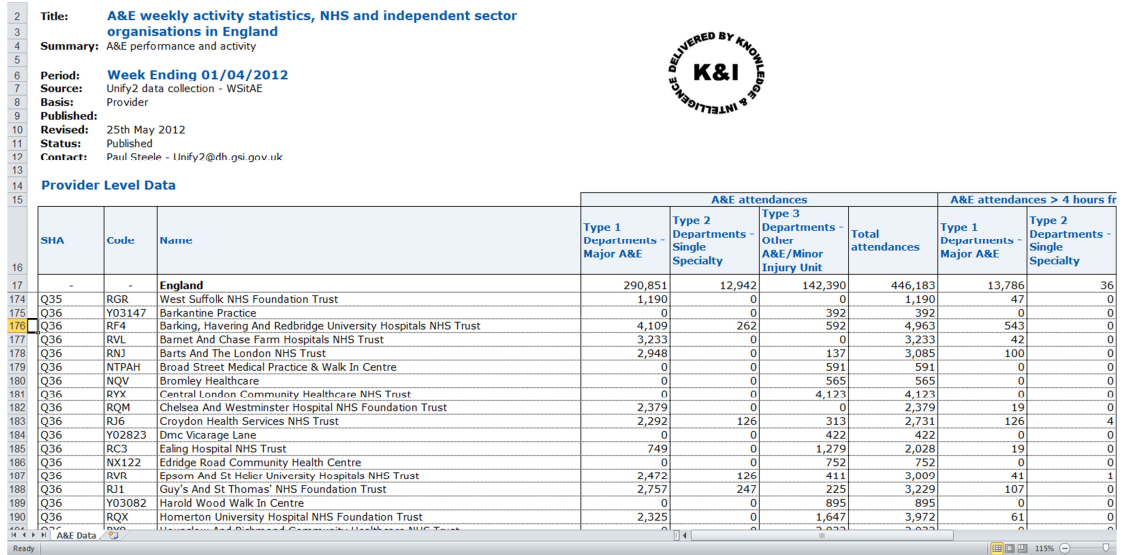


Figure 34: Screenshot of published online A&E attendance data spreadsheet (source: NHS England(b))

that were consistently part of the UK’s DoH data collection throughout 2011-12 (www.dh.gov.uk) there were considered. The data has been aggregated and distributed normally to all A&E hospitals across Greater London. Therefore, all departments have the same workload and capacity. The bed capacity and the number of resources values were decided according to a busy A&E department in London. The configurations of the models are: triage beds = 5, minors beds = 12, majors beds = 24, clinical staff = 15, walk-in arrivals = 12.60 per hour. A summary of the A&E model specifications is shown in Table 9.

**5.2.2.3 A&E model verification and validation**

The same verification procedure took place in this model, too. That is, the code was being checked by the author for errors and inconsistencies throughout the coding process and, upon the completion of the programming stage, the model was checked by an expert Java programmer with experience in M&S.

Table 9: A&amp;E model specifications

<b>A&amp;E model</b>			
<b>walk-in inter-arrival time normal distribution</b>		<b>patient condition</b>	
mean	4.81	minors	35%
SD	0.59	majors	65%
<b>time in triage normal distribution (with staff)</b>		<b>need treatment</b>	
mean	7.00	yes	60%
SD	2.00	no	40%
<b>time in minors normal distribution (with staff)</b>		<b>number of staff</b>	
mean	30.00		15
SD	10.00	<b>triage capacity</b>	5
<b>time in majors normal distribution (with staff)</b>		<b>minors capacity</b>	
mean	40.00		12
SD	10.00	<b>majors capacity</b>	24

One key performance indicator for A&E departments is the total duration of patient journey in the departments. The target for the UK's A&E departments is to treat all patients within four hours. According to published data in the financial year 2011-12 for London strategic health authorities, 94 per cent of the attendances spent up to four hours in the A&E departments. The total time in the system that was extracted from the distributed simulation runs is shown in Figure 35.

The graphs in Figure 35 show that the overall simulation behaves as expected according to its design. For example, the aggregation of all processes within the departments leads to less total time in an A&E department since there are not any queues in-between the various processes, exactly as anticipated. Furthermore the fact that 40 per cent of the attendances need only advice (The Press Association, 2011) leads to high volume of discharges in less than 30 minutes, which mean that they just go through the triage service.



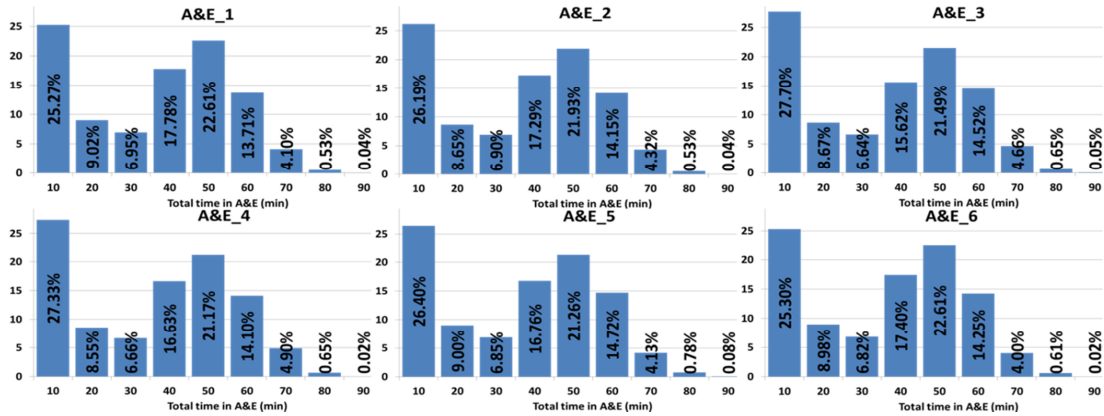


Figure 35: Total time spent in the six A&amp;E departments

### 5.2.3 Time management strategy

From the proposed stepped methodology for DS project building, at this point the middleware implementation begins. Before commencing the coding for the RTI, the time advance strategy of the distributed system should be decided.

As mentioned before, there are two schemes for synchronisation, namely conservative and optimistic. In this project a conservative approach is adopted. That is, the system does not allow the local simulations, i.e., federates, to process any events in future time than the current federate simulation time. The lookahead value for the current system is one simulation time unit. The lookahead value of one is decided because in the ABS model the sequence of events cannot be predicted and therefore there is a risk for potential conflict in the availability of an A&E department. For example, as explained in Chapter 2, by the definition of lookahead, federates are not allowed to generate any event with timestamp less than the current simulation time plus the lookahead. However, the events that have timestamp less than the current time plus lookahead will be processed. If these events include an ambulance agent seeking a hospital and selecting one with the last available re-

source, and a walk-in patient arrives in the chosen hospital, they will both try to seize the one last available resource, but as illustrated in Figure 36, this action is not allowed by the lookahead. At the next update, the local arrival to the hospital (i.e., walk-in arrival) will seize the resource and the A&E model will update the federation that its availability is zero and, therefore, the ambulance model will select an alternative A&E.

The logical time advance service that is implemented in this project is the time advance request. It was explained that TAR is used when the simulation is time-driven. In the developed hybrid simulation, federates are implemented in two different simulation paradigms. As identified in Chapter 4, Section 4.3, ABS is a time-driven simulation while DES is an event-driven one. Therefore, in order to facilitate both ABS and DES, TAR is selected as the most appropriate time advance service despite the fact that TAR tends to slow-down the simulation execution time.

### 5.2.4 Middleware implementation

The next step, after defining the time management strategy is to implement the RTI. In this project the poRTIco open source software is used for this purpose. The current implementa-

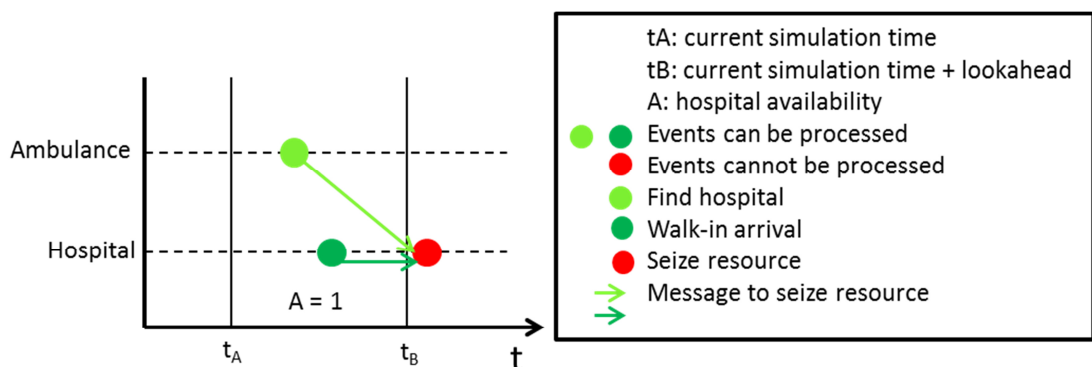


Figure 36: Lookahead

---

tion is the very basic HLA interface. In future research it is planned to enhance the functionality by adding HLA features, such as management object model (MOM) which can control the functioning of the system components, i.e., federation, federates and RTI.

poRTIco 2.0 can be used with both C++ and Java programming languages. Since Java-based Repast simulator is used for developing the federate models, Java implementation was selected for the RTI, too. Repast toolkit is compatible with Java version 1.6, and therefore, for compatibility purposes, the first implementation was created in the earlier version of poRTIco 1.0, which implements HLA 1.3 and compiles with Java version 1.6. The second version of the current DS is created with the new version of poRTIco 2.0 that implements HLA 1516e (Evolved) along with the backward compatibility for HLA 1.3. HLA 1516e, however, is restricted to be used with the latest version of Java 1.7. Repast Symphony Java API is compatible with Java 1.6 but there are some issues with the Java 1.7 compiler. For example, the change in the array sorting algorithm in Java 1.7 has as a result not to be able to use the schedule prioritisation functionality of the Repast `Schedule` class. The prioritisation functionality is optional and allows the modellers to control the order of the scheduled methods execution when there is more than one method scheduled for the same simulation time unit. If no prioritisation is set, by default, Repast will execute the methods in a random order. In the first implementation of this project, schedule prioritisation was used for checking the queues in the DES models. The two queues that require prioritisation are the queue for the minors department and the queue for the majors department. Obviously, the majors queue has higher priority over the minors queue. The second version of the current DS uses the HLA 1516e RTI implementation, hence there is an incompatibility between the simulator and the RTI software. To remedy the Repast and poRTIco 2.0 incompatibilities, the default random ordering of Repast is used. This may compromise slightly the functionality of the DES models, however it does not affect the distributed system per-

formance. Another remedy option would be to recompile poRTico with Java 1.6, which supports the earlier versions implementation.

The run-time interface involves some additional artefacts, as shown in Figure 37. Two types of object models are involved in HLA, the FOM and the SOM. Both objects follow the same documentation approach that is defined by the HLA OMT. However, the contents of FOM and SOM are not defined by HLA. The SOM is internal object in a federate and describes the information that the federate shares within the federation, while, the FOM belongs to the federation and describes the shared information across the federation. In HLA 1.3 the FOM is passed to the RTI as a file, yet, in HLA 1516e the FOM should be created in an XML file format.

The run-time services that are provided by the RTI to federates and by the federates to the RTI are described by the HLA interface specification. There are six groups of services:

- i. Federation management – involves basic functions for creation and operation of a

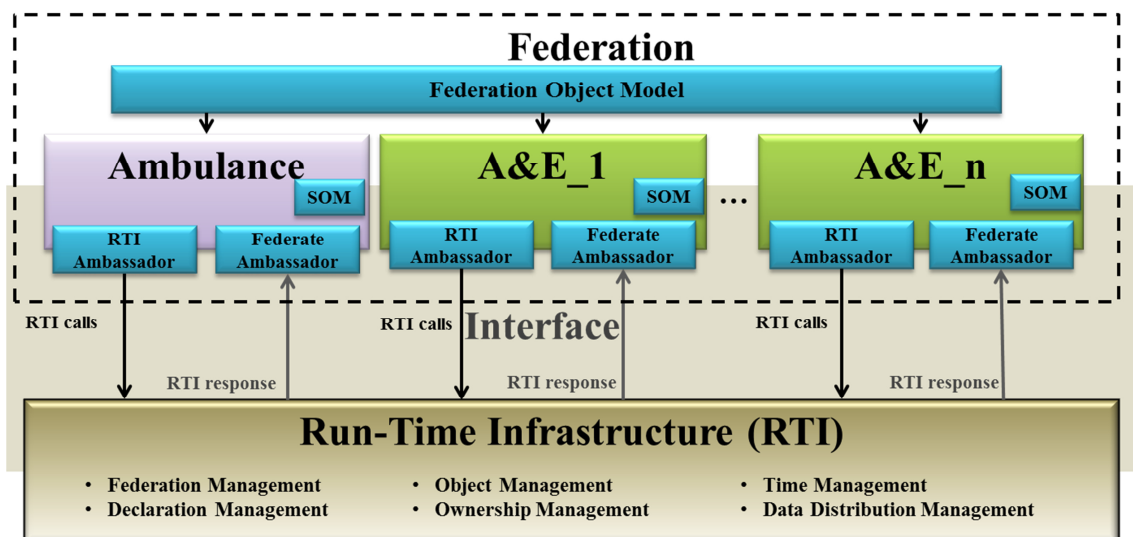


Figure 37: DS implementation

- federation. More specifically, it defines how federates create and join federations, and how federates are synchronised to the same time.
- ii. Declaration management – involves the way that federates declare the shared information. That is, federates state their intention to subscribe data and interactions.
  - iii. Object management – involves the way that federates use the objects that they have ownership of them, i.e., deletion/creation of objects, etc.
  - iv. Ownership management – involves the ownership of the registered objects and the way that federates can dynamically acquire ownership.
  - v. Time management – involves synchronisation of data exchange during run-time.
  - vi. Data distribution management – involves the way that data is transferred within the federation.

In each federate model there are two classes included for achieving communication with the RTI, an RTI ambassador class which is responsible for sending information to the RTI and a federate ambassador class which is responsible for receiving information from the RTI (see Figure 37). A diagrammatic representation of the interactions between each federate and the RTI interface is shown in the sequence diagram of Figure 38. The ambulance model federate and each A&E model federate send a time advance request to the RTI at every simulation time unit (in Repast this is called tick) and wait for the RTI to send the TAG response to permit time advance. When there is a change in the level of A&E availability, this A&E federate sends its availability to the ambulance federate through the RTI. In a similar way, when the ambulance federate needs to transfer a patient to an A&E, it sends the patient instance to the selected A&E federate. All A&E federates have unique identifiers. Before any update is committed, the identifier of the A&E that sends or receives the message is verified.

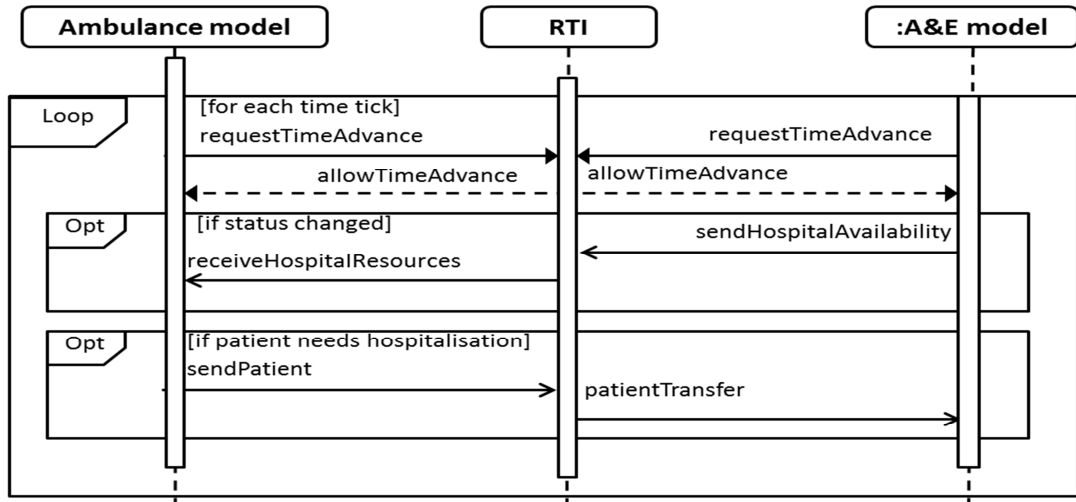


Figure 38: Sequence diagram showing the interactions via the RTI

During the first time advance all hospital models update the ambulance model about their availability. The ambulance model always keeps an updated log of availability for its local hospital agents that are part of the ABS environment, this information is sent out by the hospital federates through RTI and get refreshed at every time tick, if necessary. Once an ambulance has an emergency and needs to transfer a patient to an A&E, it looks for the nearest hospital for availability. The selection of the hospital is done on the basis of the distance from the patient's location and the availability of the hospital. Once a hospital is selected by the ambulance, a notification is sent through the RTI to the particular hospital's A&E to book a place, namely, to reserve resources. All communication among federates are time controlled by RTI using TAR.

#### 5.2.4.1 Middleware verification and validation

The middleware implementation was verified throughout the coding process. The program was checked for errors and inconsistencies.

Some characteristics of the distributed system were used in order to validate that the data and time synchronisation was correct. For example, the hospitals and patients IDs were crosschecked between the interacting federates. Furthermore, the delay during the handover of a patient is not included in the model. Hence, referring to Table 7, the time relationship  $t_{a6} = t_{h1}$  (where,  $t_{a6}$  is the time that the ambulance arrives at the hospital location in the ABS model and  $t_{h1}$  is the time that the patient arrives by ambulance to the ambulance entry point in the DES model) is always true, when the patient arrives at the hospital by ambulance. This feature was used to validate the time synchronisation.

### 5.3 Experimental design

To test the functionality of the prototype hybrid ABS DES DS of EMS, two sets of experiments were conducted. Firstly, the execution time performance was tested over a network and a single processor. At this stage the experiment involves a comparison of the two implemented versions of HLA, namely, HLA 1.3 and HLA 1516e. Secondly, the scalability of the prototype system was tested by performing three different scenarios. The first scenario demonstrates an increase in the emergency calls and the walk-in arrivals by 10, 20 and 30 per cent. The second scenario demonstrates an increase of only the emergency calls by 10, 20 and 30 per cent. Lastly, the third scenario demonstrates an increase of only the walk-in arrivals by the same percentages. Each scenario runs under controlled settings in order to maintain consistency in the communication level between federates. For example, the emergency calls are generated in the same pattern and the incidents happen at the same location. A&E walk-in arrivals follow the same pattern, too. Following the distributed simulation methodology that was presented in Chapter 3, this is the last step in the DS project building process.

---

### 5.3.1 Network settings

The first set of runs involves execution of the distributed system over a 2-core processor and over a homogeneous network connected with 1Gbps network card. The network consists of seven computers on a non-dedicated network interconnected via LAN. Each PC contained an i5-2500 processor at 3.30GHz speed and 4.00GB RAM. Each PC had Microsoft Windows 7 with Java 1.7 JRE installed with poRTico package for both HLA 1.3 and HLA 1516e.

### 5.3.2 Results

In the following subsections, there is a presentation and discussion of the experimental results. Each experiment ran for one month simulation time and the average of five runs was recorded for each scenario. In Mustafee *et al.* (2009) and Taylor *et al.* (2002b), five replications were used to reduce variance due to communication network and the operating system. It was also mentioned that many DS studies present results of a single run.

#### 5.3.2.1 Performance testing

The performance testing experiments are divided in two levels. The first level involves the creation of two federations. One federation is created with RTI implementation based on the earlier HLA 1.3 version and the other federation is created with RTI implementation based on the most recent HLA 1516e version. The second level involves the execution of both federations in two different environments, i.e., single-node and distributed environment. In each environment and for each type of federation six tests are conducted, increasing the number of the participating federates by one. Starting from two federates, the ABS ambulance model and one DES A&E department model, and incrementing up to seven, the ABS ambulance model and six DES A&E department models that constitute the 1:5 scaled-



down London EMS. The average simulation execution time spanned from approximately 36 minutes to approximately 63 minutes, as shown in Table 10.

Figure 39 shows the resulting graphs of the different scenario execution time for a single-node and a distributed environment, respectively. The graph depicts the execution time (Y-axis) in relation to the number of connected federate models (X-axis).

Table 10: HLA 1.3 and 1516e simulation execution time when executed on a single-node and on a homogeneous network

	Single-node environment				Distributed environment			
HLA version	1.3		1516e		1.3		1516e	
	Execution Time (min)							
Number of Federates	Average of 5 runs	SD	Average of 5 runs	SD	Average of 5 runs	SD	Average of 5 runs	SD
2	38.05	0.87	35.80	2.70	40.38	0.87	40.95	0.86
3	40.45	1.32	43.83	0.86	41.82	2.61	41.10	0.66
4	46.52	2.39	44.78	1.52	42.60	2.26	41.98	0.19
5	51.13	1.75	53.53	2.30	44.88	0.85	44.48	0.26
6	53.65	2.70	53.67	2.87	46.38	1.21	44.97	0.72
7	60.12	0.48	62.92	1.04	51.65	1.27	45.65	0.32

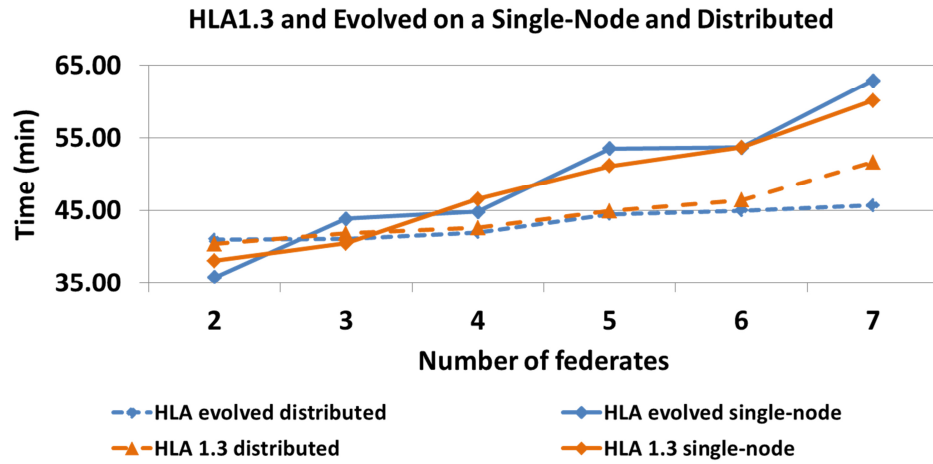


Figure 39: Comparison of execution time on a single node and a distributed environment with different number of federates

As can be seen from Figure 39, the HLA 1.3 version in a single-node environment (solid lines) behaves in a more linear manner. The increase in time is steadier with the increase of federates. On the other hand, HLA 1516e has an upward “zigzag” increase in time with the increase of federates. It is interesting to note that the execution time for even number of federates is comparatively lower than the execution time taken for odd number of participating federates. This behaviour can be caused from the utilisation of 2 core processors.

From Figure 40, the gradual increase in the difference of execution time over a single-node run environment and a distributed run environment is clearly seen. Noticeably, as the number of federates increases the execution time difference between the two run environments increases considerably. Moreover, when there is an odd number of participating federates, HLA-1516e speedup is greater.

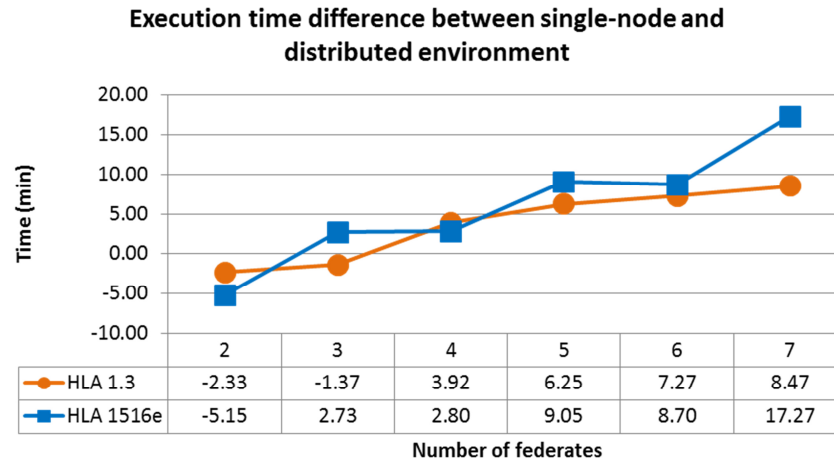


Figure 40: Time increase difference

### 5.3.2.2 Scalability testing

To test the scalability of the distributed system, three different scenarios are performed. This time the RTI implementation is based on the HLA 1516e version and the runs are executed on a dedicated network.

The first scenario involves an increase of both the emergency calls in the ambulance service federate and the walk-in patient arrivals in all the A&E federates. For the second scenario, there was an increase in emergency calls only. Finally, in the third scenario, the increase takes place only in the walk-in patient arrivals at all A&E department federates. The gradual increase involves 10 per cent, 20 per cent and 30 per cent rise of throughput.

In Table 11, the average execution times of five runs for each scenario are presented. As it is expected, the execution time increases steadily as the throughput of the federates increases. Also, from the standard deviation (SD) values, it can be seen that there is little var-

Table 11: Scalability results

Networked environment, HLA version 1516e, 7 Federates						
	Execution Time (min)					
	Emergency calls and walk-in patients		Emergency calls		Walk-in patients	
Workload increase	Average of 5 runs	SD	Average of 5 runs	SD	Average of 5 runs	SD
0%	45.65					
10%	50.21	1.01	48.08	0.74	49.10	0.48
20%	59.13	1.88	49.93	0.59	52.74	0.52
30%	62.99	1.04	51.33	1.07	55.49	0.25

iation among the five runs. Therefore, it can be concluded that the distributed system with seven federates is reasonable stable.

From the graphic representation of the produced results, shown in Figure 41, it can be observed that when there is dual increase, namely the emergency calls and the walk-in patient arrivals increase, it causes the highest rise in the DS execution time, a fact that was expected, as the number of interactions increases in both ways. Interestingly, the increase in the number of walk-in patient arrivals causes higher increase in execution time than the increase of emergency calls, despite the fact that the object that passes from one federate to another is the emergency incident patient. This can be explained by the design of the DS.

### Effects of increase in load using HLA Evolved with seven Federates

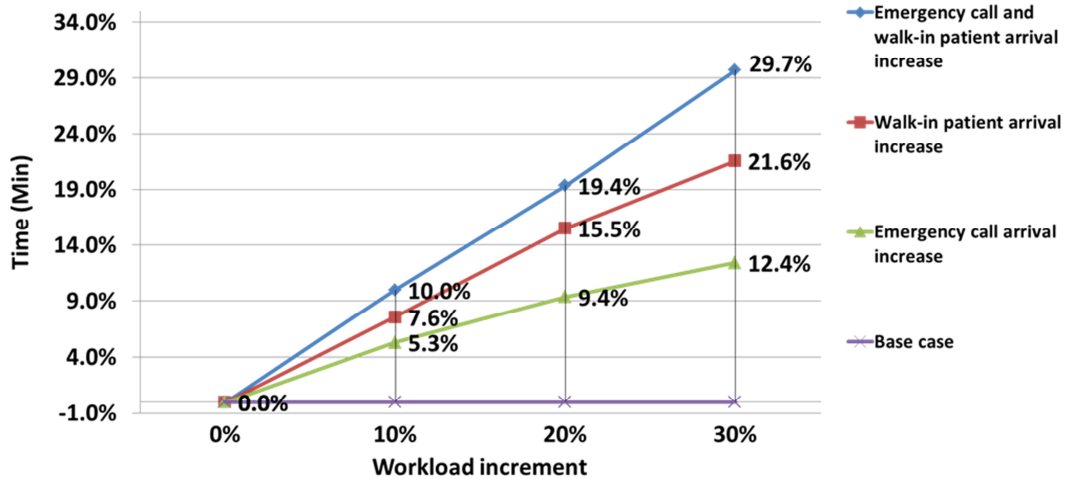


Figure 41: Scalability results graph

As stated earlier, when the availability of an A&E department changes, this is always communicated to the ambulance service model, and therefore, a busier hospital federate will send more interactions through the RTI to the ambulance federate.

### 5.3.3 Conclusions

The results of the two sets of experiments that conducted with the EMS prototype hybrid DS model show that the presented technology has the potential to perform analysis of EMS while executing efficiently on a networked platform.

In addition, the proposed methodology for building a DS project was followed faithfully and the subsequent steps provided an invaluable guide for completing the project.

## 5.4 Summary

Chapter 5 presented the realisation phase of the EMS hybrid ABS DES DS model. There was a walk-through the details of each realisation stage, namely, the ABS ambulance service simulation, the DES A&E simulation, and the middleware implementation.

For each model, an analysis of the system, the coding process and the data collection strategy was provided. Furthermore, the verification and validation of the simulation models were stated.

The product prototype model was used to test the experimentation phase of a DS system. The results indicate that the proposed technique can be utilised for modelling large-scale systems and enhance the simulation performance by distributing the required computational power over a network.

The next Chapter will demonstrate further performance testing of a larger scale case study and evaluate the feasibility of the proposed technology for large-scale complex systems analysis.

---

## **Evaluating the proposed DS methodology and the FIELDS framework**

*This chapter presents experimental results of a large-scale EMS DS model based on the London EMS. The hybrid distributed model constitutes an EMS half the size of the actual London EMS. The hypothesis that it is feasible to build a DS model of large-scale EMS that consists of independent individual models is tested further. Finally, the proposed DS methodology and the FIELDS framework are evaluated.*

## 6.1 Overview

In Chapter 3, the proposed distributed simulation methodology for conducting large-scale hybrid DS modelling studies was presented. In Chapter 4, the FIELDS conceptual framework for hybrid ABS-DES DS EMS was analysed. Based on the aforementioned DS methodology and the FIELDS framework, the development of a prototype hybrid ABS-DES EMS DS model was presented in Chapter 5. Furthermore, in Chapter 5, there was an analysis of the validation process of the prototype model. Lastly, the experimental results on performance and scalability testing were presented.

In this Chapter, there is further experimentation on a larger scale EMS. The purpose of the further experimentation is to support the testing of the hypothesis. In Chapter 5, the prototype model indicates that the hypothesis can be accepted. In this Chapter, the hypothesis will be tested in larger-scale models against the feasibility criteria of performance and scalability of the proposed technology. The London EMS is the system on which the framework will be tested. However, due to time and resources constraints, experimentation with the full-scale London EMS was not possible. Therefore, the experiments are performed in a DS hybrid EMS model half the size of the London EMS. Nevertheless, evidence shows, from the results of the half London EMS, that the implementation of the full-scale model is feasible. The presented results evaluate the performance and scalability of the distributed system.

Finally, the proposed distributed simulation methodology and the FIELDS framework are revisited and the issues that were raised during the process of developing and experimenting with the DS models are discussed. The proposed distributed simulation methodology and the FIELDS framework are evaluated and refined.



## **6.2 Distributed system performance and scalability testing**

To test the hypothesis, a DS model of an EMS of approximately half the size of the London EMS is used for experimentation. The configurations of the model, based on the London EMS context that was set in Chapter 5, are: area of coverage 320 mi<sup>2</sup>, 35 ambulance stations, 187 ambulance vehicles, 16 A&E departments, and the average speed across Greater London is 15mph. Similarly, the data collection period is the financial year 2011-12 (April 2011 to March 2012) and the data is published online by the DoH. The HLA standard version that is used for this model is the latest version HLA-1516-2010.

The experimentation of the distributed system is divided in two parts. The first part involves experimentation on the DS model performance and the second part involves experimentation on the DS model scalability testing.

### **6.2.1 Network settings**

The first part of the experimentation on performance testing involves execution of the distributed system over a single node and over a network. The single-node CPU is a 2-core processor at 3.17GHz speed and 4.00GB RAM. The distributed system ran over a homogeneous network connected with 1Gbps network card. The network consists of seventeen computers on a non-dedicated network interconnected via LAN. Each node contains an i5-2500 processor at 3.30GHz speed and 4.00GB RAM. Each PC has Microsoft Windows 7 with Java 1.7 JRE installed, and the poRTico package version HLA-1516-2010.

### **6.2.2 Experimental results**

In this section the results on performance and scalability of a large-scale DS model are presented and discussed.

### 6.2.2.1 Performance testing

Two scenarios are executed in line with performance testing of the distributed EMS simulation model. The first scenario involves the increase of the number of federates that are connected to the federation and the second scenario involves the increase of the simulation time. The test includes runs of one, two, three and four weeks of simulation time. Both scenarios are executed on a single-node and on a distributed environment.

The results shown in Table 12 are the average and the standard deviation of five runs. The shadowed grey areas indicate that it was not possible to run the experiment on a single node for the indicated load, a fact that supports the point that as the models grow in size and complexity, it is extremely difficult, or even impossible, to run as a standalone simulation on a single node. From the results, it is also inferred that the DS has insignificant variation in execution time as the number of federates increases.

The graphical representation of the performance results is depicted in Figure 42. The graphs illustrate the execution time for a) one week b) two weeks, c) three weeks and d) four weeks simulation time runs, respectively. The X-axis shows the number of federates and the Y-axis shows the execution time in minutes. As can be seen, for a short simulation time run, i.e., one week, the execution time variation between the single-node and the distributed environments is not very prominent, especially for small number of federates. However, as the simulation duration increases the execution time increases in both the single-node and the distributed simulation. This can be explained by the fact that as the simulation time extends the number of events that are scheduled to occur in future times are increased and therefore the computations of the models are becoming more complex. Interestingly, but as expected, the increase of the participating federates affects the execution time in the single-node environment but rather leaves unaffected the execution time in the distributed environment.

Table 12: Performance results

Simulation time	1 week				2 weeks				3 weeks				4 weeks			
Execution time	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD	Mean (min)	SD
	Single-node		Distributed		Single-node		Distributed		Single-node		Distributed		Single-node		Distributed	
3 federates	7.79	0.32	5.31	0.34	25.98	0.50	16.46	1.19	59.41	2.48	34.53	0.58	104.12	5.04	58.09	2.62
5 federates	8.73	0.15	5.39	0.24	30.36	0.79	16.11	1.00	62.54	0.83	34.10	0.80	111.79	0.32	57.22	2.28
7 federates	10.03	0.34	5.46	0.23	32.39	0.46	16.09	0.08	69.93	1.83	34.23	0.90	122.04	2.63	54.94	0.19
9 federates	11.33	0.16	5.31	0.07	36.07	0.70	15.99	0.41	76.16	4.29	32.72	1.80	133.91	0.69	56.19	1.09
11 federates	12.85	0.24	5.40	0.10	39.21	0.86	16.26	0.93	83.02	1.36	35.20	1.78	146.96	3.63	57.29	1.70
13 federates	14.97	0.98	5.58	0.12	43.81	0.52	16.38	0.40	91.88	1.00	33.41	1.67			56.36	1.54
15 federates	15.77	0.33	5.64	0.21	49.24	1.61	17.29	0.72			32.99	1.00			60.24	1.73
17 federates	19.06	0.37	5.50	0.14			16.77	0.22			34.46	1.12			58.25	2.76

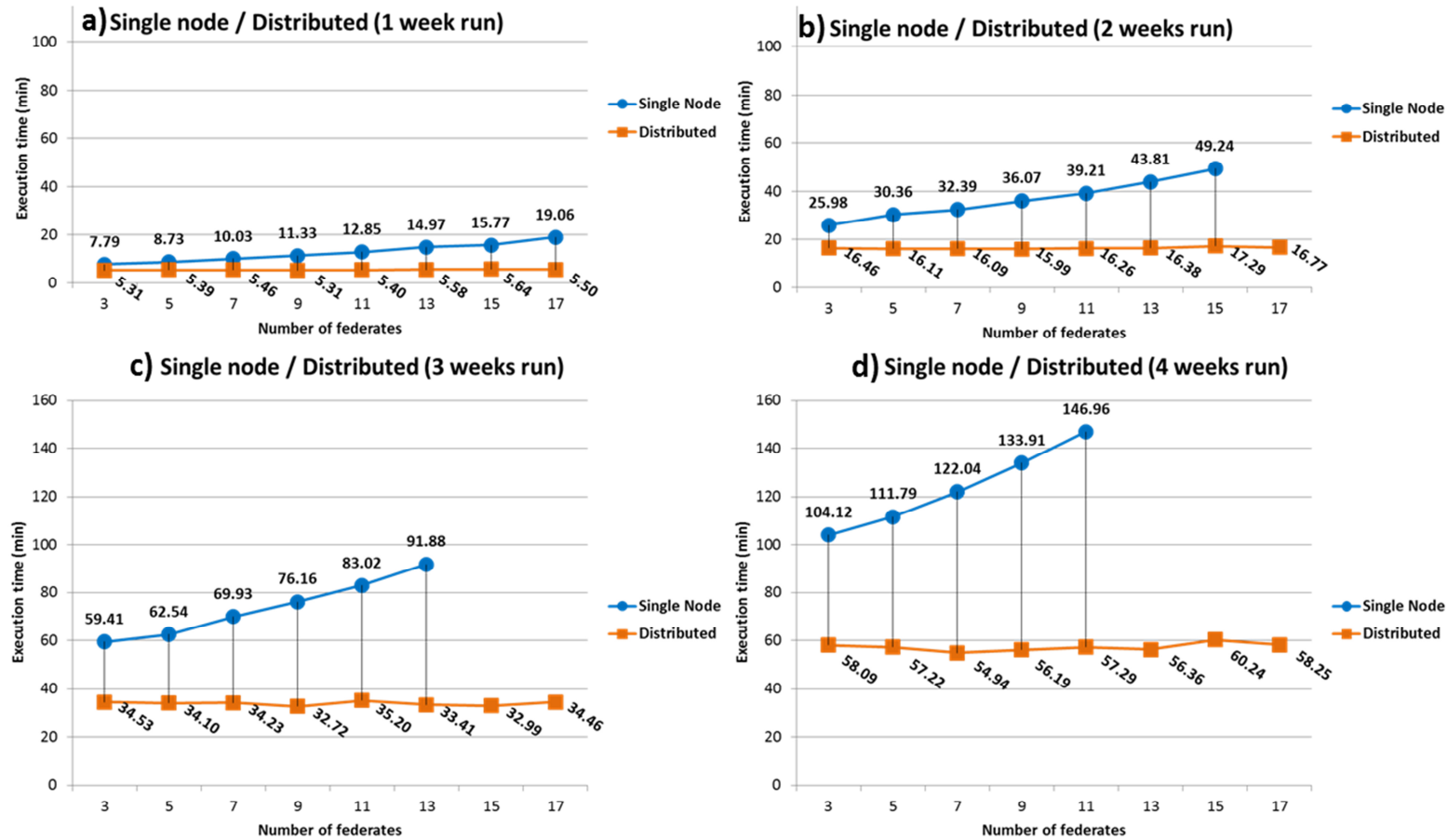


Figure 42: Execution time on a single-node versus distributed environment for a) 1 week, b) 2 weeks, c) 3 weeks and d) 4 weeks simulation time

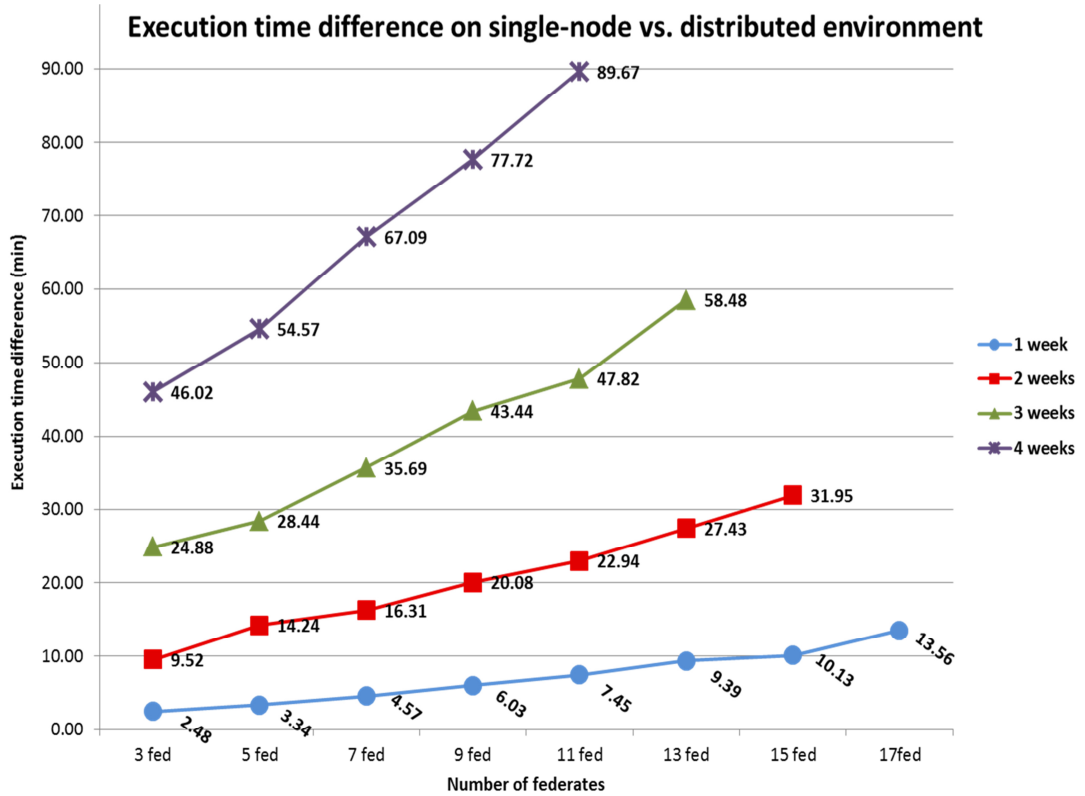


Figure 43: Execution time difference on single-node versus distributed environment

Consequently, the difference in execution time between the single-node and the distributed environment follows a more steep curve increase as the simulation run time and the number of federates increase, as can be seen in the graph of Figure 43, where the X-axis shows the number of federates and the Y-axis shows the execution time difference in minutes. For example, for four weeks simulation time elapse and with 11 federates participating in the model, the simulation takes approximately 89 minutes longer to run on a single-node than on a distributed environment. This number represents a 156.52% increase in execution time.

The speedup can be calculated by the formula.

$$\text{Speedup} = \frac{\text{Standalone execution time}}{\text{DS execution time}}$$

The speedup results for the different scenarios can be seen in Figure 44, where the X-axis shows the number of federates and the Y-axis shows the speedup.

From the graph, it is derived that as the model becomes more complex (increase of federates), DS speedups the simulation execution considerably.

### 6.2.2.2 Scalability testing

To test the scalability of the DS EMS model, two scenarios are simulated. The first scenario

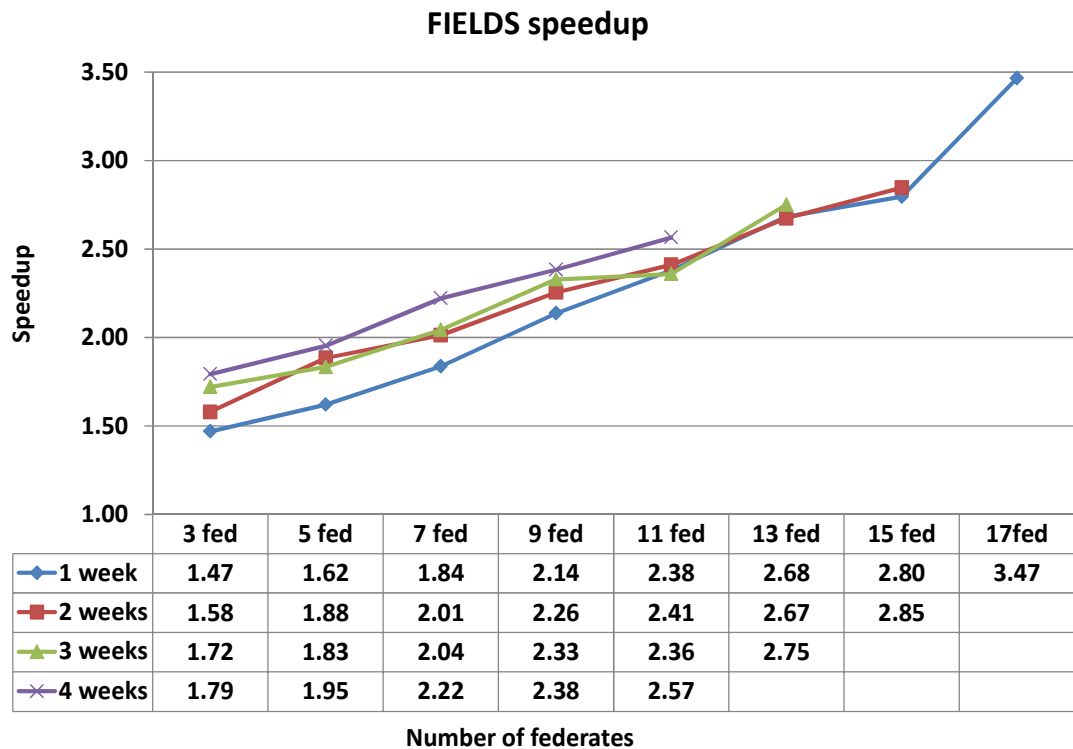


Figure 44: FIELDS speedup results

involves the increase of emergency calls arrivals in the ABS ambulance service model by 10 percent, 20 percent and 30 percent. The second scenario involves the increase by 10, 20 and 30 percent of the emergency calls arrivals in the ABS ambulance model and the walk-in patient arrivals in the DES accident and emergency models, respectively. The simulation ran for four weeks simulation time and the results of the average of five runs can be seen in Table 13.

The graph in Figure 45 shows the percentage of the execution time increase (Y-axis) while the workload increases by 10 percent, 20 percent and 30 percent (X-axis) in relation to the base case where there is the normal workload on emergency call arrivals for the ambulance

Table 13: Scalability results

	Execution time (min)					
	Base case: 58.25 min					
Workload increase	10%		20%		30%	
	Mean	SD	Mean	SD	Mean	SD
Emergency calls arrival	66.62	2.83	69.48	2.78	73.02	5.43
Execution time increase	14.37%		19.28%		26.77%	
Emergency calls and walk-in patients arrival	62.07	0.65	67.76	1.10	73.84	2.32
Execution time increase	6.55%		19.28%		25.35%	

model and walk-in patients arrivals for the A&E department models, respectively. The results show that with a cost in execution time the distributed model can respond adequately

to an increase in workload. This, for example, indicates confidence in using the FIELDS framework for testing scenarios on crisis management and major incidents where the demand on an EMS system increases significantly.

### 6.3 Revisiting the proposed DS methodology and FIELDS framework

In this section, an evaluation of the proposed distributed simulation methodology and FIELDS framework for hybrid ABS-DES DS for EMS modelling is provided.

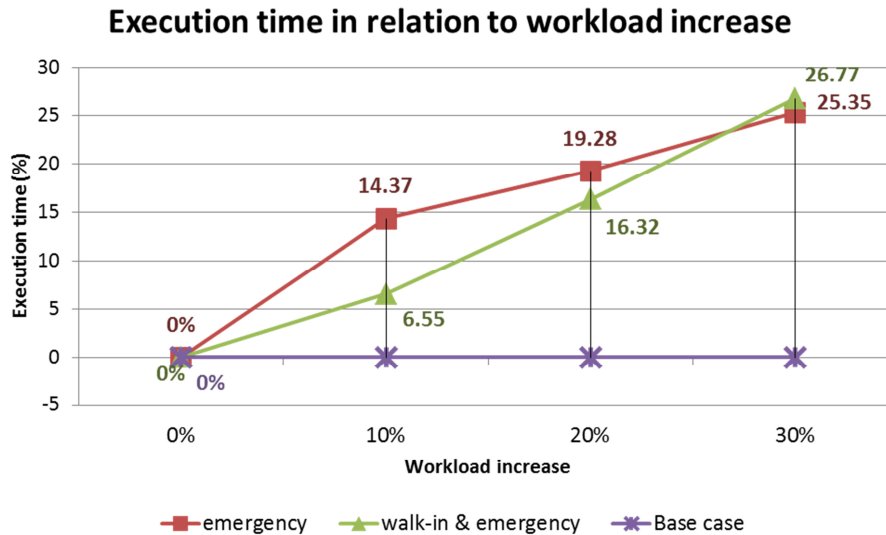


Figure 45: Execution time in relation to workload increase

#### 6.3.1 Distributed simulation methodology revisited

In Chapter 3, the proposed distributed simulation methodology was introduced and the underlying rationale for its development process was analysed. The resulting three phases of the distributed simulation methodology were discussed. In this section, there will be an



evaluation of this methodology, after conducting the case study based on it, as regards to the three phases.

#### **6.3.1.1 Planning phase revisited**

The first phase involves the problem definition and the planning of the DS project. The real-world problem that is under study is framed at this stage and the questions that are to be answered from the simulation model are formed and decided.

This phase usually involves background research and stakeholders involvement. The first phase in a DS project is identical with any standalone simulation project.

#### **6.3.1.2 Development phase revisited**

This second phase involves the development of the DS, from the conceptual design to the actual programming of the computer simulation. In this phase, there is the conceptualisation, data collection and realisation of the DS. Each individual model, or federate, is developed independently, or modified if it already exists. Each model verified and validated as a standalone model first and, therefore, can be executed autonomously. Furthermore, the synchronisation protocol is decided and implemented and, subsequently, the DS verified and validated.

However, after conducting the case study, this phase of the methodology was refined. The distributed system can be seen as a whole that consists of independent, complete, parts. Unlike a standalone model, that can consist of modular parts, but these parts cannot be seen as independent models and complete simulations, the federates of a distributed system are simulations that can be standalone models themselves. Therefore, the data collection process in the methodology was revised.

---

It was found out that there were required two sets of data, one for each individual model and one for the whole system. The individual model data was analysed in the simulation methodologies that were reviewed in Section 2.3. This is an iterative process that starts at the conceptualisation phase of a model development and continues until the simulation is verified. However, in building DS models, there is another set of data that is required for the development of the whole system simulation. This data includes the context of the whole system, the interactions and the time relationships between the interacting models, the simulation packages that the individual models are built, variable names that the interoperating models hold, and time advance mechanisms within the interoperating models. The DS data collection process is again a parallel activity that begins at the DS conceptualisation step and can be revisited until the DS model is verified.

### **6.3.1.3 Experimentation phase revisited**

The third phase involves the experimentation with the DS model. As mentioned in Chapter 3, in this phase the computer network that the distributed system will execute should be decided.

### **6.3.1.4 Refined distributed simulation methodology**

The refined distributed simulation methodology that is proposed in this thesis is shown in Figure 46. As identified in the previous sections, the data collection process is refined further after the application of the methodology in the EMS case study.

As mentioned earlier, the data collection process can be split in in two distinct levels. First, there is the data collection step for each of the individual participating simulation models and second, there is the data collection step for the whole system of the DS model. For example, in the EMS distributed model, there was a process of data collection for the ambulance service and the A&E departments individual models. In addition there was a

---

data collection process for the whole EMS distributed model. In the case of the prototype model development, the simulated EMS was a 1:5 scaled-down city based on the London EMS. However, when modifying the distributed EMS model to represent a city half the size of the actual London EMS the data collection process for the system was revisited.

Therefore, there should be a distinction between the individual models internal data and the holistic DS data. In Figure 46, there was an illustration of the refined distributed simulation methodology, where the green highlighted boxes indicate the two distinct data collection processes. Both processes are iterative procedures that occur in parallel with the model building activities, starting from the model conceptualisation step and stretching up to the model verification step. It should be noted that the validation of the models is happening after all data is collected.

Furthermore, it is important to repeat that the documentation process is taking place throughout the distributed project development. By starting documentation as early as in the planning phase, the possibility of any potential omissions from the final report is nearly eliminated.

### **6.3.2 FIELDS framework**

In Chapter 4, the FIELDS conceptual framework for hybrid ABS-DES DS of EMS was developed.

The framework views an EMS system as a complete hospital city that there is an integrated service of ambulance, and A&E departments within hospitals. To the best of the author's knowledge, the existing EMS simulation model the ambulance service and just acknowledge the A&E congestion. However, by incorporating the hospital simulations in the model and connecting all the components by DS techniques can provide a holistic framework for EMS analysis.

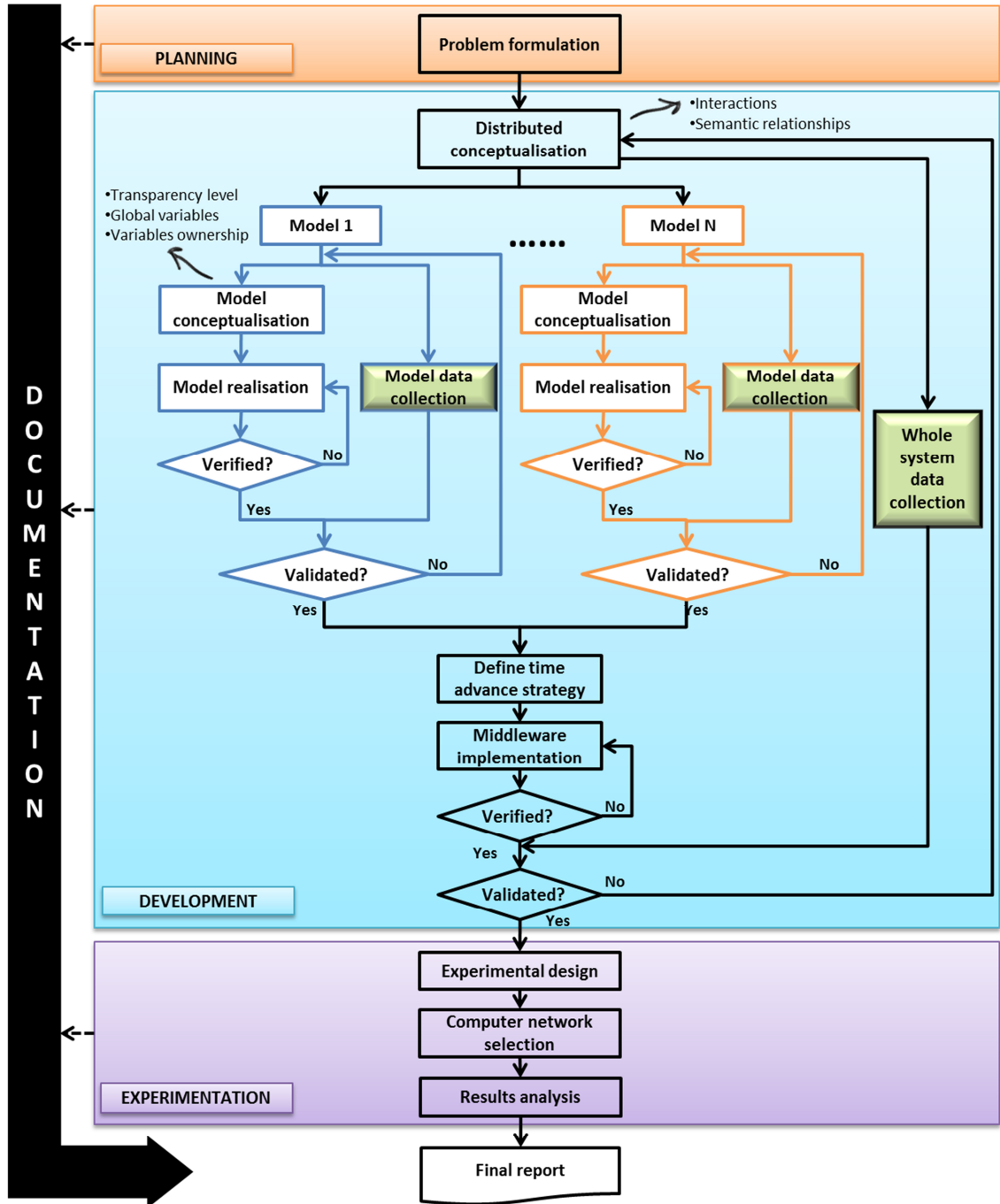


Figure 46: Refined distributed simulation methodology

The proposed FIELDS framework consists of an ABS model of the ambulance service

and as many DES models of A&E departments as there exist is the area of the ambulance service coverage. The interaction points between the participating simulations are defined in the IRM.

The FIELDS framework supports the component model re-use. By defining the boundaries of each participating model and the interactions between the participating models, they can be plugged in the distributed system on demand. For example, if a new A&E department is built within the area of a particular ambulance service, the A&E model can be add-

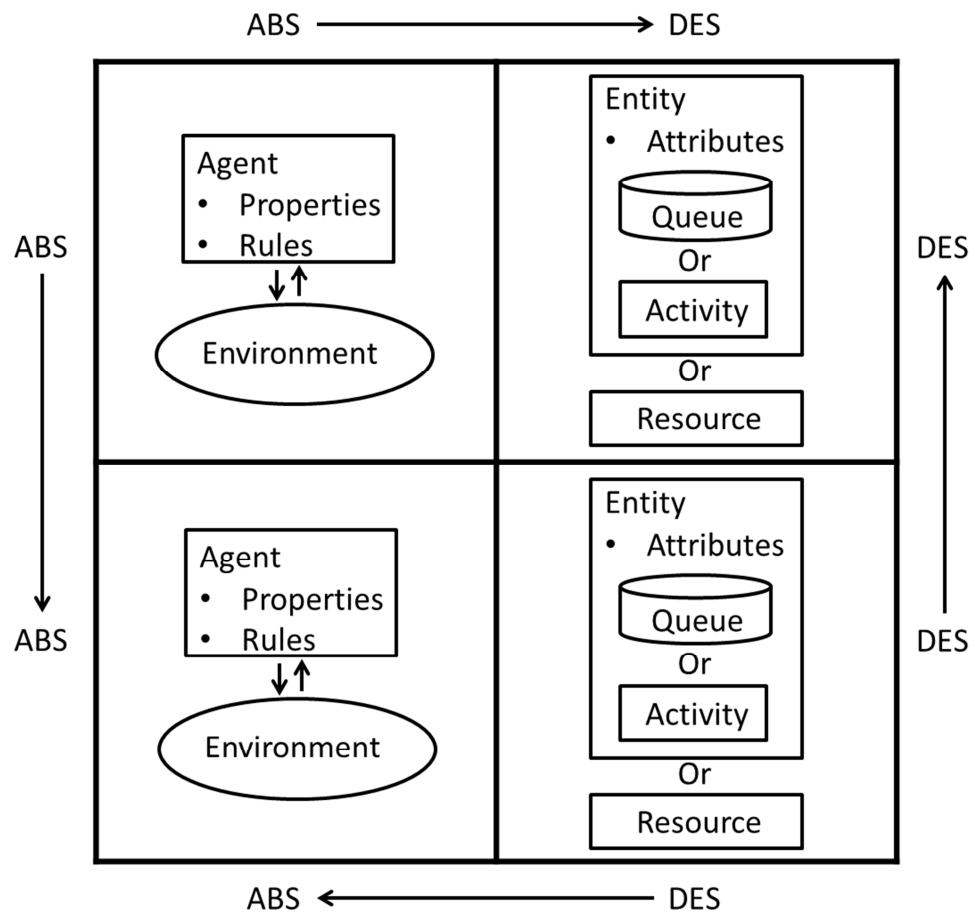


Figure 47: Simulation techniques combinations

---

ed in the DS. In the same time, the individual models are kept locally in the organisations and connected through a network only when they participate in the whole EMS system.

In the FIELDS framework the models are designed in a high level of abstraction, a fact that constitutes the system insufficient for studying the operations of the individual organisations, i.e., ambulance service and A&E departments. However, regardless the internal complexity and the details level of the participating models the interaction points are not changing. Therefore, the framework can be used for connecting any ambulance service model with any A&E department model, even if the internal complexity beyond the boundaries of the models is changed.

Furthermore, the framework can be applied even if the combination of ABS and DES simulation techniques that is used to model the subsystems alternates. For example, if the ambulance service simulation is modelled using DES, there will be interaction between DES models. In this way, there can be interaction between any combination of ABS and DES models. The possible combinations are: ABS ambulance model and DES A&E models, DES ambulance model and DES A&E models, ABS ambulance model and ABS A&E models, DES ambulance model and ABS A&E models, ABS ambulance model and a combination of ABS and DES A&E models, or DES ambulance model and a combination of ABS and DES A&E models.

Figure 47 portrays the possible combinations of ABS and DES modelling techniques for an ABS-DES hybrid DS. When a simulation object passes from an ABS model to a DES model, the agent with its properties will become an entity with the respective attributes, or a resource. The entity can either be added to a queue (or storage) or to an activity.

From an ABS model to an ABS model, the agent will pass with its properties to the receiving ABS model and become a new agent that adopts the rules that are meaningful in the

---

receiving ABS model. Another combination is the sending simulation to be modelled in DES and the receiving simulation to be modelled in ABS. In this case, an entity with its attributes, or a resource, becomes an agent that adopts the rules that are meaningful in the ABS receiving model. Finally, the sending and receiving simulations can be both modelled in DES. In this case, an entity with its attributes, or a resource, from the sending model becomes a new entity, to be added in a queue or an activity, or a resource in the receiving model.

## 6.4 Summary

In this Chapter, the results of a large-scale hybrid DS of an EMS model based on the London EMS were discussed. The EMS DS that was tested was half the size of the London EMS system. The experimental design involved two parts. The first part was focused on performance testing and was based on two scenarios. First, there was an increase of the number of federates that participated in the federation. Second, there was an increase of the simulation run time. The test included runs of one, two, three and four weeks of simulation time. Both scenarios are executed on a single-node and on a distributed environment. The results showed that the distributed environment can support adequately large-scale models simulation runs. Importantly, when the simulation increases in complexity and size a single node cannot afford the computation demands. The second part aimed on scalability testing. Two scenarios were simulated. First, there was an increase of emergency calls arrivals in the ABS ambulance service model by 10 percent, 20 percent and 30 percent. Second, there was an increase by 10, 20 and 30 percent of the emergency calls arrivals in the ABS ambulance model and the walk-in patient arrivals in the DES A&E models, respectively. The results showed that at a cost in execution time the distributed model can respond adequately to an increase in workload.

Furthermore, the proposed methodology for DS development and the FIELDS framework for distributed EMS modelling conceptualisation were discussed and refined. After revisiting the distributed simulation methodology, there was a modification in the data collection process. Moreover, there was a discussion about the FIELDS framework and the possibilities of recombining the ABS and DES simulation techniques.

In the next Chapter, a summary of this research will be provided. The way that the aim, that was presented in the introductory Chapter 1, was addressed will be discussed. Reflections on the objectives identified in Chapter 1 and how they assist in concluding this project will be noted. Importantly, the contributions of this thesis will be analysed. Finally, the limitations and future research work of the thesis will be discussed.



# CHAPTER

# 7

---

## Conclusions and future work

*This chapter presents a summary of the thesis. There is an explanation of how the aim, stated in the first Chapter, was achieved and how the identified objectives in Chapter one assisted in reaching the aim. The contributions of this research are highlighted. Finally, the limitations and future work are discussed.*

## 7.1 Overview

In the previous Chapter, the results of a large-scale hybrid distributed EMS model were presented. The experimental design included performance and scalability testing of the DS. Further, there was an evaluation and refinement of the proposed distributed simulation methodology, where the process of data collection was amended. The proposed FIELDS conceptual framework for hybrid ABS-DES DS modelling of EMS systems was discussed and the possible combinations of ABS and DES were analysed.

This Chapter provides a summary of the thesis. Thereafter, the aim of this research is discussed and the way that the objectives, stated in Chapter 1, were met in order to achieve the aim. Subsequently, the contributions of the thesis are highlighted and discussed. Finally, the limitations of this work are discussed together with the further work.

## 7.2 Research summary

This thesis was motivated by the fact that in the healthcare sector M&S has not had the scale of adoption seen in other sectors, although it has been a valuable tool for systems' analysis in many industries, and especially military applications.

In line with the above, the hypothesis that was tested was the feasibility of a hybrid ABS-DES DS model of EMS that enables holistic analysis of the system and supports model re-use.

The research methodology that was adopted to conduct this study and test the hypothesis was a combination of empirical research and design science research. One of the methods that empiricism employs is the case study design which is the method that was conducted here. From the range of case study methods available, a feasibility case study was de-

ployed, in agreement with the hypothesis to be tested and the research questions to be answered.

A hybrid ABS and DES DS model of EMS was developed that enables model reusability. The EMS system was viewed as a holistic system that incorporates the ambulance services and the A&E departments in the area of coverage. The model consists of an ABS ambulance service model and DES models of the A&E departments in the coverage range. The individual models are linked by DS technologies, more specifically, the HLA standard. The participating models can run as part of the distributed system as different nodes in a network or as standalone models of the single organisations. In this way the models of subsystems of a complex system can be reused and form a larger DS model and, at the same time, they stay locally in the individual organisation. Therefore, data protection and privacy issues, as well as data updates issues are addressed. Moreover, the computation load of execution of the large-scale simulation model is distributed over a computer network.

Computer simulation is a dynamic modelling approach that analyses how a system changes as time progresses. In this thesis, ABS and DES were deployed for modelling the different parts of an EMS system. ABS views the system at an individual level, consists of independent agents that interact with each other and their environment, and is decentralised in the decision making process. DES is a process-oriented technique that does not allow liberties to the entities, but rather the entities are driven by the processes of the system. The different simulation techniques and the different simulation software tools have an underlying philosophy for the simulation software implementations; these are known as simulation world views.

DS is the execution of a number of simulation models, linked with a middleware, that run on different nodes of a computer network. The middleware is responsible for coordinating data and synchronising time of the distributed system. Importantly, before implement-

ing programmatically the distributed system, the boundaries of each model and the connections and interactions between the simulations should be defined. The IRM developed in an attempt to standardise the way that the interactions between the simulations in a DS system are represented.

Another aspect of this project is the simulation models re-use. Model re-use can be classified in three levels: simulation software, simulation model and conceptual model support levels, each of which have a different degree of complexity and re-use frequency.

Based on previously published simulation methodologies for standalone projects, the proposed distributed simulation methodology was developed. The rationale of its development is grounded on the LCIM model and the layered view of simulation model interoperability. The proposed methodology incorporates the semantic relationships between different simulation paradigms. Furthermore, it includes the IRM development between the interacting simulations and the time management strategies of the DS system.

Moreover, the FIELDS framework for integrated EMS large-scale DS was developed as a conceptual framework for hybrid ABS-DES DS of EMS. FIELDS addresses the issues of ambulance services and A&E departments simulations in a very high level of abstraction. However, it is interested in the interactions between the components of an EMS system rather than the individual models themselves. The conceptual framework was built following the steps of the proposed distributed simulation methodology. The FIELDS framework provides the IRM model of a hybrid ABS-DES model of EMS.

A prototype model of a large-scale distributed EMS simulation was implemented. The design of the prototype model was based on the London EMS and was a 1:5 scaled-down model of the actual system. The model was populated with data published online by the UK's DoH. The prototype distributed model was validated against published performance

measures of the LAS and A&E departments. Experiments on performance and scalability analysis indicated the feasibility of the proposed framework.

Further experiments were run on a larger scale EMS DS case study, which constitutes half the size of the London EMS, in order to evaluate the proposed approach and support the hypothesis testing. Several scenarios were run and the produced results showed that it is feasible to use DS technology to model large-scale EMS systems that support individual model reusability. Furthermore, the proposed distributed simulation methodology was modified after evaluating it by the implementation of the system. The FIEDLS conceptual framework for hybrid ABS-DES DS of EMS was evaluated, too, and all the combinations of ABS and DES models were analysed.

### **7.3 Research aims and objectives**

The aim of this research was to explore new approaches to EMS simulation modelling by using DS techniques to support model reusability and to develop a novel simulation methodology for developing hybrid DS models. In doing so, the following research questions were addressed:

- Which simulation techniques are more appropriate for EMS modelling?
- Is DS better than standalone simulation for holistic EMS modelling?
- Is it feasible to support model re-use with DS?

To achieve the aim of this thesis and to address the research questions, six objectives were identified. These objectives were met through the Chapters of this thesis as follows:

**Objective 1:** *To formulate the research hypothesis and establish the aim of this thesis. Furthermore, to identify the appropriate research methodology that will be the vehicle to reach the aim and test the hypothesis.*

In Chapter 1, there was a statement of the underlying research context and rationale that established the research aim and hypothesis. Also, in the same Chapter, there was a justification of the adopted research methodology that guided the hypothesis testing.

**Objective 2:** *To review the normative literature and investigate issues in the EMS modelling field that can be studied efficiently by applying M&S.*

Chapter 2 provided the theoretical background and the literature on all the relevant aspects of the proposed framework. Also, Chapters 2 and 3 provided a review of the literature of EMS M&S. In both Chapters 2 and 3, the justification of the selected simulation techniques for EMS was established.

**Objective 3:** *To develop a distributed simulation methodology for constructing hybrid ABS and DES distributed simulation models.*

In Chapter 3, the process of developing the proposed DS methodology for building a hybrid distributed system was analysed. The methodology supports the re-use of existing, and building anew, component models.

**Objective 4:** *To develop a framework for developing hybrid ABS-DES DS of EMS systems based on the distributed simulation methodology as stated in objective three.*

The development of the proposed FIELDS conceptual framework for developing hybrid ABS-DES DS models of EMS systems and the associated IRM were discussed in Chapter 4.

**Objective 5:** *Based on the framework, to develop a prototype model of the London EMS and test the feasibility of the framework developed as specified in objective four.*

The development of the prototype ABS-DES DS model was presented in Chapter 5. The prototype was a scaled-down hybrid DS model based on the London EMS. Experiments were run on validation of the prototype model and on performance and scalability of the system to test the feasibility of the framework.

**Objective 6:** *To evaluate the distributed simulation methodology, stated in objective three, and the framework, stated in objective four, by applying it to a case study. This will be achieved by developing a realistic large-scale EMS model and experiment with different scenarios.*

The proposed distributed simulation methodology and the FIELDS framework were evaluated by further experiments with a larger scale hybrid distributed EMS model. The distributed simulation methodology was refined and the FIELDS framework was analysed further.

## 7.4 Research contributions

This thesis generated the following significant contributions, firstly to the M&S methodological field and, secondly to the EMS analytical tools area.

- ***Distributed simulation methodology***

The first contribution to the M&S methodology was the distributed simulation methodology that can be used to develop hybrid DS projects. This methodology provides a general guide, with well-defined steps, for building hybrid distributed simulation models of any system. Building a simulation model is a heavily laborious endeavour even for experienced

simulation experts (researchers and practitioners) and it is far worse for the newly introduced in the area. Especially, when the system to be modelled presents high complexity and is large in size, the modeller's effort is impressively massive. Another degree of difficulty is added when the system requires different simulation techniques in order to be realistically represented. Thus, having a well-structured methodology as a guide can contribute greatly to the M&S community and to future research.

- ***FIELDS framework for ABS-DES DS for EMS modelling***

The second contribution lies in the area of DS for EMS modelling. The proposed framework is a detailed and area-specific concept for building ABS-DES distributed simulation models of EMS. The FIELDS framework provides a conceptual framework for hybrid ABS-DES DS models of EMS. FIELDS frames the interactions between the subsystems of an EMS. Furthermore, the IRM development defines the simulation objects that are communicated between the interoperating component simulations.

- ***Prototype EMS analysis tool***

The third contribution of this thesis lies in the area of the EMS tools for systems analysis and is a prototype EMS model. This model, with some details enhancement, can be used to analyse holistically the London EMS. Many scenarios can be tested for improvements in the system. Also, it can be modified and adapted for other emergency services.

## **7.5 Research limitations and future work**

In this section the research limitations and future work are discussed.



### 7.5.1 Research limitations

One limitation of the proposed framework is that it was applied in a single case study, albeit in systems of a different scale. To fully test the feasibility of a large-scale hybrid DS technique in analysing EMS, it would be desirable to apply the framework in more than one service, i.e., implement and experiment with EMS hybrid DS models of other regions. Moreover, the FIELDS framework was not tested in other combinations of simulation techniques. It would have given a holistic view of hybrid ABS-DES DS for EMS, if all the models were implemented in ABS, or DES, or different ABS-DES combinations.

Another limitation can be considered the high level of abstraction of the individual models. This level of abstraction does not allow thorough analysis of the operations of the participating organisations, namely the ambulance service and the A&E departments, and therefore, improvement scenarios could not be studied. However, the approach taken in the thesis focuses on the framework development rather than the comprehensive modelling means that improvement scenarios can be studied in the future.

In the proposed framework, only one aspect of entity transfer is implemented. That is the general entity transfer which is described by IRM A.1. The author acknowledges the limitation of emulating the real world EMS operations by simply passing a patient object from the ambulance service to an A&E department. Patient handover and ambulance diversion are two big issues, and EMS analysts have put a lot of effort into finding efficient solutions. However, these issues cannot be modelled using the FIELDS framework in its current state. The implementation of IRM A.2, i.e., “bounded receiving element” that describes the behaviour of a distributed simulation when the receiving federate has a limited queue size, is needed in order to be able to capture the behaviour and the waiting time of the ambulance service process of handing over a patient to an A&E. Moreover, ambulance diversion is not currently included in the framework and, therefore, this behaviour cannot be analysed.

However, the A&E model's logic can be modified in the future in order to be able to divert ambulances.

Finally, all participating simulations in the distributed system, although they can run independently as standalone models and easily modified to participate in the DS, were developed with this in mind from the inception of the idea. The framework was not tested with already existing models that had been developed just for one organisation. This limits the confidence that model reusability can be fully supported. Ideally, the framework would have been used to support current EMS models. However, as these do not exist across an entire region, the proof of concept in this work demonstrates the feasibility of the approach.

### **7.5.2 Future work**

This thesis can be the springboard for a number of future projects in the different aspects of the research context.

In the context of M&S, the implementation of the ambulance service and A&E departments simulations in all possible combinations of ABS and DES can provide a thorough analysis of the suitability of the simulations paradigms for the EMS systems. Furthermore, a comparison of the different techniques and their underlying simulation world views will contribute largely to the M&S field. Another path for future research is to implement the simulation models in different simulation software, including commercial packages, and consequently test the proposed framework in a cross-platform environment.

In the context of interoperability reference standards for DS, the implementation of the remaining types of IRM that identified in SISO-STD-006-2010, namely Type A2 bounded receiving element, Type A3 multiple input prioritisation, and Type B general shared resource, can contribute to the expansion of the standards for hybrid DS using not only commercial simulation packages but open-source simulation software, too. Also, implementing

the models with different COTS can provide an analysis of the issues of cross-platform COTS interoperability.

In the context of model re-use, as mentioned in the limitations section, the application of the framework in already existing models of EMS would give a valuable analysis of the required modifications and effort needed in order to “plug” the models in the distributed simulation methodology.

In the context of EMS analysis, further research to extend the developed system and include more details in the models would provide a tool for studying improvement scenarios for the London EMS. Also, research on optimisation techniques and incorporation of GIS would improve the realistic imitation of the system. Moreover, the use of the framework in other regions would contribute to the generalisation of the FIELDS framework.

### **7.5.3 Reflections**

This research started as a very ambitious project with the vision to develop a comprehensive distributed simulation of a large-scale emergency medical service system, namely the London EMS. This would involve the development of more than 30 simulation models, using different M&S techniques, and the development of the distributed simulation middleware.

During the process, the author realised that it was not possible to master all the involved technologies, gather vast amounts of data from many organisations, develop and experiment with the distributed system during the course of a three year PhD study.

Therefore, this thesis provides the starting point of a very interesting and novel way of studying EMS systems. The proposed framework, when extended to capture all interface issues of ambulance services and emergency units, can be used to analyse interoperability

---

between the components of EMS and improve communication among the organisations that comprise the emergency services.

With the advances in network computing and the availability of resources, e.g., cloud computing, the author believes that distributed simulation is the way forward for studying large-scale systems.

## **7.6 Summary**

Chapter 7 presented the summary of the current thesis. Furthermore, there was a discussion on the aim and objectives of the current research, initially identified in Chapter 1. How the objectives were met in order to reach the thesis's aim was discussed. Importantly, the contributions of this research work were emphasised and discussed. Additionally, the research limitations were mentioned and, lastly, the future research possibilities that derived based on this thesis were discussed.

---

## References

- Abo-Hamad, W. and Arisha, A. (2013) "Simulation-Based Framework to Improve Patient Experience in an Emergency Department." *European Journal of Operational Research*, Vol. 224, No. 1, pp. 154-166.
- Aboueljinane, L., Sahin, E. and Jemai, Z. (2013) "A Review on Simulation Models Applied to Emergency Medical Service Operations." *Computers & Industrial Engineering*, Vol. 66, No. 4, pp. 734-750.
- Aboueljinane, L., Jemai, Z. and Sahin, E. (2012) "Reducing Ambulance Response Time Using Simulation: The Case of Val-De-Marne Department Emergency Medical Service." In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, pp. 943-954, Berlin, DE, December 09-12.
- Anagnostou, A., Nouman, A. and Taylor, S.J.E. (2013) "Distributed Hybrid Agent-Based Discrete Event Emergency Medical Services Simulation." In *Proceedings of the 45<sup>th</sup> Winter Simulation Conference (WSC14)*, Edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill and M. E. Kuhl, pp. 1625-1636, Washington, DC, December 08-11.
- Aringhieri, R. (2010) "An Integrated DE and AB Simulation Model for EMS Management." In *IEEE Workshop on Health Care Management (WHCM 2010)*, pp. 1-6, Venice, Italy, February 18-20.
- Ashour, O.M. and Okudan-Kremer, G.E. (2013) "A Simulation Analysis of the Impact of FAHP-MAUT Triage Algorithm on the Emergency Department Performance Measures." *Expert Systems with Applications*, Vol. 40, No. 1, pp. 177-187.

- Baker, T. (1999) "Time Management in Distributed Simulation Models." In *Proceedings of Fourth International Simulation Technology and Training Conference (SimTecT'99)*, March 1999.
- Balci, O., Arthur, J.D. and Nance, R.E. (2008) "Accomplishing Reuse with a Simulation Conceptual Model." In *Proceedings of the 40<sup>th</sup> Winter Simulation Conference*, Edited by S.J. Mason, R.R. Hill, L. Monch, O. Rose, T. Jefferson and J.W. Fowler, pp. 959-965, Miami, FL, December 07-10.
- Banks, J., Carson, J.S., Nelson, B.L. and Nicol, D.M. (2000) "Discrete-Event System Simulation." Prentice Hall Inc, USA, ISBN: 0-13-088702-1.
- Bell, D., Mustafee, N., de Cesare, S., Taylor, S.J.E., Lycett, M., and Fishwick, P.A. (2008) "Ontology Engineering for Simulation Component Reuse." *International Journal of Enterprise Information Systems (IJEIS)*, Vol. 4, No. 4, pp. 47-61.
- Bilandzic, M. and Venable, J. (2011) "Towards Participatory Action Design Research: Adapting Action Research and Design Science Research Methods for Urban Informatics." *The Journal of Community Informatics*, Vol. 7, No. 3.
- Brailsford, S. (2014) "Discrete-Event Simulation is Alive and Kicking!" *Journal of Simulation*, Vol. 8, No. 1, pp. 1-8.
- Brailsford, S., Viana, J., Rossiter, S., Channon, A. and Lotery, A.J. (2013) "Hybrid Simulation for Health and Social Care: The Way Forward, or More Trouble Than It's Worth?" In *Proceedings of the 45<sup>th</sup> Winter Simulation Conference*, Edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill and M. E. Kuhl, pp. 258-269, December 08-11.
- Brailsford, S., Bolt, T., Connell, C., Klein, J.H. and Patel, B. (2009a) "Stakeholder Engagement in Health Care Simulation." In *Proceedings of the 41<sup>st</sup> Winter Simulation*

- 
- Conference*, Edited by M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin and R.G. Ingalls, pp. 1840-1849, Austin, TX, December 13-16.
- Brailsford, S.C., Harper, P.R., Patel, B. and Pitt, M. (2009b) "An Analysis of the Academic Literature on Simulation and Modelling in Health Care." *Journal of Simulation*, Vol. 3, No. 3, pp.130-140.
- Carlsson, S.A. (2006) "Towards an Information Systems Design Research Framework: A Critical Realist Perspective." In *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (DESRIST '06)*, Claremont, CA, February 24-25.
- Chen, D., Theodoropoulos, G.K., Turner, S.J., Cai, W., Minson, R., and Zhang, Y. (2008) "Large Scale Agent-Based Simulation on the Grid." *Future Generation Computer Systems*, Vol. 24, No. 7, pp. 658-671.
- Davis, J.P., Eisenhardt, K.M. and Bingham, C.B. (2007) "Developing Theory through Simulation Methods." *Academy of Management Review*, Vol. 32, No. 2, pp. 480-499.
- Davis P.K. and Anderson R.H. (2003) "Improving the Composability of Department of Defense Models and Simulations." *RAND Corporation*: online (last accessed January 2014) <http://www.rand.org/pubs/monographs/MG101.html>
- Demeyer, S. (2011) "Research Methods in Computer Science." In *Proceedings of the 27<sup>th</sup> IEEE International Conference on Software Maintenance (ICSM)*, Williamsburg, VA, September 25-30: online (last accessed January 2014) [http://win.ua.ac.be/~sdemey/Tutorial\\_ResearchMethods](http://win.ua.ac.be/~sdemey/Tutorial_ResearchMethods)
- Derrick E.J., Balci O. and Nance, R.E. (1989) "A Comparison of Selected Conceptual Frameworks for Simulation Modelling." In *Proceedings of the 21<sup>st</sup> Winter Simulation*

- 
- Conference*, Edited by E.A. McNair, K.J. Musselman and P. Heidelberger, pp.711-718, Washington, DC, December 04-06.
- Eatock, J., Clarke, M., Picton, C. and Young, T. (2011) "Meeting the Four-Hour Deadline in an A&E Department." *Journal of Health Organization and Management*, Vol. 25, No. 6, pp. 606-624.
- Fonseca-i-Casas, P. (2008) "SDL, A Graphical Language Useful to Describe Social Simulation Models." In Proceedings of the 2<sup>nd</sup> Workshop on Social Simulation and Artificial Societies Analysis (SSASA'08), November 20-21: online (last accessed January 2014) [http://ceur-ws.org/Vol-442/p4\\_Fonseca.pdf](http://ceur-ws.org/Vol-442/p4_Fonseca.pdf)
- Fujimoto, R.M. (2000) "Parallel and distributed simulation systems." New York, NY: Wiley, ISBN: 0-471-18383-0.
- Fujimoto, R.M. (1998) "Time Management in the High Level Architecture." *Simulation*, Vol. 71, No. 6, pp. 388-400.
- Fujimoto, R.M. and Weatherly, R. (1996) "Time Management in the DoD High Level Architecture." In *Proceedings of the 10<sup>th</sup> Workshop on Parallel and Distributed Simulation*, IEEE CS Press, pp. 60-67, Los Alamitos, CA, May 22-24.
- Goldsman, D., Nance, R.E. and Wilson, J.R. (2010) "A Brief History of Simulation Revisited" In *Proceedings of the 42<sup>nd</sup> Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hukan and E. Yucesan, pp. 567-574, Baltimore, MD, December 05-08.
- Gruene-Yanoff, T. and Weirich, P. (2010) "The Philosophy and Epistemology of Simulation: A review." *Simulation & Gaming*, Vol. 41, No. 1, pp. 20-50.
- Hagtvedt, R., Griffin, P., Keskinocak, P., Ferguson, M. and Jones, G.T. (2009) "Cooperative Strategies to Reduce Ambulance Diversion", In *Proceedings of the 41<sup>st</sup> Winter*



- 
- Simulation Conference*, Edited by M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin and R.G. Ingalls, pp. 1861-1874, Austin, TX, December 13-16.
- Hawe, G.I., Coates, G., Wilson, D.T. and Crouch, R.S. (2012) "Agent-Based Simulation for Large-Scale Emergency Response: A Survey of Usage and Implementation." *ACM Computing Surveys*, Vol. 45, No. 1, Article 8.
- Henderson, S.G. and Mason, A.J. (2004) "Ambulance Service Planning: Simulation and Data Visualization." In *Operations Research and Health Care: A Handbook of Methods and Applications*, Edited by M. L. Brandeau, F. Sainfort and W.P. Pierskalla, pp. 77-102, Kluwer Academic, Boston.
- Heath, S.K., Brailsford, S.C., Buss, A. and Macal, C.M. (2011) "Cross-Paradigm Simulation Modeling: Challenges and Successes." In *Proceedings of the 43<sup>rd</sup> Winter Simulation Conference*, Edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White and M. Fu, pp. 2788-2802, Phoenix, AZ, December 11-14.
- Hoefer, A. and Tichy, W.F. (2007) "Status of Empirical Research in Software Engineering." In *Empirical Software Engineering Issues: Assessment and Future Directions*, Edited by Basili et al., pp. 10-19, Springer-Verlag.
- Holland, J. H., (1995) "Hidden Order: How Adaptation Builds Complexity." In Macal, C.M. and North, M.J. (2006) "Tutorial on Agent-based Modelling and Simulation Part 2: How to Model with Agents." In *Proceedings of the 38<sup>th</sup> Winter Simulation Conference*, Edited by L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto, pp.73-83, Monterey, CA, December 03-06.
- Holm, L.B. and Dahl, F.A. (2010) "Simulating the Influence of a 45% Increase in Patient Volume on the Emergency Department of Akershus University Hospital." In *Proceedings of the 42<sup>nd</sup> Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Mon-

- 
- toya-Torres, J. Hujan and E. Yucesan, pp. 2455-2461, Baltimore, MD, December 05-08.
- Ibri, S., Nourelfath, M. and Drias, H. (2012) “A Multi-Agent Approach for Integrated Emergency Vehicle Dispatching and Covering Problem.” *Engineering Applications of Artificial Intelligence*, Vol. 25, No. 3, pp. 554–565.
- IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP), IEEE Standard 1730, 2010.
- IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), IEEE Standard 1516.3, 2003.
- IEEE Recommended Practice for Verification, Validation, and Accreditation of a Federation – An Overlay to the High Level Architecture Federation Development and Execution Process, IEEE Standard 1516.4, 2007.
- IEEE Standard for Information Technology – Protocols for Distributed Interactive Simulation Applications – Entity Information and Interaction, IEEE Standard 1278, 1993.
- IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification, IEEE Standard 1516.1, 2010.
- IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, IEEE Standard 1516, 2010.
- IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, IEEE Standard 1516, 2000.
- IEEE Standard for Modelling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification, IEEE Standard 1516.2, 2010.

- Jahangirian, M., Eldabi, T., Naseer, A., Stergioulas, L.K. and Young, T. (2010) "Simulation in Manufacturing and Business: A review." *European Journal of Operational research*, Vol. 203, No. 1, pp. 1-13.
- Jones, S.G., Ashby, A.J., Momin, S.R. and Naidoo, A. (2010) "Spatial Implications Associated with Using Euclidean Distance Measurements and Geographic Centroid Imputation in Health Care Research." *Health Services Research*, Vol. 45, No.1, pp. 316-327.
- Kneebone, R., Arora, S., King, D., Bello, F., Sevdalis, N., Kassab, E., Aggarwal, R., Darzi, A. and Nestel, D. (2010) "Distributed Simulation – Accessible Immersive Training." *Medical Teacher*, Vol. 32, No. 1, pp. 65-70.
- Konrad, R., DeSotto, Kristine, Grocela, A., McAuley, P., Wang, J., Lyons, J. and Bruin, M. (2013) "Modeling the Impact of Changing Patient Flow Processes in an Emergency Department: Insight from a Computer Simulation Study." *Operations Research for Health Care*, Vol. 2, No. 4, pp. 66-74.
- Kuhn, J.J.R., Courtney, J.F., Morris, B. and Tatara, E.R. (2010) "Agent-Based Analysis and Simulation of the Consumer Airline Market Share for Frontier Airlines." *Knowledge-Based Systems*, Vol. 23, No. 8, pp. 875–882.
- Law, A.M. and Kelton, W.D. (2000) "Simulation Modeling and Analysis." 3<sup>rd</sup> Edition, McGraw-Hill Book Co, Singapore, ISBN: 0-07-116537-1.
- Lee, T., Jang, H. Cho, S.H. and Turner, J.G. (2012) "A Simulation-Based Iterative Method for a Trauma Center – Air Ambulance Location Problem." In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, pp. 955-966, Berlin, DE, December 09-12.
- Lim, M.E., Worster, A. Goeree, R. and Tarride, J.E. (2013) "Simulating an Emergency Department: The Importance of Modeling the Interactions between Physicians and Dele-

- 
- gates in a Discrete Event Simulation.” *BMC Medical Informatics & Decision Making*, Vol. 13, No. 59, pp. 1-11.
- Loefstrand, B., Bystroem, J. and Johansson, M. (2003) “HLA Design Patterns for Agent Based Federates.” In *Proceedings of the Euro Simulation Interoperability Workshop (03E-SIW-076)*, June 2003.
- London Ambulance Service: online (last accessed January 2014)  
<http://www.londonambulance.nhs.uk/>
- Macal, C.M. and North, M.J. (2010) “Tutorial on Agent-Based Modelling and Simulation.” *Journal of Simulation*, Vol. 4, No. 3, pp. 151-162.
- Macal, C.M. and North, M.J. (2006) “Tutorial on Agent-based Modelling and Simulation Part 2: How to Model with Agents.” In *Proceedings of the 38<sup>th</sup> Winter Simulation Conference*, Edited by L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto, pp.73-83, Monterey, CA, December 03-06.
- Malik, A.W., Khan, S.A. and Hassan, R. (2010) “An HLA Based Real Time Simulation Engine for Man-in-Loop Net Centric System.” *Pakistan Journal of Engineering and Applied Sciences*, Vol. 7, July 2010, pp. 47-54.
- March, S.T., and Smith, G.F. (1995) “Design and Natural Science Research on Information Technology.” *Decision Support Systems*, Vol. 15, No. 4, pp. 251-266.
- Miller D.C. and Thorpe, J.A. (1995) “SIMNET: The Advent of Simulator Networking.” In *Proceedings of the IEEE*, Vol. 83, No. 8, pp.1114 -1123.
- Minson, R., and Theodoropoulos, G.K. (2008) “Distributing RePast Agent-Based Simulations with HLA.” *Concurrency and Computation: Practice and Experience*, Vol. 20, No. 10, pp. 1225-1256.

- Moeller, B. and Loeffstrand, B. (2009) "Getting Started with FOM Modules." In *Proceedings of the 2009 Fall Simulation Interoperability Workshop (09F-SIW-082)*, Simulation Interoperability Standards Organisation, Orlando, FL, September 21-25.
- Mustafee, N., Katsialaki, K. and Taylor, S.J.E. (2010) "Profiling Literature in Healthcare Simulation." *Simulation*, Vol. 86, No. 8-9, pp. 543-558.
- Mustafee, N., Taylor, S.J.E., Katsialaki, K. and Brailsford, S. (2009) "Facilitating the Analysis of a UK National Blood Service Supply Chain Using Distributed Simulation." *Simulation*, Vol. 85, No. 2, pp.113-128.
- Mustafee, N. and Taylor, S.J.E. (2006) "Investigating Distributed Simulation with COTS Simulation Packages: Experiences with Simul8 and the HLA." In *Proceedings of the Operational Research Society 3<sup>rd</sup> Simulation Workshop (SW06)*, pp. 33-42.
- Nance, R.E and Sargent, R.G. (2002) "Perspectives on the Evolution of Simulation." *Operations Research*, Vol. 50, No. 1, pp. 161-172.
- Nance, R.E. (1995) "Simulation Programming Languages: An Abridged History." In *Proceedings of the 27<sup>th</sup> Winter Simulation Conference*, Edited by C. Alexopoulos, K. Kang, W.R. Lilegdon and D. Goldsman, pp. 1307-1313, Arlington, VA, December 3-6.
- Nance, R.E. (1981) "The Time and State Relationships in Simulation Modeling." *Communications of the ACM*, Vol. 24, No. 4, pp.173-179.
- Naoum-Sawaya, J. and Elhedhli, S. (2013) "A Stochastic Optimization Model for Real-Time Ambulance Deployment." *Computers & Operations Research*, Vol. 40, No. 8, pp. 1972-1978.
- NHS England (a): online (last accessed January 2014)  
<http://www.england.nhs.uk/statistics/statistical-work-areas/ambulance-quality-indicators/ambqi-data-2011-12/>

- NHS England (b): online (last accessed January 2014) <http://www.england.nhs.uk/statistics/statistical-work-areas/ae-waiting-times-and-activity/>
- NHS UK: online (last accessed January 2014) <http://www.nhs.uk/>
- North, M.J. and Macal, C.M. (2007) “Managing Business Complexity.” New York: Oxford University Press, Inc.
- Nouman, A., Anagnostou, A. and Taylor, S.J.E. (2013) “Developing a Distributed Agent-Based and DES Simulation Using poRTico and Repast.” In *Proceedings of the 17<sup>th</sup> IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2013)*, pp. 97-104, Delft, NL, October 30-November 01.
- Nygaard, K. and Ole-Johan, D. (1978) “The Development of the SIMULA Languages.” In *Proceedings of The first ACM SIGPLAN conference on History of programming languages (HOPL)*, pp. 245-272, Los Angeles, CA , June 01-03.
- Overstreet, M.C. and Nance, R.E. (2004) “Characterizations and Relationships of World Views.” In *Proceedings of the 36<sup>th</sup> Winter Simulation Conference*, Edited by R.G. Ingalls, M.D. Rossetti, J.S. Smith and B.A. Peters, pp. 279-287, Washington, DC, December 05-08.
- Page, E.H., Briggs, R. and Tufarolo, J.A. (2004). “Toward a Family of Maturity Models for the Simulation Interconnection Problem.” In *Proceedings of the Spring 2004 Simulation Interoperability Workshop*, IEEE CS Press.
- Paul, J.A. and Lin, L. (2012) “Models for Improving Patient Throughput and Waiting at Hospital Emergency Departments.” *Administration of Emergency Medicine*, Vol. 43, No. 6, pp. 1119-1126.

- Paul, R.J. and Taylor, S.J.E. (2002) "What Use is Model Reuse: Is There a Crook at the End of the Rainbow?" In *Proceedings of the 34<sup>th</sup> Winter Simulation Conference*, Edited by E. Yucesan, C.H. Chen, J.L. Snowdon and J.M. Charnes, pp. 648-652, San Diego, CA, December 08-11.
- Pawlaszczyk, D. and Strassburger, S. (2009) "Scalability in Distributed Simulations of Agent-Based Models." In *Proceedings of the 41<sup>st</sup> Winter Simulation Conference*, Edited by M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin and R.G. Ingalls, pp. 1189-1200, Austin, TX, December 13-16.
- Pegden, C.D. (2010) "Advanced Tutorial: Overview of Simulation World Views." In *Proceedings of the 42<sup>nd</sup> Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yucesan, pp. 210-215, Baltimore, MD, December 05-08.
- Petty, M.D. and Weisel, E.W. (2003) "A Composability Lexicon." In *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, IEEE CS Press.
- Pidd, M. (2004) "Computer Simulation in Management Science." 5th Edition, John Wiley & Sons Ltd., Chichester, UK, ISBN: 0-470-09230-0.
- Pidd, M. (2002) "Simulation Software and Model Reuse: A Polemic." In *Proceedings of the 34<sup>th</sup> Winter Simulation Conference*, Edited by E. Yucesan, C.H. Chen, J.L. Snowdon and J.M. Charnes, pp. 772-775, San Diego, CA, December 08-11.
- Pidd, M. (1995) "Object-Orientation, Discrete Simulation and the Three-Phase Approach." *Journal of the Operational Research Society*, Vol. 46, No. 3, pp. 362-374.
- Pidd, M., de Silva, F.N. and Eglese, R.W. (1993) "CEMPS: A Configurable Evacuation Management and Planning System – A Progress Report." In *Proceedings of the 24<sup>th</sup>*

- 
- Winter Simulation Conference*, Edited by G.W. Evans, N. Mollaghasemi, E.C. Russell and W.E. Biles, pp. 1319-1323, Los Angeles, CA, December 12-15.
- Pollacia, L.F. (1989) "A Survey of Discrete Event Simulation And State-Of-the-Art Discrete Event Languages." *ACM SIGSIM Simulation Digest*, Vol. 20, No. 3, pp. 8-25.
- Rahmat, M.H., Annamalai, M., Halim, S.A. and Ahmad, R. (2013) "Agent-Based Modeling and Simulation of Emergency Department Re-triage." In *IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pp. 219-224, Langkawi, Malaysia, April 07-09.
- Ramirez, A., Fowler, J.W. and Wu, T. (2011) "Design of Centralized Ambulance Diversion Policies Using Simulation-Optimization." In *Proceedings of the 43<sup>rd</sup> Winter Simulation Conference*, Edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White and M. Fu, pp. 1251-1262, Phoenix, AZ, December 11-14.
- Reese, R. and Wyatt, D.L. (1987) "Software Reuse and Simulation." In *Proceedings of the 19<sup>th</sup> Winter Simulation Conference*, Edited by A. Thesen, H. Grant and W.D. Kelton., pp. 185-192, Atlanta, GA, December 14-16.
- Robinson, S. (2005) "Discrete-Event Simulation: From the Pioneers to the Present, What Next?" *Journal of the Operational Research Society*, Vol. 56, No. 6, pp. 619-629.
- Robinson, S. (2004) "Simulation – The Practice of Model Development and Use." John Wiley & Sons Ltd: Chichester, UK.
- Robinson, S., Nance, R.E., Paul, R.J., Pidd, M. and Taylor, S.J.E. (2004) "Simulation Model Reuse: Definitions, Benefits and Obstacles." *Simulation Modelling Practice and Theory*, Vol. 12, No.7-8, pp. 479-494.



- Siebers, P.O., Macal, C.M., Garnett, J., Buxton, D. and Pidd, M. (2010) "Discrete-Event Simulation is Dead, Long Live Agent-Based Simulation!" *Journal of Simulation*, Vol. 4, No. 3, pp. 204-210.
- Silva, P.M.S. and Pinto, L.R. (2010) "Emergency Medical Systems Analysis by Simulation and Optimization." In *Proceedings of the 42<sup>nd</sup> Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yucesan, pp. 2422-2432, Baltimore, MD, December 05-08.
- Silver, G.A, Miller, J.A., Hybinette, M., Baramidze, G. and York, W.S. (2011) "DeMO: An Ontology for Discrete-event Modeling and Simulation." *Simulation*, Vol. 87, No. 9, pp. 747-773.
- SISO Standard for Base Object Model (BOM) Template Specification, SISO-STD-003-2006.
- SISO Guide for BOM Use and Implementation, SISO-STD-003.1-2006.
- SISO Standard for COTS Simulation Package Interoperability Reference Models, SISO-STD-006-2010.
- Steinbach, R., Cairns, J., Grundy, C. and Edwards, P. (2012) "Cost Benefit Analysis of 20 mph Zones in London." *Injury Prevention*, Vol. 19, No. 3, pp. 211-213.
- Su, S. and Shih, C.L. (2003) "Modeling an Emergency Medical Services System Using Computer Simulation." *International Journal of Medical Informatics*, Vol. 72, No. 1-3, pp. 57-72.
- Sumari, S., Ibrahim, R., Zakaria, N.H. and Ab Hamid, A.H. (2013) "Comparing Three Simulation Model Using Taxonomy: System Dynamic Simulation, Discrete Event Simulation and Agent Based Simulation." *International Journal of Management Excellence*, Vol. 1, No. 3, pp. 54-59.

- Tatara, E., Teymour, F. and Cinar, A. (2007) "Control of Complex Distributed Systems with Distributed Intelligent Agents." *Journal of Process Control*, Vol. 17, No. 5, pp. 415-427.
- Taylor, S.J.E., Brailsford, S., Chick, S.E., L'Ecuyer, P., Macal, C.M. and Nelson, B.L. (2013) "Modeling and Simulation Grand Challenges: An OR/MS Perspective." In *Proceedings of the 45<sup>th</sup> Winter Simulation Conference*, Edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill and M. E. Kuhl, pp. 1269-1282, December 08-11.
- Taylor S.J.E., Turner S.J., Strassburger, S. and Mustafee, N. (2012a) "Bridging The Gap: A Standards-Based Approach to OR/MS Distributed Simulation." *ACM Transactions on Modeling and Computer Simulation*, Vol. 22, No. 4, Article 18.
- Taylor, S.J.E., Fishwick, P.A., Fujimoto, R., Uhrmacher, A.M., Page, E.H. and Wainer, G. (2012b) "Panel on Grand Challenges for Modeling and Simulation." In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, pp. 2614-2628, Berlin, DE, December 09-12.
- Taylor, S.J.E., Eldabi, T., Riley, G., Paul, R.J. and Pidd, M. (2009a) "Simulation Modelling is 50! Do we need a reality check?" *Journal of the Operational Research Society*, Vol. 60, No. S1, pp. S69-S82.
- Taylor, S.J.E., Mustafee, N., Turner, S.J., Pan, K. and Strassburger, S. (2009b) "Commercial-Off-The-Shelf Simulation Package Interoperability: Issues and Futures." In *Proceedings of the 41<sup>st</sup> Winter Simulation Conference*, Edited by M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin and R.G. Ingalls, pp. 203-215, Austin, TX, December 13-16.
- Taylor, S.J.E., Bruzzone, A., Fujimoto, R., Gan, B.P., Starssburger, S. and Paul, R.J. (2002a) "Distributed Simulation and Industry: Potentials and Pitfalls." In *Proceedings*

- 
- of the 34<sup>th</sup> Winter Simulation Conference*, Edited by E. Yucesan, C.H. Chen, J.L. Snowdon and J.M. Charnes, pp. 688-694, San Diego, CA, December 08-11.
- Taylor, S.J.E., Sudra, R., Janahan, T., Tan, G. and Ladbrook, J. (2002b) "GRIDS-SCF: An Infrastructure for Distributed Supply Chain Simulation." *Simulation*, Vol. 78, No. 5, pp. 312-320.
- The Department of Health: online (last accessed January 2014) [www.dh.gov.uk](http://www.dh.gov.uk)
- The MITRE Corporation: online (last accessed January 2014) [www.mitre.org](http://www.mitre.org)
- The portico Project: online (last accessed January 2014) [www.porticoproject.org](http://www.porticoproject.org)
- The press Association (2011) "40% of A&E Patients Do Not Need Treatment." *Health Service Journal*: online (last accessed January 2014) <http://www.hsj.co.uk/news/acute-care/emergency/40-of-ae-patients-do-not-need-treatment/5024271.article>
- The Repast Suite: online (last accessed January 2014) [repast.sourceforge.net](http://repast.sourceforge.net)
- Tocher, K.D. (1965) "Review of Simulation Languages." *Operational Research Quarterly*, Vol. 16, No. 2, pp.189-217.
- Tocher, K.D. and Owen, D.G. (1960) "The Automatic Programming of Simulation." In *Proceedings of the Second International Conference on Operational Research*, English Universities Press: UK, pp. 58-60. In: Taylor, S.J.E., Eldabi, T., Riley, G., Paul, R.J. and Pidd, M. (2009) "Simulation Modelling is 50! Do we need a reality check?" *Journal of the Operational Research Society*, Vol.60, No.S1, pp.S69-S82.
- Tolk, A. and Muguira, J.A. (2003) "The Levels of Conceptual Interoperability Model." In *Proceedings of the IEEE Fall Simulation Interoperability Workshop*, IEEE CS Press: Orlando, FL, September 14-19.

- Tu, Z., Zacharewicz, G. and Chen, D. (2011) "Developing a Web-Enabled Federate Based on poRTIco RTI." In *Proceedings of the 43<sup>rd</sup> Winter Simulation Conference*, Edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White and M. Fu, pp. 2289-2301, Phoenix, AZ, December 11-14.
- Turnitsa, C., Padilla, J.J. and Tolk, A. (2010) "Ontology for Modeling and Simulation." In *Proceedings of the 42<sup>nd</sup> Winter Simulation Conference*, Edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yucesan, pp. 2422-2432, Baltimore, MD, December 05-08.
- Ulgen, O.M., Black, J.J., Johnsonbaugh, B. and Klungle, R. (1994) "Simulation Methodology – A Practitioner’s Perspective." *International Journal of Industrial Engineering, Applications and Practice*, Vol. 1, No. 2: online (last accessed January 2014) [http://pmcorp.se/PublishedPapers/Simulation%20Publications/Sim-Methodology/SimulationMethodology\\_APractitionersPerspective.pdf](http://pmcorp.se/PublishedPapers/Simulation%20Publications/Sim-Methodology/SimulationMethodology_APractitionersPerspective.pdf)
- Van Buuren, M., Aardal, K., Van der Mei, R. and Post, H. (2012) "Evaluating Dynamic Dispatch Strategies for Emergency Medical Services: TIFAR Simulation Tool." In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, pp. 509-519, Berlin, DE, December 09-12.
- Wang, Y., Luangkesorn, K.L. and Shuman, L. (2012) "Modeling Emergency Medical Response to a Mass Casualty Incident Using Agent Based Simulation." *Socio-Economic Planning Sciences*, Vol. 46, No. 4, pp. 281-290.
- Wang, L. (2009) "An Agent-Based Simulation for Workflow in Emergency Department," In *Proceedings of the 2009 IEEE Systems and Information Engineering Design Symposium*, pp. 19-23, Charlottesville, VA, April 24.

- 
- Wang, W.G., Tolk, A., Wang, W.P. (2009) “The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to Modeling and Simulation.” In *Proceedings of the Spring Simulation Multiconference*, Society for Modeling & Simulation International (SCS): San Diego, CA, March 22-27.
- Yin, R.K. (2009) “Case Study Research, Design and Methods.” 4<sup>th</sup> ed., Sage Publications, CA. ISBN 978-1-4129-6099-1.
- Zulkepli, J., Eldabi, T. and Mustafee, N. (2012) “Hybrid Simulation for Modelling Large Systems: An Example of Integrated Care Model.” In *Proceedings of the 44<sup>th</sup> Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose and A.M. Uhrmacher, pp. 758-769, Berlin, DE, December 09-12.

## Appendix

### A1. Ambulance Service Model: Context Class

```
package ambulanceservicemodel;

import hla.rti1516e.exceptions.RTIException;

import java.util.List;
import java.util.Random;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import cern.jet.random.Normal;
import ambulanceservicemodel.EmergencyCall;
import ambulanceservicemodel.AmbulanceStation;
import repast.simphony.context.Context;
import repast.simphony.context.DefaultContext;
import repast.simphony.context.space.continuous.ContinuousSpaceFactoryFinder;
import repast.simphony.context.space.grid.GridFactoryFinder;
import repast.simphony.dataLoader.ContextBuilder;
import repast.simphony.engine.environment.RunEnvironment;
import repast.simphony.engine.environment.RunState;
import repast.simphony.engine.schedule.ISchedulableAction;
import repast.simphony.engine.schedule.ISchedule;
import repast.simphony.engine.schedule.ScheduleParameters;
import repast.simphony.essentials.RepastEssentials;
import repast.simphony.random.RandomHelper;
import repast.simphony.space.continuous.ContinuousSpace;
import repast.simphony.space.continuous.RandomCartesianAdder;
import repast.simphony.space.continuous.StrictBorders;
import repast.simphony.space.grid.Grid;
import repast.simphony.space.grid.GridBuilderParameters;
import repast.simphony.space.grid.SimpleGridAdder;

public class AmbulanceServiceBuilder extends DefaultContext<Object>
    implements ContextBuilder<Object>{
    // Initialisation (Environment):
    // (i) parse the data files,
    // (ii) create the used space, grid instances, and
    // (iii) populate the grid with both AmbulanceStation and Hospital
    agents.
    public static final String CONTEXT_ID = "ambulanceservicemodel";
    public static final String EMERGENCY_CALL_CONTEXT = "emergencycallcon-
text";
    public static final String AMBULANCE_CONTEXT = "ambulancecontext";
    public static final String SPACE_ID = "space";
    public static final String GRID_ID = "grid";
```

```

    public static final String AMBULANCE_STATION_DATA_FILE = "AmbulanceStation.Data";
    public static final String HOSPITAL_DATA_FILE = "Hospital.Data";
    public static final String EMERGENCY_INFO_DATA_FILE = "EmergencyInfo.Data";
    public static final int DATA_FILES_HEADER_LINES = 3;
    public static final int GRID_SIZE_X = 60;
    public static final int GRID_SIZE_Y = 40;
    public static final int SPEED = 15;
    public static final double CORRECTIVE_COEFFICIENT = 1.32;
    public static final String OUTPUT_FILENAME = "ambulanceOutput.csv";

    protected static double currentTimeInHours;

    // Read the ambulance station data from a specified file
    public static final List<AmbulanceStation> ambulanceStationData =
        readAmbulanceStationDataFile(AMBULANCE_STATION_DATA_FILE);
    // Read the hospital data from a specified file
    public static final List<Hospital> hospitalData = readHospitalDataFile(HOSPITAL_DATA_FILE);
    // Read the emergency coordinates data from a specified file
    public static final double[][] emergencyInfo = readEmergencyInfoDataFile(EMERGENCY_INFO_DATA_FILE);
    //Variables for scheduling EmergencyCall arrivals
    static ISchedule schedule;
    ISchedulableAction nextArrival;
    ////////////Distributions
    static Normal arrivalDist;
    double meanArrivalDist=2.52;
    double sdArrivalDist=0.09;
    static Normal treatmentDist;
    double meanTreatmentDist=22.52;
    double sdTreatmentDist=10.54;
    static int[] conditionDist;
    static int[] hospitalisationDist;
    ////////////End distributions
    static int callID=0;
    static int ambStaID=0;
    static int ambID=0;
    static int hosID=0;
    static Federate f;
    private static Context<Object> context;
    @Override
    public Context<Object> build(final Context<Object> context) {
        context.setId(CONTEXT_ID);
        final ContinuousSpace<Object> space = ContinuousSpaceFactoryFinder
            .createContinuousSpaceFactory(null) // No hints
            .createContinuousSpace(SPACE_ID, context,
                new RandomCartesianAdder<Object>(),
                new StrictBorders(), GRID_SIZE_X, GRID_SIZE_Y);
        final Grid<Object> grid = GridFactoryFinder
            .createGridFactory(null).createGrid(GRID_ID, context,
                new GridBuilderParameters<Object>(
                    new repast.simphony.space.grid.StrictBorders(),
                    new SimpleGridAdder<Object>(),
                    true,
                    GRID_SIZE_X, GRID_SIZE_Y));

```

```

        createConditionDist();
        createHospitalisationDist();

        //initialise ambulance stations and hospitals
        createAmbulanceStations(context, ambulanceStationData, grid,
space);
        try
        {

            f = new Federate(context,space,grid,"Ambulance");

            context.add(f);
        }
        catch( RTIException rtie )
        {
            rtie.printStackTrace();
        }
        catch( Exception e )
        {
            e.printStackTrace();
        }

        createHospitals(context, hospitalData, grid, space);

        ISchedule sschedule= RunEnvironment.getInstance().getCurrentSchedule();
        sschedule.schedule(ScheduleParameters.createOneTime(0),this,"createOutputFile");
        schedule = RunEnvironment.getInstance().getCurrentSchedule();
        arrivalDist = RandomHelper.createNormal(meanArrivalDist,
sdArrivalDist);
        treatmentDist = RandomHelper.createNormal(meanTreatmentDist,
sdTreatmentDist);
        schedule.schedule(ScheduleParameters.createOneTime(0), this,
"arrivalCalls");

        return context;
    }

    public void createOutputFile(){
        try{
            String filename = OUTPUT_FILENAME;
            FileWriter fw = new FileWriter(filename);
            BufferedWriter out = new BufferedWriter(fw);
            out.write("Tick"+", "+"Call-IncidentID"+",
"+"IDinPat"+", "+"IDinAmb"+", "
                    +"AmbulanceStationID"+", "
                    +"AmbulanceID"+",
"+"AmbulanceAvailability"+", "+"PatientCondition"+", "
                    +"NeedHospitalisation"+",
"+"HospitalID"+", "+"HospitalAvailability"+", "
                    +"T1: CallGenerated"+", "+"T2: Ambu-
lanceFound"+", "+"T3: ArriveAtScene"+", "
                    +"T4: HospitalFound"+", "+"T5: Depart-
FromScene"+", "

```



```

+ "T6: ArriveAtHospital"+", "+ "T7: Ar-
riveAtStation\r\n");
        out.close();
    }catch (Exception e){
        System.err.println("Error: " + e.getMessage());
    }
}
@SuppressWarnings({ "unchecked" })
public void arrivalCalls() {
    Context<Object> context =
RunState.getInstance().getMasterContext();
    Grid<Object> grid = (Grid<Object>) con-
text.getProjection(AmbulanceServiceBuilder.GRID_ID);
    ContinuousSpace<Object> space = (ContinuousSpace<Object>)
context.getProjection(AmbulanceServiceBuilder.SPACE_ID);
    AmbulanceServiceBuilder.createEmergencyCalls(context, grid,
space);
        double arriveTime = sched-
ule.getTickCount()+arrivalDist.nextDouble();
        while(arriveTime<=schedule.getTickCount()){
            arriveTime = sched-
ule.getTickCount()+arrivalDist.nextDouble();
        }
        nextArrival = sched-
ule.schedule(ScheduleParameters.createOneTime(arriveTime, 10), this, "arrival-
Calls");
    }
    static void createEmergencyCalls(final Context<Object> context, fi-
nal Grid<Object> grid,
        final ContinuousSpace<Object> space) {
        assert (context != null);
        assert (grid != null);

        for (int i = 0; i <1; ++i){
            final EmergencyCall emergencyCall = new Emergen-
cyCall(grid, space);
                context.add(emergencyCall);
                emergencyCall.setCall_X(emergencyInfo[callID][0]);
                emergencyCall.setCall_Y(emergencyInfo[callID][1]);
                space.moveTo(emergencyCall,
(int)emergencyCall.getCall_X(), (int)emergencyCall.getCall_Y());
                grid.moveTo(emergencyCall,
(int)emergencyCall.getCall_X(), (int)emergencyCall.getCall_Y());
                final Patient patient = new Patient(grid, space);
                context.add(patient);
                patient.setCondition((int)emergencyInfo[callID][2]);
                patient.setTreatmentTime(treatmentDist.nextDouble());
                while(patient.getTreatmentTime()<=0){
                    pa-
tient.setTreatmentTime(treatmentDist.nextDouble());
                }

                if (emergencyInfo[callID][3]==0){
                    patient.setNeedHospitalisation(false);
                }
                else if(emergencyInfo[callID][3]==1){
                    patient.setNeedHospitalisation(true);
                }
            }
        }
    }
}

```

```

    }
    callID += 1;
    emergencyCall.setCallID(callID);
   emergen-
cyCall.setTCallGenerated(RepastEssentials.GetTickCount());
    patient.setCallID(emergencyCall.getCallID());
    space.moveTo(patient, (int)emergencyCall.getCall_X(),
(int)emergencyCall.getCall_Y());
    grid.moveTo(patient, (int)emergencyCall.getCall_X(),
(int)emergencyCall.getCall_Y());
    }
}
private void createAmbulanceStations(final Context<Object> context,
final List<AmbulanceStation> ambulanceStationData,
final Grid<Object> grid,
final ContinuousSpace<Object> space) {
    assert (context != null);
    assert (ambulanceStationData != null);
    assert (grid != null);

    for (final AmbulanceStation asd : ambulanceStationData) {
        context.add(asd);
        ambStaID += 1;
        asd.setAmbulanceStationID(ambStaID);
        space.moveTo(asd, asd.getX(), asd.getY());
        grid.moveTo(asd, asd.getX(), asd.getY());

        for (int i = 0; i < asd.getCapacity(); ++i){
            final Ambulance ambulance = new Ambulance(grid,
space);

            context.add(ambulance);
            ambID += 1;
            ambulance.setAmbulanceID(ambID);
            space.moveTo(ambulance, asd.getX(), asd.getY());
            grid.moveTo(ambulance, asd.getX(), asd.getY());
            ambulance.ambStartingX = asd.getX();
            ambulance.ambStartingY = asd.getY();
            ambulance.setAmbAvailability(true);
        }
    }
}

private void createHospitals(final Context<Object> context,
final List<Hospital> hospitalData, final Grid<Object>
grid,
final ContinuousSpace<Object> space) {
    assert (context != null);
    assert (hospitalData != null);
    assert (grid != null);

    for (final Hospital hsd : hospitalData) {
        context.add(hsd);
        hosID += 1;
        hsd.setHospitalID(hosID);
        space.moveTo(hsd, hsd.getX(), hsd.getY());
    }
}

```

```

        grid.moveTo(hsd, hsd.getX(), hsd.getY());
        hsd.setHosAvailability(hsd.capacity);
    }
}

//Code to get the current lists of all agents
protected static ArrayList<EmergencyCall> getEmergencyCallList() {
    @SuppressWarnings("unchecked")
    final Iterable<EmergencyCall> emergencyCalls =
RunState.getInstance().getMasterContext()
        .getObjects(EmergencyCall.class);
    final ArrayList<EmergencyCall> emergencyCallList = new Ar-
rayList<EmergencyCall>();

    for (final EmergencyCall emergencyCall : emergencyCalls) {
        emergencyCallList.add(emergencyCall);
    }
    return emergencyCallList;
}

protected static ArrayList<Ambulance> getAmbulanceList() {
    @SuppressWarnings("unchecked")
    final Iterable<Ambulance> ambulances =
RunState.getInstance().getMasterContext()
        .getObjects(Ambulance.class);
    final ArrayList<Ambulance> ambulancelist = new Ar-
rayList<Ambulance>();

    for (final Ambulance ambulance : ambulances) {
        ambulancelist.add(ambulance);
    }
    return ambulancelist;
}

public void createConditionDist(){
    int[] myNumbers = {1, 2};
    int[] myProb = {26, 74};
    int[] myComProb = new int[2];
    conditionDist = new int[500000];

    int i, j, k;

    Random myRandom = new Random(10);
    myComProb[0] = myProb[0];

    for (i =1; i< 2; i++){
        myComProb[i] = myComProb[i-1]+myProb[i];
    }

    for (j = 0; j <500000; j++){
        k = myRandom.nextInt(myComProb[1]);
        for (i =0; k>myComProb[i]; i++);
        conditionDist[j]=myNumbers[i];
        System.out.print(conditionDist[i]);

    }
    System.out.println();
}

```

```

    }
    public void createHospitalisationDist(){
        int[] myNumbers = {0, 1};
        int[] myProb = {38, 62};
        int[] myComProb = new int[2];
        hospitalisationDist = new int[500000];

        int i, j, k;

        Random myRandom = new Random(0);
        myComProb[0] = myProb[0];

        for (i =1; i< 2; i++){
            myComProb[i] = myComProb[i-1]+myProb[i];
        }

        for (j = 0; j <500000; j++){
            k = myRandom.nextInt(myComProb[1]);
            for (i =0; k>myComProb[i]; i++);
            hospitalisationDist[j]=myNumbers[i];
            System.out.print(hospitalisationDist[i]);
        }
        System.out.println();
    }

    //Read from ambulance station data file
    public static ArrayList<AmbulanceStation> readAmbulanceStationData-
File
    (final String ambulanceStationDataFileName) {
        if (null == ambulanceStationDataFileName) {
            throw new IllegalArgumentException(
                "Parameter ambulanceStationDataFileName
cannot be null.");
        }
        if (ambulanceStationDataFileName.isEmpty()) {
            throw new IllegalArgumentException(
                "A file name cannot have an empty
name.");
        }
        ArrayList<AmbulanceStation> ret = new Ar-
rayList<AmbulanceStation>();
        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileRead-
er(ambulanceStationDataFileName));
            // Skip header
            for (int i = 0; i < DATA_FILES_HEADER_LINES; ++i) {
                br.readLine();
            }
            // Read lines, parse data and add a new AmbulanceSta-
tionData for each one
            String line = null;
            while ((line = br.readLine()) != null) {
                // Split the line around whitespaces
                final String[] data = line.split("\\s+");
                // Check if current line seems all right
                if (data.length != 3) {

```

```

        throw new IllegalArgumentException(
            "File %s contains a mal-
            ambulanceStationDataFile-
            tion(String.format(
                formed input line: %s",
                Name, line));
    }
    try {
        int idx = 0;
        final int x = Inte-
        final int y = Inte-
        final int capacity = Inte-
        ret.add(new AmbulanceStation(x, y, capac-
        ity));
    } catch (final NumberFormatException e) {
        e.printStackTrace();
        throw new IllegalArgumentException(
            "File %s contains a mal-
            ambulanceStationDataFile-
            tion(String.format(
                formed input line: %s",
                Name, line), e);
    }
} catch (final FileNotFoundException e) {
    e.printStackTrace();
} catch (final IOException e) {
    e.printStackTrace();
} finally {
    if (br != null) {
        try {
            br.close();
        } catch (final IOException e) {
            e.printStackTrace();
        }
    }
}
return ret;
}

//Read from hospital data file
public static ArrayList<Hospital> readHospitalDataFile(final String
hospitalDataFileName) {
    if (null == hospitalDataFileName) {
        throw new IllegalArgumentException(
            "Parameter ambulanceStationDataFileName
            cannot be null.");
    }
    if (hospitalDataFileName.isEmpty()) {
        throw new IllegalArgumentException(
            "A file name cannot have an empty
            name.");
    }
    final ArrayList<Hospital> ret = new ArrayList<Hospital>();

```

```

        BufferedReader br = null;
        try {
            br = new BufferedReader(new FileReader(
er(hospitalDataFileName));
            // Skip header
            for (int i = 0; i < DATA_FILES_HEADER_LINES; ++i) {
                br.readLine();
            }
            // Read lines, parse data and add a new HospitalData
for each one
            String line = null;
            while ((line = br.readLine()) != null) {
                // Split the line around whitespaces
                final String[] data = line.split("\\s+");
                // Check if current line seems all right
                if (data.length != 3) {
                    throw new IllegalArgumentException(
tion(String.format(
                        "File %s contains a mal-
formed input line: %s",
                        hospitalDataFileName,
line));
                }
                try {
                    int idx = 0;
                    final int x = Inte-
ger.parseInt(data[idx++]);
                    final int y = Inte-
ger.parseInt(data[idx++]);
                    final int capacity = Inte-
ger.parseInt(data[idx++]);
                    ret.add(new Hospital(x, y, capacity));
                } catch (final NumberFormatException e) {
                    e.printStackTrace();
                    throw new IllegalArgumentException(
tion(String.format(
                        "File %s contains a mal-
formed input line: %s",
                        hospitalDataFileName,
line), e);
                }
            }
        } catch (final FileNotFoundException e) {
            e.printStackTrace();
        } catch (final IOException e) {
            e.printStackTrace();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (final IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return ret;
    }
}

```

```

//Read from emergency coordinates data file
public static double[][] readEmergencyInfoDataFile(final String
emergencyInfoDataFileName) {
    if (null == emergencyInfoDataFileName) {
        throw new IllegalArgumentException(
            "Parameter emergencyInfoDataFileName can-
not be null.");
    }
    if (emergencyInfoDataFileName.isEmpty()) {
        throw new IllegalArgumentException(
            "A file name cannot have an empty
name.");
    }
    final double[][] ret = new double[23000][4];
    BufferedReader br = null;
    try {
        br = new BufferedReader(new FileRead-
er(emergencyInfoDataFileName));
        // Skip header
        for (int i = 0; i < DATA_FILES_HEADER_LINES; ++i) {
            br.readLine();
        }
        // Read lines, parse data and add new X, Y coordinates
and condition, hospitalisation for each one
        String line = null;
        int idxR = 0;
        while ((line = br.readLine()) != null) {
            // Split the line around whitespaces
            final String[] data = line.split("\\s+");
            // Check if current line seems all right
            if (data.length != 4) {
                throw new IllegalArgumentExcep-
tion(String.format(
                    "File %s contains a mal-
formed input line: %s",
                    emergencyInfoDataFileName,
                    line));
            }
            try {
                int idxC = 0;
                final double x = Dou-
ble.parseDouble(data[idxC++]);
                final double y = Dou-
ble.parseDouble(data[idxC++]);
                final double c = Dou-
ble.parseDouble(data[idxC++]);
                final double h = Dou-
ble.parseDouble(data[idxC++]);

                ret[idxR][0]=x;
                ret[idxR][1]=y;
                ret[idxR][2]=c;
                ret[idxR][3]=h;
                idxR++;
            } catch (final NumberFormatException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```
tion(String.format(
formed input line: %s",
line), e);
    }
    } catch (final FileNotFoundException e) {
        e.printStackTrace();
    } catch (final IOException e) {
        e.printStackTrace();
    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (final IOException e) {
                e.printStackTrace();
            }
        }
    }
    return ret;
}
}
```