ABSTRACT

AKSHAY, VENKITASUBRAMANIAN. Object-Oriented Framework for Healthcare Simulation. (Under the direction of Dr. Stephen D. Roberts and Dr. Jeffery A. Joines).

This thesis addresses the problem of modeling complexity of healthcare simulation projects. From an extensive literature review, three problems associated with healthcare simulation were elicited: lack of flexibility among healthcare simulation, lack of implementation methodologies to support building of complex integrated multi-facility simulation models, and lack of stakeholder involvement in simulation projects.

The first problem was attributed to lack of flexibility in healthcare simulation software. This can be deduced from sheer number of simulation projects resorting to develop their own simulation using general purpose languages. The root cause of the problem has been identified as the lack of conceptual framework in development of healthcare simulation libraries. In addition to this, we present four intuitive axioms to assist future simulation software developers and modelers in assessing the utility of simulation toolkit and models.

The second problem was attributed to lack of integrative framework for complex facilities simulations. Healthcare is a highly interconnected environment and multi-facility simulations will help in relaxing the fixed boundaries assumed in facility-specific simulation projects. In this thesis, we present a model development process that takes advantage of expertise of simulation community in facility-specific models and extend it to the development of complex integrated multi-facility systems.

Finally, we address the problem of stakeholder involvement and lack of implementation in simulation projects in healthcare sector. We present a causal loop diagram to visualize the dynamics of four simulation success factors: Verification, Validation, Fidelity and Credibility. Using these factors, we develop the concept of simulation project lifecycle to describe the evolution of simulation modeling in a project.

© Copyright 2012 by Venkitasubramanian Akshay

All Rights Reserved

Object-Oriented Framework for Healthcare Simulation

by
Venkitasubramanian Akshay

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Industrial Engineering

Raleigh, North Carolina

2012

APPROVED BY:

Dr. Stephen D. Roberts
Committee Co-Chair

Dr. Jeffery A. Joines
Committee Co-Chair

Dr. Yunan Liu
Committee Member

DEDICATION

To my parents, brother and Dr. Parasuram Balasubramanian

BIOGRAPHY

Akshay Venkitasubramanian was born on September 19th, 1988 in a quaint, little town of Neyveli, India to two bankers. Hence, he was brought up in a dynamic environment where they shifted places every 4 years. This change put him through 4 different school systems and helped him understand that his academic forte lied in mathematics and programming. In addition to this, growing up in diverse societies helped in cultivating his strong personal interests for social development. Combining his personal experiences with academic interest, he chose to study Electronics and Biomedical Engineering at Government Model Engineering College, Cochin where he graduated with FIRST class in June 2009.

During these four years, he actively became involved with the Young Indians (Yi) organization which provided a platform for students to work alongside leaders and entrepreneurs designing and implementing initiatives to improve their local communities. For his contributions to the organization, he was awarded Most Valuable Member of the Year (All India) in 2009 and was given invited to give a talk on the topic of social development and youth involvement at the prestigious Indian School of Business, Hyderabad. This is where he came in touch with late management professor Dr. C.K.Prahalad (University of Michigan-Ross Business School) and the following interactions with Dr. C.K.Prahalad taught him that technology is only a part of the healthcare problem and that an equally-important focus should be on operations strategy.

After graduation, he decided to pursue a series of developmental internships in Nigeria and India to understand the role of healthcare systems while he pondered over the choice of graduate program. His uncle, Dr. Parasuram Balasubramanian (CEO, ThemeWorks Analytics), who holds a doctorate in Industrial Engineering from Purdue University was a huge help and he finally chose the graduate program in Industrial Engineering at North Carolina State University.

At North Carolina State University, Akshay is the chapter president of Institute for Healthcare Improvement-NCSU and also a principal investigator on a behavioral study of

infection control practices. He is also an active member of Alpha Phi Mu, Institute for Industrial Engineers, Society for Health Systems, System Dynamics Society, and INFORMS. He hopes to pursue doctoral studies in management science focusing on public health systems development.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to his advisors: Dr. Stephen D Roberts and Dr. Jeffery A Joines for tirelessly working with him and instilling a culture of academic research. In addition to this, he is also grateful to them for supporting him through research and teaching assistantships. The author also expresses his thanks to Dr. Yunan Liu for his time and support in helping refine this work. Last but not the least, the author acknowledges Dr. Richard Wysk, Dr. Julie S Ivy, Dr. Reha Uzsoy and Dr. Michael G. Kay whose encouragement and advice helped him navigate graduate school.

TABLE OF CONTENTS

LIST OF TABLES.....	x
LIST OF FIGURES	xi
READING STYLE GUIDE	xvi
1. INTRODUCTION.....	1
1.1 THE POWER TO SEE	1
1.2 HEALTHCARE AND SIMULATION	2
1.2.1 <i>Timeliness of Care</i>	2
1.2.2 <i>Quality of Care</i>	3
1.2.3 <i>Errors in care delivery</i>	4
1.2.4 <i>Complexity in healthcare delivery</i>	4
1.2.5 <i>Value of healthcare services</i>	5
1.3 SIMULATION MODELING PARADIGMS	6
1.3.1 <i>Discrete Event Simulation (DES):</i>	7
1.3.2 <i>System Dynamics (SD):</i>	7
1.3.3 <i>Agent-Based Modeling (ABM):</i>	7
1.4 SIMULATION MODELING SOFTWARE	8
1.4.1 <i>Simulation software history- An overview</i>	8
1.4.2 <i>Healthcare Simulation Packages</i>	10
1.4.2.1 Medmodel™	10
1.4.2.2 Flexsim HC™	11
1.5 THESIS OUTLINE	11
2. LITERATURE SURVEY	13
2.1 OBJECT ORIENTED SIMULATION (OOS)	13
2.2 SYSTEMATIC REVIEW OF HEALTHCARE SIMULATION LITERATURE.....	16
2.3 PATIENT FLOW AND OPTIMIZATION	18
2.3.1 <i>Outpatient Scheduling</i>	18

2.3.1 Inpatient Scheduling	20
2.3.2 Emergency Department (ED) Simulations.....	21
2.3.3 Specialist Clinics.....	23
2.3.4 Physician and Healthcare Staffing	24
2.4 HEALTHCARE ASSET ALLOCATION	25
2.4.1 Bed Sizing and Planning.....	25
2.4.2 Room Sizing and Planning.....	25
2.4.3 Staff Sizing and Planning.....	26
2.5 HEALTH POLICY SIMULATIONS	27
2.6 MULTI-FACILITY SIMULATION MODELS	31
2.7 SIMULATION TUTORIALS.....	32
2.7.1 Modeling Tutorials.....	32
2.7.2 Implementation Tutorials.....	34
2.8 RESEARCH SCOPE DEVELOPMENT.....	36
2.9 CHAPTER SUMMARY	38
3. HEALTHCARE TOOLKIT DEVELOPMENT.....	39
3.1 SIMIO – A BRIEF INTRODUCTION	39
3.1.1 Base Classes.....	40
3.1.2 Building New Class.....	42
3.2 BASE CLASSES OF SIMIO-HC	43
3.3 FIXED GROUP	46
3.3.1 Patient Care (PC)	46
3.3.2 Patient Care with Lab Care Class (PC_Labcare).....	50
3.3.3 Hospital Workstation	54
3.3.4 Hospital Workstation with Lab Care Class (HWS_Labcare).....	58
3.3.5 Redirect Object Class (RO)	60
3.3.6 Waiting Room (WR).....	62
3.3.7 Registration Desk (RD).....	63
3.3.8 Statistics Collector Class (SC).....	66

3.4 NODE	67
3.4.1 Hospital Transfer node	67
3.5 TRANSPORTER CLASS	68
3.5.1 Hospital Transporter Class (HT).....	69
3.5.2 Hospital Worker (HW).....	73
3.6 ENTITY CLASS.....	74
3.6.1 Patient Entity Class (PE).....	74
3.6.2 Specimen Entity (SE).....	77
3.7 CHAPTER SUMMARY	78
4. COMPLEX INTEGRATED MULTI-FACILITY SYSTEMS SIMULATION.....	79
4.1 AN INTRODUCTION TO CIMS	79
4.2 BUILDING CIMS MODEL	80
4.2.1 Emergency Department Model	81
4.2.2 Oncology Clinic model	84
4.2.3 Laboratory Services.....	87
4.2.4 Radiology Services.....	89
4.2.5 Facility Integration	91
4.3 MODEL DEPLOYMENT.....	91
4.3.1 Experimentation Setup and Results	93
4.4 SIMULATION PROJECT LIFECYCLE.....	97
4.5 CHAPTER SUMMARY	100
5. CONCLUSIONS AND FUTURE WORK	101
BIBLIOGRAPHY	104
APPENDICES.....	149
Appendix A SIMIO Framework:	150
A.I Object-Oriented Programming Concepts	150
A.II SIMIO and OOP concepts	151

A.III Object Definitions	153
A.IV SIMIO Standard Library	158
A.V Additional Readings	159
Appendix B : SIMIO-HC Toolkit	161
B.I Base Class Overview	161
B.II Patient Entity	163
B.III Specimen Entity	167
B.IV Hospital Transfer node	168
B.V Redirect Object	169
B.VI Waiting room Object	173
B.VII Registration desk	177
B.VIII Patient Care Object	179
B.IX Patient Care with LabCare (PC_LabCare)	185
B.X Hospital Workstation	189
B.XI <i>Hospital Workstation</i> with Lab Care (HWS_Labcare)	200
B.XII Statistics Collection Sink:	207
B.XIII Hospital Transporter	209
B.XIV <i>Hospital Worker</i>	217
Appendix C : CIMS Model Development	220
C.I Emergency Room Model.....	220
C.II Oncology Clinic	228
C.III Laboratory Services	235
C.IV Radiology Services	241
C.V Multi-Facility Model.....	246

LIST OF TABLES

Table 3-1: Description of SIMIO Base Class	41
Table 3-2: <i>Patient Care</i> Object Label Definitions s	48
Table 3-3: <i>Patient Care</i> Object Resource State Definitions.....	49
Table 3-4: <i>Hospital Workstation</i> Resource States	57
Table 3-5: Agent Class Resource States	71
Table 3-6: <i>Patient Entity</i> Activity States.....	77
Table 4-1: Patient Care Treating & Occupied Resource State Averages	95
Table 4-2: Laboratory utilization breakdown	95
Table 4-3: Radiology Department utilization breakdown	95
Table 4-4: Percent Time statistics breakdown for Patient's activity	96
Table 4-5: Total Time statistics (in minutes) for patient's activity	97
Table B-1: <i>Patient Entity</i> Lists.....	163
Table B-2 : <i>Patient Entity</i> Function Table	163
Table B-3: <i>Patient Entity</i> States	165
Table B-4: Specimen Entity's States	167
Table C-1: ED Patient Arrival Table (<i>ED_ArrivalTable</i>).....	221
Table C-2: Data Table for Oncology Patients.....	230
Table C-3: Processed Specimen routing out expressions	240
Table C-4: Phase Duration Table for <i>HWS_Labcare</i> class in Radiology Department	244

LIST OF FIGURES

Figure 1-1: Abstraction Levels For Dynamic Simulation Paradigms.....	6
Figure 1-2: Block Diagram Of A DES Package [Adopted from Pidd and Carvalho(2006)] ..	9
Figure 2-1 Breakdown of Literature Survey Papers	37
Figure 3-1: Simio Base Classes	40
Figure 3-2: Utility as function of Hierarchy and Class Count	44
Figure 3-3: Class Hierarchy Tree of SIMIO-HC	45
Figure 3-4: Decision Making Structure for Specimen Collection	52
Figure 3-5: Utility Of a Class.....	54
Figure 3-6: Relationship between Class Hierarchy and Its Complexity	65
Figure 3-7: Utility of Class as a function of Complexity And Object Count	66
Figure 3-8- Routing Logic Flowchart Of Hospital Transporter.....	70
Figure 3-9 : Utility Function as a function of Run Speed and Complexity	72
Figure 4-1: Hypothetical Hospital Model	81
Figure 4-2: Emergency Department Conceptual Model	82
Figure 4-3: Oncology Department Conceptual Model	84
Figure 4-5- Treatment Patient Flow.....	86
Figure 4-4- Check Up and Follow Up Patient Flow	85
Figure 4-6: Treatment with Checkup Patient Flow.....	86
Figure 4-7: Laboratory Department Conceptual Model	88
Figure 4-8: Radiology Department Conceptual Model	90
Figure 4-9: Patient Time in System for 25 scenarios generated by OptQuest.....	94
Figure 4-10: K-N Ranking and Selection Best Scenario	94

Figure 4-11: Simulation Project Lifecycle.....	99
Figure A-1: Standard Library of SIMIO.....	158
Figure B-1: Patient Related References Storage.....	166
Figure B-2: Specimen Related Reference Storage.....	168
Figure B-3: Modified Process for <i>Hospital Transfer node</i>	169
Figure B-4: Process on Entering the Object	171
Figure B-5: Redirect Object Processing Logic	171
Figure B-6: Paperwork Batching Logic.....	172
Figure B-7: Results Batching Process.....	172
Figure B-8: Entity Trying To Enter <i>Waiting Room</i>	175
Figure B-9: Waiting Room Process-I	175
Figure B-10: Waiting Room Process-II	176
Figure B-11: When an Entity Balks.....	176
Figure B-12: Renege Process to evaluate renegeing conditions.....	177
Figure B-13: When an entity enters <i>Registration Desk</i>	178
Figure B-14: Processing Entities In <i>Registration Desk- I</i>	178
Figure B-15: Processing Entities in <i>Registration Desk- II</i>	179
Figure B-16: Upon Initializing Run.....	181
Figure B-17: When a seize request is evaluated	182
Figure B-18: When capacity of the object is allocated	182
Figure B-19: Upon Capacity Released	183
Figure B-20: Patient Treatment Process -I.....	183
Figure B-21: Patient Treatment Process -II	184
Figure B-22: Treatment State Monitor Process	184

Figure B-23: When a Patient Exits	184
Figure B-24: When an Entity Enters <i>pc_LabCare</i> -I.....	187
Figure B-25: When an Entity Enters <i>pc_LabCare</i> -II	187
Figure B-26: Process for providing secondary care.....	188
Figure B-27: Continue Treatment Process post visit to lab	189
Figure B-28 Results Arrive and Batch Process.....	189
Figure B-29: Run Initialization Process.....	191
Figure B-30: Seize Request Process	192
Figure B-31: Capacity Changed Process	192
Figure B-32: On Entering the Setup Buffer	193
Figure B-33: Performing Setup Activity- I.....	193
Figure B-34: Performing Setup Activity-II.....	194
Figure B-35: Specimen Batching Process.....	195
Figure B-36: Creating and Batching Parent <i>Specimen Entity</i> Process.....	195
Figure B-37: Logic of Processing Activity-I	196
Figure B-38: Logic of Processing Activity-II.....	197
Figure B-39: Upon exiting processing buffer	197
Figure B-40: On Entering Post-Production Buffer	198
Figure B-41: Perform Post Production Activity Process	199
Figure B-42: Post Production Buffer Exited.....	199
Figure B-43: Initialization of the <i>HWS_LabCare</i>	202
Figure B-44: Implementing Failure	202
Figure B-45: On Failed Process.....	202

Figure B-46: On Repaired Process	203
Figure B-47: On Entered Setup Buffer	203
Figure B-48: Perform Setup Activity.....	204
Figure B-49: <i>Specimen Entity</i> Create-I.....	204
Figure B-50: <i>Specimen Entity</i> Create-II.....	205
Figure B-51: Processing Buffer Exited.....	205
Figure B-52: Batching Addition to Perform Processing Activity Process	206
Figure B-53: Results on Entering the Results Station	207
Figure B-54: Waiting for Results Process	207
Figure B-55: Collection of Statistics.....	209
Figure B-56: On creating the transporter	211
Figure B-57: On Initializing the Simulation Run	211
Figure B-58: Rider Loading Process.....	212
Figure B-59: Rider Loaded	212
Figure B-60: Rider Unloading Process	213
Figure B-61: Deciding if the Node is Destination Node-I.....	213
Figure B-62: Pickup or Drop-off	214
Figure B-63: Parking Decision Process	215
Figure B-64: Seize Assistance Resources.....	216
Figure B-65: Release Assistance Resources	216
Figure B-66: Worker Capacity Changed Process	217
Figure B-67: Upon Worker Capacity Released	218
Figure B-68: Initialization Of Worker	218

Figure B-69: Implementing Fixed Routing.....	219
Figure C-1: Emergency Department Process Layout	221
Figure C-2: Transferring patients to different entrances.....	223
Figure C-3: Diverting Ambulance if waiting room is full	223
Figure C-4: Refusal of Service if Waiting Room is full at <i>Registration Desk</i>	224
Figure C-5: Custom Treatment logic	226
Figure C-6: Sample Warning for failing transfer from a node	228
Figure C-7: Oncology Clinic Process Layout.....	230
Figure C-8: Splitting entities between two transfer node	231
Figure C-9: Modeling Nurse Triage	233
Figure C-10: IV Setup Process	234
Figure C-11: Post Treatment Process to remove IV line	234
Figure C-12: Laboratory Patient Flow	236
Figure C-13: Patient States Assignments for eliciting patient choices	238
Figure C-14: Radiology Department Model Layout.....	242
Figure C-15: Deciding random patient entity assignments for radiology service	243
Figure C-16: Patient behavior post service completion.....	245

READING STYLE GUIDE

Certain formatting styles have been adopted in this paper to facilitate easy reading.
All the fonts and their keys are explained below.

Legend	Key
<u>Processes</u>	Times new roman, 12 font, underlined represents SIMIO processes
<i>ELEMENTS</i>	Courier New, 12 font, ALL CAPS Italics represents elements
EventNames	Courier New, 12 font, First Letter Caps, represents elements
<i>States</i>	Courier New, 12 font, Sentence Case, Italics represents state variables
Functions	Cambria, 12, represents function definitions
<i>Expressions</i>	Cambria 12, Italics represents expressions
<i>Properties</i>	Times new roman, 12 , Italics represents properties as well as drop down selections
<u>List Names</u>	Courier New, 12 font, Sentence Case, Underlined represents lists
“Output Node”	Times new roman 12, Sentence case represents items of external view
“Concept”	Times new roman 12, Sentence case, Italics represents concepts
BASE CLASS	Times new roman 12, Bold, All Caps represents SIMIO Base Classes
Standard Library	Times new roman 12, Bold Sentence case represents SIMIO Standard library
<i>Hospital Worker</i>	Times new roman 12, Bold Italics represents SIMIO HC Base class
<u>HW LC Xray</u>	Times new roman 12, Italics, Underlined represents SIMIO-HC children class (not different from base as they vary by very few parameters)
STEPS	Copper Plate Gothic Light, 12, lowercase, represents SIMIO steps

Chapter 1 Introduction

1.1 The Power to See

Just over a decade ago, the medical community was confronted by the Institute of Medicine (IOM) *To err is human* report (IOM, 2000) and *Crossing the Quality Chasm* Report (IOM, 2001). The reports attributed major shortcomings of the US healthcare system to the growing complexity of medical practice and technology amplified by a poor organization healthcare delivery system. In a nutshell, these reports shifted the focus of medical community from just treating the disease to treating the entire person which ushered in the patient safety revolution. The core focus of this ongoing revolution is to improve the patient safety, quality, and effectiveness of the healthcare system.

A decade into the patient safety movement, the results are discouraging. A recent study carried out by the Institute of Healthcare Improvement (IHI) indicates the number of adverse events is maybe 10 times the reported value (Classen et al., 2011). Errors in healthcare has become the eighth leading cause of death in the USA (Dhillon, 2008). Moreover, current healthcare spending exceeds 17% of the GDP, which has further exacerbated the current healthcare systems condition creating a sense of urgency for developing efficient healthcare system models (Kaplan and Porter, 2011),.

In order to develop an efficient and effective healthcare system, we need to develop a systems perspective, which is easier said than done. Let's consider the field of medical science which has undergone radical change over the last century. For over 2000 years, physicians believed *bloodletting* would cure their patients. It was not until the late 19th century that the medical providers accepted that this method was actually worsening their patients. The invention of the microscope in the late 1600s coupled with the discovery of x-ray radiations in the late 1800s revolutionized the field of medicine. For the first time in human history, scientists were able to observe microorganisms and conduct a systematic study to understand the underlying dynamics of diseases which gave birth to the concept of

Evidence based Medicine and what we today call “Modern Medicine” (Claridge and Fabian, 2005). In short, visualizing the system is critical in developing a systems perspective. The development of a systems perspective helps us understand the structure of the system which is responsible for the behavior of the system. Thus, in order to fix the behavior of the system, we need to fix the structure of the system and simulation modeling will help us just do that.

1.2 Healthcare and Simulation

McGlynn et al. (2003) carried out an empirical investigation of the quality of healthcare delivered to adults in the USA and found out that only $54.9 \pm 0.6\%$ adults received recommended care. Furthermore, their study also reported huge variation in the quality of care depending on patient's condition. In fact, the literature on healthcare is laden with articles like these that emphasize on the importance of the systems view of healthcare problems (Barker et al., 2002; Berwick, 2002; Cook, 2001; Fordyce et al., 2003; Wears, 2000; Wears, 2003; Young et al., 2004). The dissatisfaction with healthcare service can be broadly classified in to five categories: Timeliness of Care, Quality of Care, Errors in Care Delivery, Complexity of healthcare system , and Cost effectiveness of care. These problems are briefly discussed in the next sections and we suggest how simulation and systems thinking can assist in fixing them.

1.2.1 Timeliness of Care

Invariably, one of the main causes of dissatisfaction among patients is the unprecedented waiting times for patients, as highlighted in IOM (2001). This unpredictability in waiting time can be frustrating as well as affect the effectiveness of care for the patient and the care provider. For the patient, the waiting time may cause him/her to come in contact with other infections or various environmental factors may interact to further complicate the patient's condition. As for the delay in treatment, this delay factor can reduce the effectiveness of the treatment and medication delays are an example of this type of delay. For the care providers, this unpredictability in waiting times produces an unpredictable workload for them. It has been well-documented in the service fields such as call centers, the

importance of achieving time-stable performance of workload (or utilization) to provide consistent quality of service (Feldman et al, 2008). Thus we can extend the same concept to healthcare service. Furthermore, empirical research results presented by Kuntz, Mennicken, and Scholtes (2012) show that the utilization of care providers should be moderate (i.e. 86-87%) to minimize the chance of adverse event and indirectly, mortality rate.

Simulation modeling has been used for over 60 years in the service industries to appropriately staff as well predict customer flows in order to maximize utilization of clinics while providing best service. We intend to extend the very same methodology to healthcare systems in order to improve its timeliness of care factor.

1.2.2 Quality of Care

In the healthcare field, patient satisfaction should be the fundamental measure of quality of care (Berwick, 2002). Quality problems in healthcare services in both managed care and fee for service care systems can be divided into three categories: underuse, overuse, or misuse. “*Underuse*” is defined as the failure to provide healthcare when it would have provided desirable outcome. “*Overuse*” is defined as the provision of service that can produce an undesirable outcome. “*Misuse*” is defined as choosing appropriate treatment but failure to avoid preventable complications. In short, the healthcare delivery process is not well understood which makes quality of healthcare also a systematic problem (NRHQ, 1998).

The first step in solving any problem is to understand it and represent it appropriately. Simulation modeling as a tool has been used in the Operations Research (or Management Science)¹ field to do the very same task of representing the problem in a language that we can comprehend. Moreover, visualization of simulation model has helped in effective problem solving where there are diverse stakeholders such as in the healthcare field.

---

¹ Will be abbreviated in the thesis as OR/MS

1.2.3 Errors in care delivery

In the United States, at least 44,000 people and maybe as many as 98,000 people die due to preventable medical errors (IOM, 2000). To put these numbers in perspective, the mortality in hospitals due to preventable medical errors exceeds mortality due to motor vehicle accidents, breast cancer, and AIDS. The most often cited reason for such errors is decentralized structure of healthcare service or the non-existence of systematic approach to delivery of healthcare. In layman terms, a patient does not die because one medical person made a mistake but because one person's mistake sets off a chain reaction in the patient's care chain compounding errors. Thus, the fatal errors are generated by faulty systems whose processes and conditions inhibit the system's stakeholders from preventing it.

In order to fix errors, we should perform systematic analysis of the causes so that we fix the system. If this approach is not taken, fixing the errors will only further complicate the system. Simulation modeling can play a critical role in analyzing the system and eliciting errors. Moreover, with today's computing power and advances in simulation languages, we can perform "what-if" analysis to yield valuable insights in to the system's behavior.

1.2.4 Complexity in healthcare delivery

The Boeing 777 was considered to be too complicated to be flown by human pilot until a veteran US air force pilot extended the remarkable simple concept of a checklist to commercial aviation. Today this concept has been extended to the medical world and has been hailed in the medical community as a remarkable breakthrough as it has reduced the surgical errors by a third worldwide (Gawande, 2010). The checklist provided surgeons a systematic way to deal with the complexities of surgical procedures by standardizing it. Moreover as complexity increases, latent errors also survive longer in the system and in the case of healthcare, this translates to fatalities (Wears, 2003).

The only way to deal with complexity is to break it down in to simple steps that human beings can understand. It may not be possible for us to observe certain functions of

the healthcare system for extended periods of time making simulation an attractive substitute. Furthermore, simulators have been used extensively in the aviation industries to formulate and test their checklists making an even stronger argument to use simulation models in the field of healthcare given both are high precision industries.

1.2.5 Value of healthcare services

The classical economics definition of value is the intersection of purchaser's perception of goods or service and the cost which the purchaser is willing to pay for it. For example, if you paid \$10 for a slice of pizza at a restaurant compared to spending the same money on chef's special pasta at the restaurant, the latter would feel more satisfying. If we further try to extrapolate this concept to healthcare services, we can express the value of healthcare service as the cost effectiveness of care received.

$$Value = \frac{Quality}{Payment}$$

Given the fact that quality of healthcare services is at best ambiguous as it is heavily dependent on individual factors such as patient's age, caregiver's experience among multifarious factors, this equation can be considered as not well defined (HFMA, 2011).

If we look to other industries, they handled variability in quality by designing processes using tools of lean and six sigma methodologies (Juran, 1992). Nonetheless, simulation modeling has been integrated in the development of these processes and has proved to be extremely effective since managers get to understand the dynamics of any changes made and to put in place effective change management policies (El-Haik and Al-Aomar, 2006). With the increasing trend of the healthcare field adopting lean and six sigma culture to standardize healthcare quality, adoption of simulation modeling methodology is also going to increase (Young et al., 2004). Moreover, the process of building a simulation model enforces a systematic review of the process in order to quantify the value of healthcare service.

1.3 Simulation Modeling Paradigms

Among the various simulation modeling paradigms, four modeling paradigms stand out in the healthcare literature (Fone et al., 2003; Katsaliaki and Mustafee, 2011): Discrete Event Simulation (DES), System Dynamics (SD), Monte Carlo Simulation (MCS), and more recently, Agent based model (ABM). With the exception of Monte Carlo Simulation, all other aforementioned paradigms are dynamic modeling paradigms and can therefore capture the dynamics of complex systems.

Among the dynamic modeling paradigms, discrete event and system dynamics take a top down approach where you define the rules of the physical world and the entities are modeled in the system. Agent based models take a bottom up approach where each agent (or entity) logic is defined and these intelligent agents interact with each other in the world based on these rules to develop system behavior. For a more detailed comparison of these paradigms, refer to Borshchev and Filippov (2004).

Figure 1-1: Abstraction Levels For Dynamic Simulation Paradigms

The choice of modeling paradigm is also based on the nature of the problem. If the problem requires a macro approach, then system dynamics will be the method of choice. On the other hand, if you have a problem that requires operational level detail with supporting process level data, discrete event modeling paradigm will be the most appropriate. If you have only individual level data (for example: person's immunity, contact rate) and want to incorporate *Artificial Intelligence (AI)*, then agent based modeling is ideally suited. Figure 1-1 correlates modeling paradigms with the level of abstraction.

1.3.1 Discrete Event Simulation (DES):

Discrete Event Simulation (DES) technique is the most popular among the simulation modeling techniques (Robinson, 2005). The roots of this modeling paradigm can be traced back to the 1950s and offers a “transaction-flow world view” of the system. In other words, we visualize the system as discrete units of traffic (or transaction) flow from point to point competing for system's resources. The model's state changes only at discrete points of time called event times. The event times can be randomly generated and it is possible for multiple units of traffic to be manipulated simultaneously. For a better understanding of DES, the reader may refer to Schriber and Brunner (2009).

1.3.2 System Dynamics (SD):

Developed by Jay.W.Forrester in the 1960s, this simulation paradigm helps us understand the behavior of complex systems. This simulation paradigm can capture the internal complexity of the system in order to represent them as feedback processes to explain the behavior of an entire system. The system dynamics models capture the nonlinearity using feedback (or causal) loops and stocks and flows. For a deeper understanding on the subject, the reader may refer to Sterman (2000) .

1.3.3 Agent-Based Modeling (ABM):

Agent-Based modeling simulation paradigm is relatively young as compared to the DES and SD. The popularity of this simulation paradigm is low in OR/MS communities as

compared to computer science, social science, and economics fields (Siebers et al. 2010). Nonetheless, the problems tackled by this paradigm are unique compared to SD or DES and has a wide range of applications in real world problems.

ABM is a complex system composed of simple autonomous decision making agents who follow simple behavior rules defined by the modeler. The system properties emerge as a result of the interaction of these agents with each another and the environment which is of interest to the modeler. These agents mimic their counterparts in real world and thus make it an attractive method for modeling human behavior. For a more detailed tutorial on this methodology, refer to Bonabeau (2002).

1.4 Simulation Modeling Software

Among the OR/MS tools available today, Simulation has been acknowledged to be the most popular technique. (Fildes and Ranyard, 1997; Jeffrey and Seaton, 1995; O’Kane, Spenceley, and Taylor, 2000). Simulation has become an indispensable tool in addressing a wide range of problems posed by complex systems in a cost effective way. The growth in power and popularity of simulation methodology is highly correlated with developments in computing world.

1.4.1 Simulation software history- An overview

Although simulation did exist in pre- digital computing era, it was limited to academic circles. The applications of methodology were limited due to high level of complexity involved in analog computing machinery which proved to be cost ineffective. The onset of the digital computer revolution after World War II reshaped the prospects of computer simulation as the methodology. Gordon E Moore, co-founder of Intel predicted the computing power nearly doubled every 18 months which created an explosion of programming languages. This lead to development of more readily accessible and powerful simulation languages such as FORTRAN based SIMSCRIPT, ALGOL based SIMULA, Geoffrey Gordon’s GPSS among many others. GPSS process interaction view set the

standard for the forthcoming generation of simulation languages such as GASP, SLAM, SIMIAN/ARENA, SIMIO, etc. which made simulation a powerful, useful and cost effective OR/MS technique. Detailed accounts of aforementioned developments are presented in these papers if the reader wishes to explore this topic further; B.W. Hollocks (2006), Goldsman, Nance, and Wilson (2010), Ross (1967) and S. Robinson (2005).

Today, simulation has gained formal acceptance in every academic field. In fact, the field of climate science is entirely based on simulation models, even though some of them are difficult to validate. Furthermore, there is plethora of simulation languages and packages to address a wide spectrum of problems. In a recent issue *OR/MS today* magazine, Swain (2011) presented a survey of simulation software packages which boasted 55 simulation products from 29 vendors.

Figure 1-2: Block Diagram Of A DES Package [Adopted from Pidd and Carvalho(2006)]

Given the diversity in simulation packages, we can establish standards to define a good simulation package. A good simulation package must boast a wide range of modeling

tools which comprise of simulation objects with predefined methods and variables², a graphical library to visualize the system and integration capabilities in to corporate systems such as ERP systems, databases and spreadsheets. Secondly, a simulation package must be capable of executing simulation models with provisions for debugging, run control, and meaningful animations. Finally, the simulation package must have capabilities to perform analysis such as model experimentation and optimization, output analysis and interactive graphics to represent the results. The block diagram presented in Figure 1-2 accurately represents building blocks of a good simulation package.

1.4.2 Healthcare Simulation Packages

Although any general simulation language can be used to model a healthcare system, it may be difficult for the modeler to model given the inherent complexity embedded in healthcare services. This concern probably spurred the development of specialized healthcare simulation packages to improve the cost effectiveness of employing simulation. In order to be precise, we define a healthcare simulation package as a simulation package consisting of object libraries and visualizations to model healthcare related situations with ease. Under this definition, Medmodel^{TM} and Flexsim HC^{TM} are the two popular software packages that can be categorized as healthcare simulation packages.

1.4.2.1 Medmodel^{TM}

Developed in the late 1990s by Promodel Corporation, Medmodel package included functionality of ProModel (the general simulation package by Promodel) to match lab results to patients, preempting resources, staff schedules, subroutines and an extensive 2D healthcare symbol library. Moreover, this package had a drag and drop feature which made it easy to use. The package also has output analysis and optimization functionality along with spreadsheet integration. A detailed tutorial on Medmodel^{TM} is presented in Harrell and Lange (2001).

---

² Methods represent processes and variables are a set of states, elements, events, lists, object properties

1.4.2.2 Flexsim HC^{TM}

Flexsim HC^{TM} is the healthcare extension of Flexsim, general simulation package. The Flexsim HC^{TM} provides 3D custom objects to represent patient queuing, processing and resources with an in-depth focus on providing statistics in a graphically interactive way. Furthermore, Flexsim also has event triggers that allow the user to specify custom behavior from a drop down list when the corresponding system event is triggered. This construct can be used to develop preempt logic, among various other customized logic. It also provides a flow chart view to allow the modeler to represent the model logic without getting bogged down in the details of 3D figures. In terms of output analysis, Flexsim has a built-in experimentation model as well as provides an add-in for Optimization and statistical output analysis. Flexsim has constructs to support Microsoft Excel integration, movie creation and interactive graphical presentations. For further information, refer to

<http://www.flexsim.com/flexsim-healthcare/>

1.5 Thesis Outline

Although, these healthcare simulation packages and paradigms have existed for some time, their application to real world healthcare problems has been only around 8% (Eldabi, 2009). In the same timeframe simulation has emerged as a popular decision support in the domains of manufacturing and services industries (Kuljis, Paul, and Stergioulas, 2007). This suggests that there are problems associated with current packages as well as paradigms which form the motivation behind this body of work.

Chapter 2 presents an extensive literature survey to study simulation software and healthcare industry. In order to understand problems associated with simulation packages, the literature survey spanning the domain of object-oriented simulation which covers the evolution of simulation packages. To cover the healthcare aspect of this problem, we conduct a systematic review of healthcare simulation papers presented in *Winter Simulation Conference*.

The next phase of this thesis describes solutions to close the gaps in healthcare simulation modeling elicited in previous chapters. In Chapter 3, we present the SIMIO-HC Toolkit along with the simulation development guidelines for developers and modelers. After the development of the tools, we address the issue of implementation of simulation projects in healthcare environment (Chapter 4). To demonstrate this, we apply the SIMIO-HC toolkit to build a Complex Integrated Multi-Facility System (CIMS) simulation model. We also explore the nature of simulation modeling engagements in this chapter. Finally, Chapter 5 presents the limitations of the framework and the work presented in this thesis. It also presents future directions for research in the domain of healthcare simulation.

In order to make this document complete, we have included appendices. A detailed note on SIMIO's Framework relevant to this thesis is presented in Appendix A. In Appendix B, we document the SIMIO-HC object development and in Appendix C, we provide a step by step approach to construct the hypothetical hospital model presented in Chapter 4.

Chapter 2 Literature Survey

The aim of this literature review is twofold: the first aim is to explore the domain of object oriented simulation and the second one is to perform a systematic review of the healthcare simulation domain. The ultimate goal of this review is to help the reader understand the scope of object-oriented frameworks for healthcare simulation.

2.1 Object Oriented Simulation (OOS)

Concepts of OOS first appeared in the Programming Languages journal in the paper *SIMULA- an ALGOL based simulation language* authored by Ole-Johan Dahl and Kristen Nygaard (Dahl and Nygaard, 1966). The landmark paper introduced SIMULA's basic concepts of classes, inheritance, polymorphism and run time (or dynamic) binding which are now considered as basics of Object oriented programming (OOP) and, by implication, OOS. The paper presents a very consistent framework and implementation of these concepts in ALGOL.

Roberts and Dessouky (1998) present a comprehensive review of the OOS field. They present a brief introduction to the concept of OOS and focus on polymorphism and dynamic binding concepts. They provide a detailed discussion on simulation libraries using C++, Self, Smalltalk, CLOS, Eiffel, Modula-3, objective-c and Java. The paper also provides a visual as well as textual comparison between procedural and OOS. Finally, the paper provides a list of future issues with OOS which includes OOS approaches to distributed and complex modeling, modeling and maintaining OOS systems, and validation of OOS models.

One of the key advantages of OOS is the ability to use multiple processors to be executed in parallel which allows for a higher utilization of processing and memory resources making the simulation cost-effective. This concept is coined as Object Oriented Parallel Simulation (OOPS) and is discussed in Lomow and Baezner (1991). The paper also provides a comparison between OOP, OOS, and OOPS. In addition to the concepts of OOP and OOS, it also discusses the key concepts of parallel simulation which are determinism,

scalability and portability. A tutorial using Sim++ of all the above concepts is presented in this work.

For the reader to fully appreciate the elegance of OSS, an object-oriented view of the world is presented in Pegden (2010). This discussion presents the three views of simulation: Event, Process and Object. In the event orientation, the system is viewed as a series of instantaneous events that alter the state of the system over time. Alternatively, in the process orientation view of the world, the entity flow is described in series of process step which change the state of the system over time. Finally, in the object-oriented view we see the world composed of objects that interact to alter the dynamics of the system. The object oriented view of the world is tandem with the way we view our world, which makes it easier to model complex problems.

On the topic of development of OOS languages, Bischak and Roberts (1991) present a comprehensive outlook on the central issues. The paper discusses world views and explains the OOS world view which is centered on an object. They also discuss scope, visibility and object creation and destruction and polymorphism in the context of OOS. In addition to this, they describe topics in object communication, problems and potential for OOS and controversies surrounding OOS which refer to treatment of processes, control of simulation time, and statistics collection.

Following up on previous work, Joines and Roberts (1997) present a tutorial on the development of OOS in C++. The paper presents a differentiation between Object-oriented versus Object-based concepts as well as topics on composition where a new class of objects can be created by combining existing classes. The paper presents an example using YANSL (Yet ANother Simulation Language) and also covers the basic principles of YANSL. Joines and Roberts (1998) builds on their previous paper where the authors differentiate between the procedural style and the object-oriented style of simulation. A user of object oriented simulation is a language designer in their own way. As they develop the simulation model,

they modify objects which can be reused and in their own respect have developed a new object library.

J. A. Joines and Roberts (1998) is a one stop reference on OOS. The chapter covers topics from Object oriented thinking to implementation of these concepts using YANSL (Yet ANother Simulation Language). In addition to all the basic concepts of OOS, it also discusses on the development of OOS languages which introduce the concept of *frame* and *framework*. A “*frame*” is a collection of classes that provide a level of abstraction for the OOS language and “*framework*” is a collection of classes that provide the language a specific modeling ability.

Cubert and Fishwick (1998) explore the development of OOS in different simulation paradigms which include Finite State Machine (FSM), Functional Block Model (FBM), Equation Constraint Model (EQN) and System Dynamics (SD). The paper also explores the concept of multi-modeling and simulation application framework which is utilized in SIMIO. The paper also provides a detailed reference of object-oriented concepts to integrate model geometry and dynamics as well as the importance of visual elements to model building. The paper also discusses elements of OOS which include translator, simulation engine, and distributed model repository.

An earlier examples which is similar to this thesis is demonstrated in Malloy and Chen (1998) who extend **SIM**ulation in **P**roduction, **L**ogistics and **E**ngineering (SIMPLE++) to model Personal Communication Service Network (PCS). The authors extended the SIMPLE++ base object library to create an object library for modeling PCS networks. Another example of the usefulness of OOS in healthcare environment is described in the Birch and Cook (1991) where a simulation to model of a hospital’s bed resource utilization was developed using SMALLTALK, an OOS language. The authors concluded that the visual user interface helped in gaining a higher user acceptance. Moreover, the administration did a periodic review of the simulation model and the changes were implemented with ease given the OOS nature of the language. The paper also described the performance limitations

of the computer by reducing the scope of the simulation model which should be irrelevant given today's computing power.

The work presented in this thesis is a framework of healthcare objects; a collection of classes that can model healthcare operations. The classes were developed by extending classes from SIMIO's bases classes. A special emphasis was placed on visualization given that simulation model's credibility is derived from the engaging stakeholders. In addition to this, a framework's utility was also factored in to the development which is influenced by the modeling capability and extensibility of the classes.

2.2 Systematic Review of Healthcare Simulation Literature

The field of healthcare simulation has exploded in the past decade to virtually include every segment of the healthcare industry. The simulation paradigms employed by these models have diversified from a handful to many different methodologies owing to a rise in computing power as well as the availability to user friendly software which has reduced the technical nature of modeling.

One of the earliest systematic reviews of the healthcare field was carried out by England and Roberts (1978) which reviewed ninety two simulation models and reported limited success of application of simulation models. Fone et al. (2003) performed a subsequent literature review from the year 1980-1999 which assessed the value of computer simulation modeling in healthcare. The paper classified literature into five categories: hospital scheduling and organization, infection and communicable disease, cost of illness and economic evaluation, screening and miscellaneous. The publication concluded that despite the growth of literature in the domain, limited evidence of implementation was found. Another contemporary review during this era can be found in Jun, Jacobson, and Swisher (1999) but with a focus on DES on healthcare clinics. The review divides the papers in to three sections namely patient flow, allocation of resources and future directions.

Building on the previous work, Jacobson, Hall, and Swisher (2006) presented a much more detail literature review of the DES packages in healthcare systems over the past forty years. The review divides published work in two main categories which are patient flow optimization and healthcare asset allocation. The review also has a section which deals with recent advances in modeling of healthcare systems. The main conclusion of their literature survey indicated there were few publications in the domain of complex integrated multi-facility systems or those that looked in to inter-departmental relationships. Their suggestions also included that the focus of DES packages should be on optimization add-ons, increased visualization, and object-oriented paradigm.

Günel and Pidd (2010) performed a systematic survey of DES literature and organized papers on the modeling detail required for accident and emergency units, outpatient and inpatient clinics. They also covered other types of hospital units and whole hospital simulations in their study. Their study also concluded that most studies are unit-specific and facility-specific (i.e. customized for hospitals). They also report that literature is still vague on model implementation and use. A more general literature survey is presented by Katsaliaki and Mustafee (2011) where they extended the scope of modeling and application areas. Their study also deals with the topics of funding, implementation, and model use in the healthcare domain.

Our literature review is survey of modeling techniques used in the healthcare domain. Since, the domain of healthcare simulation is expanding at a rapid rate, following a database search methodology will be flawed as we will miss landmark papers in simulation (Brailsford et al., 2009). Thus, we chose the *Winter Simulation Conference* which is a premier forum for simulation and chose to review the papers presented under the healthcare application section from the year 1997-2010 given their availability on the conference site.

The papers were still classified following the Jacobson et al. (2006) areas with two more areas added to account for papers that represented the larger set of healthcare problems

and simulation tutorials. Thus, the areas specified below are explored in the rest of the chapter.

1. Patient Flow and Optimization
  - a. Outpatient Scheduling
  - b. Inpatient Scheduling and Admission
  - c. Emergency Room Simulation Models
  - d. Specialist Clinics
  - e. Physician and Healthcare Staff Scheduling
2. Healthcare Asset Allocation
  - a. Bed Sizing and Planning
  - b. Room Sizing and Planning
  - c. Staff Sizing and Planning
3. Health policy modeling
4. Multi-facility modeling
5. Tutorials
  - a. Modeling tutorial
  - b. Implementation tutorial

2.3 Patient Flow and Optimization

A highly efficient hospital will be providing the a high quality of service for patients, as measured by patient related metrics such as patient waiting time, patient flow time, and patient service refusal rates. Patient Flow and optimization studies focus on improving the patient satisfaction by optimizing these patient related metrics by varying appointment scheduling systems, hospital resource allocation methodology, and reorganization of care processes to better serve the patient.

2.3.1 Outpatient Scheduling

Outpatient clinics serve specialized patient needs, where patients can be classified into categories of care. In order to serve them, there is limited number of physician hours. These problems deal with effective allocation of this physician time in order to maximize patient satisfaction.

Guo, Wagner, and West (2004) presented a patient centric simulation approach to outpatient scheduling. They detail their development of the Patient Scheduling Simulation Model (PSSM) which can handle the four determinants of an outpatient department: External demand for appointments, Supply of provider slots, Patient flow logic (and accounts for internal generated demand) and a Scheduling algorithm. This paper also presents analysis strategies for scheduling algorithms which are extremely important.

Giachetti et al. (2005) present a discrete and continuous time simulation approach to analyze appointment scheduling at a dermatology clinic. The discrete event simulation model analyzed and created patient schedules, and the system dynamics model investigated behavioral factors behind no show patients. This paper is a good example on how to combine and use discrete and system dynamics methodologies to analyze a system. This paper also provides an excellent summary on appointment scheduling policies. Giachetti (2008) investigates appointment scheduling policies for reducing patient wait time using a system dynamics model with queuing theory. This paper proves that when the no show rates are high, the system is operating below optimal capacity and suggests that pooling of appointment types is a better alternative to overbooking.

Wijewickrama and Takakuwa (2005) develop a special purpose data generator to validate their outpatient scheduling model for an internal medicine department. The paper also investigates the performance impact of realistic environment factors (e.g. no shows, variance in consultation time and variance of walk-in) on the system. Wijewickrama and Takakuwa (2008) extend the study of outpatient systems across departments and then relax the assumptions from their previous paper. The study concludes that the patient sequencing and appointment rules are still significant factors that impact patient satisfaction. Both the papers are excellent references for types of statistical analyses that can be performed on outpatient scheduling systems. Helm, Lapp, and See (2010) present a genetic algorithm approach to outpatient appointment scheduling and apply pareto efficiency charts to evaluate GA solutions and the tradeoffs between patient performance metrics. The study assumes that

the hospital is operating in steady state (or under normal condition) which might not be true always.

2.3.1 Inpatient Scheduling

Operating Rooms (OR) or Surgical units are considered to be one of the most expensive resources in a hospital. Thus, it must be allocated to the patients in a way that will increase its utilization rate while minimizing patient waiting times. Given the multifaceted nature of such problems, simulation has evolved to be a popular approach to solve them.

Lowery and Davis (1999) develop simple OR model using MEDMODEL software to evaluate block scheduling policies. Ferrin et al. (2004) develop incentives that can be analyzed using a simulation model which are: Case Cart Completion, Pre Admission Testing and Room Turnaround time (includes setup and cleanup time). Denton et al. (2006) present a monte carlo simulation model and applied simulated annealing procedure to multi OR surgical scheduling based on data collected from the Mayo Clinic. Huschka (2007) perform a bi-criteria evaluation OR scheduling problem where they analyzed the effects of competing objectives of mean patient waiting time versus overtime in the OR. The paper results suggest that the scheduling efficiency is highly sensitive to surgical mix. Ferrand, Magazine, and Rao (2010) study the allocation of OR room policies for elective versus emergency surgeries where they compare fixed and flexible policies.

Inpatient Scheduling heuristics are also a focus of research in the simulation community. Iser, Denton, and King (2008) analyze the relationship between OR scheduling and Post Anesthesia Care Unit (PACU) using heuristics. This paper can also serve as an excellent reference for those interested in heuristics. Arnaout and Kulbashian (2008) present a heuristic for OR scheduling which is derived by modeling OR as a parallel machine scheduling problem with sequence dependent setup times with an objective of minimizing makespan.

There are also other performance metrics on which efficient inpatient schedules can be generated. Reindl et al. (2009) develop a simulation model using SLX tool for a cataract surgical suite where they minimize the patient waiting time. The paper presents a comprehensive analysis of performance metrics for OR scheduling. Stanciu, Vargas, and May (2010) present a revenue based approach to OR scheduling where the aim is to maximize revenue per unit time. The paper modifies the EMSRb algorithm to set near optimal protection level for each patient class. Pearce et al. (2010) develop a DES model to analyze the dynamics of preoperative room and its effect on OR. They also suggest a novel agent based modeling approach to improve the accuracy of the simulation model.

2.3.2 Emergency Department (ED) Simulations

Emergency departments are on the frontline of healthcare service delivery. These services are critical to the public health infrastructure and a supply of patients for the hospital. The demand for emergency services is unpredictable. Nonetheless, the services they provide are time critical where excessive waiting times could prove fatal. ED departments respond to these challenges by experimenting with their staffing levels and patient flow. This makes them excellent problems to be solved using simulation, different scenarios and experiments can be examined without causing harm to anyone.

Rossetti, Trzcinski, and Syverud (1999) develop a staff scheduling simulation model in ARENA and is an excellent reference for verification and validation techniques for healthcare simulation problems. Baesler, Jahnsen, and DaCosta (2003) developed simulation model to estimate the maximum capacity of an ED using ARENA by estimating the patient's time in system. Based on the previous stage's result, design of experiments was conducted to compute minimum number of staff required to serve the maximum demand. Centeno et al. (2003) develop a simulation model and superimposed an Integer Linear Programming model to optimize staffing levels in ED.

Most of the ED simulations are planning tools whereby they are used for testing new and innovative policies before implemented in practice. Samaha, Armel, and Starks (2003) developed a simulation model in ARENA to test new ED policies to estimate the one that would reduce the length of stay (LOS). The paper developed a *fast track* policy to free beds for critically ill patients by expediting service for non-critical patient. Mahapatra et al. (2003) test a triage system for categorizing ED patients using a simulation model developed in ARENA in order to estimate patient waiting times in the ED. M. J. Miller, Ferrin, and Szymanski (2003) test six sigma policies using a simulation model of the ED employing the EXTEND simulation package. Davies (2007) tests the controversial “see and treat” policy versus “see” and “treat” policy using a SIMUL8 simulation package, where the former policy outperforms the latter one. The paper also discusses the problem of convincing the system’s stakeholders using simulation model. Ruohonen, Neittaanmaki, and Teittinen (2006) test the effect of triage team on the ED which according to their simulation model increases the ED efficiency by nearly 25%. Khurma, Bacioiu, and Pasek (2008) study the effect of new lean policies using a DES model and estimate savings in a future time horizon. Kolb et al. (2008) develop a simulation model in ARENA to test a five buffer concept which aims at reducing the overcrowding in the ER. Miller, Ferrin, and Shahi (2009) and Holm and Dahl (2010) estimate the impact of patient surge on ED operating efficiency.

ED simulation projects are the most common type of facility simulation projects which has given rise to a class of new class of simulation software, customized for its needs. M. J. Miller, Ferrin, and Messer (2004) present a study of the using EDSIM package in ER and the insights of using the tool. Sinreich and Marmor (2004) develop a simple and intuitive tool to analyze ED which flexible in design to account for difference in patient needs. Gunal and Pidd (2006) introduce DGHPsim simulation framework which can multitask and allocate doctor on the basis of experience. Holm and Dahl (2009) developed an ED model using FLEXSIM package where they simulate the effect of triage system.

Apart from process interaction approach, there are other approaches to model the ED. Takakuwa and Shiozaki (2004) present a functional approach for the ED where by patients

are classified by the type of services required. Medeiros, Swenson, and DeFlitch (2008) present a Provider Directed Queuing system model for ED using ARENA which provides estimates of performance measures under various patient loads. Meng and Spedding (2008) model the ED using a combination of Fourier and autoregressive process which resulted in more realistic representation of the system. Beck, Balasubramanian, and Henneman (2009) present a simplified ED approach to model ED. Marmor et al. (2009) developed a real time decision support tool for ED using the offered load approach. Ismail, Abo-Hamad, and Arisha (2010) apply the balanced scorecard approach to optimize ED performance. Chockalingam, Jayakumar, and Lawley (2010) model ED's patients and resources flow using Petri nets. Using this model, they develop stochastic differential equations which are used for determining the time to divert ambulances to avoid ED overcrowding.

2.3.3 Specialist Clinics

Specialist clinics are facilities that provide tertiary medical care such as Chemotherapy centers, Laboratory, MRI imaging facility, etc. These clinics are highly specialized in the services they provide, hence bringing in a new set of problems that can also be addressed by simulation. Most simulation studies that belong to the specialist clinic types focus on the facility design.

Johnson (1998) study the requirements of a new maternity ward and account for the future increase in load. Centeno et al. (2001) is another study that focuses on maternity ward and process design. This study accounts for the patient's delivery method which affects the patient's LOS in the ward. Sepulveda et al. (1999) study the design of cancer ward to improve patient throughput. They also identify patient appointment scheduling also as a potential area of improvement in such clinics. Bosire et al. (2007) presents a simulation model that deals with design of a new Computed Tomography (CT) center while Ramakrishnan et al. (2004) dealt with an existing CT center that needed to upgrade its process to efficiently incorporate digitalization processes. Vila-Parrish, Ivy, and King (2008) study the inventory costs of a hospital pharmacy and develop a model to minimize the costs.

The models accounts for the cost for medicine shortages, medicine expirations, and the holding cost.

In addition to facility design, Specialist clinics simulation studies also deal with clinical quality management. Baesler and Sepulveda (2001) formulate a multi-objective optimization model for performance improvement in an oncology clinic. The objective was to minimize patient waiting time while maximizing resource and worker utilization. Martin, Gronhaug, and Haugene (2003) develop model to reduce overcrowding in geriatric clinic using PROMODEL by reducing patient waiting times. Centeno et al. (2010) develop a DES model to explore the factors affecting the performance of an endoscopy center. Interestingly, early arriving patients were causing the system to operate under capacity. Housseman et al. (2009) study the impact of RFID technologies in a cryo-conservation center and incorporate human error in to their simulation model. Ramis et al. (2008) model the imaging center as a pull system and minimize the patient waiting times. Yurtkuran and Emel (2008) develop a simulation based decision making tool to optimize pharmacy operations. Song, Bair, and Mingchang (2008) study the effect of physician starting time in a physical examination routine and patient arrival times using simulation with goal programming model.

2.3.4 Physician and Healthcare Staffing

In resource constraint situations, physician and healthcare staffing becomes the focus of simulation problems that better serve the patient. These problems are usually multi-objective problems where the simulation model minimizes provider idle times subject to patient constraints. Balasubramanian et al. (2007) convert a staffing problem in an assignment problem and solve it using a multi-period meta-heuristic simulation optimization model. The patients are allocated to a preferred provider while trying to minimize patient waiting times and physician overtimes. Wijewickrama and Takakuwa (2006) developed a simulation model which evaluates doctor schedule mixed with appointment schedules to determine optimal staffing levels. Takakuwa and Wijewickrama (2008) developed a staffing model for an outpatient department to maximize patient satisfaction. . Lach and Vazquez

(2004) study the effectiveness of a tele-medicine program and conclude the poor scheduling of physicians and ineffective process flow is the root cause for poor system performance. Centeno et al. (2000) study the staffing policies using MEDMODEL in radiology department.

2.4 Healthcare Asset Allocation

Healthcare is an ever changing field as it has to keep up with technological developments as well as demographic developments. In order to stay in line with these developments, hospital management teams rely heavily on planning and allocation of hospital assets. Simulation modeling is an attractive tool for these problems given the flexibility embedded in these tools and reduced assumptions required to model the scenarios.

2.4.1 Bed Sizing and Planning

Bed availability is a major issue in hospitals as they must be available to treat patients. Moreover, bed availability in primary care facilities such as ICU and PACU will determine the efficiency of ED and OR respectively. On the other hand, having excess capacity will mean resource under-utilization and added maintenance costs which also is not acceptable solution. Cahill and Render (1999) develops a simulation model in ARENA for bed availability in the ICU unit. They further state that although they optimized the ICU bed utilization, it increased the backlog in downstream medical units highlighting the need for multi-facility approach to such problems.

2.4.2 Room Sizing and Planning

Room sizing and planning problems deal with facility design. Hospitals have to upgrade their facilities to match their demand and this brings up a range of problems which can be addressed using simulation. Ramis, Palma, and Baesler (2001) extend the planning simulation models to include the whole facility. In the aforementioned paper, they create a simulation model of a proposed ambulatory care center and experiment with clinic closing times and scheduling of surgeries to evolve the best fit model. Their results state that Longest

Processing Time (LPT) schedule is the near-optimal schedule for operating rooms assuming longer operations have a longer recovery time. Ashby et al. (2008) use simulation as a tool for planning patient flow from the current hospital to replacement facility. Their work helps the administrators identify non-value added tasks which transforms this hospital in to a more efficient one. Alexopoulos et al. (2001) develop a simulation model to analyze a low cost vaccination clinic for the poor. The model is designed to incorporate other services in the future if they need to be offered.

Wiinamaki and Dronzek (2003) use simulation to validate an architectural design of an ED which would have impact on the downstream department. The model estimated future capacity requirements of the ED using length of stay of each patient. Miller et al. (2007) develop a simulation model to study the impact of merging six EDs in to a single one. This model identified potential bottlenecks in the ED simulation and was further used to test workarounds for the new facility.

Huschka et al. (2008) developed a DES model of relocation of interventional services for pain medicine at Mayo Clinic. This paper further states the importance of using simulation in early stages of facility development as their model is critical for efficient facility design. de Mendonca et al. (2010) study the improvements to blood processing center using MEDMODEL package. The simulation model also accounts for human error in the order collection system which causes delays in the overall treatment of the patient.

2.4.3 Staff Sizing and Planning

Complementary to rooms and beds, hospitals have to ensure they have quality service providers to utilize the facilities. Staff sizing and planning models help in estimating the future staffing requirements in order for the hospital to operate efficiently and effectively. Few studies focus on evaluation of new technology to assist staff. Rossetti, Kumar, and Felder (1998) present a model developed to understand the impact of mobile robot technology in the hospital which can create a 56% cost savings and 3% performance

improvement from the solution. Wong et al. (2003) study an electronic medical ordering system and its effect on pharmacy response time and correct medication delivery.

Weng and Houshmand (1999) present a case study on simulation application in a local clinic. They developed a model to estimate tradeoffs between patient performance measures and staffing cost to find the near-optimal staffing levels. Osidach and Fu (2003) also study the performance versus staffing cost tradeoffs in a mobile examination center. Spry and Lawley (2005) experiment with the staffing schedules in a pharmacy department as well as test the effect of a dispensing system to improve the facility's effectiveness. Pirolo et al. (2009) studies the cardiovascular laboratory worker schedules which affects patient length of stay in ED. The simulation insights were considered in the redesign of the laboratory. Rohleder et al. (2010) developed a simulation model to test alternative care team design to improve the efficiency of the Mayo Clinic's Outpatient department

Some studies in this category focus on the generalized staffing problems in whole system scenarios. Thorwarth, Arisha, and Harper (2009) derive a general analytic expression for staffing that leads stable queuing systems using simulation. Decision makers can use these expressions to design staffing policies that maximize service levels. See et al. (2009) study the staffing levels at an urgent care clinic during pandemic outbreaks and identify that clinical efficiency is sensitive to doctor staffing levels. Faller, Flynn, and Ferrin (2009) simulate the staffing levels for healthcare services in prison systems and provide a new process to improve the throughput of the system.

2.5 Health Policy Simulations

"The thing [medicine] that still startles me is how fundamentally human an endeavor it is", a quote from Atul Gawande's bestseller "Complications" (Gawande, 2002). Healthcare is an imperfect science, as it based on concrete observations and physician's experience. Neuroscience studies have proved the variability in human thought process. So, a patient might take several treatment paths under different doctors. Hence, the fundamental question

that we seek to answer using the simulation can be aptly framed as: “Which is the best course of treatment?” Simulation models presented in this section are decision tools to influence physicians or policy makers and improve the healthcare field’s quality of treatment.

One of the most common applications of simulation modeling in decision support is cancer screening and treatment. Baldwin, Eldabi, and Paul (1999) present a flexible Markov chain model for designing adjuvant breast cancer treatment trial. Based on this model’s insights, health policy makers can collect the relevant data to study the interactions in breast cancer treatment trials. Tafazzoli et al. (2005) compare the screening methodologies for colorectal cancer. This paper provides a deterministic and probabilistic cost effectiveness analysis. Yaesoubi and Roberts (2008) develop a simulation model to analyze the willingness to pay cost of health insurance providers for colorectal cancer, which is the cost of additional one life year gained from screening.

Organ transplant policies are another field where simulations are widely used to decide on equitable allocation of organs. Harper et al. (2000) develop a liver transplant simulation model for Unified network for Organ Sharing (UNOS) organization and incorporate general organ rejection rate into UNOS Liver allocation model. Baldwin et al. (2000) study the prioritization criteria for waiting list patients on UNOS Liver allocation model. Taranto et al. (2000) develop a national kidney allocation model and take in to account post-transplant events. Abellan et al. (2004) develop a discrete event simulation model of patient waiting behavior for renal transplantation. They use Bayesian inference to model inputs to the model. Davies (2006) study the equitable kidney allocation problem for different risk groups.

Another topic of healthcare policy simulations cover is the study of cost effectiveness of treatments. S. H. Jacobson, Sewell, and Weniger (2001) study the value of combination vaccination for children. They use Monte Carlo as the simulation method and also factor in the human costs when computing the value of the vaccination process. Koppenhaver et al.

(2009) study HIV treatment in drug scarcity environment. The paper develops a simulation model to evaluate the current WHO policies.

Healthcare policy simulations are also used in public health preparedness problems which deal infectious disease outbreaks and public health response system preparation. Brandeau (2008) discusses the role of simulation in infectious disease control policy. The paper present simulation case studies on HIV, Contact tracing and hepatitis B infection progression. Hughes, Currie, and Corbett (2006) develop a compartment model to simulate TB disease spread in the presence of HIV where they assume the HIV disease progression to be static. Mellor et al. (2007) develop a model to test strategies to detect TB in high prevalence of HIV environment and conclude that active contact tracing is the better method for these scenarios. Wynter and Ivy (2009) study the public health response to an E.coli outbreak in North Carolina State fair and identify that lab capacity to be a major bottleneck. Worth et al. (2010) develop a DES model in ARENA to model an outbreak of infectious disease and the response of public health department. The model also accounts for secondary infection spread by creating a submodel for contact tracing. Yarmand et al. (2010) study the cost effectiveness of isolation versus vaccination for an outbreak of H1N1 in a university campus. The study methodology involved development of a simulation model to model the disease progression, and an optimization model to minimize vaccination costs. The study results were analyzed using grid search methodology to identify feasible solutions. Andradóttir et al. (2010) develop stochastic compartmental model to model pandemics in a mid-sized North American city and study the social and economic impact of vaccination at different levels. Lizon, Aleman, and Schwartz (2010) develop a SIR (Susceptible Infected and Recovered) model for SARS infection where they take in to account effectiveness of public healthcare system along with different thresholds of infections. Silva and Pinto (2010) study the effectiveness of emergency medical systems using simulation and optimize it to maximize the number of life saved by the services.

Apart from DES, Agent based modeling in combination to DES is gaining momentum in the research community. Aleman, Wibisono, and Schwartz (2009) develop an

agent based model to model individual behaviors during pandemic responses. Their model also accounts for close and casual contact which affects the way disease progresses in the society. Carr and Roberts (2010) develop an agent based model in C++ to model an infectious disease progression. They superimpose a facility allocation problem which identifies the best spots to open clinics to effectively control infections. Barnes, Golden, and Wasil (2010) develop a dynamic patient network model for spread of hospital acquired infection taking the dynamic nature of hospital care worker in to consideration. Soorapanth and Chick (2010) present a differential equation model to study the behavior of behavioral interventions for HIV-infected persons. The compartment infection model includes HIV behaviors and compares intervention policies. Ekici, Keskinocak, and Swann (2008) study food distribution logistics response to pandemic avian flu using a dynamic update approach to build the food network.

Ambulance diversions are another class of problems investigated by healthcare simulation models. Maxwell, Henderson, and Topaloglu (2009) develop a markov decision process to model the decision to divert ambulance and solve it using a dynamic programming approach. Hagtvedt et al. (2009) apply game theory to study the ambulance diversion problem which takes in to account the economic choices for each player. Ramirez, Fowler, and Wu (2009) go one step further and develop a DES model to study the financial impact of ambulance diversion on hospitals. Nafarrate, Fowler, and Wu (2010) develop a simulation model to perform a bi criteria evaluation of Ambulance diversion policies thereby evaluating the direct impact of ambulance diversion on the emergency department. The criteria under evaluation are mean waiting time and percentage of ambulances diverted.

Healthcare policy studies also span the domain of pharmaceutical industry. McGarvey et al. (2007) present a semi markov chain model to predict patient enrollment for clinical trials. The model is accurate and fast compared to current methods to predict clinical trial enrollment. Blau and Orcun (2009) develop a model to personalize drug regimen based on minimally invasive clinical testing. Pasdirtz (2009) studied the drug markets and the effect of direct to consumer advertising. The study concludes that drug markets don't behave as a

competitive market and controlled advertising will only affect sales and not the cost of the drug.

2.6 Multi-facility Simulation Models

Until this point in this literature survey, we have presented the papers that only dealt with problems that are specific to a particular facility or for an operation type or function. In all of these models, our modeling insights are restricted to the operation under analysis and ignore the cascading effects of improvements on the connected parts of healthcare system. For example, improvement of ED throughput may increase the back log at inpatient units or may upset the scheduling of OR departments. Another good example is the experimentation of physician schedules. Physicians are shared between hospital departments and changing them in one department may have a pronounced effect in another department where the doctor's works. These complex relationships that exist between facilities are investigated in the papers presented in this section.

Ballard and Kuhl (2006) develop a general method to estimate the maximum capacity of OR using simulation model while accounting for capacity constraints of PACU, Inpatient and outpatient departments. James Swisher et al. (1997) develop a simulation of physician network by taking in to account the staffing levels at each clinic which affects patient throughput. The patients use a call center for appointment scheduling which allocates patients based on their needs and clinical availability. This model was developed in a Visual simulation environment using an object-oriented approach. Multi-facility simulation systems are data intensive, which can be overcome using electronic medical records as presented in Takakuwa and Katagiri (2007). They develop a detailed simulation model of the whole hospital by harnessing the power of electronic medical records.

Martin Pitt (1997) presents a whole hospital model as a state transition network to compute the bed usage hospital-wide. Gunal and Pidd (2007) present a conceptual model for whole hospital model composed of ED, Inpatient and Outpatient Department sub-models.

Ashby et al. (2008) simulate the complexities involved in transitioning of patients from the inpatient units of one hospital facility to another for the purposes of renovations. Kolb et al. (2007) study the effect on patient related metrics by coupling of ED and inpatient unit.

2.7 Simulation Tutorials

Every field needs practitioners that can advance the practice. Significant work presented in WSC belonged to the category of simulation theory. Simulation theory can be grouped in to two sections: Modeling and Implementation which will be discussed in the following sections.

2.7.1 Modeling Tutorials

To propel the field of simulation forward, there needs to development in modeling techniques which can be brought around by development

- new modeling tools enhancing current modeling capabilities
- development of new software to address requirements of new domains
- new modeling techniques to capture the dynamics of complex systems

These papers help simulation community stay abreast with latest advancements in the field of simulation modeling.

Modeling techniques present new ways to model systems with existing tools. Mabry, Rodriquez, and Heffernan (1997) present modeling logic to build a medical analysis system. Workman (2000) present a roadmap on how simulation models can be built to understand the drug development processes. Yaesoubi and Roberts (2007) derive the important factors in design of colorectal cancer screening using simulation based on cost, QALY and cost per QALY. The paper also presents a comprehensive review on experiment design. M. J. Miller et al. (2008) present a tutorial on allocation of outpatient clinics services using simulation, linear programming and spreadsheet analysis to deal with space assignments, schedule

configurations and throughput targets. Eldabi et al. (2010) present a tutorial on model development using SysML package. They also introduce concepts relate to life cycle of simulation projects in healthcare services field. Kasaie et al. (2010) present an agent-based approach (developed in NETLOGO) to resource allocation problem. The resource allocation is modeled as a function of disease level, where the disease is modeled using ABM. Ibrahim and Whitt (2010) develop real-time delay predictors that can effectively cope with customer abandonment and time-varying arrival rates to make accurate delay announcements.

Human behavior is also a field of simulation modeling research. S. C. Brailsford, Sykes, and Harper (2006) discuss modeling logic for incorporating human behavior in computer simulations namely; the health belief model and theory of planned behavior. Katzper (2005) present a tutorial in modeling pain for a clinical trial environment. They develop a pain transition model based on system thinking. Bisset et al. (2009) present a partially observable Markov model for wide variety of human behavior and solve it using high performance computing oriented computer simulation. Their approach facilitates realistic simulations of over 100 million nodes and 6 billion edges thus capturing human behavior along with disease dynamics. Schwaab and de Freitas Filho (2009) present a dynamic simulation model to plot mortality curves. Incorporation of fuzzy module to treat uncertainty of aging and death produces realistic reproduction of curves. Charfeddine and Montreuil (2010) develop a ABM where healthcare is considered a function of population and healthcare services. The model integrates population and healthcare delivery methodology reflecting human behavior and disease progression.

There is significant interest in the simulation community to learn about new simulation software that can address specific requirements. Adams et al. (1999) present an epidemiological tool (GERMS) for studying infectious disease spread. The paper is also an excellent reference on compartmental modeling and micro simulations. Barth-Jones, Adams, and Koopman (2000) describe HIVSIM, a specially designed simulation system for modeling HIV disease dynamics. The paper couples HIVSIM with Monte Carlo Simulation analysis to study the effectiveness of vaccine trial design. Rao and Chernyakhovsky (2008) discuss

SERAMUS++ software package that is based on parallel simulation to stimulate infectious disease spread. Gunal and Pidd (2008) describe the DGHPSim simulation software and its approach for multi-facility modeling in the healthcare context. Troy and Rosenberg (2009) present a Java library to model peri-operative processes using a process interaction simulation. The tools could be used for short term scheduling or long term capacity planning. Bohk, Ewald, and Uhrmacher (2009) present a probabilistic population projection simulator which can accurately compute a future population size along with composition by age and sex. Hay, Valentin, and Bijlsma (2006) present a new emergency care modeling framework that is not patient centric or entity driven. Instead the simulation framework is driven by clinical priority and medical resource's decision on most appropriate task based on the skills

In methodology development, several papers discuss frameworks for design of hybrid simulation experiments. Chahal and Eldabi (2008) discuss a hierarchy in designing healthcare simulations where DES is used at operational level and SD at the strategic level. Brailsford (2008) paper is an introductory tutorial in SD for healthcare situations. Dibble (2010) integrate computation laboratories, spatial agent based simulation and real time situation updates into a simulation framework to provide realistic pandemic risk assessments. The simulation framework also facilitates study of optimal intervention and prevention strategies. Brailsford, Desai, and Viana (2010) also describes a hybrid framework combining DES and SD

2.7.2 Implementation Tutorials

Healthcare is a field with diverse stakeholders and each of them has a different view of the system. Successful execution of simulation projects in healthcare requires the modeling team engage their stakeholders in addition to building an accurate simulation model.

Implementation tutorials cover organizational factors that influence successful implementation of simulation. Hermans, Sluijs, and Aartsen (1998) emphasize on the need to

have flexible model to account for dynamic policy changes and need for animation for effective communication between modeler and stakeholders. Lange (1999) presents seven “*chaotic conditions*” of healthcare domain and also suggests workaround to tame them. Sanchez et al. (2000) is a collection of viewpoints of healthcare simulation experts on the potential of the healthcare simulation practice. They address the issue of the low implementation of healthcare simulation projects based on their experiences in healthcare simulation. Brady (2003) proposes a simulation based approach to develop, test, and refine emergency response plans. T. Young et al. (2009) present three critical challenges with healthcare simulation practice and methods to address model fidelity, linking modeling and decision making process, and cultural barriers in healthcare for simulation practice. Eldabi (2009) classifies healthcare problems as wicked problems, and proposes some approaches for the modeling community to overcome them. S. C. Brailsford et al. (2009) identifies the barriers to stakeholder engagement in healthcare simulations and presents guidelines to engage stakeholders. Jahangirian, Taylor, and Young (2010) study the economics of healthcare simulation and develop a set of guidelines for healthcare system stakeholders to assess the cost effectiveness of the simulation project.

Second focus area within implementation tutorials is the approach toward applying simulation modeling in healthcare. Eldabi and Paul (2001) point out key challenges to modeling healthcare systems which are data inconsistencies, lack of understanding the problem and conflicting interest of stakeholders. They propose a participative framework for healthcare simulation engagements to overcome these challenges Morrison and Bird (2003) identified critical success factors for modeling ambulatory healthcare units which were visual mapping and simulation tools, resource mapping, data collection methods, minimize on site involvement and more importantly, having a healthcare professional on the team. Kuljis, Paul, and Stergioulas (2007) draws parallels from business and manufacturing in to healthcare to identify mismatches. Eldabi and Young (2007) present an approach that support integrative modeling for healthcare scenarios. Gibson (2007) also presents another approach to planning and design of hospitals using DES. Efe, Raghavan, and Choubey (2009) presents

a taxonomy of movable healthcare assets that can help healthcare simulation modelers understand the domain as well as presents simulation framework for each of the movable healthcare assets.

2.8 Research Scope Development

The fairly broad literature survey from 1997-2010 presented some interesting insights in to simulation world (see Figure 2-1). Fifty percent of the simulation projects dealt with facility specific issues.

Based on the papers reviewed, simulation methodology has also diversified to include agent based modeling and system dynamics in addition to discrete event and Monte Carlo. Twenty-one percent of the problems dealt with health policy issues with more papers presented in latter half of the review period. Disappointingly, only five percent of the papers were related to multi-facility modeling which is in line with earlier reviewers (Günel and Pidd, 2010; Jun et al., 1999). Significant papers reviewed in this study developed their own simulations in generic programming languages which points to the lack of flexibility in modeling tools currently in market. Lack of flexibility term encompasses the inability for the modeler to modify software constructs, inability to modify software output generation to provide custom statistics and visualizations as well as the inability to use the tool in more than one simulation paradigm. These issues can be addressed by using an object-oriented approach to a simulation framework development in a multi-paradigm supporting simulation environment of SIMIO. Furthermore, there needs to be guidelines on best practices in simulation modeling software development to promote development of good simulation software. These developments of these guidelines are also a focus of research of this thesis.

Secondly, in 50% of the papers focused on facility specific issues whereby the boundaries are rigid and ignore the complex interactions between facilities which have a marked effect on the real system which could be the reason for low implementation of healthcare simulation models (Fone et al., 2003; Katsaliaki and Mustafee, 2011). Simulation

modeling. in healthcare needs to move in to a more integrative framework to encourage complex systems thinking for facility specific issues. In a nutshell, modelers need to transform their facility-specific issues in to multi-facility problems. For this, we propose an object-oriented approach to development of simulation models where each facility can be modeled independently and then combined together to form a whole hospital model. This concept was proposed earlier in Günal and Pidd (2007).

Literature Survey Breakdown

Figure 2-1 Breakdown of Literature Survey Papers

Finally, the majority of simulation tutorials focused on addressing issues related to low stakeholder engagement and low implementation levels of healthcare simulation projects. Healthcare is inherently complex and structured differently compared to business and manufacturing industry. Successful implementation of healthcare simulation projects need to have clear understanding of the interaction of influencing factors of simulation

projects in healthcare environment. Secondly, the low stakeholder engagement and low implementation levels are interlinked and the nature of this relation will also be a focus of this study.

2.9 Chapter Summary

In the first part of this chapter, we surveyed the field of object oriented simulation to understand if similar work has been carried out in healthcare field or other application domain. The second part of this chapter dealt with healthcare simulation papers. A systematic review of papers presented at Winter Simulation Conference from 1997-2010 was conducted and organized in to 5 categories which are Patient Flow and Optimization, Healthcare Asset Allocation, Health Policy Simulation, Multi-Facility Simulation and Simulation Tutorials.

Based on this extensive survey, we drew conclusions that formed the pillars of our work. A significant number of papers chose to develop their own simulation language which indicates the lack of flexibility among the present packages. This can also be attributed to lack of formal rules to capture the utility of simulation toolkit. Secondly, we observed that the number of papers in complex systems analysis has increased. However, the number of works that particularly dealt with multi-facility systems showed no change. This prompted us to work on a framework to support complex integrated multi-facility systems development. Finally, the healthcare simulation field still suffers from low implementation levels. This problem has been attributed to multiple factors in the past papers. In this thesis, we explore the dynamics of these factors and discuss their interdependencies.

Chapter 3 Healthcare Toolkit development

In the previous chapter we had identified that healthcare simulation field lacks a flexible modeling tool and framework which has contributed to an increase in development of application-specific simulations and made simulation modeling an expensive tool for healthcare problems (refer to Section 2.8). Our development of healthcare toolkit in SIMIO (SIMIO-HC) is designed to overcome these barriers and make simulation a feasible and attractive option for healthcare problems.

This chapter will discuss the design philosophy behind the health care object library in SIMIO (SIMIO-HC) as well as describe the inspiration, development process and utility of SIMIO-HC. We intend this chapter to be an *architect's plan* for healthcare simulation software development. In other words, this chapter is meant to be a guide for healthcare simulation software developers to articulate the important design features and set the direction for future work in this domain.

Finally, this chapter will also develop guidelines for OOS class development. For this we define three attributes for each class: “*Usability*”, “*Complexity*” and “*Utility*”. “*Usability*” is defined as the user friendliness of the class. In broader terms, a class that is easy is to understand and use. “*Complexity*” is defined as the class’s ability to execute advanced logic. Alternatively, this concept can be thought of a class that can model many different processes. “*Utility*” is defined as the resultant value of the class to the modeler. This is a function of complexity and usability. We shall explore the nature of this function in this chapter.

3.1 SIMIO – A brief introduction

Before we delve in to the development of SIMIO-HC, we present a quick introduction for readers who aren’t familiar with SIMIO. SIMIO is an acronym for **S**IMulation modeling framework based on **I**ntelligent **O**bjects (SIMIO) which provides an object oriented approach to simulation modeling. This modeling approach essentially helps modelers break a complex

problem in to manageable problems. Since healthcare problems are often complex with no definitive solution, they are often classified as wicked problems making a strong case for object-oriented approach to modeling healthcare problems (Eldabi, 2009).

A detailed discussion of SIMIO framework can be found in the (Appendix A) for readers who wish to become familiar with SIMIO's internal workings. This section will discuss the standard library and basic concepts in creating new objects in SIMIO providing the user a good understanding as we follow the class-based programming approach.

3.1.1 Base Classes

SIMIO is composed of six basic classes of objects derived from one super class- Intelligent Object. These basic classes provided modelers a starting point for developing and extending intelligent objects within SIMIO. The base classes of SIMIO have bare minimum native intelligence but have the capability to extend and gain more functionality. All the base classes with the exception of the *Entity* and *Transporter* class are derived from the *Intelligent object* which lays the foundation for the SIMIO modeling paradigm. It can be seized, released or follow a schedule .The seven base classes are displayed as a tree in Figure 3-1 and is explained in the Table 3-1.

Figure 3-1: Simio Base Classes

Table 3-1: Description of SIMIO Base Class

Class	Description	Standard SIMIO Sub-classes
FIXED	Derived from INTELLIGENT OBJECT , this class represents stationary items such as gas station, MRI scanner etc	Server, Resource, Combiner, Separator, Workstation, Source and Sink
AGENT	These intelligent objects have the ability to move freely through three-dimensional spaces.. Agents can be used to implement ABM paradigm, though at present, SIMIO provides only basic support for ABM	ENTITY and TRANSPORTER
LINK	These intelligent objects are used to build network to facilitate the flow of entities within the model.	Connector, Time path and Conveyor
NODE	Represents the fixed points used to model a network. The node class combined with the link class can formulate complex network structures.	Transfer node and Basic node
ENTITY	Ability to move through the links and nodes of the network as well as in and out of objects, this class is used to model dynamic objects of a model	Model Entity
TRANSPORTER	Class representing a specialized entity with the ability to pick-up, carry and drop off entities.	Vehicle and Worker

A detailed representation of the standard object library is shown in Appendix A.

3.1.2 Building New Class

One of the hallmark features of an Object-Oriented Programming's language is the concept of extension of base classes to build new classes with complex behavior. In SIMIO, the user can build new class in two ways: "*Composition*" and "*Inheritance*"

In "*Composition*", we build a new object by combining two or more component classes. For example, a two **Server** classes can be combined to create a class to model two stage processes. This type of class building is hierarchical and is useful in creating smaller models which can be used to build larger models. So, every model you build in SIMIO can be used as an object. Most other simulation languages, only provide for this type of object building.

The second method *Inheritance* is more powerful. This method is useful when you want to modify the behavior of an existing object (i.e. create more specialized object). When you sub-class an existing object, it inherits all the methods and variables of its parent object. The user has the power to override existing processes to modify them as well add new variables to create new object behavior. For example, in order to create a transporter that needs an external assistance to move, you need to subclass the **Vehicle** class and add properties and variables to seize a resource when its capacity is allocated.

In addition to subclassing from the standard library, SIMIO allows the user to add methods and variables to build a new class onto one of SIMIO's base object classes. Nonetheless, this method inherits all the properties of SIMIO's base object class. Furthermore, this approach allows you build objects from scratch, thus offering modeling flexibility as well as improving run time performance for the object.

Regardless of the employed method; the new object can be used in exactly the same way as SIMIO standard objects. The choice of method to build new objects is based on the

nature of the new object. Composition is an effective method if your new object's behavior can be generated by combining two or more objects, while inheritance is an effective method when your new object's behavior can be generated by specializing or adding methods or variables from an existing object. The issue of inheritance versus composition will be further explored in the following sections as we describe the SIMIO-HC classes.

3.2 Base Classes of SIMIO-HC

Healthcare, as an industry, lacks standardization in their service organization and delivery due to its heavy dependence on humans on both sides of the service table. This lack of standardization has lead hospitals to adapt processes to suit their organization and to best cater to their demographic. Nonetheless, this poses a unique challenge to healthcare simulation modelers as they need to adapt standard objects to reproduce these complex behaviors, which affect the value of simulation model. Given these constraints, good healthcare simulation software should be flexible enough to incorporate the modeler's requirement with some ease. This concerns makes class based programming for software development ideal.

A class based programming approach to object oriented software development defines strongly typed classes with rigid structure which supports the development of robust classes (Anderson and Drossopoulou, 2003). What does this mean for healthcare simulation software development? Well, adopting Class based programming allows for the definition of strong base classes, which are collections of objects that can provide a full spectrum of healthcare operations. These bases classes can be customized by the user depending on their modeling requirements through subclassing. Nonetheless, the critical factor for the simulation software developer is to correctly identify the base classes. Otherwise the class hierarchies may grow exponentially making the simulation library inefficient and difficult to use (Gottlob, Schrefl, and Rock, 1996). Moreover, base classes define the utility of the simulation package, which makes their design even more critical for successful use of the

simulation software package. The concept is captured graphically in Figure 3-2 and is framed as “*Guideline 1*”.

Guideline 1: *The incremental utility of a framework gradually decreases with an increase in hierarchy*

Figure 3-2: Utility as function of Hierarchy and Class Count

It is obvious that the number of object classes in an OO-toolkit is a function of hierarchy (number of class levels in the toolkit). In other words as the hierarchy increases, the number of object classes will increase. In this thesis, we have assumed an exponential increase in each class gives rise to at least two sub classes. Nonetheless, the increase in object classes will increase the utility (i.e. how much can the toolkit accomplish) up to a certain threshold. Beyond this point, new classes become redundant as the same functionality can be

achieved using existing classes. Hence, the relative utility of these classes are very small (near 0 but positive).

The base classes in SIMIO-HC were derived from the common themes that have arisen in simulation literature survey pertaining to healthcare operations. We limited the scope of this toolkit's first release only for healthcare operational modeling, since 53% of the surveyed papers belong to this category. More importantly, operational inefficiencies cost the American healthcare system nearly \$6 billion which is 0.2% of the total cost of the system itself (IOM, 2010). The SIMIO-HC base classes can be grouped in to four groups which are *Fixed*, *Entity*, *Node*, and *Transporter* each of which will be discussed in detail in the following sections (Figure 3-3). It should be noted that these four classes have analogous base classes in SIMIO itself.

Figure 3-3: Class Hierarchy Tree of SIMIO-HC

3.3 Fixed Group

FIXED group classes are defined to be the class which when instantiated, gives structure to a simulation model. SIMIO-HC has six classes that belong to this group which are *Patient Care*, *Redirect Object*, *Waiting Room*, *Registration Desk*, *Statistics Collector*, *Hospital Workstation*, *Patient Care with Lab Care*, and *Hospital Workstation with Lab Care*. A hierarchical representation of these classes along with implementation notes is presented in Appendix B.

3.3.1 Patient Care (PC)

Patients need to be held at a specific location in the healthcare facility when they receive clinical service such as ward beds, treatment clinics, etc. Secondly, every healthcare facility serves the patient according to their needs, which influences the organization of patient processing systems. Finally, within every healthcare facility the service delivery is highly dependent on departmental resources which further customize the treatment delivery. The *Patient Care (PC)* object class in the SIMIO-HC toolkit has been designed to provide a construct that can address the patient processing modeling concerns when patients are processed at a fixed location.

Looking under the hood of the *Patient Care* object will reveal that it is in fact just a simple processing station but with no input or output buffer. The reason behind adoption of this design is to accurately represent the hospital environment. It is ridiculous to visualize patients waiting outside a ward bed for it to be free so that they can occupy it. Alternatively, patients do not wait for assistance to move on further in their treatment cycle outside the ward bed. It is always the case that patients wait at a designated location until they can move further in to another stage of their care process. Thus it makes sense for the *Patient Care* object not to have input or output buffers. In order to facilitate transfer between successive *Patient Care* objects in the simulation model, a *Redirect Object (RO)* is required to control the flow of patients. This design paradigm allows the modeler to take in to account patient

preferences for healthcare services in the system (refer to Section 3.3.5 for more information).

In a simulation, an operation is modeled as a sequence of delays following any resource seizes. The healthcare corollary to the previous statement would mean that time the patient spends in a treatment station, after the required medical resources are available in order to treat the patient is the treatment process. Given this view of the world, it makes it easier for one to model preemption and multitasking logic of medical resources.

Preemption is defined as seizing resources from a lower priority task when the resource is busy (except when failed). For example, if a head trauma patient arrives in an emergency department the patient will receive all the resources, even the ones that are treating minor injuries. Multitasking is when a resource serves more than its capacity by alternating between the jobs. From a simulation modeling perspective, these are one and the same as the resource's capacity is seized by interrupting another job it is working on. However, the situations for which a medical worker may preempt or multitask are variegated and impractical for the developer to have foresight to account for all of them. Furthermore, in the spirit of flexibility in design the focus of the developer should be to provide the constructs to interrupt the delays so the modeler can harness and implement the specific logic as the needs want.

In SIMIO-HC, we implement preemption and multi-task constructs using the built in interruptible `DELAY` steps. All the process delays are interruptible except for the ***Redirect Object*** class. This simple modification gives the modeler complete control over defining preemption logic using the `INTERRUPT` step. For more detailed documentation on the `INTERRUPT` step, refer to SIMIO documentation.

In order to facilitate modeling of variations in treatment delivery, we provide three constructs which are *Secondary Seizes/Releases*, *Add-on Processes* and *Custom Treatment Logic*. “*Secondary Seizes*” and “*Secondary Releases*” are built-in constructs to seize or

release resources at specific events as specified by their name. The “*Add-on Processes*” allow for the user to add processes that define specialized logic for that event in the event calendar. For the *Patient Care* object, Table 3-2 provides the list at which these constructs are defined. With the exception of *Treatment Resources* construct, seize and release steps are not coupled. This uncoupling means the user will have to specify the releases appropriately giving the modeler complete flexibility in designing the resource seizing and releasing processes.

Table 3-2: *Patient Care* Object Label Definitions For Seize/Release and Add-On Processes

Logic	Seize-Release Label	Add-on-Process Label
When an patient occupies the <i>Patient Care</i> object	<i>On Entering</i>	<i>Entered</i>
Just before the treatment procedures	<i>Pre Treatment</i>	<i>Pre Treatment</i>
During the Treatment	<i>Treatment Resources</i>	<i>Treating</i>
Just after treatment procedures are executed	<i>Post Treatment</i>	<i>Treated</i>
After the patient exits the <i>Patient Care</i> Object	<i>On Exiting</i>	<i>Exited</i>

Although the *Seize/Release* and *Add-on Processes* construct provide considerable flexibility to the modeler to define a large combination of treatment delivery methods, there are situations where this construct might not be easy to implement. For example, situations exist where the modeler cannot define the boundaries at which resources must be seized or released due to lack of information or the given constructs might not fit the healthcare scenario under consideration. Thus, in order to provide the modeler an extra degree of freedom, the *Patient Care* object embeds the *Custom Treatment Logic* construct which gives the modeler full ownership over modeling the treatment logic.

The *Custom Treatment Logic* construct requires that the user define their own process element and link it to this property. Furthermore, the modeler will need to assign the patient treatment status by firing the treatment status monitor in order to collect meaningful patient statistics. It is quite common in today's medical practice for a patient to have a follow up visit to assess their condition. For a follow-up visit, the medical services are dispensed by the same medical worker in most cases. However, this detail is often missed in simulation models due to the complexity in modeling logic. In order to facilitate this logic, the ***Patient Care*** object has built-in object reference state variables: *Last Doctor Seized* and *Last Nurse Seized* which can be used to seize patient-specific resources if the ***Patient Entity*** is allowed to acquire ***Hospital Worker*** (refer to section 3.6.1 for more information). Due to software limitations, the current capabilities can either keep a Nurse or Doctor as defined by the modeler. Furthermore, it is not uncommon for a room to be held for the patient to return. This concern can be easily implemented using a boolean state variable, provided the software developer provides the right constructs in the object. If the clinic is supposed to be reserved for a particular patient then this can be translated in to a simulation model by allowing the ***Patient Entity*** to own the object's capacity until the state variable (*Hold Clinic*) is set false. Therefore, generation of complex model behavior can be simplified by defining correct sequences of object reference states. These states can be modified by either the user or model to effectively manage complexity without adversely affecting usability and in effect, maximize utility of the simulation framework.

Table 3-3: ***Patient Care*** Object Resource State Definitions

Label	Definition
<i>Idle</i>	The object has no patients in its processing station
<i>Allocated</i>	The object's capacity has been allocated and is awaiting patient
<i>Treating</i>	The object has seized all the necessary resources for treating the patient
<i>Occupied</i>	The object is awaiting one or more resources to treat the patient
<i>On Hold</i>	The object is awaiting a specific patient to return

State variables can be also used to keep track of resource states and automatically generate state statistics that track resource utilizations. Here again, developers must pay special attention in defining the resource states (e.g. Idle, Allocated, Busy etc) as they need to be compatible for any class that can be subclassed from the base class. For the *Patient Care* object, the resource state definitions are presented in Table 3-3.

Simulation software developers must strive to standardize their resource state metrics to avoid confusion for the software users. Standardization allows for users to combine states of different classes to provide custom metrics that may interest their clients. Finally, Classes can also encapsulate functions which are handy tools for the simulation modeler who needs to track specific information about the object in order for them to define their custom logic for the object. In order to be precise, a function evaluates a specific expression defined by the developer and returns a value provided the user inputs correct parameters. The information returned by the function could be an object reference, number or boolean value depending on the function definition which is also defined by the simulation software developer. Hence, the developer must clearly explain the context of usage for the function to be used effectively. The *Patient Care* object encapsulates three functions namely:

1. Current Resource State: returns the numerical value of the resource state
2. Currently Treating: returns true if the resource state is set to *Treating*
3. Current Patient ID: returns the object reference to the patient occupying the object.

3.3.2 Patient Care with Lab Care Class (*PC_Labcare*)

Modern medicine heavily relies on diagnostic services like laboratory testing and diagnostic imaging as well as many other forms of testing for treatment. The types of services have been standardized and can be broken down in to three different sub-processes: Specimen Collection, Specimen Processing and Result Dispatching. In this section, we develop modeling paradigms that can handle the interactions at specimen collection and the dispatching of results processes effectively and efficiently.

Specimen Collection process is triggered when a medical worker requests it. At this stage, the patient has to take one of the two actions;

- The patient can wait in the same location until a representative of the laboratory logic arrives to collect the specimen. This would be the case if a patient who needs a blood test where to wait in their bed until the phlebotomist can come around to collect the sample.
- The patient can be redirected to a specific location for testing. For example, a patient needs an X-ray which means he or she must go to the Radiology department where the X-ray is located for testing.

Following this stage, depending on the type of test and state of the patient will be instructed to do one of the three actions for the results collection stage:

- Wait for results at the present (or previous if redirected) holding station
- Wait for results at another designated location
- Exit the system

Assuming the patient needs laboratory services, the decision tree presented in Figure 3-5 summarizes the patient decisions.

One of the ways to model this decision making structure at each stage is to take advantage of the built-in state variables under the ***Patient Entity*** class (refer to Appendix B.I for state variable definitions), which allows us to accurately capture patient to patient variability. Secondly, this approach represents a natural way of thinking for the modeler, since the decision is associated with a patient. Thirdly, this approach allows the developer to re-use the ***Patient Entity*** states in other objects to define more advanced logic.

In this case, the result collection stage is dependent on specimen collection states. If the ***Patient Entity*** decides to stay in the clinic, then the corresponding results are automatically routed to the clinic. Alternatively, if the patient decides not stay in the clinic

but wait at a different location then the results are routed to the designated location. And finally, if the patient does not opt to wait for results then the results can be routed depending on the model requirements. On the other hand³, if the patient is redirected to another department for testing or any other patient processing then the same state variable above can be re-used to implement the patient treatment logic.

Figure 3-4: Decision Making Structure for Specimen Collection

³ This branch is not shown in the Figure 3-5.

For the implementation of specimen collection logic, we need to define a new object that will carry the message from one object to another in the model. In real hospital situations, it is common to have a patient subjected to more than one test from the same specimen. Thus, we need to create a new entity and in SIMIO-HC these are referred to as a ***Specimen Entity*** (refer to Section 1.5.2 for the design philosophy). Once created, the ***Specimen Entity*** needs to be tracked so that it can be matched with its parent (i.e. real patient represented by a ***Patient Entity***) at post processing. In other words, a specimen belongs to a patient (where one patient can have multiples of specimen entity) and once processed, the results should also be delivered to the same patient. The simulation software developer will need to assign parent references so that this process can be automated. Thankfully, In SIMIO every instance of an object has a unique ID once instantiated and the simulation software developer can use this unique object ID to match entities. In SIMIO-HC, we assigned a ***Patient Entity*** reference to every ***Specimen Entity*** and use batching logic to combine results as members related to a single specific patient.

Finally, the reader may have observed that this class is referred to as ***PC_Labcare***. The reason behind this naming convention is that this class is subclassed from the ***Patient Care*** class. In short, we extended the ***Patient Care*** class to add logic for laboratory tests. So an inquisitive reader may ask what is the use of ***Patient Care*** Class if ***PC_Labcare*** can do everything a ***Patient Care*** class can do and more? To answer this question, we use the three factors that determine success of simulation software: Usability, Complexity and Utility.

Usability of a simulation object class is inversely correlated to class complexity which means a high complex class requires more input from the user to define its behavioral boundaries making it less user friendly. Thus, the utility of the object is the same as low complexity which has high usability. Given these relations, the software developer must be careful in choosing the complexity of the class and designing the user interface (usability) so he/she may maximize the utility of the class (Figure 3-4). This also leads us to form our second guideline.

Guideline 2: *The utility of an object class proportional to its complexity and usability*

Figure 3-5: Utility Of a Class

From the literature survey, we observed that majority of papers did not account for patient's secondary interactions which prompted us to provide two ***Patient Care*** classes. For those who are simulating a simple system, we have provided the ***Patient Care*** class. Alternatively, the above finding can also be due to lack of modeling frameworks which inhibit the implementation of secondary treatment logic. Hence, for modelers who are interested in incorporating the secondary interactions, we have the ***PC_LabCare*** class.

3.3.3 Hospital Workstation

After a specimen is collected from the patient, the specimens undergo processing at the laboratory. The laboratory processes are generally standardized which make it ideal to simulate. Specimens are grouped by type of tests required and then they are loaded into

corresponding machine. When it comes to the machine, some can run more than one type of tests and this sometime requires additional time to change the machine configuration. Moreover, some tests are processed in batches where size of the batch may or may not influence processing time. In addition to all this, laboratory tests require materials like beakers, test solutions, etc. to run them which should be also accounted for as they pose a standard inventory problem; which can affect operating costs of the facility. Finally, the last step of a laboratory process involves creation of paperwork and dispatching it. In this section, we lay out a roadmap to develop an object that can handle the complexities of specimen processing (or lab processes).

In essence, the lab processes can be broken down in to three stages: setup, processing and post processing. The setup stage represents the processes that transform the specimen in to an acceptable form to be processed by the machine. The processing stage logic encapsulates the logic for analyzing the specimen and performing the required tests. The post processing stage captures the result preparation logic which could range from preparing paperwork to processing an image or even just packaging the processed results.

Why model the three stages of lab processes as one single object? One might argue that the inherent complexity will negatively affect the utility of the class. However, this is modeling approach makes the task of modeling laboratory technicians work patterns easier. Laboratory technicians are cross trained and often work throughout the entire process of a specimen's lifecycle. This observation means that a simulation modeler must carefully plan resource seizes and releases to avoid the classic "*Resource Deadlock*" problem if the object is split into different objects.

"*Resource Deadlock*" happens when two resources are interlocked such that no one can complete the task (Havender, 1968). For example, two separate batches of blood-work specimens at different stages in their lab cycle require the same two lab technician to complete their processing and each one has seized only one of them; then both tests are deadlocked with each other. In other words, they need the other's resource to complete their

process. To avoid resource deadlock, we need to incorporate multi-tasking logic in the **Worker** objects or just design the **fixed** object such that we can take advantage of its structure. In the case of SIMIO-HC, we opted for the latter since we followed the “*system centric*” development (refer to Section 3.6.1 where we discuss the “*system centric*” concept).

In order to effectively and efficiently manage the class development complexity, we implement the concept of modules. In simulation, a module encapsulates functions, states and variables to execute a specific task. In the ***Hospital Workstation*** class, we have a setup module, processing module, and post processing module. Modularization of a complicated logic assists the developer in structuring the logic which becomes extremely handy when it comes to debugging. Moreover for the modeler it will be easier to understand the inner workings of the class. In SIMIO, the concept of modules can be implemented using the *ACTIVITY* element and modules can be combined using the *OPERATION* element. Moreover, the internal structure of *OPERATION* element allows the user to manage activity resources and avoid the deadlock problem. In addition this, the built-in structure of the *OPERATION* element facilitates automatic implementation the changeover times and sequence dependent processing times.

Thirty-one percent of hospital operating costs, or in dollar figures 16.5 billion dollars a year, are directly related to inventory management (Schneller, 2006). This figure alone should be enough to justify the need for material management constructs in healthcare simulation software. The ***Hospital Workstation*** object logic exploits the built-in construct of the *MATERIAL* element (Refer to Appendix A.III.5) in SIMIO to allow modelers to define inventory problems at each stage of the specimen processing problem. This concern is just like in the case of the ***Patient Care object*** where we use resource states to keep track of ***Hospital Workstation*** object utilization. The states that accurately track the object’s functionality are presented in the Table 3-4. Although the ***Hospital Workstation*** class is not designed to fail at any time, the *Failed* state is only there for subclassed elements to utilize.

The *Waiting for Secondary Resource* is only applicable if the resources are defined under *Activity Resources*.

Table 3-4: ***Hospital Workstation*** Resource States

Resource Label	Definition
<i>Idle</i>	The object has no specimen in its processing station
<i>Allocated</i>	The object's capacity has been allocated and is awaiting specimen
<i>Blocked</i>	The object is awaiting one or more specimen to move on to the next stage
<i>Off-shift</i>	The object's current capacity is 0 assuming the capacity is based on a schedule
<i>Setup</i>	The Object is executing setup processes for one or more specimens
<i>Processing</i>	The object is analyzing one or more specimens
<i>Post-Processing</i>	The object is performing post-production on one or more specimens
<i>Waiting For Secondary Resource</i>	The object is awaiting a secondary resource to continue its operation
<i>Waiting for Material</i>	The object is waiting for materials to execute the process
<i>Failed</i>	The object has failed and is undergoing failure and repair events

The ***Hospital Workstation*** class is designed to process specimens in a batch if required. In fact many of hospital lab specimen processing equipment operate under batch processing mode, which makes this feature useful. Here again, we demonstrate the need for the developer to have a strong understanding of the domain knowledge in developing class objects. The batching process in the ***Hospital Workstation*** requires the user to set a simple

property to indicate batching and corresponding condition to configure the internally defined *BATCHING* element. The workaround would require the user to construct a combination of a **Combiner**, **Waiting Room** and **Workstation** object into a model to achieve the same functionality.

3.3.4 Hospital Workstation with Lab Care Class (HWS_Labcare)

Not all types of patient's secondary visits are specimen controlled; some require the patient to be present too. For example, a patient with a fractured limb is always directed to an X-ray imaging service and this requires the patient visit the service center to utilize service. The *HWS_Labcare* object class models these classes of systems in the SIMIO-HC simulation framework.

In any type of described facility, such as a stress testing center, MRI imaging facility among many others, the patient flow is standardized. As a patient walks in to the center, he or she has to undergo a preliminary check-up of their vital statistics or be briefed on the procedure. Once this is done, they are directed to the processing room where they utilize the service and leave. During their service time, some form of testing material is collected from patient. Following this, the patient is redirected to do as per their dispatching logic (discussed in *PC_Labcare*). The *Specimen Entity* which is created (hence the suffix Labcare) is directed to post-processing buffer for post processing work such as X-ray image development, EMG test report preparation, etc. and then this dispatched as per the aforementioned result dispatching logic.

In some services where the post-processing stage is outsourced. Outsourcing can also be modeled using the same object by allowing the release of the resource after *Patient Entity* leaves the processing stage. Thereby, the post processing is done without the use of the resource and capacity, and can be controlled by using the buffer size. In these types of facilities, there are some exceptions to the dispatching of results logic. There are some cases where patient reports are instantly handed over to them after the tests are performed, which

fall under *Wait at Current Object* redirection logic. To incorporate this combination, we have added an additional result batching logic to this class that implements the same functionality without the additional ***Waiting Room*** and ***Combiner*** logic.

Finally, the major addition to this object is the concept of “*Failures*”. Most of the machines modeled by ***HWS_Labcare*** are composed of a number of electro-mechanical parts whose reliability is often under question. Failures can also conceptually represent machine downtimes, such as when the operator is on a break or off-shift. Thus, failures of these facilities are known to be a major hassle for managers and play an important role in facility operations. The ***HWS_Labcare*** accounts for four types of failures⁴: Calendar Time, Processing Time, Processing Count and Event Count. The implementation method behind each of the failures is identical. A monitor is assigned to a particular event internally where the event specifications depend on failure type.

A “*Calendar Time based failure*” is a type of failure where the machine is bound to fail at a certain time. For example, this failure can be used to model a clinic in a remote region in the world that is known to have electricity shortages. It faces power cuts certain time of the day or a certain part of the day when a operator is scheduled to take a break.

A “*Processing Time based failure*” is where the machine fails after processing for a certain time. The time between failures is referred to as *Uptime between failures* and is specified by the user. For example, we can model a failure of a part of imaging device which would render the device to be failed and estimate its impact on the service. This information can be used by purchasing managers to make purchasing decisions and improve the facility profitability.

A “*Processing Count based failure*” is where the machine fails after processing a certain number of parts. The user is required to specify the count and is ideal for modeling

---

⁴ These failures are similar to the ones defined in SIMIO standard library objects.

cases where information on processing amount is readily available. For example, a facility manager can report the number of rolls consumed by the EMG machine before the printer fails, which also means the EMG facility has also failed

Finally, an “*Event Count based failure*” is for the failures that do not fit any of the other three categories listed above. The user specifies an event for which the machine must fail. This specification gives the user a chance to model complex failure logic, such as when a setup and post processing stages must fail, or when a compounded machine processing failure with count is present. All failures in *HWS_Labcare* are implemented using the built in *Timer* construct (Refer to Appendix A.III.5) where the *Timer* event executes Failure logic which consists of FAIL-DELAY-REPAIR step. The DELAY step of failure is non-interruptible which makes sense because you cannot use a failed machine for a time critical task.

3.3.5 Redirect Object Class (RO)

Having discussed the medical structures in the previous sections, we now turn our attention to one of the most important objects that keep these other objects glued together: the ***Redirect Object***. In SIMIO-HC, patient queuing is decoupled from patient processing queues to allow for more realistic simulation of the system (discussed in Section 3.3.1). The ***Redirect Object*** is a simple fixed object to hold patients and manage the routing of the patients using their preferences as well as taking the clinical preferences into account. Developers must try to provide as much flexibility as possible in the patient routing stage in order to allow for a wide range of modeling applicability because it is impractical to enlist all the options. Nonetheless, when providing flexibility, the developer must provide structure to the logic creation in order to reduce logical errors.

In SIMIO, the flexibility expression is obtained by using a built-in construct of *Selection Goal*, *Selection Expression* and *Selection Condition*. The *Selection Condition* determines which objects can be evaluated by the *Selection Expression*. The *Selection*

Expression evaluates the pruned list and follows the rules selected in the *Selection Goal* stage. The *Selection Goal* is an enumerated list of evaluation criterion which consists of Smallest Distance, Largest Distance, Cyclic, Random, Smallest Value and Largest Value. As an example, we might select the nearest clinic provided it is empty. To model this logic, we will write a *Selection condition* to evaluate if the clinics are empty and then select *Selection Goal* to smallest distance for the *Selection expression* to select the nearest one among the ones that were selected in the *Selection condition*.

An “*enumeration*” is a list of symbolic names for a set of values (or functions). In SIMIO, an “*enumeration*” is treated as a data type. Enumeration reduces errors due to mistyping or mistakes in defining the function correctly since user has to choose a value from a pre-defined list. SIMIO provides a large selection of enumerations which offer the user considerable flexibility and functionality as well as making the software robust to use. Although in SIMIO-HC, we did not develop any custom enumerated data types, we did extensively use it in the toolkit development to improve SIMIO-HC readability and forward compatibility.

For the routing of **AGENTS (ENTITIES and TRANSPORTERS)**, the *Redirect Object* has two options:

- **Transfer node** from the standard library (Refer to Appendix A)
- *Hospital Transfer node* (refer to Section 3.4.1)

The reason we needed to provide two transfer nodes was based on tradeoff between developer’s priorities versus modeler’s convenience. In other words, we could have designed the *Hospital Transfer node* to incorporate exceptions for entities that just pass through the node and provide a different routing logic for the **TRANSPORTER** class as well as exception for batching logic (discussed in Section 3.3.2). This would require the creation of a new **NODE** class over a simple modification to the subclassed **Transfer node**. Moreover, the simple design of this object allows us to incorporate the batching logic and an another

Transfer node to handle the rest of the exceptions making it much easier to create the object to represent administrative functions such as a Cashier's Desk, Waiting Rooms, Doctor's office, etc. In fact, some of them have been implemented in this release of SIMIO-HC.

Secondly, the implementation of separate nodes is also advantageous for the modeler. In hospitals, the travel pattern of hospital employees is very different from a patient's travel pattern. By separating the **WORKER/ TRANSPORTER** path networks from **ENTITY** path networks, the modeler can govern the flow of both of these intelligent agents using separate rules. Furthermore, in directing this flow, it also reduces the logical complexity of using these objects which enhances the utility of the object.

3.3.6 Waiting Room (WR)

The decoupling of patient waiting from processing requires that the toolkit provide an object to facilitate modeling of patient waiting. Waiting is a primary factor of interest in the majority of hospital facility simulation. The ***Waiting Room*** class is subclassed from the ***Redirect Object*** class since the ***Waiting Room*** object possesses the same internal and external structure. However, the class's implementation of the internal structure is slightly different in order to provide more specific functionality to emulate waiting rooms namely: renegeing and balking.

A patient is said to renege if he or she leaves the waiting room after waiting for a certain amount time. A developer can easily provide constructs to facilitate modeling of this logic using the concept of a delay followed by logical evaluation of condition under which a patient decides to renege, such as number of people ahead in the queue, time waited, etc. Here again, we use the expression construct to allow for the modeler to define their own renegeing logic. On the other hand, a patient is said to be balking if they refuse to wait at the facility upon arriving. In simulation modeling terms, the developer must destroy any entity that arrives when the waiting room capacity is zero. In the ***Waiting Room*** class, the user has

to set the *Allow Balk* property, as well as change the *Initial Capacity* property to some real number less than infinity to emulate balking.

By providing these advance constructs is not helpful unless we can generate the correct type of statistics for the system automatically. To generate statistics pertaining to reneging, balking and waiting; one has to be familiar with the concept of “*Tally*” and “*State*” statistics. “*Tally statistic*” is an observation-based statistic or a statistic that weights each entity equally. *Average Waiting Time* is a tally statistic since we sum the individual entity waiting times and divide by number of entities in the above summation. Mathematically it can be represented as:

$$\text{Average Waiting Time} = \frac{\sum_{i=1}^N (\text{Waiting Time})_i}{N}$$

On the other hand, a “*State statistic*” is a time persistent statistic which means we average the time-weight observations over total run time. The *Average number waiting* represents the average number of entities in the waiting room at a given time is an example of state statistic as seen below where T stands for final simulation run time

$$\text{Average Number Waiting} = \frac{\int_0^T \text{Number Waiting}}{T}$$

There are lot of different types of these statistics that a developer can provide, but one must keep in mind the golden rule of output generation which states not all of output is useful to the user. Hence, a developer must assess the usefulness of the output statistic and see if it has any practical use before investing time in developing logic to generate it.

3.3.7 Registration Desk (RD)

In 2007, the administrative costs were estimated to be \$156 billion dollars and are projected to rise to \$315 billion by 2018 (IOM, 2010). Woolhandler, Campbell, and

Himmelstein (2003) performed an extensive study which reported an inverse correlation between complexity in administrative services and quality of service. Given the magnitude and impact of administrative services on healthcare, design of an administrative class is a necessity within a healthcare simulation framework. In SIMIO-HC, the **Registration Desk** object is a representative class of administrative structures in the hospital like billing desk, records, reception, etc. At an administrative service, patients can queue up in front of the desk and wait for their chance to be served. Although the functional requirements of such an object might inspire the developer to develop a new class, it can be avoided if the developer carefully designs the **Waiting Room** class.

The **Registration Desk** object's external and internal structures (i.e. stations, queues, states and properties) are exactly same as the **Waiting Room** so we can extend the **Waiting Room** object by subclassing. However, the implementations of these constructs are significantly different. Patients wait in the input buffer from where they can decide to balk or renege. Secondly, when they move to the processing buffer, they undergo service as compared to waiting in the **Waiting Room**. Furthermore, we need to modify the placement of statistics collection logic to account for the variations. This addition is a classic example of how inheritance can greatly reduce the complexity in development process. This concept is described in Guideline 3.

Guideline 3: *The incremental complexity of class development reduces with increasing depth in object hierarchy.*

In other words, it becomes easier to add a layer of logic to a class over developing it from scratch. The graphical representation of this concept is presented in Figure 3-7. Combining this guideline with Guideline 1, we can develop the tradeoffs between utility, complexity, and number of objects, as seen in Figure 3-8. Based on the graphical representation, we can say that the marginal utility of the class diminishes with hierarchy.

Furthermore, we can say that the utility of a class can also be determined by complexity and object number.

Figure 3-6: Relationship between Class Hierarchy and Its Complexity

Another implication of the Figure 3-8 is that it conveys the need for the base class to be simple, yet encapsulate the essential methods and variables with a fixed structure that can be modified to generate different behaviors. For example, a simple ***Redirect Object***'s structure was used to develop ***Waiting Room*** logic whose logic was only slightly modified to develop ***Registration Desk*** logic. If we extend the hierarchy by one more level, we can develop a ***Billing Desk*** class by just modifying reneging and balking properties.

Figure 3-7: Utility of Class as a function of Complexity and Object Count

3.3.8 Statistics Collector Class (SC)

The primary objective of a simulation is to understand the system, and sometimes from the patient perspective. In all of hospital operational simulations reviewed in Section 2.2, the factor of interest has often been to improve some aspect of patient's treatment process like the response time for treatment, reducing turnover time from lab, and so on and so forth. In order for a simulation toolkit to provide such statistics, it should have either classes or functions capable of computing these statistics. In SIMIO-HC, the *Statistics Collector* class is designed to collect these types of predefined statistics automatically from the *Patient Entities* that are directed to exit the system through this object. The definitions of pre-defined statistics can be found in Section 1.5.1. There are two versions of tally statistics that should be collected here for each statistic: “*Total Time*” and “*Percent Time*”.

“*Total Time*” is defined as average time spent by the patient population in a specific state. Total Time statistic is an absolute value and gives the modeler an estimate of time spent by the entity at each stage. However, total time by itself doesn’t offer much insight in to system performance unless you compute the “*Percent Time*”.

$$Percent\ Time\ in\ State\ (i) = \frac{Total\ Time\ in\ State\ (i)}{\sum_{i=0}^n Total\ Time\ in\ State\ (i)}$$

“*Percent Time*” is defined as the percentage of time spent at state over the total time in the system. This statistic helps the modeler understand the relative time spent in a state and help capture the internal dynamics of the system. For example, if you reduce the turnover time of your lab work, it might increase patient’s waiting time as the doctor’s might be overburdened. This result may be evident from the Percent Time statistic. In SIMIO-HC, both of these statistic groups were computed using a TALLY step just before the entities were about to be destroyed.

3.4 Node

The node class represents a class of objects that allow a physical connection between fixed objects. The standard library treatment of this class is discussed in Appendix A .Hence we restrict our discussion to the subclassed *Hospital Transfer node*.

3.4.1 Hospital Transfer node

In SIMIO-HC, we have separated entity queuing from processing in order to create waiting spaces that emulate hospital settings. However, consider the situation where we have three clinics which have a common *Waiting Room*. If we allow the entities to queue first come first served (FCFS) basis and route them on that basis to the nearest one that is free, the system will automatically assign every entity as soon as it enters the object to a clinic. In other words, the entities are virtually waiting in front of a specific clinic, just not visually. Given the stochastic nature of a simulation, this order might not prove be correct (be in

increasing stochastic order of waiting times) and the entities will not be routed on first come first served basis. This concern inspired us to design *Hospital Transfer node*.

The *Hospital Transfer node* class is directly subclassed from the **Transfer node** class with the ability to regulate the flow of entities on first come first served basis by controlling the node's transfer capacity. By regulating this capacity we can essentially shut the node down if all the clinics are allocated and ensure entities have to wait at the node till the first one is freed to start the allocation.

In SIMIO, node capacity regulation can be implemented by manipulating the *Current Capacity* state variable. Every time an entity crosses this object, the *Current Capacity* decrements by one, and every time an entity finishes service at the corresponding patient processing object, the *Current Capacity* is incremented.

This section brings us to our next conceptual question: When to subclass? To frame it differently, when to create objects via inheritance? Subclassing is applicable if and only if the changes to derive the new class are minor. In addition to this, we strongly urge the developer to also consider the implementation of all the inherited methods and variables from the parent class. If not, these methods and variables (as you cannot delete them in a subclass) are going to be a burden for the subclass as they increase the size of the class without any value addition. This size increase will only further reduce the class's utility, given that complexity of the class will rise and inhibit usage or subclassing. In the case of *Hospital Transfer node*, we did not add any new methods or variables. We only modified the On Exited process using inherited variables to produce this new behavior, making subclassing perfectly acceptable in terms of utility and complexity.

3.5 Transporter Class

The transporter class refers to those objects that can move freely in three dimensions during model. Subclassed from **AGENT** class, the **Transporter** class represents everything

that can move in a hospital environment. The transporter class can be divided in to two sections: *Hospital Transporter* and *Hospital Worker*.

3.5.1 Hospital Transporter Class (HT)

The *Hospital Transporter* class represents those objects that can transport patients in and around the hospital environment like a stretcher, wheel chair, ambulance, etc. Agent based design is clearly the most efficient and effective design paradigm for a freely moving object. In this section, we specifically discuss the conceptual design of routing algorithm for agents and its implementation in SIMIO. In order to create an efficient algorithm to define the movement of an **AGENT**, we need to break down movement logic in to a set of sequential decisions for the object as represented in the Figure 3-8.

The first and most obvious one is to check whether the node the where the transporter is currently located is the destination node. If not, then it can only mean that the transporter is in transit and has to pass through this node to an outbound link. If it is the destination node, then the transporter has to park in order to avoid the node from blocking the movement of other agents. In reality, if a stretcher is dropping off or picking up a patient do they perform this activity in the hospital hallway? In the *Hospital Transporter* class, the logical checks are implemented using the DECIDE step and the parking is executed by PARK step which are built in SIMIO. The next step is to check for the one task type which could either be pickup or drop-off.

If it is a pickup task, the transporter checks if the patient needs the transporter to operate in assistance mode. In assistance mode, the *Hospital Transporter* needs to seize a *Hospital Worker* in order to execute its functions. In the hospital context, you can visualize this situation as patients who need stretchers being are pushed around by a *Hospital Worker*. In SIMIO-HC's *Hospital Transporter* class, the assistance mode is triggered if and only if the *Seize Assistance* state variable of the *Patient Entity* class is set, which indicates the patient needs assistance and the selected transporter is defined as *Assisted Type*. We

assume that the modeler is familiar with selection condition construct of SIMIO to define an expression that will select an *assisted transporter* from a list of transporters if not all are of the assisted type.

Figure 3-8- Routing Logic Flowchart of Hospital Transporter

By defining the ***Hospital Transporter*** as an *assisted type*, the user must also define the resources that need to be seized by the ***Hospital Transporter*** in order for it to transport the patient. In reality, an ambulance is usually associated with a paramedic team who transports patients to hospital. If assistance is not required or if they have already been seized by the transporter, the next logical step is to pick up the patient and transport the patient its destination. In SIMIO, we can implement the logic using a combination of a PICKUP and SET NODE steps. Following these steps, the ***Hospital Transporter*** exits the node and repeats the same logic from the start at every node to determine if it is the destination node.

Given that the patient's is not being picked up and the ***Hospital Transporter*** is at the destination node, the only option left for the transporter is that current location is the drop off location for the rider. In order to transfer a patient, the patient's destination location has to be available to help the patient. If it is not free, then it parks and waits till the resource becomes free to take the patient to the destination. With the assisted type, even the seized workers will wait till the patient is dropped off at the location.

Table 3-5: Agent Class Resource States

State	Description
<i>Idle</i>	The agent is not performing any action
<i>Busy</i>	The agent is currently being seized by another resource
<i>Blocked</i>	The agent's is unable to perform its function
<i>Transporting</i>	The agent is currently transporting another agent
<i>Waiting For Handler</i>	The agent is seeking assistance
<i>Off shift</i>	The agent is on a schedule and at present its capacity is 0
<i>Off shift transporting</i>	The agent current capacity turned 0 while transporting.
<i>Off shift busy</i>	The agent current capacity turned 0 while servicing

Following the patient transfer, the transporter decides whether it should park at the current location or go back to its home node (i.e. location where we initialized the transporters). If it is the former option, the workers are immediately released if the vehicle is an assisted transporter and has seized medical workers. If it is going home, then the resources stay with the assisted transporter until they reach the home node. In SIMIO-HC's design of ***Hospital Transporter*** class, the drop off logic was implemented using the DROPOFF step to transfer the ***Patient Entity*** to its destination location.

Finally, we need to discuss the statistics collection on these types of agents. Just like we did in the **FIXED** class, a list state was used to implement statistics collection. The states are defined in Table 3-5 . There are four states; *Busy*, *Off shift*, *Off shift busy* and *Off shift*. Transporting will not be utilized in the ***Hospital Transporter*** class but will be used for implementing the ***Hospital Worker***.

Figure 3-9 : Utility Function as a function of Run Speed and Complexity

Now, we extend the concept of class complexity and size to simulation run speed. Simulation run speed is an important factor in complex facility simulations and is inversely correlated to model size. Model Size is linearly correlated to class complexity. Therefore, the graphical relation between Model Size and Run speed is shown in Figure 3-9. This concept is described in Guideline 4.

Guideline 4: *The utility of the model is a tradeoff between run speed and model complexity.*

From the conceptual graph, we observe that as complexity increases, run speed slows at an exponential rate. This decline affects the class utility as it inhibits the run time performance of complex simulations. In the case of agents, this factor plays a critical role in the design of the simulation model. If the run speed is extremely low, it hinders experimentation and ultimately the value of the simulation model.

3.5.2 Hospital Worker (HW)

Healthcare, in particular, is a human intensive system where human decisions, behavior and setup make a huge difference in how the healthcare system operates. Many of the problems that are usually handled in the reviewed literature deal with staffing problems. In fact, you cannot model a healthcare system without having a human component, which brings us to the discussion on design of the ***Hospital Worker*** Class.

A ***Hospital Worker*** can be thought of as a resource but one that can move around in the system as well as be seized to execute a process. It should have the functionality to allow the user to define schedules for the workers because that is how medical workers are staffed. Furthermore, there are situations like medicine rounds, doctor rounds require a different type of routing scheme which a ***Hospital Worker*** class must provide. Lastly, there are conditions where the ***Hospital Worker*** may help to carry out the task. From this description, it must be evident to the reader that the design specifications closely match the ***Hospital Transporter*** class. In fact, with the exception of routing, schedules and additional capabilities to execute

processes, a worker is a transporter for the simulation model. Hence, the ***Hospital Worker*** class can be subclassed from the ***Hospital Transporter*** Class.

In order to provide the additional routing logic to model rounds, scheduled deliveries can be implemented using the concept of *Routing Sequences* where the worker has to follow a list of nodes and serve them in the order specified. In SIMIO, this can be implemented using the *Sequence Table* that is used to specify a *Fixed Routing*. By default, the ***Hospital Transporter*** class only moves when an entity or object seizes them.

To implement worker schedules, we need to modify the design of the allocation of capacity of the worker. In other words, we have to actually check if the worker is active (or is on-shift). If it is, then we allocate its capacity. In some cases, the worker's shift may get over, or he or she is busy with a patient for which it is logical to make the assumption that the worker goes off-shift only after finishing the job. Nonetheless, this overtime must be recorded by the system for such events. In order to implement this capacity allocation logic, we modify the *On Capacity Allocated* process of the *Transporter* class and for the overtime statistics we utilize the pre-defined states of *Off-Shift*, *Off-shift Transporting* and *Off-shift Busy*.

3.6 Entity Class

The **ENTITY** Class models freely flowing agents that can be dynamically created and destroyed in the system. With respect to SIMIO, an **ENTITY** can move in and out of **FIXED** object, travel on networks created by nodes and links as well as ride on **TRANSPORTER** objects. These agents in simulation represent transactions in other simulation languages, and in SIMIO-HC, entities represent patients, specimens, and paperwork.

3.6.1 Patient Entity Class (PE)

The ***Patient Entity*** class, as the name suggests, represents the patients in a healthcare system. This class is central to a design of an effective healthcare simulation toolkit, given

the fact that every healthcare system's operational efficacy is also strongly influenced by patient behavior, such behavior should be replicated by the logic embedded in this class. Another factor that must be given attention is to describe the stages that each entity has undergone. This information provides a meaningful breakdown of the system's performance.

As defined earlier, agent's behavior is governed by model structure. For example, what are the options available to the patient, once he or she enters a **Waiting Room** where the **Waiting Room** provides model structure context. The entity's behavior in the model can either be defined inside the agent (which can be referred to as "*agent centric*") or be embedded in the system structure (which can be referred to as "*system centric*"). Applying agent centric development to earlier example, the **Patient Entity** realizes that he/she has just entered an object of **Waiting Room** type and triggers the **Waiting Room** logic inside the **Patient Entity** class. In system centric development, the **Waiting Room** object realizes that a new **Patient Entity** has arrived and executes processes that are associated with entry, dependent on patient's properties such as health condition, preferences etc. Nonetheless, the end result of both types will be the generation of an intelligent agent. But the process inside the simulation engine is different and has an effect on the performance of the simulation model.

All types of processes and methods increase complexity of a class which directly correlates to an increase in class size and inversely correlate to run time performance. Embedding the logic inside an agent class would mean a heavier agent and during the simulation run if we generate hundreds and thousands of them, then the simulation run time performance will significantly deteriorate as illustrated in (Figure 3-9). This relationship will not facilitate complex systems simulation given that variability in simulation performance will be high.

On the other hand, if we define the logic inside a system structure then the agent class size would be small and result in a heavier system structure class. This system design would perform significantly better than the aforementioned design given the fact we have only finite

number of these system structures and the size of entity class is negligible, so it allows us to determine the size of the model. This methodology ensures that even in complex systems simulation, the run time performance will be deterministic and have a fixed upper and lower bound adding stability to our simulation model. Nonetheless, this type of design requires that we accurately keep track of agent states and properties which can be implemented using states, events and properties. In a nutshell, the system structure defines the agent behavior.

The two approaches have their advantages and disadvantages. Encapsulation of methods and variables inside an agent class allows for centralization of control as compared to the alternative method which decentralizes the agent control. Secondly, the former method does not require the developer to track state references, whereas the latter method depends entirely on the state references. If the referencing is not implemented correctly, the agent behavior will degenerate. Lastly, the system centric approach shifts the burden to the developer as they have to be careful in maintaining the right sequence of state referencing.

The next issue in the design of the *Patient Entity* class is the state definitions which help in quantifying the time spent by the patient population in a certain activity state for the operation system under simulation study. As stated in earlier sections, the patient activity states help in understanding the system's performance as well as assist in comparing different system outcomes such as waiting times, on time performance etc. For this, a patient's activity can be broken down in to resources states as shown in Table 3-6.

In SIMIO-HC, the *Patient Entity* class has been subclassed from **AGENT** class with activity states added in as a list state resource. The *Patient Entity* class has been designed for system centric framework where the *Patient Entity* frame is used to hold the current patient state and resource state references. A process for storing object references was developed which is based on the current activity state of the *Patient Entity* which can store preferences of the patient such as which previous medical worker diagnosed the patient, which ward was assigned to the patient, which wheel chair was assigned to patient, among many others as described in Appendix B.I.

. Table 3-6: *Patient Entity* Activity States

State Name	Definition
<i>Travel</i>	The entity is on transit
<i>Riding</i>	The entity is on transit and is using a transporter
<i>Treatment</i>	The entity is undergoing medical procedure
<i>Administrative</i>	The entity is performing administrative work
<i>Waiting for Medical Resource</i>	The entity is waiting for an object that can perform medical care
<i>Waiting for Non- Medical Resource</i>	The entity is waiting for an object that can perform administrative work
<i>Waiting for Medical Personnel</i>	The entity is waiting for one of the medical personnel resource group
<i>Lab Turnaround Time</i>	Throughput time for the lab specimen
<i>Radiology Turnaround Time</i>	Throughput time for the radiology system

3.6.2 Specimen Entity (SE)

For every patient who enters a healthcare service, there is paperwork generated. These paperwork entities represent patients and most of the processes in the hospital await paperwork to begin service and generate paperwork at the end of service. Secondly, a majority of medical service providers heavily depend on secondary medical services such as lab testing, radiology, stress tests, etc. to assist them with their treatment diagnosis and action plans where there is usually a specimen extracted from the patient. This specimen undergoes processing in a different environment, where a patient sometimes is not required to be present. For post-processing, the specimen gets converted in to paperwork which should be

then matched with the right patient. In both cases, the agents are replicas of patients and a good healthcare toolkit must provide provisions for modeler to model these processes.

In SIMIO-HC, we developed this class from the *Patient Entity* class given that this entity should be a logical replica of the patient. However, this class must have intelligence to identify its patient (parent) to whom it belongs. As the SIMIO-HC framework development can be categorized as system centric, we defined state reference variables inside the *Specimen Entity* class that references to the *Patient Entity* (*Parent Entity Reference*). This state variable is referenced in *Lab care* and *Redirect* class structures to match a patient with the corresponding paperwork. During the time the patients are waiting for results can be also recorded in the *Patient Entity* under the resource state label *Lab turnaround time* and *Radiology turnaround time* which are standard performance measures for lab and radiology services.

3.7 Chapter Summary

To sum it up, this chapter presents a conceptual framework for healthcare simulation toolkit. We created a class hierarchy for healthcare simulation along with their implementation notes in SIMIO (detailed SIMIO guidelines for these object classes are provided in Appendix B). In the process of developing of classes, we also developed four axioms that characterize the utility of simulation toolkit using Complexity, Usability, Model Run Time and Object Count. These four axioms will help future simulation toolkit developers to carefully think about design and implementation of classes in order to maximize their simulation toolkit's utility and also its adoption rate.

Chapter 4 Complex Integrated Multi-Facility Systems Simulation

This chapter will be a discussion of modeling approaches to develop complex integrated multi-facility systems simulation (CIMS) in healthcare settings using the SIMIO-HC toolkit (see Chapter 3). CIMS simulation models are composed of multiple facilities which are dependent on each other. These simulation models are more realistic representations of modern healthcare settings as no healthcare facility works by itself. Furthermore, the productive interaction between facilities is critical to provide quality medical service to patient. CIMS simulation models break down the traditional tight boundaries assumptions of individual facility simulation models and facilitate analysis across the whole healthcare system to study the systematic inter-dependence of healthcare facilities which is extremely important.

4.1 An Introduction to CIMS

Even though CIMS is important, only 3% of the reviewed literature dealt with complex systems simulation of healthcare facilities and are in line with earlier findings (Günel and Pidd, 2010; Jun, Jacobson, and Swisher, 1999). During the same time frame {1999-2012}, over 17,000 papers were published on complex systems analysis which indicates a significant surge in interest in the community⁵. Hence, the findings imply that a lack of an effective framework for implementation of CIMS simulation projects is inhibiting simulation modelers from building CIMS simulation models.

In every simulation project, the modeling team has to juggle four factors that influence the value of simulation model: Credibility, Fidelity, Verification, and Validation. “*Credibility*” can be defined as the stakeholder’s belief in the simulation model. “*Fidelity*” is defined as the modeling detail, often determined by the modeling team. “*Verification*” term captures the process of ensuring the model mechanics are working correctly inside the model.

---

⁵ The growth in interest was estimated by a Google scholar search with (Complex*adaptive *systems*healthcare) keyword which resulted in 17,000 academic papers published between the years 1999-2012 in business, engineering and social sciences categories

“*Validation*” is the process of ensuring that model is representative of the system under question. These objectives are interlinked and their relationship affects the implementation of CIMS simulation models as explained in Section 4.3.

There are two implementation methodologies for CIMS simulation models: The Big Bang approach or the building block approach. In the Big Bang approach, we build one simulation model of the entire system at a time. This approach is laborious and cost intensive making it an unattractive option for CIMS. On the other hand, the building block approach constructs CIMS model by composition of two or more facility specific models. This approach allows the simulation project to be broken down into smaller units of individual facilities. Each of the individual facilities are modeled, verified and validated independently and then combined in their final stages. Hence, we can leverage the expertise in facility specific modeling of the simulation community to develop CIMS which eliminates the potential for errors.

Nonetheless this approach has its disadvantages. It is heavily dependent on software and class structures. It is only applicable for simulation languages that are object-oriented and among them only to those that allow object composition and inheritance methodology. This currently restricts the simulation language set to SIMIO. Furthermore, this approach requires a customized framework for the modeling environment in order to standardize facility specific models so that they can be integrated to build CIMS simulation model. In our case, we developed SIMIO-HC framework for healthcare simulations in SIMIO which facilitates the adoption of the building block approach to develop CIMS simulation model using both composition and inheritance.

4.2 Building CIMS Model⁶

To clarify the usage of the toolkit in context of CIMS model building, we have implemented a simulation model using the building blocks approach. We developed an

---

⁶ Detailed Step by step Model building instructions is provided in Appendix C.

Emergency room model, Oncology Clinic, Radiology Services and Laboratory Services facility models and combined them to create a CIMS simulation model of a hypothetical hospital (see Figure 4-1).

Figure 4-1: Hypothetical Hospital Model

4.2.1 Emergency Department Model

Emergency Departments (ED) play a critical role in public health infrastructures and is often the first point of medical contact for patients. Patients who are admitted to the emergency department often are in an acute condition which means the department needs to be responsive. Among the surveyed papers, emergency departments accounted for the largest number of facility simulation projects which also indicates a strong interest among the simulation community as well. Given the time critical nature of the department, emergency department simulations typically deal with staffing, capacity and process flow problems. Each of these factors has been discussed in Section 2.3.2.

In this section, we build a model of a generic emergency department (ED) using SIMIO-HC toolkit and discuss the modeling capabilities of SIMIO-HC (Figure 4-2). We

assume that patients can either arrive in an ambulance or walk into the ED. The arrival rate varies over the day. Given this variation, it is best represented as a Non-Homogenous Poisson Process (NHPP) whose arrival rate is a function of time. All patients, except for the ambulance delivered ones, are usually directed to the **Registration Desk** object which in our model is manned by two receptionists. Following registration, they move on to wait in a **Waiting Room** object where they may decide to leave the service after waiting for a certain time (i.e. renege). Also, if the **Waiting Room** is full, the hospital is required to redirect the ambulance to another hospital. The demand lost from both of the above cases is important for capacity planning of the hospital. This data is automatically captured by an output statistic in this ED model.

Figure 4-2: Emergency Department Conceptual Model

The patients who are accepted into medical service are allocated either a room or bed depending to availability and other requirements. In our model, we choose the nearest **Patient Care** object that is available to serve the patient which models the room and bed in our model. Following their allocation, the patients wait in their bed until a service provider can visit them. The service provider's staffing levels are critical here and usually the focus

point for many simulation projects. The *Hospital Worker* class which is used to model this feature can use fixed or time varying capacity to satisfy staffing modeling requirements.

After the consult, the patients are directed to leave the emergency room. Sometimes, they are discharged which means they will be directed to billing and insurance related activity stage that is modeled using the *billing desk* object. Sometimes, they need to move into another department of the hospital for care treatment which is only captured in the CIMS model. Nonetheless, in both the cases patients leave the ED.

The above model can also be developed using the standard library where we follow the natural SIMIO approach. In the natural SIMIO approach, entities will wait for service in input buffers of the server objects. In order to represent registration, *Patient Care* and billing objects we use servers with 2, 8 and 3 capacity respectively. The *Hospital Workers* can be represented as **workers** who can be seized by a patient care **server**. The balking and reneging could be represented as processes *on entered*. Functionally, the model described here and the one above be similar with few more simplifications in order to account for the modeling challenges that were listed in the Section 3.3.1. Nonetheless, this simplification affects the “*credibility*” factor of the simulation models.

“*Credibility*” factor can be defined as the stakeholder’s belief in the simulation model. In the above described SIMIO model that utilizes standard library, the model does not provide visualizations for patient balking, reneging, different treatment locations as well as accounting for different patient treating times. Secondly, the model simplifies the clinical allotment process as well as seizing and releasing of workers (no animation is provided). To provide this animation would require an extra set of nodes added to the patient care server objects which would make them similar to structure of *Patient Care* object class. In a nutshell, a layman who is not familiar with simulation will not be able see how the standard SIMIO model accurately represents the system. Hence, it will be difficult to engage them in the model building process, which will affect your model utility and restrict its scope.

If an explicit approach like the one using SIMIO-HC is followed, the management can see the model; which will translate into a belief in the simulation model (i.e. that it is a representation of their system). This approach will help in engaging the client in the model building and deployment phases, which will improve the modeling utility. This concept is captured in the “*Credibility*” loop as seen in the Figure 4-9.

4.2.2 Oncology Clinic model

The oncology clinic model presented in this section is a good example of an outpatient clinic simulation model (see Figure 4-3). Outpatient clinics are usually specialized in the services they offer. Typically, these clinics deal with patients who have appointments. Furthermore, the specialization among the outpatient services allows a patient to choose the services they need, unlike the emergency department process. Hence, in addition to the list of simulation objectives presented for emergency department simulation, outpatient clinics also provide appointment scheduling and patient services scheduling.

Figure 4-3: Oncology Department Conceptual Model

In our clinic, there are four types of patients: Consult Patients, Treatment Patients, Treatment with Checkup Patients, and Follow up Patients. A consult and follow-up patient types have to visit the doctor. The treatment patient type visits the clinic to get his/her chemotherapy treatments. Only the treatment with checkup patient has to use all of the clinic's services (i.e. chemotherapy followed by doctor consult). Nonetheless, all arriving patients have to check-in at the check-in desk which is handled by non-medical personnel. All types of patients have to check in at **Registration Desk** object.

Following the check-in process, the consult and follow up patients are redirected to the clinic's **Waiting Room** object and the other two patient categories are redirected to the treatment **Waiting Room** object. At the clinic's **Waiting Room**, patients need a nurse to escort them to an available clinic room where the nurse collects patient history details since the last visit. Following that, a doctor visits the patient for the medical consultation where the process time is dependent on the type of visit. The field of oncology is heavily dependent on the secondary services of laboratory testing and medical imaging to form diagnosis. Hence, after the consultation process, the doctors may decide to redirect the patient to either one of the secondary services, if the patient has certain indicators. However, this service cannot be modeled at the facility-specific model so we redirect them to a **Sink**. If none of the indicators are present, then the patient is redirected to the **Billing Desk** object and exits the clinic through the **Statistics Collector Sink**. The patient flow diagram for this category of patients is presented in Figure 4-4.

Figure 4-4- Check Up and Follow Up Patient Flow

The treatment patients and treatment with checkup patients wait in treatment a **Waiting Room** until a Chemotherapist (i.e. **Hospital Worker**) escorts them to one of the treatment rooms (i.e. **Patient Care**) and initiates the chemotherapy therapy. Following that, the Chemotherapist returns to the nurse station if there are no more waiting patients or if all the treatment rooms are occupied. After treatment, the nurse is required to visit the patient room to remove the central line and discharge the patient if they are treatment patients. For treatment and checkup patients, they are redirected to the clinic **Waiting Room** where they follow the consult process. The patient flow for treatment and treatment with checkup is presented in Figure 4-5 and Figure 4-6 respectively.

Figure 4-5- Treatment Patient Flow

Figure 4-6: Treatment with Checkup Patient Flow

Each group of treatment patients have their specific treatment times at each stage, which can be accommodated with the use of data tables and treatment time references. The nurses who have to place the central line require a certain amount of time to execute this task. The time can be modeled as an unloading time as the patient is transferred to the bed after which the nurse places the central line. The routing of the patient is dependent on their type, which is reflected in the network structure.

Building on the concept of credibility, by engaging the system stakeholders indirectly helps the “*verification*” process. The “*verification*” process ensures that the simulation model is built correctly. In order to verify a simulation model, the simulation modeler must institute a review process that ensures that model building is supplemented by activities that ascertain the exactitude of simulation model structure. In a nutshell, the simulation modeler checks if the simulation model is an accurate translation of modeling objectives⁷. This concept is represented as the “*Verification*” loop in the simulation project Figure 4-9.

4.2.3 Laboratory Services

Laboratory service belongs to the class of secondary treatment facilities which have become an indispensable part of the modern day healthcare system delivery. In order to serve the patient, these facilities have to manage their supplies, in addition to all the problems of clinical departments.

The hypothetical laboratory service (Figure 4-7) operates from 7 am-4 pm and is open all days of the week. The patients arrive according to a Non Homogeneous Poisson process during these hours and all arriving patients are redirected to registration service which is staffed by a single receptionist. Using SIMIO-HC, we can accurately represent this stage using a **Registration Desk** object. After registration, they proceed to the waiting room and patiently await their chance which can be modeled using a **Waiting Room** class. We assume that there is no difference among the patient’s needs, so the **Waiting Room** operates on a first

---

⁷ Modeling objectives is a translation of customer requirements

come, first served basis. From the **Waiting Room**, patients are accompanied by a non-medical staff to the nearest station where the patient's test sample (specimen) can be obtained by a nursing staff member. A specimen is supplied; the patient can stay in the room, wait in the **Waiting Room** or leave the service center as per their wish. This stage requires that we use the **PC_Labcare** class and take advantage of its features. The **Patient Entity's** state variable is used to track the entities result options.

After the specimen is obtained, it is sent to the lab where the specimens are processed in batches of six. The specimens are batched and put through a setup where some materials are used to transform the specimen into another form for processing. The processing stage processes this batch and consumes some more materials to process the specimens. After the processing is done, the batch is un-batched and results are prepared and dispatched to the corresponding destination (i.e. dependent on patient). SIMIO-HC's **Hospital Workstation** is used to model this stage and can handle all the modeling requirements.

Figure 4-7: Laboratory Department Conceptual Model

The facility simulation model developed in this section is undoubtedly intricate and complex, which demonstrates the importance of model verification processes. Verification of the simulation model allows the simulation model to be validated. “*Validation*” is a process of

building the right simulation model. In order to validate a simulation model, one needs establish that the statistical insignificance between simulation model and the actual system performance measures. In order to establish validity of a verified simulation model requires data integration in to simulation model.

In this case, the simulation model requires data feed on setup times, processing times, and result preparation times for the laboratory work in addition to the patient specimen collection times and flow structures of both patient and specimen entity. Integrating this data in to the simulation model will help in computing if the current model is an accurate representation of the current system. In order to collect the right data for simulation model requires the assistance of medical system stakeholders who must believe that the simulation model is an accurate representation of their system. Hence, in addition to verification, we also need to establish credibility of the model to initiate the validation process of simulation model. By validating the simulation model, will allow us to experiment with it and provide alternative solution. Without Validation, the modeling insights cannot be considered reliable. This loop is represented as “*validation*” in Figure 4-9.

4.2.4 Radiology Services

Radiology services belong to the category of secondary healthcare services but are prone to face facility maintenance issues. The facility modeled here provides three services: X-ray, Gamma Imaging, and MRI Imaging (see Figure 4-8). Patients who arrive at the center are looking to use one of the three services. They wait their turn in a waiting room where a non-medical staff accompanies them to their selected service. Once they enter the service, the facility manager takes over. The waiting room is modeled using the ***Waiting Room*** class. The workers in the model are represented by a ***Hospital Worker***. The whole imaging service can be modeled by the ***HWS_Labcare*** object.

Each imaging service has to prepare their patients before they imaging process can begin. The preparation part is modeled using the *setup* options of the ***HWS_Labcare*** object.

Once prepared, patients are imaged in the facility, which is modeled by the *processing* options of the *HWS_Labcare* object. Post-processing, except for MRI service images, take some time to prepare. For MRI, the results are always handed to the patient immediately, given the fact that the images are digitally captured. A patient who opts not to wait to collect results, will have their results mailed to their homes. This choice is modeled using *Patient Entity*'s state variable and facilitated using the *result batch options* in the *HWS_Labcare* class. For the X-Ray machine, it is prone to failure and we assume that we are given the count data on the number of patients processed between failures. This also is accounted by the *HWS_Labcare* failure option under *Processing Count based failures*.

Figure 4-8: Radiology Department Conceptual Model

Suppose a modeler tried modeling this facility using a non-healthcare oriented framework, the complexities will overwhelm the modeler. It has been well-established in an earlier paper that modeling complexity reduces the overall utility of the model (Jahangirian, Taylor, and Young, 2010). In order to minimize this utility loss, the standard modeling strategy is to compensate by reducing modeling “*fidelity*”, namely the resolution of the simulation model.

“*Fidelity*” plays an important role in the modeling process and affects credibility, verification, and validation of the model. Fidelity determines how much a modeler can model, given a set of customer requirements without sacrificing model’s utility. The choice of fidelity measure is dependent on the modeling maturity in supporting the modeler’s goals. If the modeler is able to model with high fidelity, then we assume the modeling goals closely

align with customer requirements. This concept is captured as the “*fidelity*” loop in the Figure 4-9.

Let’s take this concept one step further. If your simulation model closely aligns with the customer requirements, then indirectly the scope of the simulation engagement will also increase. The fidelity factor of simulations directly affects the resolution of the simulation models. If this factor were to increase, then it will indirectly increase the capability of the simulation model being used. This feedback concept is captured in “*scope*” loop in the Figure 4-9.

4.2.5 Facility Integration

The final step in building a CIMS model consists of combining the previous four models using composition. In an object-oriented language, every model developed is also an object. Therefore, all the four facility models developed above are also objects representing Emergency Department, Oncology Clinic, Radiology, and Laboratory services. Hence, the final CIMS model will be composed of these four objects. The ED and Oncology Clinic’s laboratory and radiology transfer nodes are connected to the respective services so that redirected patients can move to these departments and complete their care cycle.

4.3 Model Deployment

The ultimate goal in building a simulation model is to experiment with it to understand the system under consideration. In order to experiment with a simulation model, it needs to be verified and validated. For the purposes of verification, the model’s process flow should be a representative of the system. Validation requires the model accurately estimate the parameters of system such as waiting time, throughput, etc as compared to real system. In the previous section, we established the symbiotic relationship between verification and validation with respect to model animation which will be described in this section.

A good model animation can be described as accurate graphical portrayal of the system which is composed of process flow animation and info graphics. Process flow animation depicts the transaction of entities in a logical manner, which is easy to understand for the system stakeholders. This understanding depends on symbols (drawings) used for depicting the system. In the case of SIMIO-HC, we developed a graphical library of objects using Google's Sketchup (<http://sketchup.google.com/>) in addition to using the Google's 3D Warehouse (<http://sketchup.google.com/3dwarehouse/>). Readers must note that usage of high polynomial count graphics (high resolution 3D images) will significantly slow down the simulation run as the SIMIO graphical engine has to redraw the image for each frame. This redraw will negatively affect the utility of the model as represented in Figure 3-9. We have shared our graphical library which has been optimized for healthcare simulation purposes on Google's warehouse for anyone to use⁸.

In addition to representing the simulation model in a way that is easy to understand, simulation modelers can also provide interesting “*infographics*”. “*Infographics*” provide an interactive interface to visualize simulation parameters. Infographics can be created using status plots, status pies, status labels linear and circular gauge. An interactive dashboard can be developed to go along with the model using these elements to provide real time statistics. In the CIMS model developed, a dashboard provided real time statistics on the bed usage, patient throughput, secondary facility usage, and patient related performance metrics.

Finally, we deal with the topic of model experimentation. Model experimentation helps us perform sensitivity analysis. Sensitivity analysis helps in quantifying the uncertainty in output analysis, more importantly it helps us understand the “nature” of our proposed solution. In addition, model experimentation facilitates testing the robustness of our solution and helps in generating alternatives. More often than not, optimal solutions aren't the practical ones or cannot be found owing to the complexity in the system. Hence generating alternatives can be helpful for the practitioner to identify near optimal scenarios.

---

⁸ Collection Name: Hospital equipment for Simulation shared by Akshay (vsa.akshay@gmail.com)

Having placed a paramount importance on model experimentation, an effective experiment depends on the right experimental design. In simulation modeling, this means identification of properties that are important to the model. In a CIMS model, the number of properties could explode making it difficult to run the experiment. Hence, to avoid this we suggest following the building block approach whereby we identify factors of interest at each stage. These factors are converted in to model properties so that they can be later used in the CIMS experimental design.

In the hypothetical hospital model design, we were interested in understanding the relationship between staffing level and patient throughput time. Hence, we converted staffing numbers at each simulation model into properties. These properties were included in the experimental design and measured against patient's throughput time.

4.3.1 Experimentation Setup and Results

A huge advantage in usage of SIMIO-HC can be attributed to the ready availability of statistics that are focused in healthcare settings. The statistics help in setting up an experiment and evaluating the results. In the case of a hypothetical hospital, we wanted to minimize the flow time of patients through the clinic subject to staffing restrictions for each type of employee. It was assumed that each employee staff count can be changed from 1 to 10. In the first stage of this experiment, we used the OptQuest Add-in for SIMIO to generate scenarios that minimized the total time in system for patient (*SC_PatientExit.TimeInSystem.Average*). We restricted the number of scenarios generated in this case to 25. The second stage employed K-N ranking and the Subset selection add-in to select best scenario with an indifference zone of 0.5 hours. This approach helped reinforce the statistical validity of the best scenario's result. The SMORE⁹ plots presented in Figure 4-9 and Figure 4-10 correspond to the first and second stages of the experiment. The experiment setup is discussed in detail Appendix C.V. Hence, we restrict this discussion to important results that we gain from the CIMS model experimentation.

---

⁹ SIMIO Measure Of Risk and Error is abbreviated as SMORE. Default confidence interval of 95% is reported.

The best scenario required three laboratory technicians, three phlebotomist, and two receptionists for the laboratory department. The radiology department requires three receptionists to assist with patient flow in the department. In the oncology department, we require two each of all the worker types. Finally, the emergency department required three nurses, three residents and three doctors to perform optimally in the larger hospital model. The smallest total flow time of 4.06 ± 0.11 hours is reported for this staffing setup.

Figure 4-9: Patient Time in System for 25 scenarios generated by OptQuest

Figure 4-10: K-N Ranking and Selection Best Scenario

One of the prime factors of interest is bed utilization in emergency department. These are the most expensive resources in the hospital and hence, it would be really useful to know how much time was the bed used for treating (Table 4-1). Given our interest in staffing levels, it will be also interesting to note the percentage time the bed was waiting for staff, which is given by the Occupied State of each *Patient Care* object.

Table 4-1: Patient Care Treating & Occupied Resource State Averages

Object Instance	Resource State in % (Treating)	Resource State in % (Occupied)
PatientCare1	63.65±1.33	25.10±1.18
PatientCare2	79.16±1.28	3.75±0.15
PatientCare3	74.66±1.67	8.15±0.25
PatientCare4	80.99±2.26	2.79±0.25
PC_MaternityBed	23.16±1.8	34.27±3.12
PC_TreatmentBed	32.82±1.67	31.27±2.46
PC_TreatmentChair	37.14±1.29	32.18±2.5
PC_WardBed	44.38±2.26	20.95±1.65

Table 4-2: Laboratory utilization breakdown

Resource State	In %
Setup	1.21±.48
Processing	30.99±.9
Post Processing	32.68±.89

Second interest factor in the simulation is resource utilization of the radiation and laboratory departments. This statistic can also be read from the resource state lists from the respective objects. It is interesting to note that Table 4-2 reports that the laboratory object is idle for close to one third of the time. From Table 4-3, we can deduce that the radiology services are being idle for nearly 50% of the time. These are strong indications for system redesign as these secondary resources are being underutilized.

Table 4-3: Radiology Department utilization breakdown

HWS Object Instance	Resource State	In %
Gamma Camera	Setup	7.35±1.29
	Processing	10.69±4.12
	Post Processing	4.08±1.43
MRI	Setup	10.6±1.48
	Processing	28.8±4.1
	Post Processing	10.65±1.39
X-Ray	Setup	10.6±1.48
	Processing	20.4±3.25
	Post Processing	20.7±1.96

Finally, we also look in to patient activity state statistics (in percentage and total time) to understand patient's activity breakdown during his/her time in the clinic. This analysis can give an overall breakdown of hospital structure and can help understand the potential bottlenecks. Table 4-4 provides percent time statistics that helps in giving the relative picture and Table 4-5 provides the absolute time average statistics in minutes. From the two tables, it is clearly evident that the major bottleneck in the hypothetical hospital model is the patient processing center (*Patient Care* object types). This bottleneck can be inferred from the fact that nearly 43.5% of patient time or over 2 hours on average is spent waiting for a medical resource.

Table 4-4: Percent Time statistics breakdown for Patient's activity

States	Average	Half Width
Administration	8.7	0.57
Lab Turnaround Time	0.69	0.07
Radiology Turnaround Time	2.57	0.28
Riding	4.3	0.25
Travel	4.14	0.51
Treatment Time	25.9	1.57
Waiting For Medical Personnel	4.43	0.31
Waiting for Medical Resource	43.56	2.55
Waiting for Non-Medical Resource	5.71	1.22

Table 4-5: Total Time statistics (in minutes) for patient's activity

States	Average	Half Width
Administration	9.372	0.168
Lab Turnaround Time	1.176	0.474
Radiology Turnaround Time	8.628	0.204
Riding	8.718	0.3
Travel	6.798	1.326
Treatment Time	43.422	3.618
Waiting For Medical Personnel	6.408	0.444
Waiting for Medical Resource	151.194	6.81
Waiting for Non-Medical Resource	8.316	2.112

Hence, based on all the above observations we can suggest central changes to the hypothetical hospital model to improve patient times. Given that the experimentation has helped us identify optimal staffing targets, we suggest adding emergency department beds, clinics and treatment beds in oncology department to reduce this waiting time. This addition will also reduce the patient's demand lost which in the current system is pegged at 113.17 ± 18.42 patients which is about 40% of total patient served. This number also justifies the financial investment of adding new patient processing centers. In addition this recommendation, the increased throughput from the above two departments will improve the utilization of laboratory and radiology departments. The hospital may need to review its policy to put clinics on hold in the future to improve the performance of laboratory department.

4.4 Simulation Project Lifecycle

In the previous sections, we discussed the role of each of the four simulation objectives: Credibility, Fidelity, Verification and Validation. In this section, we present how these objectives are interlinked to form the "*simulation project lifecycle*". The concept diagram is presented in Figure 4-9, which assumes that all modeling events are positive (or goal tending) in nature.

Before we explain the simulation project lifecycle, we shall define the variables in the conceptual model:

- “*Modeling Knowledge Bank*”: Collective applicable knowledge earned from modeling,
- “*Client Requirements*”: The goals for the simulation project from the client side,
- “*Modeling Goals*”: The goals abstracted from client requirements by factoring in the modeling knowledge for the system,
- “*Relative Modeling Gap*”: It is the difference between model and model goals,
- “*Visualization Factor*”: The visual representation of the model (functional or graphical) that modeling team presents to their stakeholders,
- “*Model Review*”: Model testing activities that establish simulation model exactness to conceptual framework (or modeling goals).
- “*Stakeholder Approval*”: Model acceptance by the clients. and
- “*Data Integration*”: Filtering data with help of client and including it in the model

All projects start with client requirements and base modeling knowledge bank that is possessed by the modeling team which determine the modeling goals for a simulation project. Modeling goals are achievable goals for the modeling lifecycle influenced by project scope and fidelity processes. Once these goals are set, they form a relative model gap which indicates that the simulation model is yet to be considered as a representative of the system.

The relative modeling gap triggers model development activity which in turn reduces this gap, given that this model has yet to be proved to be the right one. Secondly, this modeling activity also contributes to the modeling expertise of the team. As the gap reduces, a model is taking shape which has some form visualization. This further reduces the modeling gap as it gives the modeling team a chance to get the stakeholder engaged.

Stakeholder approval is based on visualization which could be either functional or graphical. If stakeholder approvals increase, it indicates that simulation model is an accurate representation of the system and further reduces the relative modeling gap.

Figure 4-11: Simulation Project Lifecycle

. As model development activities begin to take shape, the modeler will also initiate model review activities to verify the simulation model. These activities further reduce the modeling gap as it helps simulation modeler to ascertain the exactness of the model. Only a verified model should be validated. In order to validate a simulation model, requires accurate data which can be collected by engaging the stakeholder. Validating a simulation model takes us one step closer to closing the modeling gaps as it ensures we have the right model which works the right way allowing the team to experiment with it and learn from it. A happy customer has the potential to be a customer for life. In other words, if stakeholders of the

system are impressed with the simulation model, it will tend to increase the scope of the project. The team may be given embellished goals or even new ones which will contribute to customer requirements. This result will further the objectives of simulation project and will sometimes contribute to our modeling knowledge bank.

4.5 Chapter Summary

This chapter formalizes the building blocks approach for construction of Complex Integrated Multi-Facility systems (CIMS) simulation models. This approach is shown with an example of a hypothetical hospital which consists of an Emergency department, Oncology department, Radiology department, and Laboratory department where the patients can flow between departments. In the process of construction of these facility models, we also explained the role of verification, validation; credibility and fidelity factors play in simulation. These concepts were combined to describe the simulation project lifecycle which captures the interdependencies between each of the above factors. Detailed model development notes are presented in Appendix C.

Chapter 5 Conclusions and Future Work

The work presented in this thesis develops an object-oriented framework to model healthcare operations. The toolkit development followed class based design paradigm where we define robust base classes. These base classes and their logic are briefly described below.

- **Fixed Class:** Classes that give structure to the simulation model
  - ***Redirect Object*** : represents a base class where an entity can be held to be redirected
  - ***Waiting Room***: It represents ***Waiting Room*** in an hospital with balking and reneging
  - ***Registration Desk***: represents functions of an administrative object
  - ***Patient Care*** : An object with capabilities to treat a patient
  - ***Patient Care with Lab Care***: An object to treat a patient including support for tertiary care logic
  - ***Hospital Workstation***: An object that represents tertiary services such as laboratory
  - ***Hospital Workstation with Lab Care***: An object that represents tertiary services where a patient also has to visit such as radiology departments, stress test centers.
- **Entity Class:** Classes that represent freely flowing agents that can be dynamically created or destroyed in the simulation model
  - ***Patient Entity*** represents a patient
  - ***Specimen Entity*** represents an entity related to a ***Patient Entity***
- **Transporter Class:** Classes that allow for an object to freely move in 3 Dimensions
  - ***Hospital Transporter***: Represents hospital vehicles such as Wheel Chairs, Stretchers, ambulances etc.
  - ***Hospital Worker***: Represents a medical worker.

- **Node Class:** Represents fixed object that facilitates connection between two fixed objects
  - ***Hospital Transfer node:*** Represents a capacitated transfer node

Although the toolkit presented here has many of the essential base classes, there is exists ample opportunities to refine the framework. The toolkit currently release does not have costing variables and costing elements could be added in the future to keep track of costs. Secondly, the open framework of SIMIO and SIMIO-HC allows for users to take apart the classes and re-configure them. We hope this would help in further refining the toolkit and can also lead to creation domain specific specializations of SIMIO-HC. Thirdly, hospitals are filled with resources such as crash carts which are not fixed at a location. These resources can be aptly named as ***Movable Resources*** and we hope to have a class in the future to represent these resources. Finally, the animation provided in SIMIO is three-dimensional in nature. We intend to include interactive hospital graphical elements as SIMIO upgrades its graphical capabilities to engage the modeler and stakeholders. In terms of output generation, we hope to include patient type statistics collection in the future to help analyze patient statistics depending on their type.

In the process of class development, we also developed guidelines for simulation software that developers can follow. The first guideline related hierarchy of the object to the utility of the simulation package where the incremental utility of the class decreases with increasing hierarchy. This guideline implies that design of base classes is critical to success of the simulation package. The second guideline states that utility of class is product of object complexity and usability where complexity represents how much the class can do and usability is defined as the ease of use. This guideline indicates the developers must carefully think about the application of the classes in order to maximize the class utility. The third guideline states that the incremental complexity in developing a class decreases with hierarchy. This observation implies that developing a class by subclassing is easier than developing from scratch. By combining the third guideline with the first guideline, we can

say there is point at which the utility of the class can be maximized by identifying the right complexity and consequently the number of objects. The fourth guideline links complexity and run speed to model size (or class size) whereby we can determine the utility of the simulation model depending on how fast it can run and how many systems can it represent (complexity).

We hope that these guidelines will start a discussion in the simulation community about modeling philosophy and evolve to create best practices for simulation modeling. Furthermore, we have limited ourselves to developers, but these concepts can also be easily extended to include guidelines for modelers too.

Finally, we used the SIMIO-HC toolkit to develop a Complex Integrated Multi-facility simulation model. This model was developed using composition where we develop individual facility simulation models independently and combine them in their final stage to form the multi-facility model. Along with the development of the model, we also presented how the four factors of simulation credibility, fidelity, verification and validation interact with each other and developed a Simulation Project Lifecycle model to represent them. The simulation project lifecycle conceptual model characterizes the link between modeling community and stakeholders.

In the future, we hope to characterize this behavior with respect to negative modeling efforts and develop markers for simulation modelers to understand when their engagement is not moving in the desired direction. In addition, we hope develop a lifecycle model to characterize the self-correcting nature of modeling and the stakeholder beliefs to create modeling guidelines that can improve the modeling process for the modeler and instill faith in simulation models for stakeholders.

BIBLIOGRAPHY

- Abellan, J. J., Armero, C., Conesa, D., Perez-Panades, J., Martinez-Beneito, M. A., Zurriaga, O., Vanaclocha, H. (2004). Predicting the behaviour of the renal transplant waiting list in the pais valencia (spain) using simulation modeling. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 1969-1974. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Adams, A. L., Koopman, J. S., Chick, S. E., & Yu, P. J. (1999). GERMS: An epidemiologic simulation tool for studying geographic and social effects on infection transmission. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nemhhard, D. T. Sturrock and G. W. Evans. , 1549-1556. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Aleman, D. M., Wibisono, T. G., & Schwartz, B. (2009). Accounting for individual behaviors in a pandemic disease spread model. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1977-1985. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Alexopoulos, C., Goldsman, D., Fontanesi, J., Sawyer, M., De Guire, M., Kopald, D., & Holcomb, K. (2001). A discrete-event simulation application for clinics serving the poor. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1386-1391. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Anderson, C., and Drossopoulou, S. (2003). BabyJ: From object based to class based programming via types. *Electronic Notes in Theoretical Computer Science*, 82(8), 53-81.
- Andradóttir, S., Wenchi Chiu, Goldsman, D., Mi Lim Lee, Kwok-Leung Tsui, Fisman, D. N., Nizam, A. (2010). Simulation of strategies for containing pandemic influenza. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2221-2229. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Arnaout, J. -. M., & Kulbashian, S. (2008). Maximizing the utilization of operating rooms with stochastic times using simulation. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1617-1623. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ashby, M., Ferrin, D., Miller, M., & Shahi, N. (2008). Discrete event simulation: Optimizing patient flow and redesign in a replacement facility. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1632-1636. Institute for Electrical and Electronics Engineers, San Diego, CA.
- B. W. Hollocks. (2006). Forty years of discrete-event simulation: A personal reflection. *Journal of the Operational Research Society*, 57(12), 1383-1399.

- Baesler, F. F., Jahnsen, H. E., & DaCosta, M. (2003). The use of simulation and design of experiments for estimating maximum capacity in an emergency room. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1903-1906. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Baesler, F. F., & Sepulveda, J. A. (2001). Multi-objective simulation optimization for a cancer treatment center. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1405-1411. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Balasubramanian, H., Banerjee, R., Gregg, M., & Denton, B. T. (2007). Improving primary care access using simulation optimization. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1494-1500. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Baldwin, L. P., Eldabi, T., Paul, R. J., & Burroughs, A. K. (2000). Using simulation for the economic evaluation of liver transplantation. In *Proceedings of the 2000 Winter Simulation Conference*, ed., Jeffrey A. Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1963-1970. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Baldwin, L. P., Eldabi, T. A., & Paul, R. J. (1999). Simulation modeling as an aid to decision-making in healthcare management: The adjuvant breast cancer (ABC) trial. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B.

Nembhard, D. T. Sturrock and G. W. Evans. , 1523-1531. Institute for Electrical and Electronics Engineers, San Diego, CA.

Ballard, S. M., & Kuhl, M. E. (2006). The use of simulation to determine maximum capacity in the surgical suite operating room. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 433-438. Institute for Electrical and Electronics Engineers, San Diego, CA.

Barker, K. N., Flynn, E. A., Pepper, G. A., Bates, D. W., & Mikeal, R. L. (2002). Medication errors observed in 36 health care facilities. *Archives of Internal Medicine*, 162(16), 1897-1903.

Barnes, S., Golden, B., & Wasil, E. (2010). A dynamic patient network model of hospital-acquired infections. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2249-2260. Institute for Electrical and Electronics Engineers, San Diego, CA.

Barth-Jones, D. C., Adams, A. L., & Koopman, J. S. (2000). Monte carlo simulation experiments for analysis of HIV vaccine effects and vaccine trial design. In *Proceedings of the 2000 Winter Simulation Conference*, ed., Jeffrey A. Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1985-1994. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Beck, E., Balasubramanian, H., & Henneman, P. L. (2009). Resource management and process change in a simplified model of the emergency department. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1887-1895. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Berwick, D. M. (2002). A user's manual for the IOM's 'quality chasm' report. *Health Affairs (Project Hope)*, 21(3), 80-90.
- Birch, G., and Cook, S. (1991). Discrete event simulation in Smalltalk/V windows. Paper presented at the *Object-Oriented Simulation and Control, IEE Colloquium on*, 5/1-5/3.
- Bischak, D. P., & Roberts, S. D. (1991). Object-oriented simulation. In *Proceedings of the 1991 Winter Simulation Conference*, ed., W. D. Kelton, Barry L. Nelson and B. G. Lawson. , 194-203. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Bisset, K. R., Xizhou Feng, Marathe, M., & Yardi, S. (2009). Modeling interaction between individuals, social networks and public policy to support public health epidemiology. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2020-2031. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Blau, G., & Orcun, S. (2009). A bayesian pharmacometric approach for personalized medicine — A proof of concept study with simulated data. In *Proceedings of the 2009*

Winter Simulation Conference, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1969-1976. Institute for Electrical and Electronics Engineers, San Diego, CA.

Bohk, C., Ewald, R., & Uhrmacher, A. M. (2009). Probabilistic population projection with JAMES II. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2008-2019. Institute for Electrical and Electronics Engineers, San Diego, CA.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3(90003), 7280-7287. doi:10.1073/pnas.082080899

Borshchev, A., and Filippov, A. (2004). From system dynamics and discrete event to practical agent based modeling: Reasons, techniques, tools. Paper presented at the *The 22nd International Conference of the System Dynamics Society*, Oxford England.

Bosire, J., Shengyong Wang, Gandhi, T., & Srihari, K. (2007). Comparing simulation alternatives based on quality expectations. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1579-1585. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brady, T. F. (2003). Emergency management: Capability analysis of critical incident response. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J.

Sánchez, D. Ferrin and D. J. Morrice. , 1863-1867. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brailsford, S. C., Harper, P. R., Patel, B., & Pitt, M. (2009). An analysis of the academic literature on simulation and modelling in health care. *Journal of Simulation*, 3(3), 130-140.

Brailsford, S. C. (2008). System dynamics: What's in it for healthcare simulation modelers. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1478-1483. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brailsford, S. C., Bolt, T., Connell, C., Klein, J. H., & Patel, B. (2009). Stakeholder engagement in health care simulation. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1840-1849. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brailsford, S. C., Desai, S. M., & Viana, J. (2010). Towards the holy grail: Combining system dynamics and discrete-event simulation in healthcare. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2293-2303. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brailsford, S. C., Sykes, J., & Harper, P. R. (2006). Incorporating human behavior in healthcare simulation models. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 466-472. Institute for Electrical and Electronics Engineers, San Diego, CA.

Brandeau, M. L. (2008). Infectious disease control policy: A role for simulation. In *Proceedings of the 2008 Winter Simulation Conference* ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler., 1578-1582. Institute for Electrical and Electronics Engineers, San Diego, CA.

Cahill, W., & Render, M. (1999). Dynamic simulation modeling of ICU bed availability. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nembhard, D. T. Sturrock and G. W. Evans. , 1573-1576. Institute for Electrical and Electronics Engineers, San Diego, CA.

Carr, S., & Roberts, S. (2010). Planning for infectious disease outbreaks: A geographic disease spread, clinic location, and resource allocation simulation. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2171-2184. Institute for Electrical and Electronics Engineers, San Diego, CA.

Centeno, M. A., Albacete, C., Terzano, D. O., Carrillo, M., & Ogazon, T. (2000). A simulation study of the radiology department at JMH. In *Proceedings of the 2000 Winter*

Simulation Conference, ed., Jeffrey A. Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1978-1984. Institute for Electrical and Electronics Engineers, San Diego, CA.

Centeno, M. A., Dodd, H. R., Aranda, M., & Sanchez, Y. (2010). A simulation study to increase throughput in an endoscopy center. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2462-2473. Institute for Electrical and Electronics Engineers, San Diego, CA.

Centeno, M. A., Giachetti, R., Linn, R., & Ismail, A. M. (2003). A simulation-ILP based tool for scheduling ER staff. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1930-1938. Institute for Electrical and Electronics Engineers, San Diego, CA.

Centeno, M. A., Lee, M. A., Lopez, E., Fernandez, H. R., Carrillo, M., & Ogazon, T. (2001). A simulation study of the labor and delivery rooms at JMH. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1392-1400. Institute for Electrical and Electronics Engineers, San Diego, CA.

Chahal, K., & Eldabi, T. (2008). Applicability of hybrid simulation to different modes of governance in UK healthcare. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1469-1477. Institute for Electrical and Electronics Engineers, San Diego, CA.

Charfeddine, M., & Montreuil, B. (2010). Integrated agent-oriented modeling and simulation of population and healthcare delivery network: Application to COPD chronic disease in a canadian region. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2327-2339. Institute for Electrical and Electronics Engineers, San Diego, CA.

Chassin, M. R., Galvin, R. W., and the National Roundtable on Health Care Quality, and Natl Roundtable Hlth Care Quality. (1998). The urgent need to improve health care quality: Institute of medicine national roundtable on health care quality. *JAMA: The Journal of the American Medical Association*, 280(11), 1000-1005.

Chockalingam, A., Jayakumar, K., & Lawley, M. A. (2010). A stochastic control approach to avoiding emergency department overcrowding. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2399-2411. Institute for Electrical and Electronics Engineers, San Diego, CA.

Claridge, J. A., M.D., & Fabian, T. C. M. D. (2005). History and development of evidence-based medicine. *World Journal of Surgery*, (29), 547-553.

Classen, D., Resar, R., Griffin, F., Federico, F., Frankel, T., Kimmel, N., . . . James, B. (2011). 'Global trigger tool' shows that adverse events in hospitals may be ten times greater than previously measured. *Health Affairs*, 30(4), 581-589.

- Cook, R. I. (2001). The end of the beginning: Complexity and craftsmanship and the era of sustained work on patient safety. *Jt Comm J Qual Improv*, 27(10), 507-508.
- Cubert, R. M., & Fishwick, P. A. (1998). OOPM: An object-oriented multimodeling and simulation application framework. *Simulation*, 70(6), 379-395.
- Dahl, O., & Nygaard, K. (1966). SIMULA: An ALGOL-based simulation language. *Commun.ACM*, 9(9), 671-678.
- Davies, R. (2006). Use of simulation to determine resource requirements for end-stage renal failure. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 473-477. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Davies, R. (2007). “See and treat” or “See” and “Treat” in an emergency department. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1519-1522. Institute for Electrical and Electronics Engineers, San Diego, CA.
- de Mendonca, B., Phibbs, A., Vandermeer, J., & Pasek, Z. J. (2010). Use of simulation in support of analysis and improvement of blood collection process. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2352-2363. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Denton, B. T., Rahman, A. S., Nelson, H., & Bailey, A. C. (2006). Simulation of a multiple operating room surgical suite. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu, J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 414-424. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Dhillon, B. S. (2008). *Reliability technology, human error, and quality in health care [electronic resource]*. Boca Raton: CRC Press.
- Dibble, C. (2010). Effective real-time allocation of pandemic interventions. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2211-2220. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Efe, K., Raghavan, V., & Choubey, S. (2009). Simulation modeling movable hospital assets managed with RFID sensors. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2054-2064. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ekici, A., Keskinocak, P., & Swann, J. L. (2008). Pandemic influenza response. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1592-1600. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Eldabi, T. (2009). Implementation issues of modeling healthcare problems: Misconceptions and lessons. In *Proceedings of the 2009 Winter Simulation Conference*, ed., Anonymous , 1831-1839. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Eldabi, T., Jun, G. T., Clarkson, J., Connell, C., & Klein, J. H. (2010). Model driven healthcare: Disconnected practices. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2271-2282. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Eldabi, T., & Paul, R. J. (2001). A proposed approach for modeling healthcare systems for understanding. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1412-1420. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Eldabi, T., & Young, T. (2007). Towards a framework for healthcare simulation. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1454-1460. Institute for Electrical and Electronics Engineers, San Diego, CA.
- El-Haik, B., & Al-Aomar, R. (2006). Frontmatter. In *Simulation-based lean six-sigma and design for six-sigma* (pp. i-xx) John Wiley & Sons, Inc.
- England, W., and Roberts S. (1978). Applications of computer simulation in health care *Proceedings of 1978 Winter Simulation Conference* Miami Beach. 665-676.

- Faller, S. L., Flynn, C. T., & Ferrin, D. M. (2009). Simulating health care in prison systems. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2032-2041. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Feldman, Z., Mandelbaum, A., Massey, W. A., & Whitt, W. (2008). Staffing of time-varying queues to achieve time-stable performance. *Management Science*, 54(2), 324-338.
- Ferrand, Y., Magazine, M., & Rao, U. (2010). Comparing two operating-room-allocation policies for elective and emergency surgeries. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2364-2374. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ferrin, D. M., Miller, M. J., Wininger, S., & Neuendorf, M. S. (2004). Analyzing incentives and scheduling in a major metropolitan hospital operating room through simulation. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 1975-1980. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Fildes, R., & Ranyard, J. C. (1997). Success and survival of operational research groups-A review. *Journal of the Operational Research Society*, 48(4), 336-360.
- Fone, D., Hollinghurst, S., Temple, M., Round, A., Lester, N., Weightman, A., . . . Palmer, S. (2003). Systematic review of the use and value of computer simulation modelling in

population health and health care delivery. *Journal of Public Health Medicine*, 25(4), 325-335.

Fordyce, J., Blank, F., Pekow, P., Smithline, H., Ritter, G., Gehlbach, S., . . . Henneman, P. (2003). Errors in a busy emergency department¹. *Annals of Emergency Medicine*, 42(3), 324-333.

Gawande, A. (2002). *Complications : A surgeon's notes on an imperfect science*. New York: Picador.

Gawande, A. (2010). *The checklist manifesto : How to get things right*. New York: Metropolitan Books.

Giachetti, R. E. (2008). A simulation study of interventions to reduce appointment lead-time and patient no-show rate. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1463-1468. Institute for Electrical and Electronics Engineers, San Diego, CA.

Giachetti, R. E., Centeno, E. A., Centeno, M. A., & Sundaram, R. (2005). Assessing the viability of an open access policy in an outpatient clinic: A discrete-event and continuous simulation modeling approach. In *Proceedings of the 2005 Winter Simulation Conference*, ed., M. E. Kuhl, N. M. Steiger, F. B. Armstrong and Jeffrey A. Joines. , 2246-2256. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Gibson, I. W. (2007). An approach to hospital planning and design using discrete event simulation. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1501-1509. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Goldsman, D., Nance, R. E., & Wilson, J. R. (2010). A brief history of simulation revisited. In *Proceedings of the 2010 Winter Simulation Conference*, ed., Anonymous , 567-574. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Gottlob, G., Schrefl, M., and Rock, B. (1996). Extending object-oriented systems with roles. *ACM Trans.Inf.Syst.*, 14(3), 268-296.
- Günel, M. M., and Pidd, M. (2010). Discrete event simulation for performance modelling in health care: A review of the literature. *Journal of Simulation*, 4(1), 42-51.
- Gunal, M. M., & Pidd, M. (2006). Understanding accident and emergency department performance using simulation. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 446-452. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Gunal, M. M., & Pidd, M. (2007). Interconnected des models of emergency, outpatient, and inpatient departments of a hospital. In *Proceedings of the 2007 Winter Simulation*

Conference, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1461-1466. Institute for Electrical and Electronics Engineers, San Diego, CA.

Gunal, M. M., & Pidd, M. (2008). DGHPsim: Supporting smart thinking to improve hospital performance. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1484-1489. Institute for Electrical and Electronics Engineers, San Diego, CA.

Guo, M., Wagner, M., & West, C. (2004). Outpatient clinic scheduling - a simulation approach. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 1981-1987. Institute for Electrical and Electronics Engineers, San Diego, CA.

Hagtvedt, R., Ferguson, M., Griffin, P., Jones, G. T., & Keskinocak, P. (2009). Cooperative strategies to reduce ambulance diversion. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1861-1874. Institute for Electrical and Electronics Engineers, San Diego, CA.

Harper, A. M., Taranto, S. E., Edwards, E. B., & Daily, O. P. (2000). An update on a successful simulation project: The UNOS liver allocation model. In *Proceedings of the 2000 Winter Simulation Conference*, ed., Jeffrey A. Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1955-1962. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Harrell, C. R., and Lange, V. (2001). Healthcare simulation modeling and optimization using MedModel. . In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1412-1420. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Havender, J. W. (1968). *Avoiding deadlock in multitasking systems* IBM Systems Journal ,7(2), 74-84
- Hay, A. M., Valentin, E. C., & Bijlsma, R. A. (2006). Modeling emergency care in hospitals: A paradox - the patient should not drive the process. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 439-445. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Helm, J. E., Lapp, M., & See, B. D. (2010). Characterizing an effective hospital admissions scheduling and control management system: A genetic algorithm approach. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2387-2398. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Hermans, M. D., Sluijs, J. M., & Aartsen, H. (1998). Reducing inter-organizational barriers in the medical sector with simulation. In *Proceedings of the 1998 Winter Simulation Conference*, ed., D. J. Medeiros, E. F. Watson, John S. Carson and M. S. Manivannan. , 1441-1449. Institute for Electrical and Electronics Engineers, San Diego, CA.

HFMA. (2011). *Value in healthcare: Current state and future directions*. Westchester, IL:

Healthcare Financial Management Association. Retrieved from

<http://www.hfma.org/HFMA-Initiatives/Value-Project/Value-in-Health-Care--Current-State-and-Future-Directions/>

Holm, L. B., & Dahl, F. A. (2009). Simulating the effect of physician triage in the emergency department of akershus university hospital. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1896-1905. Institute for Electrical and Electronics Engineers, San Diego, CA.

Holm, L. B., & 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 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2455-2461. Institute for Electrical and Electronics Engineers, San Diego, CA.

Housseman, S., Absi, N., Feillet, D., & Dauzere-Peres, S. (2009). Impacts of radio-identification on cryo-conservation centers through simulation. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2065-2077. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Hughes, G. R., Currie, C. S. M., & Corbett, E. L. (2006). Modeling tuberculosis in areas of high HIV prevalence. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 459-465. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Huschka, T. R. (2007). Bi-criteria evaluation of an outpatient procedure center via simulation. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1510-1518. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Huschka, T. R., Denton, B. T., Narr, B. J., & Thompson, A. C. (2008). Using simulation in the implementation of an outpatient procedure center. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1547-1552. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ibrahim, R., & Whitt, W. (2010). Delay predictors for customer service systems with time-varying parameters. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2375-2386. Institute for Electrical and Electronics Engineers, San Diego, CA.
- IOM. (2000). In Linda T. Kohn, Janet M. Corrigan, and Molla S. Donaldson, editors, Committee on Quality of Health Care in America, Institute of Medicine., Corrigan J., Donaldson M. S., Institute of Medicine (U.S.) Committee on Quality of Health Care in

America. and Kohn L. T. (Eds.), *To err is human : Building a safer health system*.
Washington, D.C.: National Academy Press.

IOM. (2001). In Committee on Quality Health Care in America, Institute of Medicine.,
Institute of Medicine (U.S.). Committee on Quality of Health Care in America. (Eds.),
Crossing the quality chasm : A new health system for the 21st century. Washington,
D.C.: National Academy Press.

IOM. (2010). *Healthcare imperative : Lowering costs and improving outcomes : Workshop
series summary*. Washington, D.C.: National Academies Press.

Iser, J. H., Denton, B. T., & King, R. E. (2008). Heuristics for balancing operating room and
post-anesthesia resources under uncertainty. In *Proceedings of the 2008 Winter
Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and
J. W. Fowler. , 1601-1608. Institute for Electrical and Electronics Engineers, San Diego,
CA.

Ismail, K., Abo-Hamad, W., & Arisha, A. (2010). Integrating balanced scorecard and
simulation modeling to improve emergency department performance in irish hospitals.
In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J.
Montoya-Torres, J. Hugan and E. Yücesan. , 2340-2351. Institute for Electrical and
Electronics Engineers, San Diego, CA.

Jacobson, S., Hall, S., & Swisher, J. R. (2006). Patient flow [electronic resource] : Reducing delay in healthcare delivery. In edited by Randolph W. Hall., R. W. Hall & I. NetLibrary (Eds.), (pp. 211-252). New York: Springer.

Jacobson, S. H., Sewell, E. C., & Weniger, B. G. (2001). Using monte carlo simulation to assess the value of combination vaccines for pediatric immunization. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1421-1428. Institute for Electrical and Electronics Engineers, San Diego, CA.

Jahangirian, M., Taylor, S. J. E., & Young, T. (2010). Economics of modeling and simulation: Reflections and implications for healthcare. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2283-2292. Institute for Electrical and Electronics Engineers, San Diego, CA.

James Swisher, Brian Jun, Sheldon Jacobson, & Osman Balci. (1997). Simulation of the question physician network. In *Proceedings of the 1997 Winter Simulation Conference*, ed., S. Andradóttir, K. J. Healy, D. H. Withers and Barry L. Nelson. , 1146. Institute for Electrical and Electronics Engineers, San Diego, CA.

Jeffrey, P., & Seaton, R. (1995). The use of operational research tools: A survey of operational research practitioners in the UK. *Journal of the Operational Research Society*, 46(7), 797-808.

- Johnson, W. C. (1998). Birth of a new maternity process. In *Proceedings of the 1998 Winter Simulation Conference*, ed., D. J. Medeiros, E. F. Watson, John S. Carson and M. S. Manivannan. , 1429-1432. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Joines, J. A., & Roberts, S. D. (1997). An introduction to object-oriented simulation in C++. In *Proceedings of the 1997 Winter Simulation Conference*, ed., S. Andradóttir, K. J. Healy, D. H. Withers and Barry L. Nelson. , 78-85. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Joines, J. A., & Roberts, S. D. (1998a). Fundamentals of object-oriented simulation. In *Proceedings of the 1998a Winter Simulation Conference*, ed., D. J. Medeiros, E. F. Watson, John S. Carson and M. S. Manivannan. , 141-149. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Joines, J. A., & Roberts, S. D. (1998b). Object oriented simulation . In J. Banks (Ed.), *Handbook of simulation - principles, methodology, advances, applications, and practice* (pp. 397-426, 605-627) John Wiley & Sons.
- Jun, J. B., Jacobson, S. H., & Swisher, J. R. (1999). Application of discrete-event simulation in health care clinics: A survey *The Journal of the Operational Research Society*, 50(2), 109.

- Juran, J. M. (1992). *Juran on quality by design : The new steps for planning quality into goods and services*. New York; Toronto; New York: Free Press ; Maxwell Macmillan Canada ; Maxwell Macmillan International.
- Kaplan, R. S., and Porter, M. E. (2011). How to solve the cost crisis in health care. *Harvard Business Review*, 3-18-18.
- Business Review*, , 3-18-18.
- Kasaie, P., Kelton, W. D., Vaghefi, A., & Naini, S. G. R. J. (2010). Toward optimal resource-allocation for control of epidemics: An agent-based-simulation approach. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2237-2248. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Katsaliaki, K., & Mustafee, N. (2011). Applications of simulation within the healthcare context. *Journal of the Operational Research Society*, 62(8), 1431-1451.
- Katzper, M. (2005). Clinical trial factors in a pain transition state model. In *Proceedings of the 2005 Winter Simulation Conference*, ed., M. E. Kuhl, N. M. Steiger, F. B. Amstrong and Jeffrey A. Joines. , 2232-2236. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Khurma, N., Bacioiu, G. M., & Pasek, Z. J. (2008). Simulation-based verification of lean improvement for emergency room process. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1490-1499. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Kolb, E. M. W., Peck, J., Schoening, S., & Lee, T. (2008). Reducing emergency department overcrowding - five patient buffer concepts in comparison. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1516-1525. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Kolb, E. M. W., Taesik Lee, & Peck, J. (2007). Effect of coupling between emergency department and inpatient unit on the overcrowding in emergency department. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1586-1593. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Koppenhaver, R. T., Schaefer, A., Braithwaite, R. S., & Roberts, M. (2009). A simulation model of HIV treatment under drug scarcity constraints. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2090-2095. Institute for Electrical and Electronics Engineers, San Diego, CA.

Kuljis, J., Paul, R. J., & Stergioulas, L. K. (2007). Can health care benefit from modeling and simulation methods in the same way as business and manufacturing has? In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1449-1453. Institute for Electrical and Electronics Engineers, San Diego, CA.

Kuntz, L., Mennicken, R., and Scholtes, S. (2012). *Stress on the ward: Linking individual performance to system service quality*. Unpublished manuscript. Retrieved 1/13/2012, from http://www.health.jbs.cam.ac.uk/research/current/downloads/110718_stress_on_the_ward.pdf

Lach, J. M., & Vazquez, R. M. (2004). Simulation model of the telemedicine program. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 2012-2017. Institute for Electrical and Electronics Engineers, San Diego, CA.

Lange, V. E. (1999). The benefits of simulation modeling in medical planning and medical design. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nembhard, D. T. Sturrock and G. W. Evans. , 1564-1567. Institute for Electrical and Electronics Engineers, San Diego, CA.

Lizon, N. E., Aleman, D. M., & Schwartz, B. (2010). Incorporating healthcare systems in pandemic models. In *Proceedings of the 2010 Winter Simulation Conference*, ed.,

Anonymous , 2230-2236. Institute for Electrical and Electronics Engineers, San Diego, CA.

Lomow, G., & Baezner, D. (1991). A tutorial introduction to object-oriented simulation and sim++. In *Proceedings of the 1991 Winter Simulation Conference*, ed., W. D. Kelton, Barry L. Nelson and B. G. Lawson. , 157-163. Institute for Electrical and Electronics Engineers, San Diego, CA.

Lowery, J. C., & Davis, J. A. (1999). Determination of operating room requirements using simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nembhard, D. T. Sturrock and G. W. Evans. , 1568-1572. Institute for Electrical and Electronics Engineers, San Diego, CA.

Mabry, S., Rodriguez, S., & Heffernan, J. (1997). Integrated medical analysis system In *Proceedings of the 1997 Winter Simulation Conference*, ed., S. Andradóttir, K. J. Healy, D. H. Withers and Barry L. Nelson. , 1163. Institute for Electrical and Electronics Engineers, San Diego, CA.

Mahapatra, S., Koelling, C. P., Patvivatsiri, L., Fraticelli, B., Eitel, D., & Grove, L. (2003). Pairing emergency severity index5-level triage data with computer aided system design to improve emergency department access and throughput. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1917-1925. Institute for Electrical and Electronics Engineers, San Diego, CA.

Malloy, B. A., & Chen, B. (1998). Exploiting an object-oriented simulation tool to model a PCS network. *Simulation*, 70(6), 369-378.

Marmor, Y. N., Wasserkrug, S., Zeltyn, S., Mesika, Y., Greenshpan, O., Carmeli, B., . . .

Mandelbaum, A. (2009). Toward simulation-based real-time decision-support systems for emergency departments. In *Proceedings of the 2009 Winter Simulation Conference*, ed., Anonymous , 2042-2053. Institute for Electrical and Electronics Engineers, San Diego, CA.

Martin Pitt. (1997). A generalised simulation system to support strategic resource planning in healthcare. In *Proceedings of the 1997 Winter Simulation Conference*, ed., S.

Andradóttir, K. J. Healy, D. H. Withers and Barry L. Nelson. , 1155. Institute for Electrical and Electronics Engineers, San Diego, CA.

Martin, E., Gronhaug, R., & Haugene, K. (2003). Proposals to reduce over-crowding, lengthy stays and improve patient care: Study of the geriatric department in norway's largest hospital. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1876-1881. Institute for Electrical and Electronics Engineers, San Diego, CA.

Maxwell, M. S., Henderson, S. G., & Topaloglu, H. (2009). Ambulance redeployment: An approximate dynamic programming approach. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R.

G. Ingalls. , 1850-1860. Institute for Electrical and Electronics Engineers, San Diego, CA.

McGarvey, B. M., Dynes, N. J., Lin, B. C., Anderson, W. H., Kremidas, J. P., & Felli, J. C. (2007). A discrete event model of clinical trial enrollment at eli lilly and company. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1467-1474. Institute for Electrical and Electronics Engineers, San Diego, CA.

McGlynn, E. A., Asch, S. M., Adams, J., Keeseey, J., Hicks, J., DeCristofaro, A., & Kerr, E. A. (2003). The quality of health care delivered to adults in the united states. *The New England Journal of Medicine*, 348(26), 2635-2645.

Medeiros, D. J., Swenson, E., & DeFlitch, C. (2008). Improving patient flow in a hospital emergency department. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1526-1531. Institute for Electrical and Electronics Engineers, San Diego, CA.

Mellor, G. R., Corbett, E. L., Currie, C. S. M., & Cheng, R. C. H. (2007). Targeted strategies for tuberculosis in areas of high hiv prevalence: A simulation study. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1487-1493. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Meng, L. Y., & Spedding, T. (2008). Modelling patient arrivals when simulating an accident and emergency unit. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1509-1515. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Miller, M., Ferrin, D., Ashby, M., Flynn, T., & Shahi, N. (2007). Merging six emergency departments into one: A simulation approach. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1574-1578. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Miller, M., Ferrin, D., & Shahi, N. (2009). Estimating patient surge impact on boarding time in several regional emergency departments. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1906-1915. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Miller, M. J., Ferrin, D. M., & Messer, M. G. (2004). Fixing the emergency department: A transformational journey with EDSIM. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 1988-1993. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Miller, M. J., Ferrin, D. M., Shahi, N., & LaVecchia, R. (2008). Allocating outpatient clinic services using simulation and linear programming. In *Proceedings of the 2008 Winter*

Simulation Conference, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1637-1644. Institute for Electrical and Electronics Engineers, San Diego, CA.

Miller, M. J., Ferrin, D. M., & Szymanski, J. M. (2003). Simulating six sigma improvement ideas for a hospital emergency department. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1926-1929. Institute for Electrical and Electronics Engineers, San Diego, CA.

Morrison, B. P., & Bird, B. C. (2003). A methodology for modeling front office and patient care processes in ambulatory health care. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1882-1886. Institute for Electrical and Electronics Engineers, San Diego, CA.

Nafarrate, A. R., Fowler, J. W., & Wu, T. (2010). Bi-criteria analysis of ambulance diversion policies. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2315-2326. Institute for Electrical and Electronics Engineers, San Diego, CA.

The Urgent Need to Improve Health Care Quality: Institute of Medicine National Roundtable on Health Care Quality, 280, 11Cong. (1998).

- O’Kane, J. F., Spenceley, J. R., & Taylor, R. (2000). Simulation as an essential tool for advanced manufacturing technology problems. *Journal of Materials Processing Technology*, 107(1–3), 412-424.
- Osidach, V. Z., & Fu, M. C. (2003). Computer simulation of a mobile examination center. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1868-1875. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Pasdirtz, G. W. (2009). Controlling direct-to-consumer advertising, professional promotion and the price of pharmaceutical drugs. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2096-2106. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Pearce, B., Hosseini, N., Taaffe, K., Huynh, N., & Harris, S. (2010). Modeling interruptions and patient flow in a preoperative hospital environment. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2261-2270. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Pegden, C. D. (2010). Advanced tutorial: Overview of simulation world views. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 210-215. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Pidd, M., & Carvalho, A. (2006). Simulation software: Not the same yesterday, today or forever. *Journal of Simulation*, 1(1), 7-20.
- Pirolo, J., Ray, A., Gadzinski, M., Manese, M., Garvert, B., Scoville, G., . . . Donlon, K. (2009). Utilization of discrete event simulation in the prospective determination of optimal cardiovascular lab processes. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1916-1926. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ramakrishnan, S., Nagarkar, K., DeGennaro, M., Srihari, M., Courtney, A. K., & Emick, F. (2004). A study of the CT scan area of a healthcare provider. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 2025-2031. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ramirez, A., Fowler, J. W., & Wu, T. (2009). Analysis of ambulance diversion policies for a large-size hospital. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1875-1886. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ramis, F. J., Baesler, F., Berho, E., Neriz, L., & Sepulveda, J. A. (2008). A simulator to improve waiting times at a medical imaging center. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1572-1577. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Ramis, F. J., Palma, J. L., & Baesler, F. F. (2001). The use of simulation for process improvement at an ambulatory surgery center. In *Proceedings of the 2001 Winter Simulation Conference*, ed., B. A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer. , 1401-1404. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Rao, D. M., & Chernyakhovsky, A. (2008). Parallel simulation of the global epidemiology of avian influenza. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1583-1591. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Reindl, S., Monch, L., Monch, M., & Scheider, A. (2009). Modeling and simulation of cataract surgery processes. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1937-1945. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Roberts, C. A., & Dessouky, Y. M. (1998). An overview of object-oriented simulation. *Simulation*, 70(6), 359-368.
- Rohleder, T., Huschka, T., Egginton, J., O'Neil, D., & Woychick, N. (2010). Modeling care teams at mayo clinic. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2304-2314. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Ross, L. W. (1967). Simulation survey and literature review. *Simulation*, 8(3), 175-178.

Rossetti, M. D., Kumar, A., & Felder, R. A. (1998). Mobile robot simulation of clinical laboratory deliveries. In *Proceedings of the 1998 Winter Simulation Conference*, ed., D. J. Medeiros, E. F. Watson, John S. Carson and M. S. Manivannan. , 1415-1422 vol.2. Institute for Electrical and Electronics Engineers, San Diego, CA.

Rossetti, M. D., Trzcinski, G. F., & Syverud, S. A. (1999). Emergency department simulation and determination of optimal attending physician staffing schedules. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nembhard, D. T. Sturrock and G. W. Evans. , 1532-1540. Institute for Electrical and Electronics Engineers, San Diego, CA.

Ruohonen, T., Neittaanmaki, P., & Teittinen, J. (2006). Simulation model for improving the operation of the emergency department of special health care. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson, D. M. Nicol and R. M. Fujimoto. , 453-458. Institute for Electrical and Electronics Engineers, San Diego, CA.

S. Robinson. (2005). Discrete-event simulation: From the pioneers to the present, what next? *Journal of the Operational Research Society*, 56(6), 619-629.

Samaha, S., Armel, W. S., & Starks, D. W. (2003). The use of simulation to reduce the length of stay in an emergency department. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1907-1911. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Sanchez, S. M., Ogazon, T., Ferrin, D. M., Sepulveda, J. A., & Ward, T. J. (2000). Emerging issues in healthcare simulation. In *Proceedings of the 2000 Winter Simulation Conference*, ed., Jeffrey A. Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1999-2003. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Schriber, T. J., & Brunner, D. T. (2009). Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2009 Winter Simulation Conference*, ed., Anonymous , 216-229. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Schwaab, A. A. S., & de Freitas Filho, P. J. (2009). Dynamic-fuzzy simulation model for reproduction of mortality curves. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 2078-2089. Institute for Electrical and Electronics Engineers, San Diego, CA.
- See, B. D., Shih-Ping Liu, Yi-Wei Lu, & Qi Pang. (2009). Staffing a pandemic urgent care facility during an outbreak of pandemic influenza. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1996-2007. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Sepulveda, J. A., Thompson, W. J., Baesler, F. F., Alvarez, M. I., & Cahoon, L. E.,III. (1999). The use of simulation for process improvement in a cancer treatment center. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B.

Nembhard, D. T. Sturrock and G. W. Evans. , 1541-1548. Institute for Electrical and Electronics Engineers, San Diego, CA.

Siebers, P. O., Macal, C. M., Garnett, J., Buxton, D., & Pidd, M. (2010). Discrete-event simulation is dead, long live agent-based simulation. *Journal of Simulation*, 4(3), 204-210.

Silva, P. M. S., & Pinto, L. R. (2010). Emergency medical systems analysis by simulation and optimization. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2422-2432. Institute for Electrical and Electronics Engineers, San Diego, CA.

Sinreich, D., & Marmor, Y. N. (2004). A simple and intuitive simulation tool for analyzing emergency department operations. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 1994-2002. Institute for Electrical and Electronics Engineers, San Diego, CA.

Song, W. T., Bair, A. E., & Mingchang, C. (2008). A simulation study on the impact of physician starting time in a physical examination service. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1553-1563. Institute for Electrical and Electronics Engineers, San Diego, CA.

- Soorapanth, S., & Chick, S. E. (2010). Cost-utility analysis of behavioral interventions for HIV-infected persons to reduce HIV transmission in the USA. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2433-2443. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Spry, C. W., & Lawley, M. A. (2005). Evaluating hospital pharmacy staffing and work scheduling using simulation. In *Proceedings of the 2005 Winter Simulation Conference*, ed., M. E. Kuhl, N. M. Steiger, F. B. Amstrong and Jeffrey A. Joines. , 2256-2264. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Stanciu, A., Vargas, L., & May, J. (2010). A revenue management approach for managing operating room capacity. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2444-2454. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Sterman, J. (2000). *Business dynamics: Systems thinking and modeling for a complex world*. Boston: Irwin/McGraw-Hill.
- Swain, J. (2011, October). **Software survey: Simulation — back to the future**. *ORMS Today*, pp. 56-69.
- Tafazzoli, A., Roberts, S. D., Ness, R. M., & Dittus, R. S. (2005). A comparison of screening methods for colorectal cancer using simulation modeling. In *Proceedings of the 2005*

Winter Simulation Conference, ed., M. E. Kuhl, N. M. Steiger, F. B. Armstrong and Jeffrey A. Joines. , 2236-2246. Institute for Electrical and Electronics Engineers, San Diego, CA.

Takakuwa, S., & Katagiri, D. (2007). Modeling of patient flows in a large-scale outpatient hospital ward by making use of electronic medical records. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1523-1531. Institute for Electrical and Electronics Engineers, San Diego, CA.

Takakuwa, S., & Shiozaki, H. (2004). Functional analysis for operating emergency department of a general hospital. In *Proceedings of the 2004 Winter Simulation Conference*, ed., R. G. Ingalls, M. D. Rossetti, J. S. Smith and B. A. Peters. , 2003-2011. Institute for Electrical and Electronics Engineers, San Diego, CA.

Takakuwa, S., & Wijewickrama, A. (2008). Optimizing staffing schedule in light of patient satisfaction for the whole outpatient hospital ward. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1500-1508. Institute for Electrical and Electronics Engineers, San Diego, CA.

Taranto, S. E., Harper, A. M., Edwards, E. B., Rosendale, J. D., McBride, M. A., Daily, O. P., . . . Schmeiser, B. (2000). Developing a national allocation model for cadaveric kidneys. In *Proceedings of the 2000 Winter Simulation Conference*, ed., Jeffrey A.

Joines, R. R. Barton, K. Kang and Paul A. Fishwick. , 1971-1977. Institute for Electrical and Electronics Engineers, San Diego, CA.

Thorwarth, M., Arisha, A., & Harper, P. (2009). Simulation model to investigate flexible workload management for healthcare and servicescape environment. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1946-1956. Institute for Electrical and Electronics Engineers, San Diego, CA.

Troy, P. M., & Rosenberg, L. (2009). A java class library for simulating peri-operative processes. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1927-1936. Institute for Electrical and Electronics Engineers, San Diego, CA.

Vila-Parrish, A. R., Ivy, J. S., & King, R. E. (2008). A simulation-based approach for inventory modeling of perishable pharmaceuticals. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1532-1538. Institute for Electrical and Electronics Engineers, San Diego, CA.

Wears, R. L. (2000). Beyond error. *Acad Emerg Med*, 7(11), 1175-1176.

Wears, R. (2003). A different approach to safety in emergency medicine1. *Annals of Emergency Medicine*, 42(3), 334-336.

- Weng, M. L., & Houshmand, A. A. (1999). Healthcare simulation: A case study at a local clinic. In *Proceedings of the 1999 Winter Simulation Conference*, ed., P. A. Farrington, H. B. Nembhard, D. T. Sturrock and G. W. Evans. , 1577-1584. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Wiinamaki, A., & Dronzek, R. (2003). Using simulation in the architectural concept phase of an emergency department design. In *Proceedings of the 2003 Winter Simulation Conference*, ed., S. Chick, P. J. Sánchez, D. Ferrin and D. J. Morrice. , 1912-1916. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Wijewickrama, A., & Takakuwa, S. (2005). Simulation analysis of appointment scheduling in an outpatient department of internal medicine. In *Proceedings of the 2005 Winter Simulation Conference*, ed., M. E. Kuhl, N. M. Steiger, F. B. Armstrong and Jeffrey A. Joines. , 2264-2274. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Wijewickrama, A., & Takakuwa, S. (2008). Outpatient appointment scheduling in a multi facility system. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1563-1571. Institute for Electrical and Electronics Engineers, San Diego, CA.
- Wijewickrama, A. K. A., & Takakuwa, S. (2006). Simulation analysis of an outpatient department of internal medicine in a university hospital. In *Proceedings of the 2006 Winter Simulation Conference*, ed., L. F. Perrone, Wieland, F.P., Liu,J., B. G. Lawson,

D. M. Nicol and R. M. Fujimoto. , 425-432. Institute for Electrical and Electronics Engineers, San Diego, CA.

Wong, C., Geiger, G., Derman, Y. D., Busby, C. R., and Carter, M. W. (2003). Redesigning the medication ordering, dispensing, and administration process in an acute care academic health sciences centre. Paper presented at the *Winter Simulation Conference, 2003. Proceedings of the 2003 Winter*, 2 1894-1902 vol.2.

Woolhandler, S., Campbell, T., and Himmelstein, D. U. (2003). Costs of health care administration in the united states and canada *The New England Journal of Medicine*, 349(8), 768-775. doi:10.1056/NEJMs022033

Workman, R. W. (2000). Simulation of the drug development process: A case study from the pharmaceutical industry. Paper presented at the *Winter Simulation Conference, 2000. Proceedings. Winter*, 2 1995-1998 vol.2.

Worth, T., Uzsoy, R., Samoff, E., Meyer, A. -, Maillard, J. -, & Wendelboe, A. M. (2010). Modelling the response of a public health department to infectious disease. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Huan and E. Yücesan. , 2185-2198. Institute for Electrical and Electronics Engineers, San Diego, CA.

Wynter, S. A., & Ivy, J. E. (2009). Simulating public health emergency response: A case study of the 2004 north carolina state fair E. coli outbreak. In *Proceedings of the 2009*

Winter Simulation Conference, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. , 1957-1968. Institute for Electrical and Electronics Engineers, San Diego, CA.

Yaesoubi, R., & Roberts, S. D. (2007). Important factors in screening for colorectal cancer. In *Proceedings of the 2007 Winter Simulation Conference*, ed., S. G. Henderson, B. Biller, M. H. Hsieh, J. Shortle, J. D. Tew and R. R. Barton. , 1475-1482. Institute for Electrical and Electronics Engineers, San Diego, CA.

Yaesoubi, R., & Roberts, S. D. (2008). How much is a health insurer willing to pay for colorectal cancer screening tests? In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1624-1631. Institute for Electrical and Electronics Engineers, San Diego, CA.

Yarmand, H., Ivy, J. S., Roberts, S. D., Bengtson, M. W., & Bengtson, N. M. (2010). Cost-effectiveness analysis of vaccination and self-isolation in case of H1N1. In *Proceedings of the 2010 Winter Simulation Conference*, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. , 2199-2210. Institute for Electrical and Electronics Engineers, San Diego, CA.

Young, T., Eatock, J., Jahangirian, M., Naseer, A., & Lilford, R. (2009). Three critical challenges for modeling and simulation in healthcare. In *Proceedings of the 2009 Winter Simulation Conference*, ed., M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R.

G. Ingalls. , 1823-1830. Institute for Electrical and Electronics Engineers, San Diego, CA.

Young, T., Brailsford, S., Connell, C., Davies, R., Harper, P., & Klein, J. H. (2004). Using industrial processes to improve patient care. *BMJ (Clinical Research Ed.)*, 328(7432), 162-164.

Yurtkuran, A., & Emel, E. (2008). Simulation based decision-making for hospital pharmacy management. In *Proceedings of the 2008 Winter Simulation Conference*, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W. Fowler. , 1539-1546. Institute for Electrical and Electronics Engineers, San Diego, CA.

APPENDICES

Appendix A SIMIO Framework:

A.I Object-Oriented Programming Concepts

In order for us to understand SIMIO framework, we need to be familiar with Object-Oriented Programming (OOP) paradigm. OOP roots can be traced back SIMULA, a discrete event simulation language based on ALGOL which was developed by Ole-Joan Dahl and Kristen Nygaard in the 1960s (Holmevik, 1994). This language proved to be an effective language for systems simulation and has changed the field of simulation modeling as well as software engineering for the better¹⁰. Although the OOP paradigm revolves around the same principles as conventional procedural programming paradigm, the implementation of the code in OOP framework is radically different compared to the latter.

Instead of the traditional implementation where a code executes a sequence of tasks, an OOP code creates *objects* that interact with each other to execute the task at hand. Each object *encapsulates* a set states and method which define its behavior in the simulation. The set of methods and states *inherited* by the object is dependent on the parent *class* from which they were derived. In a nutshell, the class is a simple template for creating objects belonging to the same group. In terms of SIMIO, all the base class objects belong to the intelligent object class.

An OOP paradigm closely resembles structures of “real world systems” which makes it an ideal candidate for modeling complex behavior of real world systems (Wirth, 2005). The OOP paradigm is based on following principles, which is briefly summarized (C. D. Pegden and Sturrock, 2010):

---

¹⁰ Among commercially simulation packages currently available, Flexsim (www.flexsim.com) and AnyLogic (www.xjtek.com) are also object oriented languages.

- “*Abstraction*”: The program structure should be simple and clear. In other words, complexity is hidden in objects. A simple simulation model in SIMIO is composed of objects.
- “*Encapsulation*”: Every object has its own set of methods and states which can be modified only by the object. This allows us to maintain objects independent of each other. In terms of SIMIO, every object encapsulates its processes and definitions to generate its behavior.
- “*Polymorphism*”: Each class must have a method to respond to standard messages but the response is dependent on the nature of the object. SIMIO implements the polymorphism concept by allowing users to override processes in the sub-classed object; thus implementing the polymorphism concept.
- “*Inheritance*”: Each class can be *sub-classed* to derive new classes which will inherit methods and states of the parent class. This helps us extend existing classes by adding new methods as well as override existing methods to create new classes. In the SIMIO framework, new classes are created by sub-classing from either basic classes or object library.
- “*Composition*”: This concept refers to the creation of new classes by combining existing classes. For example, when you build a simple model in SIMIO using the standard object library, you have just created a new class by composition which can be used in larger models.

A.II SIMIO and OOP concepts

The key distinction between SIMIO and other general purpose object-oriented languages is that OOP concepts are applied to create a modeling framework rather than a programming interface. In other words, SIMIO is a graphical modeling framework that facilitates the development of object-oriented simulation models (C. D. Pegden, 2008). For example, if you build a model composed of a server and a sink, it is equivalent to defining a new class of objects under the SIMIO framework. If we correlate it to OOP principles, you have implemented the *composition* concept to create a new class of objects (server and sink), which

is an amalgamation of server and sink behavior. Thus, all you need to for defining and creating new objects in SIMIO are modeling skills and not programming skills. Under the SIMIO framework, object and model are different words for the same thing.

A.II.1 SIMIO Object Hierarchy

One of the many unique internal design features of SIMIO is their layered object design. There are three tiers to an object in SIMIO namely:

1. “*Object Definition*”: As defined in (Kelton, Smith, Sturrock, and Verbraeck, 2010), “*An object definition defines an object’s view and behavior in the system.*” The object definition in SIMIO consists of properties, states, events, external view and logic. For example, there are object definitions of server, resource, etc.
2. “*Object Instance*”: When you drag a server from the library panel into the facility window, you have just created an instance of the server object (also referred as “instantiate”). It is possible to create multiple instances of your object in your model. Instances have property values and you may define one or more symbols for animation by clicking on them in the facility window.
3. “*Object Runspace*”: An object runspace is also known as object realization. The realization holds the current value of the object’s states. An object is realized when you run the model and each one possess a unique ID and may refer to a changeable symbol. At this tier, object instances may be dynamically created and destroyed during the run of the model and may have their state variables changed. All object instances are destroyed at the end of the run. A user may refer to an object during the runspace using the following convention; InstanceName[InstanceID].StateName. For example: server[1].resourcestate, modelentity[1].priority, resource[1].currentcapacity

A.III Object Definitions

An object in SIMIO is defined by its external view, properties, states, events, elements, lists, functions and tokens.

A.III.1 External View:

The “*External View*” property tab aids in creating a graphical representation (2D/3D) of the object. SIMIO provides the user with drawing options, importing files (.skp or .dxf formats) from an external medium or using images from Google’s 3D warehouse. You can also add external nodes as well as animation add-ins such as *queue*, *status labels*, *status plots*, *status pie*, *status gauge* or *button*. In a nutshell, what you see here is what you get!

A.III.2 Properties:

(Kelton et al., 2010) define “*Properties*” as “*Properties are defined within an object to collect information from the user to customize the object’s behavior*”. For example, When you drag a *server* in to your model, you can specify *processing time*, *resource seizes*, etc. in the right hand corner box called *Properties:Server1(Server)*.

There are four different types of properties:

1. Standard: Integer, Real, Expression etc.
2. Element: Station, Network, Material etc.
3. Object: Entity, Transporter, etc.
4. Repeat Group: A repeating set of any of the above properties

The value of properties cannot change dynamically during the run. Nonetheless, you can implement this special case by referencing the property. For example, In order to represent a trauma dependent treatment time is a special case where you will need to reference the processing time of the doctor to a property.

The advantage of properties is to help in differentiating instances of the same object in a simulation model. Users can modify properties of an object to generate different behaviors. To simplify the concept, properties can be thought of as modeling inputs.

A.III.3 States:

“*States*” are dynamic variables whose value might change during the model run. The state variables can be used as an input or output parameters. For example, patient health state can be a discrete state variable that will change in due course of the patient’s movement through clinic model.

State variables are broadly classified in three categories:

1. “*Discrete*”: these state variables change at discrete points during simulation. There are 6 types of discrete state variables namely Real, Integer, Boolean, Date/Time, List¹¹ and String.
2. “*Continuous*”: these state variables change continuously over simulation run time. There are 2 types of continuous state variables:
  - a. “*Level*”: the change is controlled by time based value of *rate*. The *rate* property by definition is a discrete change variable
  - b. “*Level with Acceleration*”: the change is controlled by time based value of rate and acceleration. The *rate*, *acceleration* and *acceleration duration* parameters are changed during discrete points of simulation.
3. “*Reference*”: State variables that refer to a SIMIO element or object and change at discrete points in time. They are of two types:

---

¹¹ A list discrete state variable defines a set of integer values with a discrete set of possible values from 0 to N. The zero value is indexed to the item beginning of the list. You can collect time-persistent statistics for each state defined on the list. For example, SIMIO’s default library object use list state variable to record resource state of an object.

- a. “*Element Reference*”: A state variable that can refer to SIMIO elements (Refer to A.III.5)
- b. “*Object Reference*”: A state variable that can refer to SIMIO base class object types which are Fixed, Transporter, Entity, Node, Link or Object

A.III.4 Events:

“*Events*” are helpful in communicating to an object that an important event has occurred in the simulation run. Events can help trigger custom behavior in objects. Certain events are incorporated in elements such as *Station.Entered*, *Timer.Event*, *Ridestation.Entered*, etc. by default. SIMIO allows user to define events under this tab. User defined events can be manually triggered using the FIRE step. Events can be classified in to four main groups:

1. “*Time event*”: A Time event is fired by a timer according to a specified time pattern which can be either a stationary or non-stationary pattern. For example, time events are useful for modeling the time dependent treatment times for patients.
2. “*Logic Event*”: A Logic event is triggered on a logical occurrence. A logical occurrence is a result of underlying model logic such as the “transferred” event which is fired upon an entity transferring into its station, owned by the object.
3. “*Change Event*”: A change event is fired whenever a specified discrete change variable value is changed. This event is fired by a monitor when used under the *discrete state change* mode.
4. “*Crossing Event*”: A *cross event* is fired by a monitor element when used under *cross state mode*. Under this condition, the event is fired when the specified threshold is crossed. Users can also specify the type of crossing to monitor: positive or negative.

A.III.5 Elements:

Quoting (C. D. Pegden, 2010), “*Elements assist in creating dynamic components of an object which possess properties (Input parameters) as well as states (Run time*

changeable values)." These elements states are changeable during run time by the owned object using processes and tokens. Certain elements do also have predefined events that can be incorporated in the building the simulation object. These elements are classified in to four categories.

1. General

- i. *STATION*: Represents a capacity constrained resource within which entities can reside. *Transfer* step is used to transfer entities in and out of station.
- ii. *TIMER*: Element with capability to fire a stream of events according to a specified interval type.
- iii. *FAILURE*: Element with the ability to define failure modes. These elements can be modified using the *FAIL* and *REPAIR* step.
- iv. *BATCH LOGIC*: element that can be used along with *BATCH* step to match multiple member entities to form a batch and then attach them to the parent entity. For example, in a chip and board assembly setup; the board is represented as parent member and chip combination assembled is implemented using batch logic.
- v. *STORAGE*: These are used to represent user defined queues for custom states. Unlike *SIMIO* defined queues, these queues are manually operated using *INSERT* and *REMOVE* step.
- vi. *MONITOR*: element that fires event when a state variable undergoes a discrete change or crosses the threshold value.
- vii. *NETWORK*: a list of links that is assigned to an entity to determine the shortest path
- viii. *ROUTING GROUP*: element that can be used with *ROUTE* step to assist in routing entities.

2. Statistics

- i. *OUTPUT STATISTIC*: Element that can be used to record value of an expression at the end of each replication.

- ii. *STATE STATISTIC*: element used to record time persistent statistics
  - iii. *TALLY STATISTIC*: element used to record observation based statistics
- 3. Workflow
  - i. *OPERATION*: Sequence of activities performed on object over time
  - ii. *ACTIVITY*: a particular work performed by an object over time
  - iii. *MATERIAL*: elements that can be consumed or produced by tokens using CONSUME and PRODUCE steps.
- 4. User Defined: SIMIO allows user to create custom elements for specific requirements using SIMIO's standard template in .NET platform.
- 5. Cost Center: An element which can be used to define an area where cost might be incurred or created. Cost center elements can be defined in hierarchical format, by which we can define a set of particular cost center elements as parents so that costs can be automatically added.

A.III.6 Functions

A function encapsulates a complex expression in to a simple expression. This expression (or called function name) can be used as a reference in a simulation program to evaluate the complex expression. SIMIO Functions can return a number, string, and object or element reference.

A.III.7 Lists

A list is used to define a collection of objects. In SIMIO, users can use lists to define Strings, transporters, nodes or objects. Each member of the list is assigned a number in ascending order from 0 to length of the list. A list can be referenced in simulation program using the *List.Listname.ListMember* expression.

A.III.8 Tokens

A token is smallest simulation object in SIMIO which executes a process. Each token is created at the beginning of the process and destroyed at the end of the process. By default,

SIMIO provides a token named *Token*. Token can also store states if state variables are defined, for which users need to create custom tokens under the definitions tab of the object.

A.IV SIMIO Standard Library

Figure A-1: Standard Library of SIMIO

A.V Additional Readings

Holmevik, J. R. (1994). Compiling SIMULA: A historical study of technological genesis.

IEEE Annals of the History of Computing, 16(4), 25-37.

Joines, J. A., & Roberts, S. D. (2010). *Simulation modeling with simio - A workbook* (1st ed)

Simio {LLC}.

Kelton, W., Smith, J., Sturrock, D., and Verbraeck, A. (2010). *Simio and simulation:*

Modeling, analysis, applications (1st ed.) Learning Solutions.

Pegden, C. D., and Sturrock, D. T. (2010). Introduction to simio. Paper presented at the

Winter Simulation Conference (WSC), In *Proceedings of the 2010 Winter Simulation*

Conference, ed., B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan and E. Yücesan. 1-

10. Institute for Electrical and Electronics Engineers, San Diego, CA.

Pegden, C. D. (2008). Introduction to simio. Paper presented at the *Proceedings of the 40th*

Conference on Winter Simulation, In *Proceedings of the 2008 Winter Simulation*

Conference, ed., S. J. Mason, R. R. Hill, L. Monch, O. Rose, P. Jefferson and J. W.

Fowler. , 229-235. Institute for Electrical and Electronics Engineers, San Diego, CA.

Pegden, C. D. (2010). *System and method for creating intelligent simulation onjects using*

graphical process descriptions. PA/US:

Wirth, N. (2005). *Good ideas through the looking glass* Retrieved from
http://www.inf.ethz.ch/personal/wirth/Articles/GoodIdeas_origFig.pdf

Appendix B : SIMIO-HC Toolkit

B.I Base Class Overview

- **Fixed Class:** Classes that give structure to the simulation model
  - **Redirect Object** : represents a base class where an entity can be held to be redirected
  - **Waiting Room**: It represents **Waiting Room** in an hospital with balking and reneging
  - **Registration Desk**: represents functions of an administrative object
  - **Patient Care** : An object with capabilities to treat a patient
  - **Patient Care with Lab Care**: An object to treat a patient including support for tertiary care logic

- ***Hospital Workstation***: An object that represents tertiary services such as laboratory
- ***Hospital Workstation with Lab Care***: An object that represents tertiary services where a patient also has to visit such as radiology departments, stress test centers.
- **Entity Class**: Classes that represent freely flowing agents that can be dynamically created or destroyed in the simulation model
  - ***Patient Entity*** represents a patient
  - ***Specimen Entity*** represents an entity related to a ***Patient Entity***
- **Transporter Class**: Classes that allow for an object to freely move in 3 Dimensions
  - ***Hospital Transporter***: Represents hospital vehicles such as Wheel Chairs, Stretchers, ambulances etc.
  - ***Hospital Worker***: Represents a medical worker.
- **Node Class**: Represents fixed object that facilitates connection between two fixed objects
  - ***Hospital Transfer node***: Represents a capacitated transfer node

B.II Patient Entity

The **Patient Entity** is an agent developed to mimic a patient's behavior using states which are modified through entity interaction with objects. Along with this, the **Patient Entity** is also designed to store patient related statistics as well as resources such as workers, transporters and clinics.

B.II.1 Structure

Lists

Table B-1: **Patient Entity** Lists

List Name	Type	Members
<i>Action</i>	String	Travel, Riding, Treatment, Administration, Awaiting Medical Resource, Awaiting Non-Medical Resource, Awaiting Medical Personnel, Lab Work and Radiology Work
<i>Wait area</i>	String	Waiting Room, Hospital Queue, Treatment Ward, Miscellaneous and Travel Path
<i>Wait type</i>	String	Medical Resources, Medical Worker, Treatment Ward, Transporter, None and Assisted Transporter

Functions

Table B-2 : **Patient Entity** Function Table

Definition	Return Type	Description
<i>Current Entity State</i>	Number	Returns the value of the <i>ObjectState</i>
<i>Currently Treating</i>	Number	Returns the value of 1 if the <i>ObjectState</i> is in <i>Treating</i> state
<i>Current Clinic</i>	Object Reference	Returns the location of the clinic where the entity is undergoing treatment
<i>Current Treatment Resource</i>	Object Reference	Returns the Hospital Worker object ID which is currently seized

State Variables Descriptions

Table B-3: *Patient Entity States*

State	Type	Description
<i>Treatment Status</i>	Boolean	Indicated whether the patient is undergoing treatment
<i>Statistics</i>	List	Stores statistics related to entity Action
<i>Area State</i>	List	Stores Area based statistics on where the entity is waiting i
<i>Object State</i>	List	Stores statistics related to what type of object is the entity waiting i
<i>Valid Observation</i>	Boolean	Indicates if the entity's statistics is to be included in the calculations
<i>Last Clinic</i>	Object Reference	Stores the object reference for patient care object types
<i>Last Nurse</i>	Transporter Reference	Store transporter reference to the last Hospital Worker of nurse group seized i
<i>Last Doctor</i>	Transporter Reference	Store transporter reference to the last Hospital Worker of doctor seized i
<i>Last Transporter</i>	Transporter Reference	Store transporter reference to the last Hospital Transporter seized
<i>Entity Reneging</i>	Boolean	Indicates if the entity is reneging
<i>Seize Assistance</i>	Boolean	Indicate if the entity needs to assistance (used for assisted transporters)
<i>Seize Transporter</i>	Boolean	Indicates if the needs a transporter upon creation
<i>Hold Clinic</i>	Boolean	Indicates if the clinic is held for the entity
<i>Last Hospital Worker</i>	Transporter Reference	Stores the reference to Hospital Worker if the property to store them is set to true in the agent
<i>Need Lab</i>	Boolean	Indicates the entity needs to visit to the laboratory
<i>Stay in Clinic</i>	Boolean	Indicates the entity is staying in clinic for the laboratory testing
<i>Test Quantity</i>	Integer	Indicates the number of tests ordered for the patient
<i>Results Ready</i>	Boolean	Indicates if the Results have been batched (handed over_) to the patient
<i>Waiting For Results</i>	Boolean	Indicates if the patient (agent) is still waiting for results
<i>Need Radiology</i>	Boolean	Indicates if the patient needs radiology visit

<i>Next Destination</i>	Object Reference	To store next object name for patient
<i>Wait Time</i>	Date/Time	To keep track of average waiting time for tally steps

Elements

A *PATIENT STATUS MONITOR* monitors discrete state changes in the *Statistics* state variable and triggers the PatientStatus_OnEvent process.

B.II.2 Process Descriptions

Patient Related References Storage

This process is responsible for storing the appropriate object references based on the state of the patient. If the patient state (*Statistics*) is *riding*, then it stores the seized object as a *Last Transporter* using the expression *CurrentTransporter*. If the patient state is *treating*, the current location is stored as *Last Clinic* using the expression *Location.Parent*. If the user allows the hospital resources to be stored, then *Last Hospital Worker* will be used to store the latest seized resource using the following expression:

Math.If(SeizedResources.LastItem==Is.Transporter,SeizedResources.LastItem,Nothing)

Figure B-1: Patient Related References Storage

B.II.3 Extensions

This class is extended to develop *Specimen Entity* Class.

B.III Specimen Entity

Specimen Entity is an agent class that is created when the patient needs laboratory work or radiology work. This entity is a representation of the patient as they go through the secondary stages, which is why they are subclassed from *Patient Entity*. A few states were added along with modification of the monitor process

B.III.1 States

Table B-4: Specimen Entity's States

State	Type	Description
Parent Entity Reference	Object Reference	Refers to the <i>Patient Entity</i> , to which the <i>Specimen Entity</i> are representing
Parent Specimen Entity	Boolean	Indicates if this <i>Specimen Entity</i> is a batch of specimen entities used for multitasking
Specimen Processed	Boolean	Indicates if the specimen has been processed

B.III.2 Process Descriptions

Patient Status_On Event Process (overridden)

This process is modification of PaitentStatus_OnEventProcess by overriding it and only storing the transporter and worker identifiers.

Figure B-2: Specimen Related Reference Storage

B.IV Hospital Transfer node

The ***Hospital Transfer node*** is a sub class of a standard transfer node. The node was subclassed in order for us to facilitate the transfer of one entity at a time through the node. This behavior will help us in creating objects (such as nurse station, waiting room etc.) where entities wait at a node until the first available resource serves them.

B.IV.1 Logic:

All nodes have a *PARKING STATION* where the entities reside. If the entities are allowed to capture this space, then they are assigned to a specific route. Thus, this does not emulate a first come first served behavior. Hence, we restrict this capacity using the *Current Capacity* state variable.

The modeler needs to specify the number of outbound links from the node using the property *Outbound Link Count*, which is visible under the Crossing Logic category. This is internally assigned to *Current Capacity* state variable which holds the crossing capacity of station. Thus, we decrease this variable as each entity crosses the node. Hence, when all the servers or resources are busy, the node will have a capacity of zero. Due to this, the entities will have to wait in the allocation queue. When a server or resource becomes free, the *Current capacity* is incremented by one. Thus, the first entity in the queue can grab the station's capacity and will be automatically routed to first available station.

In order to implement the logic, reset *Initial capacity* property in order for the user to always prompt to enter a value. Next, override the OnExited Process and append the current capacity decrement step using the Assign step.

B.IV.2 Process Description

On Exited (overridden)

The Decide Step checks if *PatientEntity.EntityReneging* variable is set to true, which indicates that the entity is refusing service. If not, then the entity is exiting the object to move in to a service location which means that node's *Current Capacity* should be decremented. For this, an assign step is placed to decrement the *Current capacity* of the node by one.

Figure B-3: Modified Process for *Hospital Transfer node*

B.V Redirect Object

The **Redirect object** is object mimics the basic behavior a routing station using the *Hospital Transfer node*. In other words, when entities arrive at the object they wait in the queue until there is a server available to assist them. This object helps us visualize the entities waiting to be served. The **Redirect object** can be used to model facilities such as reception area, nurse station, registration desks, waiting rooms, etc. In addition to the waiting, it also has a batching logic to match entities if they are waiting for a result or paperwork

B.V.1 Structure

The ***Redirect Object*** is a simple object with 3 nodes:

- “Input Node”- Belonging to basic node class where the entities enter and reside in *INPUT STATION* until they can be transferred to the *PROCESSING STATION*
- “Output Node”- This is a ***Hospital Transfer node*** and is responsible to routing entities to correct service station based on routing logic requested by user
- “Paperwork Node”- This a standard transfer class node used with “Output node” in certain situation where we need to use special objects to assist the entities transfer from the ***Redirect object***. This node can be used a node to route paperwork or any ***Specimen Entity*** that needs to be batched. It is by default assigned to *PAPERWORK STATION*.

B.V.2 Logic:

The ***Redirect Object*** has three stations, *INPUT STATION*, *PROCESSING*, and *PAPERWORK STATION*. All entities enter through *Input Station* and move on to *processing station* where they are held until they can be routed out. All entities exit through the “Output node” which can be used in conjunction with Paperwork node if entity needs to be picked up. The “Paperwork node” houses results that are waiting to be batched in the *PAPERWORK STATION*.

The batching process is triggered by the *ResultsArrived* event which is fired by the *PAPERWORK ARRIVED MONITOR*. The batching process requires a *BATCHING ELEMENT*, where both ***Patient Entity*** and specimen entities are matched based on the patient id which is stored in the *SpecimenEntity.ParentEntityReference* State Variable. The batch quantity is decided by *Test Quantity* state variable inside the ***Patient Entity***. Once batched, the *PAPERWORK ARRIVED MONITOR* is reset.

B.V.3 Process Descriptions

Input Process:

This process is triggered when an entity enters *INPUT STATION*. In the base model, the *INPUT STATION* has infinite capacity. Hence, the OnEnteredObject process basically accepts the entity into the object and transfers it to the processing station using transfer step. Upon entering the node, an additional *On Entered* add-on process will be executed if defined by the user.

Figure B-4: Process on Entering the Object

Main Process

This process inserts the entity into processing station's contents queue by seizing the capacity of the input station. After seizing, it checks whether *PatientEntity.WaitingForResults* is set to see if the ***Patient Entity*** is awaiting results. If so, it executes Batch Process else it transfer to the "output node" for which the logic of operation is defined in ***Hospital Transfer node*** section.

Figure B-5: Redirect Object Processing Logic

Paperwork Station Logic

The paperwork station process is triggered when an entity enters the *PAPERWORK STATION*. Upon arriving, Boolean state of *Paperwork Arrived* is set which is monitored by *PAPERWORK ARRIVED MONITOR*. The entities batched as members using the batch step and an optional *Results Batched* Addon process is run if defined.

Figure B-6: Paperwork Batching Logic

Batch Process

This process waits for *ResultsArrived* event to be fired. Once fired, it batches the patient with the results and resets the monitor so that the next paperwork set can be batched.

Figure B-7: Results Batching Process

B.V.4 Extensions

This class is further subclassed to develop the *Waiting Room* class.

B.VI Waiting room Object

The waiting room object is a specialized ***Redirect object*** with two specific additions: Renege and Balk feature. It also has built-in logic to collect statistics related to balking and renegeing as well as the average waiting time of an entity that successfully passes thru the station.

B.VI.1 Structure

Elements

The waiting room object uses elements to compute statistics related to the object. Tally statistics elements are used for *waiting time* calculations. Output statistics are used for reporting count statistics related to the number balked and renegeed. State Statistics are used to record the average number waiting in the queue.

Functions

- Active Load function returns the total number of entities occupying the processing station capacity (*Capacity.Allocated*).
- Relative Capacity function computes the relative empty space in the object using the expression

$$\frac{Capacity.Allocated}{Capacity.Initial}$$

The value is between 0 and 1 where 0 is if the object is empty and 1 is if the object is full.

B.VI.2 Logic

The ***Waiting Room*** has two new logical processes: Balking and Renegeing which can be modeled by implementing minor modifications to ***Redirect Object*** logic

Balking

In some cases, a patient may decide not to join a queue because there is no space for him/her to wait. This phenomenon is referred to as balking. The Balking logic can be enabled using *Allow Balk* property. When the waiting room is full, the entity is destroyed. In reality, it leaves the system without being serviced.

Reneging

Sometimes people tend to leave from a queue after spending some time. This behavior is referred to as reneging. Reneging may be attributed to multiple factors. This object at present considers queue length and time to model the reneging process.

The reneging logic can be enabled using *Allow Renege* property. Users will be prompted to enter *Time to Renege* and *Maximum queue position* for this logic. Users can also enter an optional *Renege Condition* to model custom reneging logic such as patient health state.

B.VI.3 Process Descriptions

Input Process (overridden)

This process is triggered when an entity enters the “Input node” and the entered ***Patient Entity*** state is assigned to be *waiting for non-medical resource*. Before transferring it to *PROCESSING STATION*, it checks whether the entity is allowed to balk using *Allow Balk* state. If the state is set to false, then the object will be transferred to *PROCESSING STATION*.

Figure B-8: Entity Trying To Enter *Waiting Room*

Main Process (overridden)

This process is similar to the base process with modification of incorporating reneging logic and steps to collect waiting time statistics. In order to collect state statistics on the average number waiting, the assign step increments the *Number Waiting* on entering and decrements upon exiting this station. An optional *On Entered* add-on process may be executed if defined. Following which, *Allow Renege* state value is checked. If set, then Renege Process is run in parallel to this process by setting advanced token option of the

Figure B-9: Waiting Room Process-I

execute step to continue. The capacity of the waiting room is seized and *Patient Entity* statistics is set to *waiting for medical resource* and the following assign step assigns the *Patient Entity* corresponding wait time ($PatientEntity.WaitTime = TimeNow$). Waiting is distinguished between waiting for results and waiting for a medical resource. If it is latter case, then it is transferred to the output node and, if successful (which means *Current Capacity* > 0), then *Patient Entity*'s statistics is set to *Travel* and average wait is tallied using the following expression:

$$\text{Average Wait} = \text{TimeNow} - \text{PatientEntity.WaitTime}$$

Figure B-10: Waiting Room Process-II

Balk Process

The balk process checks if the room is full using the *Processing.Capacity.Remaining==0* expression. If it evaluates to true, the entity is destroyed and balk count state variable is incremented. Before destroying, *On Entity Balking* add on process may be defined to be executed. If it fails to be true, the token's request is rejected

(*Token.ReturnValue=False*).

Reneg Process

This process is executed only after *Time to renege* delay is complete. Following which, the entity is searched in the waiting room using its ID in the *Processing station contents* queue with SEARCH step and the following expression:

Figure B-11: When an Entity Balks

$$\text{Candidate.PatientEntity.ID} == \text{Patient Entity.ID}$$

If the entity is found, the maximum queue length condition is checked where the user specifies this limit *Queue Position Limit*. If this also has been exceeded (or if the maximum condition is deemed to be unimportant), the optional *Reneg Condition* is evaluated. Following that the ***Patient Entity*** satisfies at least one of the conditions (or multiple ones depending on the model), the *PatientEntity.EntityReneging* property is set to true and destroyed. An optional *On Entity Reneging* add-on process may be executed if defined.

Figure B-12: Renege Process to evaluate reneging conditions

B.VI.4 Extensions

This class is extended to develop a ***Registration Desk*** class. Two classes were created: *WaitingRoom Ushaped* and *WaitingRoom Lstraight* by changing their external images.

B.VII Registration desk

The registration desk is a modified ***Waiting room*** object. The ***Registration desk*** object has exactly same properties as the ***Waiting room*** object, but the implementation method is modified. The registration desk provides a queuing facility along with processing unlike the waiting room which modifies the implementation of On Entered and Main Process.

B.VII.1 Structure

In addition to inheriting the structure from *Waiting Room* class, this class has a tally statistic for *TOTAL TIME*.

B.VII.2 Logic

The **Registration desk** waiting time statistic gives the average time an object was held at the input station. The balking should be implemented in On Entered Object process where the entity awaits. On the other hand, reneging process should be implemented after the **Patient Entity** enters *PROCESSING STATION* thereby entity reneges during service. The *PROCESSING STATION* capacity is assumed to be number of registration desks available and can be changed using the *Station Capacity* property and time to process can be specified using *Processing Time* property.

B.VII.3 Process Description

On Entered Station (overridden)

This process is triggered when an entity enters an “Input node”. Upon entering, the entity is assigned to waiting action and checks if the object allows for balking. If yes, the Balk Process is executed. If not, it is transferred to the processing station capacity

Figure B-13: When an entity enters **Registration Desk**

Main Process (overridden)

Once entered the processing station, the entity is differentiated if it is waiting for paperwork. If the entity is not waiting for paperwork, then the entity is allowed to renege where the Reneg Process is executed. Following is action; the entity is allowed to seize the

Figure B-14: Processing Entities In **Registration Desk- I**

processing station capacity.

After seizing the processing station capacity, the entity is removed from waiting count and tallied for average waiting time. The optional *processing add-on* may be defined to be executed just before execution of process time for the registration process using the delay step. After Processing, the *Processed* add-on process may be defined and executed. The node to which the entity must be transferred is decided by the user and is selected between “output node” and “paperwork node” using the *Use Output Node Property*.

Figure B-15: Processing Entities in **Registration Desk- II**

After the transfer is successful, the processing station capacity is released, and patient is set to the travel action state and tallies the total time for the patient in the object using the tally step.

B.VII.4 Extensions

The class is extended to develop Billing Desk class, where renegeing and balking options are disabled.

B.VIII Patient Care Object

The patient care object is a simple object designed to facilitate the process of creating sophisticated patient treatment processes. This object contains the basic processes, states and structure that can be extended with ease. The object can also be used as a simple treatment station.

B.VIII.1 Structure:

The patient care object comprises of three nodes namely for “Input”, “Output”, and “Worker” as well as a *PROCESSING STATION* to house the patient. All the nodes attached to this object do not map to a station with a belief that no patient will wait for a bed at the door but only at a waiting station. Moreover, the input and output nodes operate in auto transfer mode to the processing station and outbound link respectively. The worker node is a simple transfer node that is attached to the object to help the user create a network for workers. The user may draw parking station to enhance animation.

State Variables

The patient care object has a user defined list variable: ListResourceState. This variable accumulates state statistics for the patient care object and consists of 4 states:

1. *Idle*: On run initialization the patient care object defaults to this state (OnRunIntialized). The object accepts entity visit requests when it is this state.
2. *Allocated*: The object switches to this state when the process OnCapacityAllocated process is executed. The above mentioned process is executed automatically by the simulation system when an entity is allocated the specific object. This statistic can help us estimate the average time the object is awaiting an entity to begin its service routine.
3. *Occupied*: The object switches to this state after the entity enters the system. This state is assigned in the ProcessingEntered process and helps us account for the time spent by a **Patient Entity** in a treatment station waiting for a medical personnel
4. *Treating*: The object switches to this state when all the resources required for the patient treatment process arrives at “Worker Node” of the object. When using custom treatment logic, the user has to indicate when the resource state of the object is to be set to *Treating* state using *TREATMENT STATE MONITOR* property found under Treatment Logic.

Functions

The Patient care object has a few built-in functions:

1. Current Resource State returns the value of the present state of the patient care object
2. Currently Treating returns a value of true if the patient care object resource state is set to treating
3. Current Patient ID retrieves object reference to the most recent entity that is occupied the treatment station using the expression *Processing.Contents.LastItem*

B.VIII.2 Logic:

The patient care object is designed to depict a simple treatment station. A treatment station is allocated if the resource is idle. Afterwards, the treatment process logic takes over. The treatment process must have the capability to seize and releases resources as well as execute custom logic. After treatment, the resources must be released and patient transferred to the outbound link.

B.VIII.3 Process Description

OnRunInitialization

This process runs when the model is executed. It defaults the object state to the *idle* state.

Figure B-16: Upon Initializing Run

On Evaluating Seize Request

This process rejects token requests if the object is not in the *Idle* resource state. If the resource is not idle, the resource capacity is allocated to the ***Patient Entity***.

Figure B-17: When a seize request is evaluated

On Capacity Allocated

This process is executed when an entity is assigned the object. This process changes the state of the object to allocated and fires the `ALLOCATED` event. If the user has defined an optional add-on process in the *Allocated* dialog box, it will be executed by this process.

Figure B-18: When capacity of the object is allocated

On Capacity Released

The process is executed when the *PROCESSING STATION* capacity is released and assigns the patient care object to the *Idle* resource state. This process also fires the *Released* event and executes the optional *Released* add-on process if defined.

Figure B-19: Upon Capacity Released

ProcessingEntered

This process is executed when the entity enters the *PROCESSING* station and after the *PROCESSING . ENTERED* system event is fired. The process seizes the capacity of the station if the clinic is not already held by the patient, as indicated by *PatientEntity.HoldClinic==True*. Next assign's the resource state to *Occupied* and *Patient Entity* state is assigned to *Waiting for Medical Personnel*. Depending on user preferences, the steps define the execution of the *entered* add-on and optional *secondary state assignments*. After this step, the *TREATING* event is fired and *Pre Treating* resource seizes and releases are done by using the corresponding *SEIZE* and *RELEASE* steps respectively.

Figure B-20: Patient Treatment Process -I

The next step is the decision on treatment logic which can be user controlled by the *Custom Logic* property. If this option is taken, the user has to define custom logic using *Custom Treatment Logic* property which is referenced to an execute step. Secondly, a Treatment state monitor process is developed to handle the object's resource states, which is also implemented using an execute step with the advance token option of *continue*. If the user does not opt for custom logic, the object is set to the *treatment* state and the *DELAY* step runs the treating delay specified by the user as *Treatment Time*. Post treatments, secondary resources seized using *Treatment Resources* construct are released and the *Treated*

Event is fired. In addition to this, *Post treatment* seize and release constructs are defined and implemented and the entity is transferred to the “Output node” and its resource state is set to *Travel*.

Figure B-21: Patient Treatment Process -II

Treatment State Monitor Add-on Process

The treatment state monitor process is used to set the resource state of patient care to *Treating* under custom treatment logic mode. The monitor is configured by the user to state when the parent object is treating under their logic. When the user specific conditions are satisfied, the monitor event is fired. This allows the token to move down from the wait step to assigning the resource state to treating and the ***Patient Entity*** action to treating. The following ASSIGN steps reset the monitor.

Figure B-22: Treatment State Monitor Process

ProcessingExited

This process is executed up system event *Processing.Exited* is fired. The process releases the *PROCESSING* station capacity and assigns the resource state to idle if the ***Patient Entity*** does not need to hold onto the clinic which is indicated by *PatientEntity.HoldClinic*. If not, then the clinic is assigned to the hold state and the patient

Figure B-23: When a Patient Exits

state is assigned to either *to Lab work*, *Radiology work* or *Travel* depending on the type of states indicated by Patient states. *PatientEntity.NeedLab* state corresponds to ***Patient Entity*** requires lab, hence lab work. *PatientEntity.NeedRadiology* state corresponds to radiology work state and if both the state variables are false, the ***Patient Entity*** is travelling to another treatment center. An optional *Exited add-on* process can be defined and will be executed by this execute step. The following assign step assigns secondary state assignments on exiting. The SEIZE and RELEASE steps provide capabilities to seize or release resources on exiting. If the clinic is not held, the corresponding *REDIRECT* station's capacity is incremented which is defined by the user using *Output Node* property.

B.VIII.4 Extensions

This object was extended to create *PC_MaternityBed*, *PC_WardBed*, *PC_TreatmentChair* and *PC_TreatmentBed* by just changing the external symbol. Additional Laboratory logic were added in ***PC_Labcare*** object

B.IX Patient Care with LabCare (PC_LabCare)

The ***PC_Labcare*** class is extended from ***Patient Care*** class to provide additional framework to create and batch specimen entities. This section will only describe the additional changes made to the patient care object described in Section B.VII.

B.IX.1 Structure

Station

An additional station (*PAPERWORK*) is created to store paperwork and housed inside the "Specimen Node"

Continue Treatment Monitor

A monitor indicates when the treatment can be continued if the patient is in stay in clinic mode. The state variable monitored is *Continue Treatment*.

Batching

A batching element is used for results batching where the batch quantity is defined by *PatientEntity.TestQuantity* state. The batching logic is matching parent and member, where the parent is identified by *PatientEntity.ID* and member by *SpecimenEntity.ParentEntityReference.ID*

Tokens

CALL LAB token is defined with the capability to store parent entity reference and it's states related to treatment (Clinic Hold, Wait for results and Stay in Clinic) as well as the *Hospital Worker* reference.

B.IX.2 Logic

Since this object is subclassed from *Patient Care*, the object behavior needs modification as well as addition of the treatment processes. The main treatment process needs to be overridden to include steps to check when treatment can be continued (*PatientEntity.WaitForResults==True*) and check if laboratory work is required post consultation.

If laboratory work is required, then specimens must be created and the process must model the complexities involved such as seizing and releasing resources, specimen extraction time and logic, etc. Finally, if the *Patient Entity* decides to stay in the clinic, then we need to create processes to match results. After that we allow the treatment course to continue.

B.IX.3 Process Descriptions

Processing Entered (overridden)

The processing entered process was overridden. If the clinic is held by the patient, we trigger *Continue Treatment_monitor* process to ensure the treatment is continued only after

results are in.

Figure B-24: When an Entity Enters *pc_LabCare-I*

Following consultations, the **Patient Entity** is checked if it needs to hold onto the clinic (*PatientEntity.StayInClinic*). If yes, it checks for laboratory logic (Call lab process) before proceeding with the normal processing entered process.

Figure B-25: When an Entity Enters *pc_LabCare-II*

Call Lab Process

The Call lab process uses the *CALL LAB* token to facilitate the logic setup. The **Patient Entity**'s *PatientEntity.NeedLab* state is checked if laboratory work is required. If it is not required, the token's request is rejected by setting its return value to false. If lab work is required and patient has to stay in clinic, the process Processing Entered and Treatment Monitor are suspended before the specimen is created using the suspend step and continue treatment process is executed .

Figure B-26: Process for providing secondary care

The specimen is created is assigned references to the parent entity using the CALL LAB token states. The token's is first assigned the states of *Patient Entity* and upon creation assigns these states to specimen entity. After referencing, the *Specimen Entity* is transferred to the “specimen node” and routed to laboratory. The SEIZE and RELEASE steps are owned by the patient care object and not by the entity which was implemented using advanced options of the steps.

Continue Treatment Process

The continue treatment process controls when the treatment process (Processing Entered) can continue with its logic flow. The continue treatment process decides if the patient is going to wait for their results using the state variable *PatientEntity.WaitForResults*. If they are going to wait for the result, the resource state is set to *On Hold* and waits for the ContinueTreatment event to be fired by the *CONTINUE TREATMENT* monitor. Once fired, the parent (patient) is batched to the results using the BATCH step and the *Continue Treatment* monitor is reset along with resuming the Processing Entered and Treatment State Monitor processes using the resume step. An optional *post lab results batch* add-on process is executed at the end of this process.

Figure B-27: Continue Treatment Process post visit to lab

Results Arrive Process

The process is triggered by the `ResultsArriveStation.Entered_event` which indicates a result entity (specimen entity) has entered the “Specimen Node”. The *Continue Treatment* state variable is set and specimen entity’s states for *hold clinic* and *wait for results* are reset. The batch step batches the members and execute steps executes results arrive add-on process if defined.

Figure B-28 Results Arrive and Batch Process

B.X Hospital Workstation

The ***Hospital Workstation*** object is used for modeling secondary patient care services such as laboratories, blood banks, etc. where the ***Patient Entity*** is not handled, but an entity belonging to a patient is.

B.X.1 Structure

Elements

The ***Hospital Workstation*** has 3 stations namely:

- *SETUP STATION* for setup processes

- *PROCESSING STATION* for processing services
- *POST PROCESSING STATION* for post processing services

All the stations correspond to their *ACTIVITY* element which in turn controls their properties. These activities are sequenced using the *OPERATION* element. The *OPERATION* element facilitates the definition of changeover or sequence dependent setup times using the corresponding properties. The *BATCHING* element facilitates batching of entities based on their *ParentEntityReference.ID*.

External View

The external view is composed of five nodes, of which the input node is basic node operating in auto transfer mode to setup station. The three other nodes with the exception of output node are merely present to facilitate graphical connections

Lists

There are two lists for the Hospital workstation: Resource states and Activity List. The latter one is a string list for the list property Activity List Name. This list property is referenced in the operations resources to vary the seize and release logic.

The Resources state list contains the following states to describe *Idle*, *Processing*, *Blocked*, *Failed*, *Off shift*, *Setup*, *Post Production*, *Waiting For Secondary Resource*, *Waiting For Material*, and *Allocated*.

Functions

The Active Load function returns the current number of entities that are using the system in setup, processing, and post-processing stages

Tokens

A special token called *Specimen Token* is created whose implementation is exactly the same as *CALL LAB* token described in section B.VIII.1.

B.X.2 Logic

The ***Hospital Workstation*** puts the entities through setup, processing, and post processing stages using separate processes. The capacity allocation and scheduling is implemented using built-in SIMIO standard processes. Specimen Batching process holds the batching logic of elements before or after setup which facilitates multitasking.

B.X.3 Process Descriptions

On Run Initialized

When a simulation run is initialized, processing capacity is assigned as well as checked to see if the resource state is set to `On Shift`. If so, the respective Add-on process is executed if defined.

Figure B-29: Run Initialization Process

On Evaluating Seize Request

This process checks if the Resource state is idle and if it is, it allocates the seize request and sets the object state to `Allocated`. If not, it rejects the token's request by returning a false return value.

Figure B-30: Seize Request Process

On Capacity Changed

This process is auto-triggered when the processing capacity of the object is changed by a schedule or by an external process. The first step is to check if the object is coming in to shift during this capacity change. If so, we do not allow capacity change to be greater than 1 and notify the user. Then, we assign the initial capacity of 1 to the *CurrentCapacity* state variable and resume the activities.

Figure B-31: Capacity Changed Process

If the change was to off-shift, then processing capacity (*CurrentCapacity*) is assigned to 0 and all the activities are suspended. If the change was in between shifts, then we do not allow it and notify the user.

On Entering Setup

This process is triggered by *SetupBuffer*. Entered event which fired every time an entity is transferred in to the setup station. Following the event fire, the delay steps execute the transfer in delay if specified by the user using *Transfer in Delay* property. The

end transfer step marks the successful transfer of the entity into station. If there is batching before setup which is indicated by the *batch before setup* property, then the execute step executes Specimen Batching Process. If the entity transferred is a parent entity which is indicated entity's *Parent Specimen Entity* state, then it is directly transferred to seize the workstation capacity to start the operation of the workstation. The parent ***Specimen Entity*** is created and batched with the specimens in Parent Specimen Create process.

Figure B-32: On Entering the Setup Buffer

The workstation processes are started using START OPERATION step and each activity is initiated using START ACTIVITY.

Perform Setup Activity

This process assigns the resource state to *waiting for materials* and consumes materials if defined and available to use using the CONSUME setup. Secondly, the ASSIGN step changes the resource state to *waiting for secondary resources* followed by seize step. After seizing resources, the resource state is reassigned to *setup* and *setting up* add-on is executed if defined by the user.

Figure B-33: Performing Setup Activity- I

If there are materials defined to be produced, the produce step will create this material. Since we allow for batch processing, the next logical step is to check if the current batch is the last one using the expression¹²:

Entity.Operation.Activity.BatchNumber ==
Entity.Operation.Activity.NumberBatchesRequired

Figure B-34: Performing Setup Activity-II

If the expression evaluates to true, it indicates we have finished the setup for the last batch and release secondary resources and end the activity using END ACTIVITY step. Secondary assignments defined as *On Exiting Setup* property is assigned at this stage.

The next criterion is to check if the property for Specimen batching is set to *after setup*. If it evaluates to true, we use start activity and end activity to finish processing and post processing and end the operation. We also release the capacity and execute Specimen Batching process. If not, then we just transfer it to processing buffer

Specimen Batching

The specimen batching process batches the member specimens using the BATCH step after it executes parent specimen create process and small delay to ensure that member

¹² Not to be confused with *Specimen Batching*. This property states each **Specimen Entity** represents a batch of entities. Using *Specimen Batching*, the workstation creates batches.

batching happens after parent specimen creation. The DECIDE step ensures that only one parent gets created for every batch, and which is implemented using the state variable *Parent Created*. It is set when the first member of the batch flows through the process

Figure B-35: Specimen Batching Process

Parent Specimen Create

The parent specimen create process creates a *Specimen Entity* using the CREATE step and then uses BATCH step to batch the parent and members. This step is followed by state assignment of *Parent Specimen Entity* in the *Specimen Entity* to identify it as parent entity and avoid the setup processes. After this identification, the transfer step transfers the parent *Specimen Entity* to its setup or processing buffer depending on the batch type (before or after setup respectively).

Figure B-36: Creating and Batching Parent *Specimen Entity* Process

If it has to transfer to the processing buffer, then we use dummy START ACTIVITY and END ACTIVITY steps to complete the setup activity.

Perform Processing Activity

The perform processing activity is triggered when the entity is transferred to the processing buffer. If the setup buffer station capacity is zero, then false arm of the DECIDE step will be executed to mimic the functions of Perform Setup Activity and then follows the same path as the true arm of the DECIDE step.

Figure B-37: Logic of Processing Activity-I

If the setup buffer is greater than 0, the `Entity.Transferred` event is fired using `END TRANSFER` setup. Following that, state assignments defined under *On Entering Processing Buffer* are made and *PROCESSING* activity is initiated. The following steps in this process are exactly same as described Perform Setup Activity, except using processing related properties. The last part is a transfer step to *POST PROCESSING STATION BUFFER*.

Figure B-38: Logic of Processing Activity-II

On Exited Processing

After transferring the entity to the post-production buffer, capacity is released if the token has the capacity of the workstation checked using the expression: *Token.HasAssociatedObject*. If it evaluates to true, the workstation capacity is released and set to idle if and only if post processing activities are not defined. All the false arm branches are connected to *exited* add-on process EXECUTE step. All steps except for execute are included in the run if Post Processing buffer Initial capacity is 0. This condition is included as an *Exclusion* property for the steps.

Figure B-39: Upon exiting processing buffer

On Entered Post-Processing Buffer

The process first ends the transfer of the entity and decides if the specimens have to be un-batched by checking the specimen entity's *Processed_state* variable. If it is set to false, the *on entering* assignments for post-production buffer are assigned which is defined by the user under

Figure B-41: Perform Post Production Activity Process

On Exited Post Production

When a specimen exits the post production buffer, the `PostProductionBuffer.Exited` event is fired and the resource state of the workstation is assigned to *idle*.

Figure B-42: Post Production Buffer Exited

B.X.4 Extensions

This class has been extended to create *Hospital Workstation with Lab care* class. Another class created is the HWS_CentrifugeLab where the external image has been modified.

B.XI Hospital Workstation with Lab Care (HWS_Labcare)

The *Hospital Workstation with lab care* is subclassed with Hospital Workstation with the ability to handle *Patient Entity* at the setup and processing stages and create *Specimen Entity* to handle the post processing stage. In addition to the existing structure the *HWS_Labcare* class can fail and may also batch results with patients after post-processing stages.

B.XI.1 Structure

In addition to structures in Hospital Workstation, this object has failure elements, additional stations, and a batching element for results

Failures

- *CALENDAR TIME BASED FAILURE* is a timer that triggers event based on calendar time and date
- *PROCESSING COUNT BASED FAILURE* is a timer that triggers event based on processing count
- *EVENT COUNT BASED FAILURE* is a timer that triggers an event based on the specified occurrences of an event.
- *PROCESSING TIME BASED FAILURE* is a timer for failing based on processing time

For all failures, users have to specify down time and uptime in order for the failure to be modeled. Every event triggers a Failure element which references to failed and repaired processes.

Batching

A batching element is *BATCH AT STATION* matches the patient and ***Specimen Entity*** based on their ID. This element uses the *RESULTS PREPARED* monitor to batch parents which monitors the state variable *Results Prepared*.

Station

- *RESULTS WAIT STATION* is the station where patient entities reside while they are waiting for the result
- *RESULTS STATION* is the station that collects results

External View

An additional transfer node named “Specimen Node” is added for Specimen transfer

B.XI.2 Logic

For this object, in the setup and processing, it deals with the ***Patient Entity*** thereby its execution steps differ compared to hospital workstation. The ***Specimen Entity*** is used only in post-processing stage, hence requiring the addition of dummy steps to continue with operation logic. In addition to these, the object also needs to have a batching logic for batch the results and patient, if the user sets the property *Batch results at station*

B.XI.3 Process Descriptions

On Run Initialized (overridden)

This process is overridden and enables all the types of failures. The Assign step disables randomness for failures using the expression *Run.RandomnessDisabled==False*. The subscribe step subscribes to failure timer event and runs the Failure Occurrence Logic upon failing.

Figure B-43: Initialization of the *HWS_LabCare*

Failure Occurrence Logic

The process runs for every type of failure event fired by timer. The fail step uses the failure element to activate the failing of the object which triggers On Failed Process. The resource state of the object is set to failed and the repair time is modeled as a delay which is specified as *time to repair* property. After the delay step, the repair step indicates the failure has to be deactivated and the object is ready to continue its work, which triggers On Repaired Process. The FIRE steps are used to reset all types of failure timers and ALLOCATE step is used to allow an entity to seize its capacity.

Figure B-44: Implementing Failure

On Failed Process

The on failed process suspends all types of activities using the suspend step and sets the *CurrentCapacity* of the object to 0. The *failed* add-on process is executed followed by resuming the failure occurrence logic. Just before the repair delay is about to begin, the user may define *Repairing* add-on process which is executed using an EXECUTE step.

Figure B-45: On Failed Process

On Repaired Process

The process resumes all the activities of the object and add-on process of repaired if defined. It also assigns the *Current Capacity* to be 1 using the assign step.

Figure B-46: On Repaired Process

On Entered Setup Buffer(overridden)

The process is triggered by SetupBuffer .Entered event which indicates a **Patient Entity** entering the station. Following this entry, a transfer in delay may be defined before triggering the transferred event for the entity using END TRANSFER step. **Patient Entity** is assigned to *Radiology Work* state and *on entering setup buffer* state assignments are also done. The SEIZE step seizes the processing station's capacity and starts the operation using start operation. The setup activity is started using START ACTIVITY step and executes the Perform Setup Activity.

Figure B-47: On Entered Setup Buffer

Perform Setup Activity (overridden)

This process is similar to the one described in the previous section without the batch process for the specimen.

using a series of dummy START ACTIVITY and END ACTIVITY steps reaches post processing activity. It is now transferred to post production buffer using a TRANSFER step.

Figure B-50: *Specimen Entity* Create-II

On Exited Processing (overridden)

The process is triggered when **Patient Entity** leaves the processing buffer station. If it has the workstation capacity, then it is released. This is checked by *Token.HasAssociatedObject* if and only if the post processing buffer's capacity is 0. The resource state is also assigned to idle subject to previous conditions. If none of these conditions are true, then *on exited* add-on process is executed, which is user defined.

Figure B-51: Processing Buffer Exited

Perform Post Processing Activity Process (overridden)

The process is exactly the same as described in the previous section with the addition of transfer out logic for specimen. If the specimen is to be batched with **Patient Entity** at the object, then the results are transferred to the *RESULTS STATION*, else it is transferred to the specimen node.

Figure B-52: Batching Addition to Perform Processing Activity Process

Results On Entered

When the results are ready, the post processing activity transfers them to the results station if and only if the results are batched at a station and the event *ResultsStation.Entered* is triggered. This assigns the state variable *Results Prepared* to a value of 1 to trigger the *RESULTS MONITOR* event and the members are batched using the batching element. The resource state is set to *idle*.

Figure B-53: Results on Entering the Results Station

Results Awaited

The results await station holds the **Patient Entity** until the results are ready to be batched. The ResultsAwaitStation.Entered process triggers the results process and waits for the ResultsPreparedMonitor.Event. Once fired, the monitor is reset and **Patient Entity** is batched as a parent. The **Patient Entity**'s state variable *Result Ready* is set and transferred to the output node using TRANSFER step.

Figure B-54: Waiting for Results Process

B.XI.4 Extensions

This class is extended with its external image modified to create HWS LC Xray, HWS LC GammaCamera, HWS LC PET, HWS LC CAT and HWS LC MRI classes.

B.XII Statistics Collection Sink:

The statistics collection sink object is subclassed from the standard library sink objects. The main function of this module is to collect tally statistics from the *PatientEntity.Statistics* list state variable.

B.XII.1 Logic

These tally statistics are collected by adding steps TALLY steps to the OnEnteredInputBuffer process before the entity is destroyed. In order for the tally steps to be included in the run, user has to set the *Collect Patient Statistics* property in this object. Two groups of Tally statistics are collected using these processes:

Step 1. Total Time Tally Statistics

The total time tally statistics is retrieved using the command:

$$PatientEntity.Statistics.TotalTime(i)$$

where i represents the *Patient Entity* action list member (Refer to Table B-2)

Step 2. Percent Time Tally Statistics

The percent time statistics were calculated using the formula:

$$Percent\ Time\ in\ State\ (i) = \frac{Total\ Time\ in\ State\ (i)}{\sum_{i=0}^n Total\ Time\ in\ State\ (i)}$$

This was implemented in SIMIO using the following commands:

$$\frac{PatientEntity.Statistics.TotalTime(i)}{\sum_{i=0}^n PatientEntity.Statistics.TotalTime(i)}$$

These statistics are grouped as per the tally type.

B.XII.2 Process Descriptions

On Entered Input Buffer(overridden)

The process is overridden and a few steps were added to collect tally statistics. The decide step checks if the entity's state variable *Valid Observation* is set to true. If so,

the tally statistics are collected as described in object logic, and the entity is destroyed using the DESTROY step. The tally steps will be included if and only if the sink's *collect patient statistics* property is set to true.

Figure B-55: Collection of Statistics

B.XIII Hospital Transporter

The modern day hospitals heavily depend on resources such as stretchers, wheel chairs, ambulances, crash carts among many others and are seamlessly integrated in to their service protocols. For healthcare simulation modelers, this is a challenge depending on the package use, and for SIMIO users, this means a series of modifications to **TRANSPORTER** class which might add more complexity to the modeling process. Thus, we have developed a *Hospital Transporter* (sub-classed from the new **TRANSPORTER**) object group to ease the modeling hurdles for SIMIO users.¹³

B.XIII.1 Structure:

The *Hospital Transporter* main function is to carry patients and in order to facilitate this; it has a *RIDE STATION* inherited from the base **TRANSPORTER**.

States:

- *Home Node Id*: Holds the id of the Home node specified by *Initial Node* property
- *Resource State*: List state variable to keep a track of the transporter state statistics

---

¹³ Users are requested to note that they need to change the properties of the transporter object by right click on the object in the *object navigation* window and selecting properties. In the *Model Properties* window, change the property of *Resource Object* to TRUE.

- *Assist Seized*: A Boolean state variable to indicate whether assistance has been seized.

Properties:

- *Initial Node*: It is mandatory to declare the initial node where the transporter object's instance be created.
- *Park To Load/Unload*: This property allows the object to use parking queue of the node to load and unload.
- *Idle Action*: This property allows user to specify the action for transporter object when idle
- *Park While Busy*: This property specifies where the object should use the parking queue of the node while being seized.
- *Assisted Vehicle*: Many of healthcare objects need to seize a worker to move, which is specified using this Boolean property. If selected, the user needs to use the repeating properties editor to specify the resources to be seized.

Functions

Current Resource State functions return the value of the resource state of the transporter at the time of query. It evaluates the *ResourceState* variable.

We have included more add-on processes' to increase the flexibility of object's behavior which shall be discussed in context under Process description section.

B.XIII.2 Logic

The ***Hospital Transporter*** object must have the ability to carry entities and move from one node to another. In addition to this, it should also provide modeler flexibility by signaling when its state changes, which is implemented using a series of events and add-on processes.

B.XIII.3 Process Descriptions¹⁴:

On Created

Upon Creation, the process assigns the transporter is home node and initial node.

Figure B-56: On creating the transporter

On Run Initialized

When the model is run, the process is run for executing the *Intialized* add-on and is parked at the initial node.

Figure B-57: On Initializing the Simulation Run

On Rider Loading

When the rider is being loading, this process is executed which seizes the *RIDE STATION*'s capacity. Following that, we execute *Loading* Add-on and then the process waits for the *Entity .Transferred Event* to indicate the entity has been loaded and executes the *Loaded* add-on process. In order to verify that entity is loaded, we include an epsilon

---

¹⁴ Only processes that differ from vehicle object have been explained in this section

delay step before firing `RiderLoaded` event and assign the resources state to transporting for this process.

Figure B-58: Rider Loading Process

On Rider Loaded

Upon `RiderStation.Entered` event being fired, the process completes the transfer of entity as well assigns **Patient Entity** state to *Riding*. Following that, state references of the entity are copied to the vehicle's reference states so that they can be used in the model. If the **Patient Entity** state variable *Need Assistance* is enabled, then the entity seizes assistance if and only if the transporter is of assisted type.

Figure B-59: Rider Loaded

On Rider Unloading

This process executes *Unloading* add-on, if defined and awaits `Entity.Transferred` event. Following this, it executes *Unloaded* add-on if defined and process is delayed by epsilon to ensure transfer of entity.

Figure B-60: Rider Unloading Process

On Visiting Node

This process is run every time a transporter enters a node. It is responsible for the routing logic embedded in the transporter object. This process also controls the execution of assistance logic which is implemented using the *Assistance* properties group, whereby the transporter can seize external resources before transporting the entity. Given the special nature of the OnVisitingNode process, we implemented seize and release of these resources using events which are explained under Seize Assistance and Release Assistance sections respectively.

Figure B-61: Deciding if the Node is Destination Node-I

Upon entering the node, the execute step runs the entered add-on process and checks if the current node is the destination node. If not, it assigns the node to the ignore list using the state variable *IgnoreDestinationNodeID* which is equated to

$$CurrentNodeID * ((DestinationNodeID == 0) \&\& (NumberRiders > 0))$$

If the transporter is un-parked, then the exiting node add-on process is executed.

If the transporter is visiting a node, then it checks if the vehicle's is not busy. If so, it parks using the PARK step and fires the arrived event of the rider using ARRIVE step. The select visit will be evaluated using the task selection strategy enumerator; else it will wait for *RemainInPlaceEnd*

Figure B-62: Pickup or Drop-off

If the transporter is not busy, then it means that it is available for transporting. So, the DECIDE step checks if it is loading or unloading. If it is doing either of the tasks, then it has to park before dropping of the entity and wait until the rider is unloaded. For parking, the DECIDE step checks using *IsParked* identifier. If not, then it parks and continues to pick up or drop off. For drop off, an unloading time can be specified using *unload time* property. If this drop-off fails, then it suggests the alternative: pickup. For the pickup, we check if the rider needs to seize assistance. If it evaluates to true, then *GetAssistance* event is fired and

waits for the `AssistanceSeized` event. If it has already been seized or is not required, then the rider picks the entity up using `PICKUP` step for which load time can be specified using *load time* property. It waits until the `RiderLoaded` event before initiating the next pickup from ride pickup queue. If it fails, then it means that all entities in the ride pick up queue are picked up and fires the `RiderPickupFinished` event. The destination of the transporter is chosen using the `SELECT DROP OFF` step and subsequently the `SELECT VISIT` step, both use the *task selection strategy* property.

If it is in the destination node and is not picking up or dropping off entity, then it means it is responding to a move request. Thus, we release the vehicle's capacity and use `DECIDE` step to check if the vehicle is *idle*. If yes, then we plan the next visit using plan visit which is controlled by *task selection strategy* property. If no visits are there in the ride request queue, then we park at the current node if the vehicle is supposed to park at the node. If not, then we set the vehicle to home node using `SET NODE`.

Figure B-63: Parking Decision Process

SeizeAssistance

This process is triggered by GetAssistance event where the required resources that are specified using Assistance properties are seized upon which AssistanceSeized event is fired and the AssistSeized Boolean variable is set to true. Once set, it sets resource state to *transporting*.

Figure B-64: Seize Assistance Resources

ReleaseAssistance

The process is triggered when the ride is parked and only executes further if *AssistSeized* property is set true; which indicates that the vehicle has seized an external resource for assistance. If it has, then it checks to see if the current node is the home node for the transporter. If this is true, then the resources seized are released and we set the *AssistSeized* Boolean property to false.

Figure B-65: Release Assistance Resources

B.XIII.4 Extensions

This class was further sub-classed to create the *Hospital Worker* class and *HT Stretcher* class which is an assisted transporter.

B.XIV Hospital Worker

The *Hospital Worker* is sub-classed from the *Hospital Transporter* object whereby the behavior is slightly modified to implement worker characteristics. The primary difference between a transporter and worker is that a transporter can only transport entities where as a worker can do both transporting and be seized as a resource to perform actions. For this, On Capacity Changed and On Capacity Released processes were modified. The second addition to the *Hospital Worker* is the ability to do periodic visit using a fixed routing element which requires the modifications of On Created and On Visiting Node Processes.

B.XIV.1 Process Descriptions

On Capacity Changed (overridden)

Being a worker, we provide the ability for the user to create a work schedule. When work schedules are implemented, worker may be on shift or off shift which is indicated by the capacity assigned to the object depending on the defined schedule. Thus, if it is on shift the user can make state assignments and the object may be seized for processing or used as transporter. On the other hand, if the worker is in off shift, then we proceed with state assignments and ignore the requests.

Figure B-66: Worker Capacity Changed Process

On Capacity Released (overridden)

When the capacity is released, the object checks whether its visit request queue is empty. If it is, then `RemainInPlaceEnded` event is fired and object follows the action defined for `Idle` State. If not, then the object `RemainInPlaceEnd` event is fired and the object visits the next node in the list and performs the necessary duties.

Figure B-67: Upon Worker Capacity Released

On Created(overridden)

When created, the **Hospital Worker** is checked if they are doing periodic visits. If so, their routing property will be set to a fixed routing by the user. Once set, the process seizes the vehicle's capacity and sets destination node from the sequence table attached to this worker.

Figure B-68: Initialization Of Worker

On Visiting Node(overridden)

The process has a minor modification from the one described in previous section. After `RiderPickupsFinished` event is fired, the entity is checked to see if it is on a

fixed routing. If so, the destination node is selected from the sequence table. If not, the normal process flow described in previous section is followed.

Figure B-69: Implementing Fixed Routing

B.XIV.2 Extensions

The class is extended to create HW Doctor and HW Nurse class by changing the external image.

Appendix C : CIMS Model Development

C.I Emergency Room Model

C.I.1 Model Description

The emergency room model consists of eight treatment units which serve the patients based on severity. Patients are assumed to have four scales of severity, each with a different probability and severity scale is in increasing order (1- lowest and 4- is the highest). The arrival rate varies throughout the day, and the total number of arrivals is known with a fraction belonging to the ambulance. All patients except the ambulance delivered patients are required to check in at the registration desk and proceed to wait in a waiting room. As for the ambulance, it is diverted if the waiting room is full and the registration desk also does not admit patients if the waiting room is full. The waiting room can hold 15 patients. Patients in the waiting room tend to leave the clinic after some time if there are more than 10 people waiting in front of them.

The checkup process consists of a visit by a physician or nurse after which the patient may be discharged or be referred to laboratory or radiology department. If discharged, the patient goes to the billing department and exits the hospital.

C.I.2 Model Layout

Figure C-1: Emergency Department Process Layout

C.I.3 Model Construction ¹⁵

- Step 1. Drag and drop a **Patient Entity, Source**, seven **transfer nodes**, eight **Patient Care** units, a **Waiting Room**, **Registration Desk**, a **Billing Desk**, and a **sink** to capture demand lost. Arrange as per Figure C-1. The decide steps logic are embedded in the objects
- Step 2. Go to Data → Rate Table and click on new arrival table and rename it to ED_Arrival. Remember to populate this table with numbers similar to the Table C-1 to generate patient arrivals in a time varying manner.

Table C-1: ED Patient Arrival Table (*ED_ArrivalTable*)

Start Time	End Time	Arrivals per hour
Day 1, 00:00:00	Day 1, 01:00:00	30
Day 1, 01:00:00	Day 1, 02:00:00	25
Day 1, 02:00:00	Day 1, 03:00:00	25

¹⁵ Unless specified, assume all links to be time paths and use any distribution.

Day 1, 03:00:00	Day 1, 04:00:00	10
Day 1, 04:00:00	Day 1, 05:00:00	12
Day 1, 05:00:00	Day 1, 06:00:00	15
Day 1, 06:00:00	Day 1, 07:00:00	6
Day 1, 07:00:00	Day 1, 08:00:00	11
Day 1, 08:00:00	Day 1, 09:00:00	10
Day 1, 09:00:00	Day 1, 10:00:00	12
Day 1, 10:00:00	Day 1, 11:00:00	14
Day 1, 11:00:00	Day 1, 12:00:00	5
Day 1, 12:00:00	Day 1, 13:00:00	6
Day 1, 13:00:00	Day 1, 14:00:00	3
Day 1, 14:00:00	Day 1, 15:00:00	10
Day 1, 15:00:00	Day 1, 16:00:00	12
Day 1, 16:00:00	Day 1, 17:00:00	5
Day 1, 17:00:00	Day 1, 18:00:00	8
Day 1, 18:00:00	Day 1, 19:00:00	7
Day 1, 19:00:00	Day 1, 20:00:00	12
Day 1, 20:00:00	Day 1, 21:00:00	16
Day 1, 21:00:00	Day 1, 22:00:00	40
Day 1, 22:00:00	Day 1, 23:00:00	30
Day 1, 23:00:00	Day 2, 00:00:00	35

Step 3. Change the entity type in the source to ***Patient Entity*** and arrival mode to *Time varying arrival rate* and select the *ED_Arrival Table*

Step 4. On the output node of the source, double click *On Entered* Add-on Process and define a process to split the arrivals in to 3 transfer nodes. Rename one to be *Ambulance transfer node*. This is to emulate arrivals from different ways for an ED.

Step 5. Assign *PatientEntity.Priority* state variable using ASSIGN Step to represent patient severities (1- lowest, 4 highest). Let the priorities have a discrete distribution as defined by: *Random.Discrete(1,.5,2,.75,3,.9,4,1)*

Figure C-2: Transferring patients to different entrances

- Step 6. For patients, who are transferred to the ambulance node, they need to ride an ambulance to the hospital. For this, drag and drop a hospital transporter, rename it to **HW_Ambulance**, and set its *home node* property to be *ambulance transfer node*.
- Step 7. Ensure that the ambulance transfer node is configured for entities to ride on the ambulance (set *Ride on Transporter* property to *True*) and is connected to an entrance near the hospital as well as the sink using bidirectional timepaths.
- Step 8. Create a state variable of integer type and name it *RefusalOfService*
- Step 9. To model ambulance diversion, double click on the hospital transporter's add-on process → *On Loading*

Figure C-3: Diverting Ambulance if waiting room is full

- Step 10. Use a DECIDE step to check if *ED_RO_WaitingRoom.RelativeCapacity==1*
- Step 11. If true, set the node using SET NODE step to the sink and increment the refusal of service state by one.
- Step 12. If false, set the node to an entrance transfer node which should be directly connected to **Waiting Room**'s output node.
- Step 13. Next connect the rest of the entrance nodes registration desk
- Step 14. Change the capacity of the registration desk to 2 and enter *Random.Pert(1,3,5)* as processing time with units to be minutes.
- Step 15. To model refusal to admit when waiting room full, double click on **Waiting Room**'s input node *on entered* add on process
- Step 16. Repeat Step 10. The true arm will destroy the entity using destroy step

Figure C-4: Refusal of Service if Waiting Room is full at **Registration Desk**

- Step 17. Change the property of *Use output* node in registration desk to *false*.
- Step 18. Connect the registration desk's transfer node to waiting room
- Step 19. **Waiting Room** object can hold 15 people so its *Initial Capacity* property should be set to 15. In addition to it, to model patient reneging and balking, set the *Allow Balk & Allow Renege* property to *true*.
- Step 20. For reneging, *Renege time* property is defined to *Random.Exponential(30)* minutes and *Queue Limit* is set to be 10 and default renege condition of *Is.Entity*
- Step 21. For the output node attached to the waiting room, *Initial Capacity* is 8 (which indicates that there are 8 patient care units attached to this node). Its *Ranking rule* is

Largest Value First with *Patient.Priority* state variable to give priority to severe patients.

- Step 22. Create a node list consisting of all the input nodes of the patient care objects and rename it as *PatientCareNode*
- Step 23. Configure the routing logic's *Entity Destination* property to *Select from list* and link the property *list name* to *PatientCareNode*.
- Step 24. Connect each patient care object to the waiting room output node.
- Step 25. Define *Selection Goal* to be *Smallest Value* with *Selection Expression* *Candidate.Node.InputLocation.Load* and *Selection Condition* to be *Candidate.PatientCare.ResourceState==0*
- Step 26. For each patient care object, define *output node* to be :
Output@ED_RO_WaitingRoom.CurrentCapacity in order to increment capacity of waiting room after a patient has exited
- Step 27. Drag and drop 3 ***Hospital Worker*** object and set them to a transfer node (which we shall call as *DoctorNode*) as their home node. Ensure that this node is connected to at least two other transfer nodes in opposing directions
- Step 28. Change the property *Initial Number in the System* to 3 for each of ***Hospital Worker*** object to model three doctors, nurses and residents.
- Step 29. Create a transporter list consisting of doctors and residents and name it as *MedicalWorkerList*

There are two ways to configure treatment logic. Step 30 presents using a built-in construct of *Treating Time* and *Treatment Resources* and Step 31 using *Custom Treatment Logic*. Reader may choose whichever step is more comfortable with their skill level. It is worthy to note that Step 31 provides more modeling flexibility.

- Step 30. For a few patient care objects define *treatment time* to be *Random.Pert (10,20,50) minutes* and use *treatment resources* property to seize from list:

MedicalWorkerlist. Set *Request Visit* property to *True* and link the node to individual *Patient Care* worker node.

Step 31. If you are configuring using custom logic: Set the *Custom Treatment Logic* property to be *True*.

Step 32. Define a state variable for each *Patient Care* object (*SetToTreat1*, *SetToTreat2*, etc.)

Step 33. Link the state variable to *Treatment Monitor State* variable in each of the *Patient Care* object. Define a process where you manipulate this state variable to indicate treatment state. For example: ED model uses a Boolean state variable *SetToTreat* and the monitor is *Discrete change mode*. The *Custom Treatment Logic* sets it to 1 before treatment delay but after doctor is seized

Figure C-5: Custom Treatment logic

Step 34. Inter connect each worker node of *Patient Care* object with each other to form worker pathways.

Step 35. Connect the output nodes of each patient care object to single **transfer node**.

Step 36. Connect this **transfer node** to input of *Billing Desk* object and 2 other **transfer node**, each one representing the outbound link to laboratory and radiology departments

Step 37. Select the outbound logic to be by link weight and give each link a specific weight like (Billing desk- 0.2, Radiology: 0.5, Lab: 0.3).

Step 38. Right click on the 2 other **transfer nodes** select *Bind to new external output node* in order for us to use this node in the Multi-Facility model constructed in Section C.V.

- Step 39. Configure the ***Billing Desk*** object to have an *Initial Capacity* of 3 to show that we have 3 cashiers at the billing stage of our model.
- Step 40. Bind this transfer node to an output node by right clicking on it and selecting *Bind to a new external output node* in order for us to use this node in the Multi-Facility model constructed in Section C.V.
- Step 41. Run the model. You will get warning statement describing auto transfer from one of the laboratory or radiology transfer nodes (Figure C-6), ignore it by checking the box *Don't show this warning again* and clicking on *Continue Running*.

This error indicates that the entity was tried to transfer from either of the transfer nodes (Laboratory, Radiology or Billing output) and failed to transfer as these nodes are not connected to any outbound link at this stage. We shall use these nodes in Section C.V while we build a multi-facility model.

Figure C-6: Sample Warning for failing transfer from a node

C.II Oncology Clinic

C.II.1 Model Description

This section uses the toolkit to develop a patient flow simulation model of oncology clinic to assess if addition of a floating nurse will reduce the patient waiting times at the clinic. A reduction of waiting times will be assist in healing process of the cancer patient since they have a weak immune system and any reduction will reduce their chances of being exposed to infection.

There are four types of patients: Consult Patients, Treatment Patients, Treatment & Checkup Patients and Follow up Patients. All patients arrive at the check-in desk which is

handled by single non-medical personnel. All types of patients have to check in at check in desk.

Following the check-in process, the consult and follow up patients are redirected to the clinic waiting rooms and the other two patient categories are redirected to the treatment waiting rooms. At the clinic waiting rooms, patients seek a nurse to escort them to an available to clinic room where the nurse collects patient history details since the last visit. Following which, a doctor visits the patient for the medical consultation and the process time is dependent on the type of visit. After this, the patient is redirected either to the billing station and exits the clinic or to radiology or laboratory department.

The treatment patients and treatment and checkup patients wait in treatment room until a Chemotherapist escorts them to one of the treatment rooms and initiates the chemotherapy therapy. Following which, the Chemotherapist returns to the nurse station if there are no more waiting patients or if all the treatment rooms are occupied.

When the treatment is done, the nurse is seized again to remove the central line and discharge the patient if they are treatment patients. For treatment and checkup patients, they are redirected to the clinic waiting room where they follow consult process.

C.II.2 Model Layout

Figure C-7: Oncology Clinic Process Layout

C.II.3 Model Construction

- Step 1. Drag and drop a **Registration Desk**, a **Billing Desk**, twelve **Patient Care Objects** (three for treatment group), two **Waiting Rooms** and eight **Transfer Nodes**.
- Step 2. Create a data table with *entity instance* property, *expression property* for consultation time, *real* property for patient fraction and *integer* property for patient type identifier as shown in Table C-2.

Table C-2: Data Table for Oncology Patients

Entity Type	Consult Time	Patient Mix	Patient Type
<i>Entity_Consumt</i>	<i>Random.Triangular(16.4,31.3,41.5)</i>	0.05	2
<i>Entity_Treatment</i>	0	0.6	3
<i>Entity_Checkup</i>	<i>3.24+Random.Weibull(3.83,1.89)</i>	0.1	4
<i>Entity_Followup</i>	<i>Random.Triangular(4.56,10.2,14.5)</i>	0.25	1

- Step 3. Drag and drop four **Patient Entity**, and rename them *Entity_Consumt*, *Entity_Treatment*, *Entity_Checkup* and *Entity_Followup*.

- Step 4. Link these entities to the data table's entity property by clicking on the property and selecting them from drop down.
- Step 5. Fill the rest of the table. Ensure that unique patient type identifier is numbered from 1 to 4, in the increasing order corresponding with entity order of Step 3.
- Step 6. Create a rate table to model the arrivals of patients similar to one in Table C-1.
- Step 7. Reference the entity type on source to the *Datatable.EntityProperty*.
- Step 8. Change the type of arrivals to *Time varying arrival rate* and use the rate table created in Step 6 to use the rate table.
- Step 9. On the output **transfer node** attached to the **source**, create an add-on process in the *entered* row. Split the arrivals equally between the two **transfer nodes** as shown in Figure C-8.
- Step 10. Connect the entrance **transfer nodes** to the **Registration Desk's** *input node*
- Step 11. Configure the **Registration Desk** to process time and output **transfer node's** routing *outbound link rule* to *By Link Weights*
- Step 12.** Connect the *output transfer node* to each clinic's **Waiting Room**

Figure C-8: Splitting entities between two **transfer node**

- Step 13. For the link that connects the **Registration Desk** and consult clinic **Waiting Room**, enter the link weight to be *PatientTable.UniqueID<=2* and for the other one *PatientTable.UniqueID>2*

- Step 14. Both the waiting rooms are used in default mode as we assume patient's don't balk or renege and the waiting room can accommodate the daily arrivals. This assumption is justified as in outpatient clinics, most arrivals are pre-planned.
- Step 15. Create two node lists; one containing all the input nodes of the consult clinics (*ClinicNodes*) and second one containing all the input nodes of the chemotherapy room (*TreatmentNodes*).
- Step 16. Connect each clinic (*Patient Care*) to the clinic's *Waiting Room*'s worker node and each treatment room (*Patient Care*) to treatment *Waiting Room*'s worker node
- Step 17. Connect the output node of the *Waiting Room* to the worker node of the respective *Waiting Room* using a **connector**. This connection is made in order to facilitate the transportation of patients to the clinics (or treatment) using *Hospital Worker*
- Step 18. Link the respective lists to their *Waiting Room*'s output node.

The next step is to create a selection construct that will always select the nearest empty clinic.

- Step 19. Define *Selection Goal* to be *Smallest Value* with *Selection Expression* *Candidate.Node.InputLocation.Load* and *Selection Condition* to be *Candidate.PatientCare.ResourceState==0*
- Step 20. Drag and drop two **transfer node** objects. Rename one as *ClinicNode* and *TreatmentTransferNode*.
- Step 21. Connect the *ClinicNode* to all the worker nodes of *Patient Care* deemed to be clinics (there should be nine of them)
- Step 22. Ensure the *Output Node* property of the *Patient Care* objects are set to the clinic *Waiting Room*'s output node.

- Step 23. Connect the *TreatmentTransferNode* to all the input nodes of treatment rooms (***Patient Care***). There should be three connections. Ensure all of these connections are bidirectional.
- Step 24. Ensure the *Output Node* property of the treatment ***Patient Care*** objects are set to the treatment waiting room's output node.
- Step 25. Drag and drop three ***Hospital Workers***, one for nurse, doctor and chemotherapist. Set the chemotherapist home node to a **transfer node** which is connected with the treatment stations (*ChemotherapistNode*).
- Step 26. Change the worker node's configurations of the ***Waiting Room*** (at Clinic and Treatment) to *Ride on Transporter* to *true* and assign it to nurse (***Hospital Worker***) in clinic network and chemotherapist (***Hospital Worker***) in the treatment network.
- Step 27. For the nurse, she performs an initial triage when she drops off the patient. To model this, add-on process in the unloading column. Add a delay to model the triage.

Figure C-9: Modeling Nurse Triage

- Step 28. For the consult part, a doctor needs to visit the patient which can be modeled using *Treatment Resources* construct of ***Patient Care*** where you can define the resources to be seized.
- Step 29. Connect the output nodes of the ***Patient Care*** objects to a **transfer node**
- Step 30. Configure this **transfer node** to route *by link weight* using the *Outbound link rule*
- Step 31. Connect this **transfer node** to two other **transfer node** each represent laboratory transfer point and second one represent radiology transfer point. The third outbound link is to ***Billing Desk***.

Step 32. Similarly, the chemotherapist needs to place an IV line when the patient reaches their treatment bed which can be modeled as described in Step 27

Figure C-10: IV Setup Process

Step 33. For the treatment part, we use the *Treatment Time* construct of **Patient Care** and use a random distribution to model treatment time (*Random.Triangular(6.88,13.9,41.8)* minutes).

We assume same treatment time for each patient since chemotherapy processes composed of only IV drip which are available only in standard sizes.

Step 34. Post Treatment, the chemotherapist is seized using *Secondary Resources Post Treating Seize* construct and request move to the appropriate input node of **Patient Care**.

Step 35. The delay to remove IV line and release the patient is modeled using an add-on process of *Treated* box in the **Patient Care** object.

Figure C-11: Post Treatment Process to remove IV line

Step 36. Connect the output node of the treatment **Patient Care** object to a single **transfer node** which routes using link weight where *PatientTable.UniqueID==4* are routed to clinic's waiting room. The rest are routed to the **Billing Desk** input node

- Step 37. Configure the **Billing Desk** processing time from a random distribution ($1.24 + \text{Random.Weibull}(3.83, 1.89)$ minutes)
- Step 38. Connect the output node to a **transfer node** and bind it an external node by right clicking on it and selecting *Bind to an external transfer node* in order for us to use this node in the Multi-Facility model constructed in Section C.V.
- Step 39. Run the model. If you get a transfer warning from any of the transfer nodes (Billing, Laboratory or Radiology), ignore it.

C.III Laboratory Services

C.III.1 Model Description

The lab services model consists of development of a model to represent a hospital laboratory department. Patients arrive either through external referrals or through internal referrals. Once they arrive, except for internal referrals, the patients have to check in at the registration desk and proceed to waiting room. The internal referral patients directly proceed to the waiting room. The waiting room feeds three care centers where patient's sample gets extracted. The specimen is collected and transported to the lab using a phlebotomist. After specimen collection, the patient may wait in the room, or in the waiting room or leave. If they wait, they will collect their results and have a follow up visit with the laboratory worker to get the results read and then they will leave the clinic after paying for the services. If they don't wait, their results are sent by courier to the patient's home.

The processing of the specimen requires materials to be used at each stage. In the setup stage it uses Setup Solution to prepare it for processing stage. The product from setup stage is fed in to processing stage using a laboratory worker. Centrifuge tubes are used in the processing stage and produced rough output which is fed in to printer. The printer prints the results and is dispatched as per the dispatching logic.

C.III.2 Model Layout

Figure C-12: Laboratory Patient Flow

C.III.3 Model Construction

- Step 1. Drag and drop source, **Patient Entity**, **Registration Desk**, **Waiting Room**, **three PC_Labcare**, **one hospital workstation**, sink and **Billing Desk**. Place them as shown in Figure C-12. Leave the decide modules (diamond shapes) as they will be embedded as processes.
- Step 2. Change the entity type in the source to **Patient Entity** and arrival type to *Time varying arrival rate* and specify a *rate table* under data table tab (refer to Table C-1).
- Step 3. Connect the source to **Registration Desk** and specify *registration time* (*Random.Uniform(5,8)* minutes) and set *output node* property to *true*.
- Step 4. Set the *initial capacity* of **Registration Desk**'s output node to *infinity*
- Step 5. In the **Waiting Room**, *Enable Batching* property must set for results to be batched on *first come first served basis* ranking order.

Patients who are waiting for results have a higher priority than patient who is waiting to get tested, hence the former type of patients are allocated space first

Step 6. For this go to the input node of the **Waiting Room** and change the *Ranking Rule* property to *Largest Value First* and *Ranking expression* to be *PatientEntity.WaitForResults*

Step 7. Connect the output node of the **Waiting Room** to **transfer node** using connector. This is again done in order to use an transporter object to carry the entities from the waiting room.

Step 8. Drag and drop three **Hospital Worker** objects to represent Receptionist, Phlebotomist and Laboratory worker.

Step 9. Initialize the receptionist and phlebotomist home node to a **transfer node** near **Waiting Room**.

Step 10. Ensure this receptionist **transfer node** is connected to all the **PC_Labcare** objects

Step 11. Connect the phlebotomist **transfer node** to the specimen node of the **PC_labcare** objects and the specimen node to input node of the **Hospital Workstation**.

Step 12. Configure the specimen node to *Ride on transporter* property to true and link the *transporter name* to Phlebotomist object name.

The next few steps detail the decision process used by **Patient Care** to decide how to handle the **Patient Entity** using the *PatientEntity.StateName* expression.

Step 13. For each **Patient Care** object, we create a *treating* add-on process to check if the patient's initial visit or results reading using the *PatientEntity.ResultsReady*.

Step 14. If the expression is false, then we initiate the delay for initial consult (*Random.Uniform(5,20)* minutes).

Step 15. The next step is to decide if the patient needs laboratory. We use the DECIDE step in *Probability Based* mode where 80% of the patients need lab.

Step 16. If lab is required, we assign (*PatientEntity.NeedLab==True*).

Step 17. Again, we use DECIDE step in probability mode if patient waits for results (20% do wait for results) and indicate the patient's choice using *PatientEntity.WaitForResults==True*

Figure C-13: Patient States Assignments for eliciting patient choices

Step 18. If yes, we decide if the patient has to wait in clinic which can be indicated using *PatientEntity.StayInClinic==True*, or else patient wait's in waiting room. For this we assumed only 4% of the patients waiting for results were actually waiting for their results in their allotted clinics.

Step 19. Finally, a random number of specimen is created by assigning *Patient.TestQuantity*. This is done to model multiple patient

Step 20. If the DECIDE step (*Initial visit*) is false, the DELAY step models the results reading process.

After deciding the *Patient Entity* behavior in the *PC_Labcare* object, we need to configure the *PC_Labcare* object to react to these situations. In this case, for specimen collection and result reading.

Step 21. Add *Lab Resoruces* property for each *PC_Labcare* unit to seize Phlebotomist (*Hospital Worker*) for specimen extraction

Step 22. Add a link from output node of patient care to input of *Waiting Station* and *Billing Desk*.

Step 23. The link weight for the above links should be (*Patient.WaitForResults==1 && Patient.ResultsReady==0*) and (*Patient.WaitForResults==0 || Patient.ResultsReady==1*) respectively in order to route **Patient Entity** as per their choices. The second expression latter half is used to route patient's whose paper work is completed as explained in Step 34241

Step 24. The **Hospital Workstation** needs to have *Time Duration* for setup (*Random.Uniform(2,5) minutes*), processing (*Random.Uniform(5,7,9,1) minutes*) and post-processing stages (*Random.Uniform(4,12) minutes*) for performing all the functions of a centrifuge laboratory.

It is quite common for specimens to be batched for processing or setup stage. This lab machines process specimens in batches in setup stage and processing stage.

Step 25. In order to model batching, set *Enable Specimen Batching* property true. In this case, we batch before setup hence *Before Setup* property is also set to true and *Specimen Matching Rule* to *Any Entity*.

Step 26. The laboratory worker is seized to transfer the specimens between machines. So to seize the laboratory worker in *Finished setting up* and *Processed* Add-on processes. Release the laboratory worker in *Processing* and *Specimen Unbatching* add-on processes.

One of the unique features about the laboratory setting is their use of materials for processing. Laboratories cannot run out of these materials, at the same time excess stock might become unusable. Hence, modeling materials is another important problem formulation requirements.

Step 27. You can define *MATERIALS* under elements tab and assign *initial quantity* property to 1000 for each of them (*CentrifugeTubes* for setup, *CentrifugeSolution* tubes for processing and *Paperwork* for Post-processing).

- Step 28. Use the *MATERIALS* properties in **HWS_LabCare** object to assign the material for consuming and production in setup, processing and post-production stages
- Step 29. Configure the output node of the **Hospital Workstation** to route entities by link weight under *outbound link rule*.
- Step 30. Connect the output node of the **Hospital Workstation** to the **Waiting Room**, a **transfer node** and a **sink** that represents results courier process.
- Step 31. The *selection weight* for each of the links from the above transfer node of results as defined in Table C-3.

Table C-3: Processed Specimen routing out expressions

Output Node Link	Expression
To Waiting Room	<i>Patient.WaitForResults==True && Patient.StayInClinic==False</i>
To Respective Clinics	<i>Patient.WaitForResults==True && Patient.StayInClinic==True</i>
To Results Sink	<i>Patient.WaitForResults==False</i>

If the processed specimen (which represent results) are to be routed to their respective clinics, it requires a an additional layer of routing logic.

- Step 32. Connect the 2nd link from output results node to another **transfer node** which is connected to each of the **PC_Labcare** object's specimen's node.
- Step 33. The link weight should be (*SpecimenEntity.LastClinic==PC_Labcare1*), (*SpecimenEntity.LastClinic==PC_Labcare1_1*) & (*SpecimenEntity.LastClinic==PC_Labcare1_2*) for PC_Labcare1, PC_Labcare2 and PC_Labcare3 respectively.

Step 34. Double click *Lab Results Batched* add-on process in each **PC_Labcare** object and insert a delay to model paperwork handling time.

After paperwork is received by the patients, they are sent to the billing desk for completing payment.

Step 35. Bind the **Billing Desk** output node to external visible output node by right clicking on it in order for us to use this node in the Multi-Facility model constructed in Section C.V.

Step 36. Connect the output nodes of the **PC_Labcare** objects to **Billing Desk** input node

Step 37. Configure the **Billing Desk**'s *Billing property* to some random distribution to represent processing time. In this case we used *Random.Exponential(5) minutes*.

Step 38. Drag and drop a transfer node and bind to external visible node as well as connect it to waiting room input to facilitate inpatient referrals.

Step 39. Run the model! Ignore any warnings from transfer out error originating from laboratory or radiology transfer node as explained in ED model's Step 41.

C.IV Radiology Services

C.IV.1 Model Description

The radiology services consist of three types of services: X-ray, Gamma Camera, and MRI. Only in-patient referrals are allowed to use the service. All arriving patients are made to wait in a waiting room which operates on first come first served basis. Upon their arrival, each patient is assigned an imaging service based on a probability. Each patient is accompanied to the imaging facility by a receptionist.

On their arrival to the facility, the patients are prepped for the procedure. This is the case where patient might have to change in to a specific gown or remove certain jewelry, informing the patient about the procedure, etc. can be modeled as setup stage of the process. After the setup stage, their body part is scanned in the respective facility. This scanning procedure can be modeled in the processing stage.

After scanning, a patient can decide between waiting for to collecting their results or getting it mailed home at HWS Xray and HWS GammaCamera . At HWS MRI the patient's are instantly handed over the result. These choices must be accurately captured to understand the dynamics of the system.

Finally, the X-ray machine is prone to failure due to its old condition. It is known from historic data it fails between 1 and 20 patients and it takes an exponential 40 minutes to repair the machine.

C.IV.2 Model Layout

Figure C-14: Radiology Department Model Layout

C.IV.3 Model Construction

- Step 1. Drag and drop HWS LC Xray, HWS LC MRI, HWS LC GammaCamera, two **Waiting Room** and a **sink**

The next logical steps describe the process of assigning a radiology service to the patient along with structural connections between waiting rooms and radiology services to permit flow of entities.

Step 2. Create an add-on process in **Waiting Room** object *On Entered Input Station*

Step 3. Assign *PatientEntity.NeedRadiology=1* using an ASSIGN step.

Step 4. Use DECIDE step (*probability Based*), assign *PatientEntity.NextDestination* to either one of the 3 **HWS_Labcare** objects. We assumed 50% of patients required to visit X-ray, 35% were in need of MRI service and the rest visited Gamma Imaging center.

Step 5. Connect the output node and **Transfer node** of the **Waiting Room** to facilitate patients to be accompanied by receptionists between waiting room and radiology service.

Step 6. Drag and drop a **Hospital Worker** to model the receptionist. Set the home node near the waiting room (*ReceptionistNode*).

Step 7. Configure the **transfer node** to *Ride on Transporter* which is the receptionist.

Figure C-15: Deciding random patient entity assignments for radiology service

Step 8. Connect the **transfer node** to a **basic node** which is connected to

HWS LC Xray, HWS LC GammaCamera and HWS LC MRI.

After the entity is dropped off at the basic node, it is assumed that logically it is at the entrance of either of the services. The decision of routing is based on what type of

service the patient requires. Thus, we use *PatientEntity.NextDestination* state to assist in decision making at this stage.

Step 9. Configure the **basic node** to *Outbound link rule* to be on link weight where the link weight is the expression *PatientEntity.NextDestination==HWS_LC_Xray*, *PatientEntity.NextDestination==HWS_LC_GammaCamera*, *PatientEntity.NextDestination==HWS_LC_MRI* respectively.

Step 10. Configure the time duration for each phase of the **HWS_Labcare** object as shown in the Table C-4.

Step 11. Drag and drop three **Hospital Worker** objects (*HW_LabTech*) each assigned to post production node of the **HWS_Labcare** object as their *Home Node*

Step 12. For each HWS_Labcare, create *setting up & post production* add-on processes to seize lab tech

Table C-4: Phase Duration Table for **HWS_Labcare** class in Radiology Department

Object	Setup Time (minutes)	Processing Time (minutes)	Post Processing Time (minutes)
<u>HWS_LC_Xray</u>	<i>Random.Exponential(5)</i>	<i>Random.Exponential(10)</i>	<i>Random. Exponential(15)</i>
<u>HWS_LC_Gamma Camera</u>	<i>Random.Exponential(5)</i>	<i>Random.Uniform(10,15)</i>	<i>Random. Exponential(5)</i>
<u>HWS_LC_MRI</u>	<i>Random.Pert(5,10,15)</i>	<i>Random.Pert(10,20,30)</i>	<i>Random. Pert(5,10,15)</i>

Step 13. For each **HWS_Labcare**, create *processing* add-on process to move the lab tech to processing node using the MOVE step. Since the worker is already seized by the entity in the *setting up* add-on process, move step allows to request visit to another

node . In this case we can say that the hospital worker takes the finished product from setting up stage and carries it to processing stage of the machine.

- Step 14. Only for X ray and Gamma Camera objects, create *processed* add-on release the lab tech because we assume that post-production work is where the hospital worker's resource state is *busy* and not *treating*.
- Step 15. Following the above release, use a DECIDE step (*probability based*) to assign *PatientEntity.WaitForResults* to decide if the patient should wait for results.

Figure C-16: Patient behavior post service completion

- Step 16. For MRI object, just release the lab tech in *processed* add-on
- Step 17. For each *HWS_Lab Care*, Create *post production processed* add-on to release the lab tech.
- Step 18. Connect X ray and Gamma Care output node to the second waiting room and a transfer node (radiology output node). Configure the output node route on *By link weight* which uses the expression in Step 19.
- Step 19. The link weight to waiting room link is *PatientEntity.WaitForResults==1* and for the output node for *PatientEntity.WaitForResults==0*
- Step 20. The use of the second waiting room is to batch results with patients who are waiting for them. For batching results, the second waiting room's *Enable Batching Logic* should be set to true and its output node (*Initial Capacity=Infinity*) should be connected to radiology output node. We set the *Initial Capacity* of its output node to infinity as it is not routing to any specific patient care center or services which is capacitated.

- Step 21. For MRI, *Batch at Station* property should be true so that results can be batched at the MRI station.
- Step 22. For MRI, the output node is directly connected to radiology output node.
For the other two radiology services, we need to handle the routing of the specimen entities depending on the patient choices.
- Step 23. Only for X ray and Gamma Camera, the specimen node should be connected to courier sink and specimen node of the waiting room
- Step 24. For each link, the link weight should be *SpecimenNode.WaitForResults==0* and *SpecimenNode.WaitForResults==1* respectively.
- Step 25. Bind first waiting room's input node and radiology output node to external node so that they can be used in the CIMS model developed in the forthcoming section.

C.V Multi-Facility Model

C.V.1 Model Description

This is an integrated model consisting of emergency department, oncology clinic, radiology department and laboratory department. Each department is interconnected to another. Patients from Emergency department are transferred to the laboratory using stretcher. The stretcher is operated by an attendant 24 hours a day.

In order for this model to be a decision making tool, the model must be interactive and display useful statistics. Finally, the model must allow the management to experiment with staffing levels at each department.

C.V.2 Model Layout

Refer to Figure 4-1: Hypothetical Hospital Model for the model layout.

C.V.3 Model Construction

- Step 1. Click on each of the facility¹⁶ model's **Hospital Worker** object and right click on the *initial number in system* and select *set property reference*. This step allows us to convert the staffing levels in to variables which can be manipulated in the CIMS model.
- Step 2. Drag and drop each facility on to new model. This forms CIMS model structure.
- Step 3. Connect the laboratory and radiology **transfer node** of Emergency and Oncology Clinic to laboratory and radiology department's input node respectively. This routes all the patients who need either of the service to the respective departments.
- Step 4. Drag and drop HT Stretcher and a **Hospital Worker** with Emergency Department Laboratory **transfer node** as home node. Assign the **Hospital Worker** to the stretcher under assistance logic.
- Step 5. Ensure that laboratory transfer node of emergency department is configured to *ride on transporter* and the *transporter name* is HT Stretcher.
- Step 6. Drag and drop a **Statistics Collector** object and set *Collect Patient Statistics* to true
- Step 7. Connect each facility's output node to the **Statistics Collector's** input node.

The following steps detail on development of an interactive statistics display module for the CIMS model.

- Step 8. Click on *dashboard* tab

First, we are interested in collecting the overall utility of the secondary services.

---

¹⁶ Emergency Department, Oncology Department, Laboratory department and Radiology department

Step 9. Drag and drop a **status pie** and use it in *list states* mode for each of the workstations: Laboratory Services, Xray, Gamma Camera and MRI using the expression: *LabServices1.HospitalWorkstation1.ResourceState*,

Step 10. *RadiologyServices1.RS_HWS_Xray.ResourceState*,

Step 11. *RadiologyServices1.RS_HWS_GammaCamera.ResourceState*

Step 12. *RadiologyServices1.RS_HWS_MRI.ResourceState* respectively

For the emergency department, we are interested in overall utilization of the department against each bed useful. In other words, we are looking for aggregate resource usage statistics for emergency department.

Step 13. Drag and drop **status pie** and use it in *repeat groups* mode. Each repeat group is assigned to each state which is the sum of all 8 emergency patient care objects in same state. For example: *Idle State*=

$$\begin{aligned} & EmergencyDepartment1.ED\_PatientCare1.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PatientCare2.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PatientCare3.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PatientCare4.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PC\_MaternityBed1.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PC\_TreatmentBed1.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PC\_TreatmentChair1.ResourceState.PercentTime(0) + \\ & EmergencyDepartment1.ED\_PC\_WardBed1.ResourceState.PercentTime(0) \end{aligned}$$

Step 14. Repeat the above expression at each resource state level.

Although, the above statistic tells us the overall utilization, we would also like to know how many emergency beds can be served in parallel (undergoing some form of medical procedure).

Step 15. For this use a **Linear Gauge** and assign to *CurrentlyTreating* function for each bed Emergency Department to get an idea on how many patients are treated at once. For example:

Step 16. *EmergencyDepartment1.ED_PatientCare1.CurrentlyTreating +*
EmergencyDepartment1.ED_PatientCare2.CurrentlyTreating +
EmergencyDepartment1.ED_PatientCare3.CurrentlyTreating +
EmergencyDepartment1.ED_PatientCare4.CurrentlyTreating +
EmergencyDepartment1.ED_PC_MaternityBed1.CurrentlyTreating +
EmergencyDepartment1.ED_PC_TreatmentBed1.CurrentlyTreating +
EmergencyDepartment1.ED_PC_TreatmentChair1.CurrentlyTreating +
EmergencyDepartment1.ED_PC_WardBed1.CurrentlyTreating

Step 17. Hospitals are worried about overcrowding. Even if one department is operating optimally, it may be the case that there might be blockage in some other part of the system. To visualize this, we use **status plots**.

Step 18. Plot the number of people in the hospital using **Status Plot** and the expression being:

Step 19. *EmergencyDepartment1.PatientEntity1.Population.NumberInSystem +*
RadiologyServices1.PatientEntity1.Population.NumberInSystem +
LabServices1.PatientEntity1.Population.NumberInSystem +
OncologyClinic1.Entity_Checkup.Population.NumberInSystem +
OncologyClinic1.Entity_Consult.Population.NumberInSystem +
OncologyClinic1.Entity_Followup.Population.NumberInSystem +
OncologyClinic1.Entity_Treatment.Population.NumberInSystem

Step 20. Display a status label to count the demand lost due to blocked Emergency department using the expression:

- Step 21.   *EmergencyDepartment1.ED_RO_WaitingRoom.BalkCount +
EmergencyDepartment1.ED_RO_WaitingRoom.RenegeCount +
EmergencyDepartment1.RefusalofServiceState*
- Step 22.   Although the above statistics are useful, none of them reveal any information directly on patient times at each resource state (treating, idle, waiting, etc).
- Step 23.   To reveal this information directly, use a **status pie** to display average times for patients in each state. Ensure the **status plot** is used in *repeat groups* mode and expression for each state can be given using **Statistics Collector** *percentage time* state. For example, Administration time is collected from
SC_PatientExit.PT_Administration.Average
- Step 24.   At this point in time we are ready to run the model. So click on Run! Users can see both facility view and dashboard view simultaneously if the right click on the tab and *vertical tab group* or *horizontal Tab group*.
- The next step is creation of experiment to experiment with the model in order for the decision makers to interact with the simulation model. For the hypothetical hospital model presented in this thesis, we are looking to optimize hospital staffing levels with respect to average patient throughput time.
- Step 25.   Right click on each facility's instance in the full model and make the **Hospital Worker** number in to a property of CIMS model.
- Step 26.   Insert a new **experiment** and add a response
SC_PatientExit.TimeInSystem.Average
- Step 27.   Set the objective of this response to minimize patient's time in system. Let all other controls be in their default setup.
- Step 28.   Run the experiment!

The next step is to implement optimization experiment to find the optimum staffing levels that bring the patient's time in system around 5 hours

- Step 29. Select OptQuest Add-in from the experiment ribbon
- Step 30. Set *Primary Response* of *Opt Quest* to Patient's Time in System
- Step 31. Set the upper bound of the Patient's time in system response to 5 hours
- Step 32. Run the procedure

This step will generate a number of scenarios that meet our requirements. However, we need to select the best scenario for which we need to implement ranking and selection method. SIMIO has an inbuilt K-N Ranking and Selection Method. We shall use this to identify our scenario

- Step 33. Select K-N Ranking and Selection Add-in from the experiment ribbon
- Step 34. Set the indifference zone to 0.5 hours. In other words, if the time between scenarios is half an hour then they are equally good.
- Step 35. Run the experiment. This step will yield one scenario as the optimal one.

ⁱ Not Implemented in the current version