University of Windsor

Scholarship at UWindsor

Electronic Theses and Dissertations | Theses, Dissertations, and Major Papers

2009

Analysis of scheduling in a diagnostic imaging department: A simulation study

Brendan Eagen
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Eagen, Brendan, "Analysis of scheduling in a diagnostic imaging department: A simulation study" (2009). *Electronic Theses and Dissertations*. 7976.
https://scholar.uwindsor.ca/etd/7976

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

ANALYSIS OF SCHEDULING IN A DIAGNOSTIC IMAGING DEPARTMENT: A SIMULATION STUDY

By

Brendan Eagen

A Thesis
Submitted to the Faculty of Graduate Studies
through Industrial and Manufacturing Systems Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada
2009

© 2009 Brendan Eagen

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-57622-9
Our file Notre référence
ISBN: 978-0-494-57622-9

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

In this thesis we present an Agent-Based Modelling Tool (ABMT) for use in the investigation of the impact that operational level changes have on diagnostic imaging scheduling and patient wait times. This tool represents a novel application of agent-based modelling in the outpatient scheduling / simulation fields. The ABMT is a decision support tool with a user friendly graphical user interface that is capable of modelling a wide array of outpatient scheduling scenarios. The tool was verified and validated using data and expertise from Hotel Dieu Grace Hospital, Windsor, Ontario, Canada. The ABMT represents a technological advancement in the modelling of multi-server, multi-priority class customer queueing systems with deterministic service times and uneven distribution of server up-time.

Dedication

To my parents,

Acknowledgements

Dr. Richard Caron

Dr. Walid Abdul-Kader

Kathy Hillman

Gail Peterson

Mary-Alice Beneteau

Neil McEvoy

Dayna Roberts

Dave McKenzie

Jacquie Mummery

Brenda Schreiber

Table of Contents

List of Tables

List of Figures

List of Acronyms

ABMT –	Agent-Based Modelling Tool
CAT -	Computed Axial Tomography
CT –	Computed Tomography
FCFS –	First Come First Serve
GUI –	Graphical User Interface
HDGH –	Hotel Dieu Grace Hospital
LHIN -	Local Health Integration Network
MRI -	Magnetic Resonance Imaging
SIRO –	Service In Random Order

1. Introduction

Canada's publicly funded healthcare system is dynamic. The system, composed of 10 provincial and 3 territorial plans, has evolved into its current state over the past forty years. The goal of the system however remains unchanged; providing universal coverage for medically necessary healthcare services on the basis of need rather than the ability to pay. In recent years stress on the system has been increasing due to factors such as the high cost of new medical technology and the aging of the baby boom generation (Ministry of Health, 2005). In years to come this stress will only continue to increase as the number of senior citizens in Canada continues to climb. The percentage of the total population that were senior citizens in 2005 was 13%, however by 2036 that number is expected to nearly double to 24.5% (Turcotte & Schellenberg, 2006). Combined with the fact that seniors historically have consumed 44% (Canadian Institute for Health Information, 2008) of the healthcare spending of provinces and territories it's plain to see that Canadian healthcare system is headed into a period that will tax its resources to a new level.

One area where resources are already spread thinly is diagnostic imaging. This area is concerned with the use of MRI (Magnetic Resonance Imaging), CT or CAT Scans (Computed Axial Tomography), Ultrasounds and X-Rays. In 2004 there were on average 4.9 MRI machines and 10.2 CT Scanners for every million Canadians; by 2007 those numbers had risen to 6.8 and 12.8, respectively (Canadian Institute for Health Information, 2004) (Canadian Institute for Health Services, 2007). However, between 2006 and 2007 the demand for MRI and CT scans increased by 42.9% and 27.9%, respectively. Compounding this issue is the fact that diagnostic imaging resources are not evenly distributed across the country, for example by the end of 2006 there were 10.2 CT scanners per million people in Ontario but 21.6 per million people in Newfoundland and Labrador (Canadian Institute for Health Information, 2007). As a result of the increasing demand for and uneven distribution of diagnostic imaging equipment wait times for diagnostic imaging scans have become a concern in Canada. To that end the government of Ontario has begun an initiative to track wait times in areas throughout the province (http://www.health.gov.on.ca). The Ministry of Health has also established target wait times for patients of different acuity (sickness) levels which are used to assess healthcare providers' wait time performance.

At present, in many areas of Canada, the demand for diagnostic imaging services outstrips the ability of public healthcare to provide these services. As a result, requests for services can go unmet, except in emergency cases, until weeks after the request has been made resulting in lengthy queues.

The 'Wait Time' targets established by the government of Ontario represent the maximum period a particular class of patients can wait for service before their health will suffer. At present there are 4 categories of patient acuity; class 1 patients are the sickest and require immediate attention (within 1 day) whereas class 4 patients are less critical and can often be elective requiring attention within 28 days.

The focus of this study will be 'Wait Time' as described by provincial government of Ontario's guidelines. However, from a queueing theory perspective this is not the actual wait time, but can be more accurately described as access time. The key difference being that this thesis will examine the days between the request for service and the day of service and *will not consider* the time that a patient may wait for service on the day that he or she is scheduled as a result of interruptions in the pre-established schedule. Essentially, the thesis will ignore the fact that a patient may have to wait as long as the waiting occurs on the day that the patient is scheduled to be scanned. It should be noted that in many works the terms access time and wait time are used interchangeably, we will assume them to both mean the number of whole days a patient waits between requesting and undergoing service.

Based on these factors it is plain to see that diagnostic imaging service providers will need a means to effectively manage resources, allocate funds and control their processes if they are to cope with the increasing demand of the Canadian population for their services to be delivered in a timely manner.

1.1 Problem Description

It was the recognition of the reality facing a diagnostic imaging department that lead to the conception of this thesis. The research team, which consisted of Dr. Richard Caron, Dr. Walid Abdul-Kader and Mr. Brendan Eagen, was invited by Mr. Neil McEvoy, former CEO of Hotel Dieu Grace Hospital (HDGH), Windsor, Ontario, Canada, to study his hospital's diagnostic imaging department and its scheduling system. As a trained industrial engineer Mr. McEvoy was keenly aware of the benefits of simulation and requested that the team pursue an agent-based solution to the problem. His vision was that a tool be created for him and his staff that would assist them in the evaluation of the effects of operational level changes to their current system. Mr. McEvoy suggested the use of NetLogo™, a zero cost software used by researchers interested in agent-based modelling. These directives motivated this thesis.

1.2 Thesis Statement

Our thesis is that agent-based modelling can provide a technological tool for use by hospital decision makers to evaluate the effects that operational level changes will have on their diagnostic imaging system with specific interest in the impact the changes will have on patient scheduling and wait time.

1.3 Objectives

With the above thesis statement in mind the objectives of this thesis are to create an agent-based simulation tool with an easy to understand graphical user interface (GUI) that would allow hospital decision makers to assess the impact of potential operational level changes to the diagnostic imaging department on the department's schedule of patients. Additionally this thesis will expand knowledge of the use of agent-based modelling in the outpatient scheduling field. The tool is a decision support tool, not a model of any one specific diagnostic imaging department, and allows users to modify input parameters according to the scheduling system that they wish to model.

1.4 Research Methodology

The research methodology is implicit in the following overview of the thesis layout. Though enumerated, many of the outlined activities were carried out in parallel.

1. **Preliminary Research**
    a. **Review of literature** – An exhaustive literature review was performed and the results provided insight into the proposed research's place in the fields of healthcare simulation/scheduling and agent-based modelling. In the case of HDGH scheduling is taken to mean the assignment of a patient to a specific appointment slot on a particular CT scanner. Additionally, the literature review helped to establish the parameters that would be used in the construction of the simulation model.
    b. **Consultation with healthcare professionals** – Consultation with practicing medical professionals and healthcare administrators helped to establish the user interface requirements of the model as well as providing insight into what internal and external factors affect the scheduling process.

2. **Design and development** –The model was developed in the NetLogo™ simulation package using the parameters established in the Preliminary Research phase. In order to use the NetLogo™ simulation package it was necessary to learn the unique programming language that is used to

control it. Learning this programming language required several months of study and resulted in the development of a prototype simulation model designed to simulate scheduling of a single server. Once the programming language had been mastered, the prototype model was expanded to accommodate multi-server scenarios.

3. **Verification and Validation** – The model was presented to medical professionals and healthcare administrators to verify that the diagnostic imaging scheduling process was accurately represented in the model. Historical data was collected from the sponsoring hospital, HDGH, and used to validate the simulation results.

4. **Discussion** – The overall effectiveness of the simulation tool was assessed and observations were made and documented regarding the applicability of agent-based simulation to scheduling in healthcare.

5. **Dissemination** – The simulation tool will be shared with Canadian medical professionals, healthcare administrators and healthcare researchers.

2. Review of Literature

This review of literature assists in the determination of what methodology should be used to approach the topic of this thesis and also to determine the thesis' place in published literature. The researchers will first consider the macro level problem of what solution method they wish to use. The end result of this thesis will be a decision support tool that is transportable between diagnostic imaging scheduling systems, as this decision support system will be required to support a system that is relatively complex and also relies heavily on historical data thus a simulation-based decision support system appears appropriate. Examples of simulation successfully being applied in healthcare include the work of (McClean & Millard, 1995), (Everett, 2002) and (Aktas, Ulengin, & Sahin, 2007) who have all effectively applied simulation-based decision support in healthcare. Everett gives perhaps the best justification for choosing simulation as a decision support tool. He states that the complex web of stakeholder objectives in healthcare all but precludes the existence of an "optimal" solution to a problem. Instead he suggests that it is the system modeller's job to enable informed debate among stakeholders. To that end, he continues, the development of a simulation model for decision support is an excellent means by which to encourage communication between stakeholders and the modeller so as to accurately capture the true nature of the system. Simulation also, through use of a graphical user interface, allows the stakeholders without technical backgrounds to contribute to the development and assume ownership and commitment to the model. It should be noted that simulation was not the only option considered for modelling the diagnostic imaging scheduling system. Queueing theory / analytical options were initially considered but the complex nature of the system combined with the need for flexibility across a wide array of scenarios lead us to disregard these approaches.

2.1 Queueing Theory

Based on preliminary consultations with HDGH diagnostic imaging staff we determined that the system can most readily be compared to a queueing system in which multiple servers work in parallel to serve a single queue of customers with weighted priority on a first come first serve basis and that have deterministic service times. Figure 1 depicts a single-stage queueing system with multiple servers in parallel serving a single queue in much the same way as CT scanners service patients at HDGH.

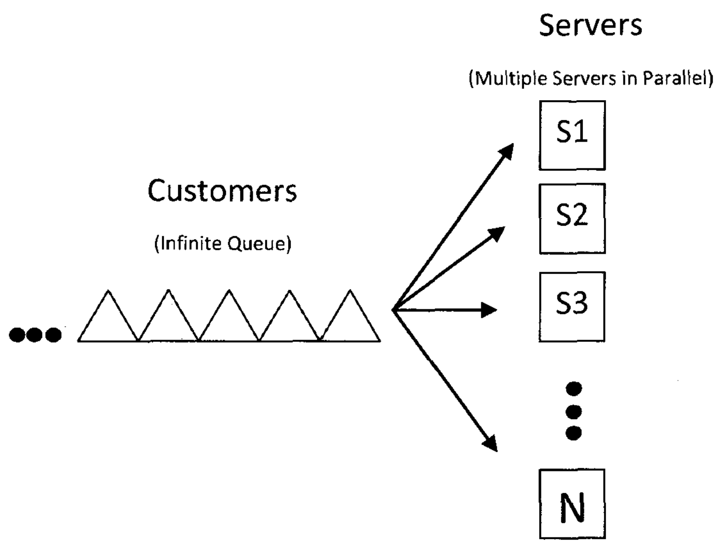


Figure 1 - Single Stage, Multi-Server Queueing System

Although discounted as a solution to this particular problem, queueing theory still provides a useful means by which to describe the situation under consideration.

In order to understand queueing theory notation and its ability to describe the current problem one must be familiar with the basic components of a queue and the way in which it functions. A brief overview of queueing as well as the importance of the exponential distribution is provided by (Winston, 2004). A queue is essentially a waiting line in which customers wait to receive service from a server. Queueing theory helps one to describe and understand the relationships between customers, queues and servers.

Customers, be they people, automobiles, manufacturing equipment, etc. require 'service' of some sort. For example, people are serviced at a bank or in a grocery store, cars are serviced by a mechanic and a broken welding robot is serviced by a technician. In these cases the number of customers often exceeds the number of servers, that is, the number of people requiring banking services exceeds the number of bank tellers for example. In situations such as there queues form. The order in which customers in a queue are serviced by the servers is known as the queue discipline. The most common queue discipline is First Come First Serve but others exist such as Last Come First Serve and Service In Random Order.

Understanding how customers come to be in the queue is another important aspect of queueing theory. Customers 'arrive' in the system; this is known as the arrival process and is the input for the system. The rate at which customers arrive is known as the 'arrival rate' and in general can be modelled by a mathematical distribution, the most common of which is the exponential distribution.

Exponential distributions are used to model interarrival times because of their no-memory property. That is,

$$P(\boldsymbol{A} > t + h \,|\boldsymbol{A} \geq t) = \frac{P(\boldsymbol{A} > t + h \cap \boldsymbol{A} \geq t)}{P(\boldsymbol{A} \geq t)} = \frac{e^{-\lambda(t+h)}}{e^{-\lambda t}} = e^{-\lambda h} = P(\boldsymbol{A} > h)$$

"The no-memory property of the exponential distribution is important, because it implies that if we want to know the probability distribution of the time until the next arrival, then it does not matter how long it has been since the last arrival." (Winston, 2004)

Other factors also affect the arrival process such as whether or not more than one customer can arrive in the system at a time and also the total number of customers that the system services.

Modelling the time required for a customer to receive service is also a key element of queueing theory. The Erlang distribution is commonly used to model services times, however, other distributions are also common. In some cases, when the same actions are repeated for every customer, the service time will always be the same. In these situations the service time is said to be deterministic.

In order to summarize all of the information required to describe a queue Kendal developed a standard notation (Kendall, 1951). Known as the Kendall notation, this method describes queues based on 6 characteristics.

1) The arrival process
2) Service times
3) # of parallel servers
4) Queue discipline
5) Max. # of customers in the system
6) Size of the population

Standard abbreviations were assigned to each characteristic, for example, M denotes an exponential distribution and D denotes a deterministic process.

Thus,

$$M / D / 2 / FCFS / \infty / \infty$$

denotes a queueing system whose customers arrive based on an exponential distribution of interarrival times which are served by two servers at a deterministic rate in a first come first serve manner. The customers come from an infinite supply and are unlimited in the number that can occupy the system.

While a model of the imaging department at HDGH as an $M/D/2/FCFS/\infty/\infty$ queue might provide insight, it would fail to capture complexities such as multiple patient classes that cause a violation of the FCFS queue discipline; and scanner downtime so that the servers are not continuously available. This reasoning leads us to the conclusion that simulation would be a better modelling technique.

2.2 Simulation

Simulation is the imitation of the operation of a real-world process or system over time (Banks et al, 2005). Simulation can provide a means by which to forecast the future of the diagnostic imaging schedule based on those past known events. The benefits of simulation are many fold as presented by (Shannon, 1992):

- Simulation can be used to explore new policies, operating procedures, decision rules, organizational structures, information flows, etc. without disrupting the ongoing operations.
- New hardware designs, physical layouts, software programs, transportation systems, etc. can be tested before committing resources to their implementation.
- Hypothesis about how or why certain phenomena occur can be tested for feasibility.
- Simulation allows us to control time. - Time can be easily compressed, expanded etc. allowing us to quickly look at long time horizons or to slow down a phenomenon for study.
- Simulation can allow us to gain insight into which variables are most important to performance and how these variables interact.
- Simulation allows us to identify bottlenecks in material, information and product flows.
- The knowledge gained about a system while designing a simulation study may prove to be invaluable to understanding how the system really operates as opposed to how everyone thinks it operates.
- Through simulation we can experiment with new situations about which we have limited knowledge and experience so as to prepare for what may happen. Simulation's great strength is its ability to let us explore "what if" questions.

Shannon's first point holds significant weight in the case of this thesis. It is not feasible or safe to interrupt the current diagnostic imaging scheduling process as doing so may adversely affect the health of the patients relying on the system. Many forms of simulation also have the added benefit of providing the modeller with a visual representation of the system which can be useful when presenting the model to those whose knowledge of the system or simulation is lacking (Banks et al, 2005).

2.2.1 Simulation Paradigm

The simulation field is composed of many different approaches or paradigms. A system can be modelled in many different ways ranging from simulations performed by hand to complex multi-scenario simulations that require more computing power than the average desktop PC has to offer. For ease of calculation and timeliness this study focused on computer simulation. For the purpose of this investigation we considered 3 central simulation paradigms; discrete-event simulation, agent-based simulation and system dynamics simulation.

Discrete-event simulation can be described in terms of its components; entities, resources, control elements and operations (Schriber & Brunner, 1997). Entities interact with system resources based on the rules established by control elements to perform operations.

Agent-based simulation functions somewhat differently than discrete-event simulation. In agent-based simulation agents are the primary focus. Agents are independent decision makers in a system that react dynamically based on their characteristics and surroundings in a simulated environment (Macal & North, 2007). Agent-based simulation is then the evolution of the behaviour of the agents and their environment over time.

System dynamics simulation functions in a significantly different manner than discrete-event simulation or agent-based simulation. System dynamics is primarily concerned with an aggregate level of detail. It is not focused on individual entities or agents but aggregate behaviour of groups. It functions by considering aggregate 'stocks' and their flow within a system based on feedback loops (Coyle, 1996).

In Figure 2 (Borshchev & Filippov, 2004) provide a useful frame of reference for the simulation paradigms considered. Figure 2 shows a comparison of the three paradigms with respect to their appropriateness at various levels of abstraction. Borshchev and Filippov show that discrete-event simulation is most appropriate at low to mid levels of abstraction in part due to its focus on individual entities. At the opposite end of the spectrum they show that system dynamics simulation is best suited for modelling system with a high level of abstraction. In contrast to discrete-event and system dynamics, agent-based simulation can be used across all levels of abstraction with the capability to model operational level detail but also present high level trends accurately (Borshchev & Filippov, 2004). Based on this information it appears safe to conclude that regardless of the level of abstraction that modelling a diagnostic imaging scheduling process requires, agent-based simulation would be an acceptable tool.

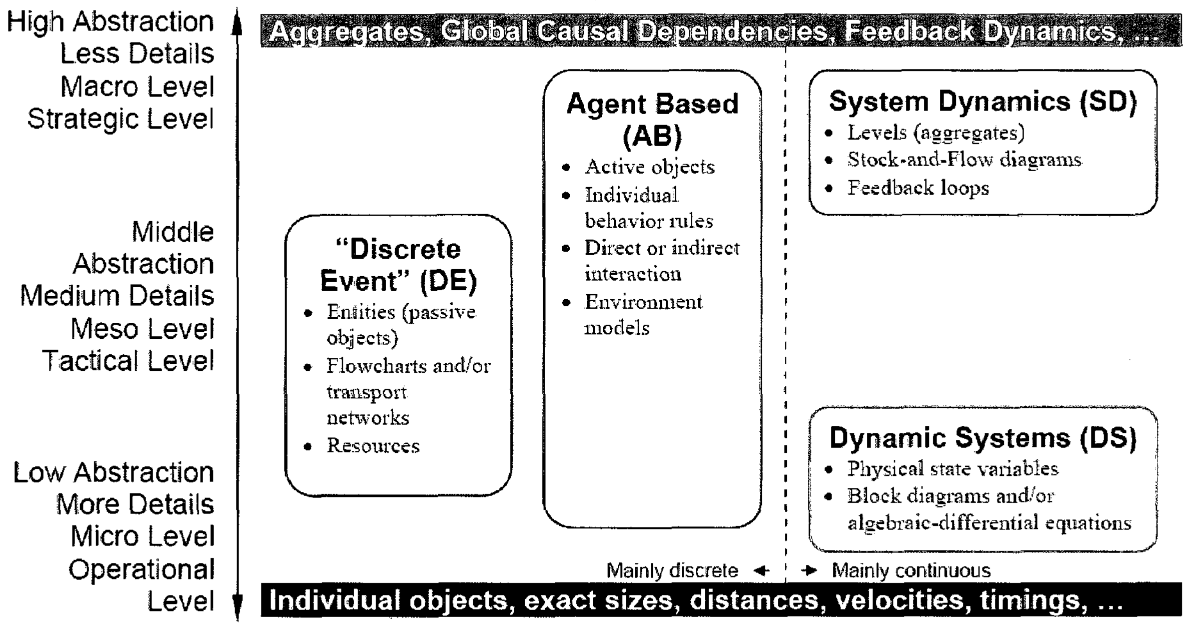


Figure 2 - Approaches (Paradigms) in simulation modelling on abstraction level scale

The decision to use agent-based simulation was influenced in part also by Mr. McEvoy who felt that this modelling technique might be especially applicable to diagnostic imaging scheduling and recommended a free simulation software package, NetLogo™, to use in the modelling process. Additionally, a preliminary review of literature revealed that using agent-based simulation to model an outpatient scheduling system would be relatively novel. In support of this approach were (Macal & North, 2007) who identify the appropriate time to use agent-based simulation with the following criteria:

- When there is a natural representation as agents
- When there are decisions and behaviours that can be defined discretely (with boundaries)
- When it is important that agents adapt and change their behaviours
- When it is important that agents learn and engage in dynamic strategic behaviours
- When it is important that agents have dynamic relationships with other agents, and agent relationships form and dissolve

- When it is important that agents form organizations, and adaptation and learning are important at the organization level
- When it is important that agents have a spatial component to their behaviours and interactions
- When the past is no predictor of the future
- When scaling-up to arbitrary levels is important
- When process structural change needs to be a result of the model, rather than a model input

The diagnostic scheduling process meets the above criteria and so it was determined that agent-based simulation would be an acceptable method to model the process. In the case of outpatient scheduling, requests for appointments are considered agents and the schedule, represented on a 2 dimensional plane (Time of the Day x Day in the planning horizon), is considered the environment.

2.2.2 Agent-Based Simulation

Agent-based simulation has a history in many fields including economics, mathematics, biology, engineering, sociology and psychology (Axelrod, 2005). The application of agent-based simulation to healthcare is a relatively novel but expanding field. However, much of that expansion is focused on modelling the transmission of infectious diseases, such as the work of (Triola & Holzman, 2003) who modelled the transmission of nosocomial diseases in intensive care units or (Teweldemedhin, Marwala, & Mueller, 2004) who study the transmission of HIV.

2.2.3 Simulation in Healthcare

Although there is a limited amount of research that has employed agent-based simulation in healthcare settings, there is a significant amount of research in healthcare using other forms of simulation. This should not be taken to mean that agent-based simulation does not have a place in healthcare; just that it is a relatively unexplored application. Although somewhat dated (Jun, Jacobson, & Swisher, 1999) survey over one hundred publications which employ simulation in healthcare. The uses of simulation they present are diverse including (but certainly not limited to) patient routing and flow schemes (Garcia et al, 1995) (McGuire, 1994) (Blake, Carter, & Richardson, 1996) and bed sizing and planning (Butler, Karwan, & Sweigart, 1992) (Lowery, 1992) (Dumas, 1985).

Of particular interest to this thesis were those publications focused on patient scheduling, including the work of (Bailey, 1952) who contributed some of the earliest work in outpatient scheduling. Outpatients are those patients who need to stay in the hospital overnight after visiting during the day. Bailey, looking at outpatients, counterbalanced patient wait times with physician utilization, developing heuristic techniques for use in batch scheduling. Although pre-dating computer simulation, Bailey's work helped pave the way for the application of a scientific approach to the study of outpatient scheduling. (Smith, Schroer, & Shannon, 1979) continue in a similar vein with their work that considers maximizing patients seen by a physician during a 3 hour session, while minimizing patient waiting time and determining the required number of nurses and examination rooms needed.

2.3 Outpatient Scheduling and Simulation

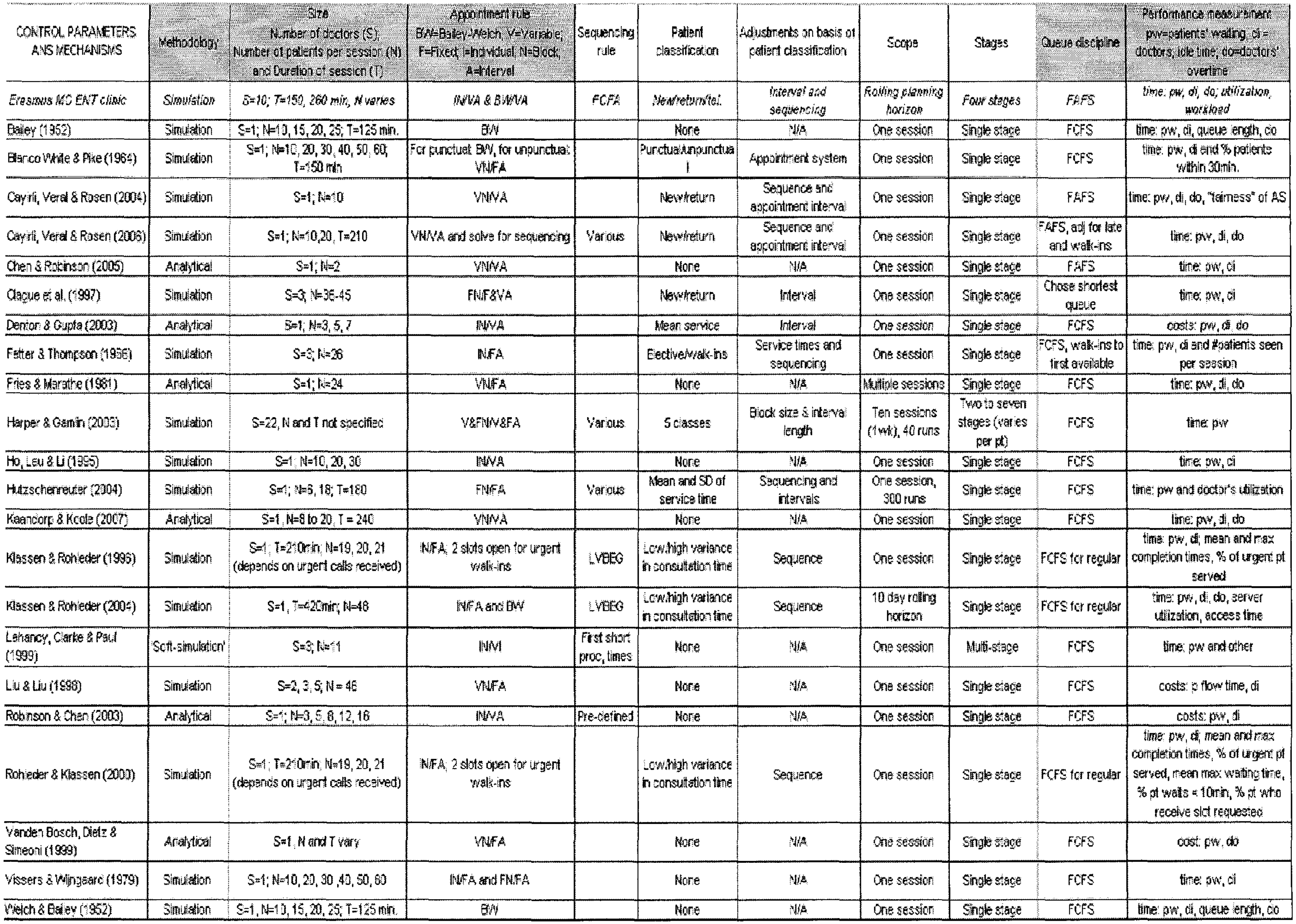
For a more current look at simulation focused specifically on outpatient scheduling we turn to (Cayirli & Veral, 2003) who survey outpatient scheduling and (Westeneng, 2007) who distils their work. Westeneng presents a useful condensed version of Cayirli & Veral's outpatient scheduling survey as part of his thesis on the evaluation of alternative appointment systems. His thesis shares commonalities with this one but differs in its goals and approach. While this thesis focuses on a standard simulation tool for outpatient scheduling in diagnostic imaging Westeneng focused on developing an optimal scheduling procedure for a single ear, nose and throat clinic.

Westeneng presents Cayirli & Veral's work in two tables (See Tables 1 and 2). Table 1 captures each works' input parameters; those parameters that are beyond the control of the simulator. These parameters could also be called outside forces or factors as they act on their respective systems from the outside, relatively uncontrolled by the system stakeholders (Note: Not all of the material referenced by Westeneg could be located, however the table has been reproduced as it appears in his thesis). Table 2 presents the control factors and mechanisms imposed on each system. These are the variables of the system that are available for manipulation by the simulator or the system stakeholder.

Westeneng's work served as a start point for establishing those internal and external parameters that effect the operation of a diagnostic imaging scheduling system. While some parameters are not applicable in the case of diagnostic imaging, others served to develop a deeper understanding of the system when considered with the assistance of healthcare professionals and hospital decision makers.

Input Parameters:	Service Time Distribution	Patient Punctuality (mean, st.dev)	No-Shows (p = no-show probability)	Walk-Ins (regular and emergency)	Doctors' Lateness	Doctors' Interruption Level
Articles:						
(Westeneng, 2007)	Gamma	N(-13, 17)	p = 0.05	Emergency only	Late N(5,15) minutes	yes (DICT)
(Bailey, 1952)	Gamma	Punctual	p = 0	None	Punctual	None
(Blanco White & Pike)	Gamma	Gamma, mu=0	p = 0, 0.09 and 0.19	None	0, 5, 10, 15 or 20 min.	None
(Cayirli, Veral, & Rosen, 2004)	Lognormal	N (-15, 25)	p = 0 and 0.15	0 to 15%, also regular	Punctual	None
(Cayirli, Veral, & Rosen, 2006)	Lognormal	N(0,25) and N (-15,25)	p = 0 and 0.15	0 to 15%, also regular	Punctual	None
(Chen & Robinson, 2005)	Randomly	Unpunctual, mu=0	p = 0	None	Punctual	None
(Clague, Reed, Barlow, Rada, Clarke, & Edwards, 1997)	Randomly	Punctual	p = 0, .2, .3	None	Punctual	None
(Denton & Gupta, 2003)	Uniform, Gamma and Normal	Punctual	p = 0	None	Punctual	None
(Fetter & Thompson, 1966)	Empirically collected	Late allowed to max. 5 min.	p=[0.04-0.22] with mean 0.14	7 to 58% with mean 38%	0, 30 or 60 min	None
(Fries & Marathe, 1981)	Negative, Exponential	Punctual	p = 0	None	Punctual	None
(Harper & Gamlin, 2003)	Not specified	Unpunctual (mean 8.3 min early, SD=14.7 min)	p > 0 (not specified)	Urgent	Unpunctual	None
(Ho, Lau, & Li, 1995)	Uniform, exponential	Punctual	p=0, 0.10, 0.20	None	Punctual	None
(Hutzschenreuter, 2004)	Triangular, Gamma	Unpunctual, (-10, 10)	p=0.10	None	Punctual	None
(Kaandorp & Koole, 2007)	Exponential	Punctual	p = 0, 0.1, 0.25, 0.5	None	Punctual	None
(Klassen & Rohleder, 1996)	Lognormal	Punctual	p = 0.05	Max 2 emergencies per session	Punctual	None
(Klassen & Rohleder, 2004)	Lognormal	Punctual	p = 0.05	10 % of patients	Punctual	None
(Lehaney, Clarke, & Paul, 1999)	Not specified	Punctual	p = 0	None	Punctual	yes
(Liu & Liu, 1998)	Uniform, exponential, Weibull	Punctual	p = 0, 0.10, 0.20	None	Uniform over [0,6] min. late	None
(Robinson & Chen, 2003)	Generalized Lambda	Punctual	p = 0	None	Punctual	None
(Rohleder & Klassen, 2000)	Lognormal	Punctual	p = 0.05	Max. 2 emergencies per session	Punctual	None
(Vanden Bosch, Dietz, & Simeoni, 1999)	Erlang	Punctual	p = 0	None	Punctual	None
(Vissers & Wijngaard, 1979)	General	In system earliness	Included by adjusting service times	Included by adjusting service times	In system earliness	None
(Welch & Bailey, 1952)	Gamma	Punctual	p = 0	None	Punctual	None

Table 1 - Westeneng's Input Parameters from Outpatient Scheduling Survey



CONTROL PARAMETERS ANS MECHANISMS	Methodology	Size Number of doctors (S); Number of patients per session (N) and Duration of session (T)	Appointment rule BW=Bailey-Welch; V=Variable; F=Fixed; I=Individual; N=Block; A=Interval	Sequencing rule	Patient classification	Adjustments on basis of patient classification	Scope	Stages	Queue discipline	Performance measurement pw=patients' waiting, di = doctors' idle time; do=doctors' overtime
Erasmus MC ENT clinic	*Simulation*	*S=10; T=150, 260 min, N varies*	*IN/VA & BW/VA*	*FCFA*	*New/return/tel.*	*Interval and sequencing*	*Rolling planning horizon*	*Four stages*	*FAFS*	*time: pw, di, do; utilization, workload*
Bailey (1952)	Simulation	S=1; N=10, 15, 20, 25; T=125 min.	BW		None	N/A	One session	Single stage	FCFS	time: pw, di, queue length, do
Blanco White & Pike (1964)	Simulation	S=1; N=10, 20, 30, 40, 50, 60; T=150 min	For punctual: BW, for unpunctual: VN/FA		Punctual/unpunctual	Appointment system	One session	Single stage	FCFS	time: pw, di and % patients within 30min.
Cayirli, Veral & Rosen (2004)	Simulation	S=1; N=10	VN/VA		New/return	Sequence and appointment interval	One session	Single stage	FAFS	time: pw, di, do, "fairness" of AS
Cayirli, Veral & Rosen (2006)	Simulation	S=1; N=10,20, T=210	VN/VA and solve for sequencing	Various	New/return	Sequence and appointment interval	One session	Single stage	FAFS, adj for late and walk-ins	time: pw, di, do
Chen & Robinson (2005)	Analytical	S=1; N=2	VN/VA		None	N/A	One session	Single stage	FAFS	time: pw, di
Clague et al. (1997)	Simulation	S=3; N=36-45	FN/F&VA		New/return	Interval	One session	Single stage	Chose shortest queue	time: pw, di
Denton & Gupta (2003)	*Analytical*	*S=1; N=3, 5, 7*	*IN/VA*		*Mean service*	*Interval*	*One session*	*Single stage*	FCFS	*costs: pw, di, do*
Fetter & Thompson (1966)	Simulation	S=3; N=26	IN/FA		Elective/walk-ins	Service times and sequencing	One session	Single stage	FCFS, walk-ins to first available	time: pw, di and #patients seen per session
Fries & Marathe (1981)	Analytical	S=1; N=24	VN/FA		None	N/A	Multiple sessions	Single stage	FCFS	time: pw, di, do
Harper & Gamlin (2003)	Simulation	S=22, N and T not specified	V&FN/V&FA	Various	5 classes	Block size & interval length	Ten sessions (1wk), 40 runs	Two to seven stages (varies per pt)	FCFS	time: pw
Ho, Lau & Li (1995)	Simulation	S=1; N=10, 20, 30	IN/VA		None	N/A	One session	Single stage	FCFS	time: pw, di
Hutzschenreuter (2004)	Simulation	S=1; N=6, 18; T=180	FN/FA	Various	Mean and SD of service time	Sequencing and intervals	One session, 300 runs	Single stage	FCFS	time: pw and doctor's utilization
Kaandorp & Koole (2007)	Analytical	S=1, N=8 to 20, T = 240	VN/VA		None	N/A	One session	Single stage	FCFS	time: pw, di, do
Klassen & Rohleder (1996)	Simulation	S=1; T=210min; N=19, 20, 21 (depends on urgent calls received)	IN/FA; 2 slots open for urgent walk-ins	LVBEG	Low/high variance in consultation time	Sequence	One session	Single stage	FCFS for regular	time: pw, di; mean and max completion times, % of urgent pt served
Klassen & Rohleder (2004)	Simulation	S=1, T=420min; N=48	IN/FA and BW	LVBEG	Low/high variance in consultation time	Sequence	10 day rolling horizon	Single stage	FCFS for regular	time: pw, di, do, server utilization, access time
Lehaney, Clarke & Paul (1999)	'Soft-simulation'	S=3; N=11	IN/VI	First short proc. times	None	N/A	One session	Multi-stage	FCFS	time: pw and other
Liu & Liu (1998)	Simulation	S=2, 3, 5; N = 46	VN/FA		None	N/A	One session	Single stage	FCFS	costs: p flow time, di
Robinson & Chen (2003)	Analytical	S=1; N=3, 5, 8, 12, 16	IN/VA	Pre-defined	None	N/A	One session	Single stage	FCFS	costs: pw, di
Rohleder & Klassen (2000)	Simulation	S=1; T=210min; N=19, 20, 21 (depends on urgent calls received)	IN/FA; 2 slots open for urgent walk-ins		Low/high variance in consultation time	Sequence	One session	Single stage	FCFS for regular	time: pw, di; mean and max completion times, % of urgent pt served, mean max waiting time, % pt waits ≤ 10min, % pt who receive slot requested
Vanden Bosch, Dietz & Simeoni (1999)	Analytical	S=1, N and T vary	VN/FA		None	N/A	One session	Single stage	FCFS	cost: pw, do
Vissers & Wijngaard (1979)	Simulation	S=1; N=10, 20, 30, 40, 50, 60	IN/FA and FN/FA		None	N/A	One session	Single stage	FCFS	time: pw, di
Welch & Bailey (1952)	Simulation	S=1, N=10, 15, 20, 25; T=125 min.	BW		None	N/A	One session	Single stage	FCFS	time: pw, di, queue length, do

Table 2 - Westeneng's control parameters and mechanisms from Outpatient Scheduling Survey

Building on Westeneng's work Table 3 (split onto 3 separate pages) combines Westeneng's key parameters and cited works with that of this thesis. This combined table also compares the work of the thesis (labelled as ABMT) with that of other published articles.

Input Parameters:	**Service Time Distribution**	**Patient Punctuality (mean, st.dev)**	**No-Shows (p = no-show probability)**	**Walk-Ins (regular and emergency)**
Articles:				
Westeneng (2007)	Gamma	N(-13, 17)	p = 0.05	Emergency only
Bailey (1952)	Gamma	Punctual	p = 0	None
Blanco White & Pike (1964)	Gamma	Gamma, mu=0	p = 0, 0.09 and 0.19	None
Cayirli, Veral & Rosen (2004)	Lognormal	N (-15, 25)	p = 0 and 0.15	0 to 15%, also regular
Cayirli, Veral & Rosen (2006)	Lognormal	N(0,25) and N (-15,25)	p = 0 and 0.15	0 to 15%, also regular
Chen & Robinson (2005)	Randomly	Unpunctual, mu=0	p = 0	None
Clague et al. (1997)	Randomly	Punctual	p = 0, .2, .3	None
Denton & Gupta (2003)	Uniform, Gamma and Normal	Punctual	p = 0	None
Fetter & Thompson (1966)	Empirically collected	Late allowed to max. 5 min.	p=[0.04-0.22] with mean 0.14	7 to 58% with mean 38%
Fries & Marathe (1981)	Negative, Exponential	Punctual	p = 0	None
Harper & Gamlin (2003)	Not specified	Unpunctual (mean 8.3 min early, SD=14.7 min)	p > 0 (not specified)	Urgent
Ho, Lau & Li (1995)	Uniform, exponential	Punctual	p=0, 0.10, 0.20	None
Hutzschenreuter (2004)	Triangular, Gamma	Unpunctual, (-10, 10)	p=0.10	None
Kaandorp & Koole (2007)	Exponential	Punctual	p = 0, 0.1, 0.25, 0.5	None
Klassen & Rohleder (1996)	Lognormal	Punctual	p = 0.05	Max 2 emergencies per session
Klassen & Rohleder (2004)	Lognormal	Punctual	p = 0.05	10 % of patients
Lehancy, Clarke & Paul (1999)	Not specified	Punctual	p = 0	None
Liu & Liu (1998)	Uniform, exponential, Weibull	Punctual	p = 0, 0.10, 0.20	None
Robinson & Chen (2003)	Generalized Lambda	Punctual	p = 0	None
Rohleder & Klassen (2000)	Lognormal	Punctual	p = 0.05	Max. 2 emergencies per session
Rohleder & Klassen (2002)	Lognormal	Punctual	p = 0.05	Max. 2 emergencies per session
Vanden Bosch, Dietz & Simeoni (1999)	Erlang	Punctual	p = 0	None
Vissers & Wijngaard (1979)	General	In system earliness	Included by adjusting service times	Included by adjusting service times
Welch & Bailey (1952)	Gamma	Punctual	p = 0	None
ABMT	Deterministic	Punctual	p = 0	Variable Rate

Table 3 - Comparison of ABMT to published works

Input Parameters:	Doctors' Lateness	Methodology	# of Servers	Patient Classification
Articles:				
Westeneng (2007)	Late N(5,15) minutes	Simulation	10	Yes, 3
Bailey (1952)	Punctual	Simulation	1	No
Blanco White & Pike (1964)	0, 5, 10, 15 or 20 min.	Simulation	1	Yes, 2
Cayirli, Veral & Rosen (2004)	Punctual	Simulation	1	Yes, 2
Cayirli, Veral & Rosen (2006)	Punctual	Simulation	1	Yes,2
Chen & Robinson (2005)	Punctual	Analytical	1	No
Clague et al. (1997)	Punctual	Simulation	3	Yes, 2
Denton & Gupta (2003)	Punctual	Analytical	1	Yes, Expected Service Time
Fetter & Thompson (1966)	0, 30 or 60 min	Simulation	3	Yes, 2
Fries & Marathe (1981)	Punctual	Analytical	1	No
Harper & Gamlin (2003)	Unpunctual	Simulation	22	Yes, 5
Ho, Lau & Li (1995)	Punctual	Simulation	1	No
Hutzschenreuter (2004)	Punctual	Simulation	1	Yes, Mean & SD
Kaandorp & Koole (2007)	Punctual	Analytical	1	No
Klassen & Rohleder (1996)	Punctual	Simulation	1	Yes, 2
Klassen & Rohleder (2004)	Punctual	Simulation	1	Yes, 2
Lehancy, Clarke & Paul (1999)	Punctual	Soft-Simulation	3	No
Liu & Liu (1998)	Uniform over [0,6] min. late	Simulation	2	No
Robinson & Chen (2003)	Punctual	Analytical	1	No
Rohleder & Klassen (2000)	Punctual	Simulation	1	Yes, 2
Rohleder & Klassen (2002)	Punctual	Simulation	1	Yes, 3
Vanden Bosch, Dietz & Simeoni (1999)	Punctual	Analytical	1	No
Vissers & Wijngaard (1979)	In system earliness	Simulation	1	No
Welch & Bailey (1952)	Punctual	Simulation	1	No
ABMT	Punctual Servers	Simulation	1 to 10	Yes, 4

Table 3 Continued - Comparison of ABMT to published works

Input Parameters:	Adjustments based on pt. class	Scope	Queue Discipline	Performance Measures PW=Patients_Wait SI=Server_Idle SO=Server_Overtime
Articles:				
Westeneng (2007)	Interval & Sequencing	Rolling Planning Horizon	FAFS	PW SO SI utilization, workload
Bailey (1952)	N/A	One Session	FCFS	PW SO SI queue length
Blanco White & Pike (1964)	Appointment System	One Session	FCFS	PW SI patients within 30 min
Cayirli, Veral & Rosen (2004)	Sequence & Appointment Interval	One Session	FCFS	PW SO SI, 'fairness'
Cayirli, Veral & Rosen (2006)	Sequence & Appointment Interval	One Session	FAFS	PW SI SO
Chen & Robinson (2005)	N/A	One Session	FAFS	PW SI
Clague et al. (1997)	Interval	One Session	Shortest Queue	PW SI
Denton & Gupta (2003)	Interval	One Session	FCFS	PW SI SO cost
Fetter & Thompson (1966)	Service Times & Sequencing	One Session	FCFS	PW SI patients/session
Fries & Marathe (1981)	N/A	Multiple Sessions	FCFS	PW SI SO
Harper & Gamlin (2003)	Block Size & Interval Length	Ten Sessions	FCFS	PW
Ho, Lau & Li (1995)	N/A	One Session	FCFS	PW SI
Hutzschenreuter (2004)	Sequencing & Intervals	One Session	FCFS	PW utilization
Kaandorp & Koole (2007)	N/A	One Session	FCFS	PW SI SO
Klassen & Rohleder (1996)	Sequence	One Session	FCFS for regular	PW SI min./max. urgent cases
Klassen & Rohleder (2004)	Sequence	10 Day Rolling Horizon	FCFS for regular	PW SI SO utilization access time
Lehancy, Clarke & Paul (1999)	N/A	One Session	FCFS	PW
Liu & Liu (1998)	N/A	One Session	FCFS	SI flow
Robinson & Chen (2003)	N/A	One Session	FCFS	PW SI
Rohleder & Klassen (2000)	Sequence	One Session	FCFS for Regular	PW SI min./max. urgent cases
Rohleder & Klassen (2002)	Sequence	Rolling Planning Horizon	FCFS for Regular	PW SI utilization, access time
Vanden Bosch, Dietz & Simeoni (1999)	N/A	One Session	FCFS	PW SO
Vissers & Wijngaard (1979)	N/A	One Session	FCFS	PW SI
Welch & Bailey (1952)	N/A	One Session	FCFS	PW SI SO queue length
ABMT	Yes	Rolling Planning Horizon	FCFS for regular	Access Time, % of patients on target, avg. wait time by class

Table 3 Continued - Comparison of ABMT to published works

2.4 Literature Review Conclusions

The literature review has established:

- That simulation is an acceptable means by which to create a decision support system, especially in those cases where the system is complex and has many stakeholders.
- The pros and cons of simulation and where it is most applicable
- That simulation in healthcare is a widely accepted practice and has the capability to yield positive verifiable and validated results.
- That agent-based simulation is appropriate for the level of abstraction required to model a diagnostic imaging scheduling system.
- That outpatient scheduling has been studied via simulation before but not through agent-based modelling.
- That when modelling outpatient scheduling there are a standard set of parameters that must be considered.
- That there is no established standard decision support tool for the scheduling of diagnostic imaging services.

It is based on these facts that we chose to build a decision support tool using agent-based simulation to assess the impact of operational level changes to a diagnostic imaging scheduling system.

3. *Agent-Based Modelling Tool (ABMT)*

This chapter describes the Agent-Based Modelling Tool (ABMT) built using NetLogo™ a programmable modelling environment well suited to complex dynamic systems. In 3.1 we introduce the ABMT Environment and in 3.2 the Patients. In 3.3 we describe the Scheduling Discipline. We end the chapter with a presentation of the User Interface.

3.1 ABMT Environment

NetLogo™ uses two different types of agents, 'patches' and 'turtles'. Patches are stationary and the collection of patches form the environment in which the turtles exist and move. In Figure 3 we see the agents used in the ABMT. Squares are patches and triangles are turtles. The colours green, blue, yellow and brown represent the different patient priority classes. Red, black and grey represent times that are not currently or cannot be used for scheduling a patient.

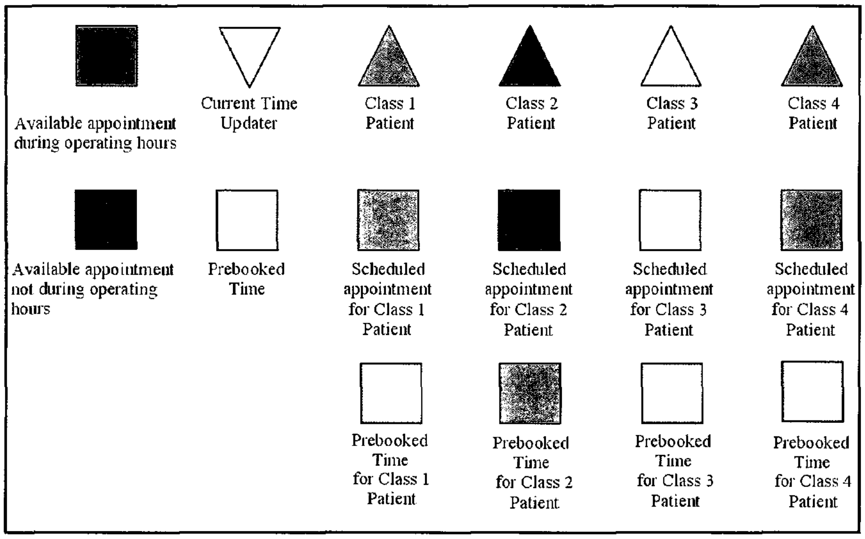

Available appointment during operating hours
Current Time Updater
Class 1 Patient
Class 2 Patient
Class 3 Patient
Class 4 Patient
Available appointment not during operating hours
Prebooked Time
Scheduled appointment for Class 1 Patient
Scheduled appointment for Class 2 Patient
Scheduled appointment for Class 3 Patient
Scheduled appointment for Class 4 Patient
Prebooked Time for Class 1 Patient
Prebooked Time for Class 2 Patient
Prebooked Time for Class 3 Patient
Prebooked Time for Class 4 Patient

Figure 3 - Patches and Turtle

The planning horizon is composed entirely of patches arranged to form a grid (See Figure 4). When configured for a single server each column represents a single day and each row a specific time of day. The number of days in the horizon is adjustable, but the number of appointment blocks in a day (red and black combined) is not. At current there are 96 blocks (patches) per day (column), each representing a 15 minute time block. Red patches are appointments that are available for scheduling and black patches are periods when patients cannot be booked. The number of operating hours per day (red patches) is controlled by the 'Scheduled-Hours-per-day' input box. Figure 4 depicts an empty planning horizon with 2.5 available hours per day.

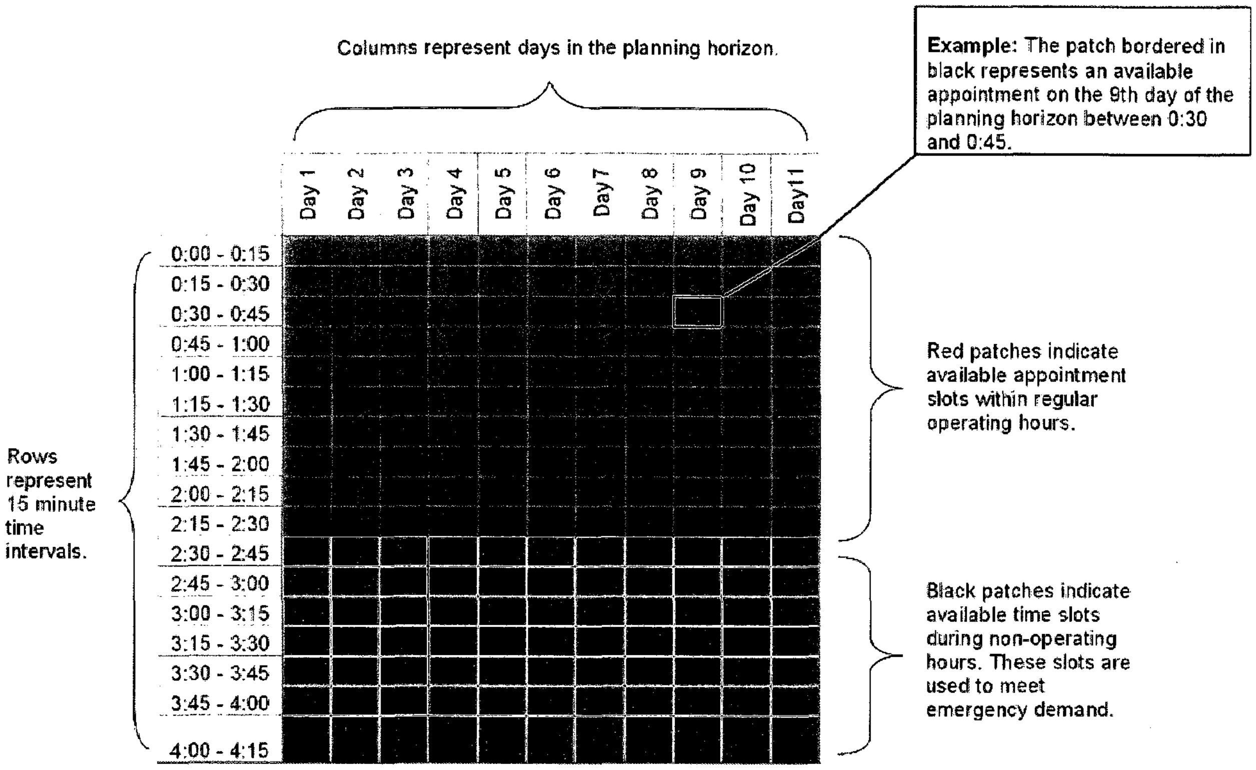

Columns represent days in the planning horizon.
Example: The patch bordered in black represents an available appointment on the 9th day of the planning horizon between 0:30 and 0:45.
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day7
Day 8
Day 9
Day 10
Day11
0:00 - 0:15
0:15 - 0:30
0:30 - 0:45
0:45 - 1:00
1:00 - 1:15
1:15 - 1:30
1:30 - 1:45
1:45 - 2:00
2:00 - 2:15
2:15 - 2:30
2:30 - 2:45
2:45 - 3:00
3:00 - 3:15
3:15 - 3:30
3:30 - 3:45
3:45 - 4:00
4:00 - 4:15
Rows represent 15 minute time intervals.
Red patches indicate available appointment slots within regular operating hours.
Black patches indicate available time slots during non-operating hours. These slots are used to meet emergency demand.

Figure 4 - Layout of Simulation Environment

For multi-server scenarios each column represents a specific server on a specific day. Figure 5 depicts a multi-server scenario with 3 servers and 2.5 hours of scheduled time per day.

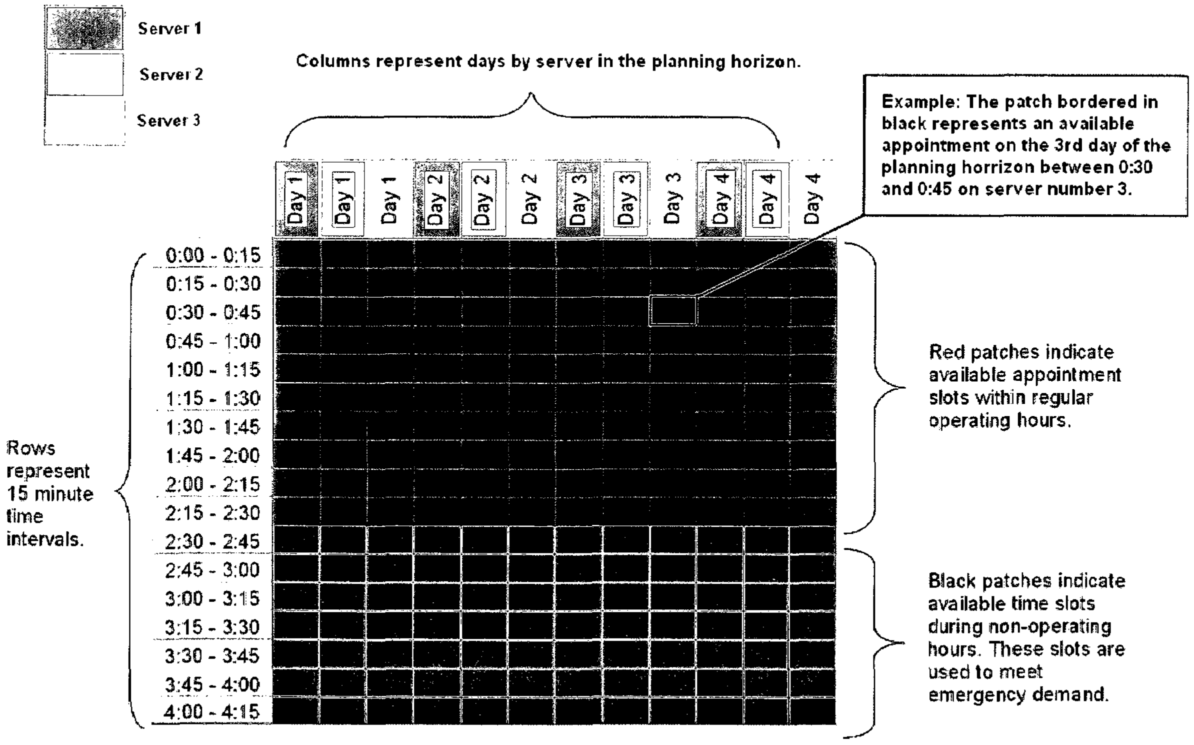

Server 1
Server 2
Server 3
Columns represent days by server in the planning horizon.
Example: The patch bordered in black represents an available appointment on the 3rd day of the planning horrizon between 0:30 and 0:45 on server number 3.
Day 1
Day 1
Day 1
Day 2
Day 2
Day 2
Day 3
Day 3
Day 3
Day 4
Day 4
Day 4
0:00 - 0:15
0:15 - 0:30
0:30 - 0:45
0:45 - 1:00
1:00 - 1:15
1:15 - 1:30
1:30 - 1:45
1:45 - 2:00
2:00 - 2:15
2:15 - 2:30
2:30 - 2:45
2:45 - 3:00
3:00 - 3:15
3:15 - 3:30
3:30 - 3:45
3:45 - 4:00
4:00 - 4:15
Rows represent 15 minute time intervals.
Red patches indicate available appointment slots within regular operating hours.
Black patches indicate available time slots during non-operating hours. These slots are used to meet emergency demand.

Figure 5 - Multi-Server Layout

Prebooked times are appointment slots set aside from the standard first come first serve scheduling process. These prebooked times are used in many cases to meet demand for patients who cannot wait for diagnostic imaging services. For example, many patients admitted to the hospital require service from the diagnostic imaging department during their stay. It is inefficient and hazardous to force them to wait for an appointment like a non-admitted patient might. To that end appointments are set aside each day to meet the potential demand for diagnostic imaging services from admitted patients. Figure 6 depicts an example of a planning horizon with prebooked time. Figure 7 shows 4 of the prebooked time controls.

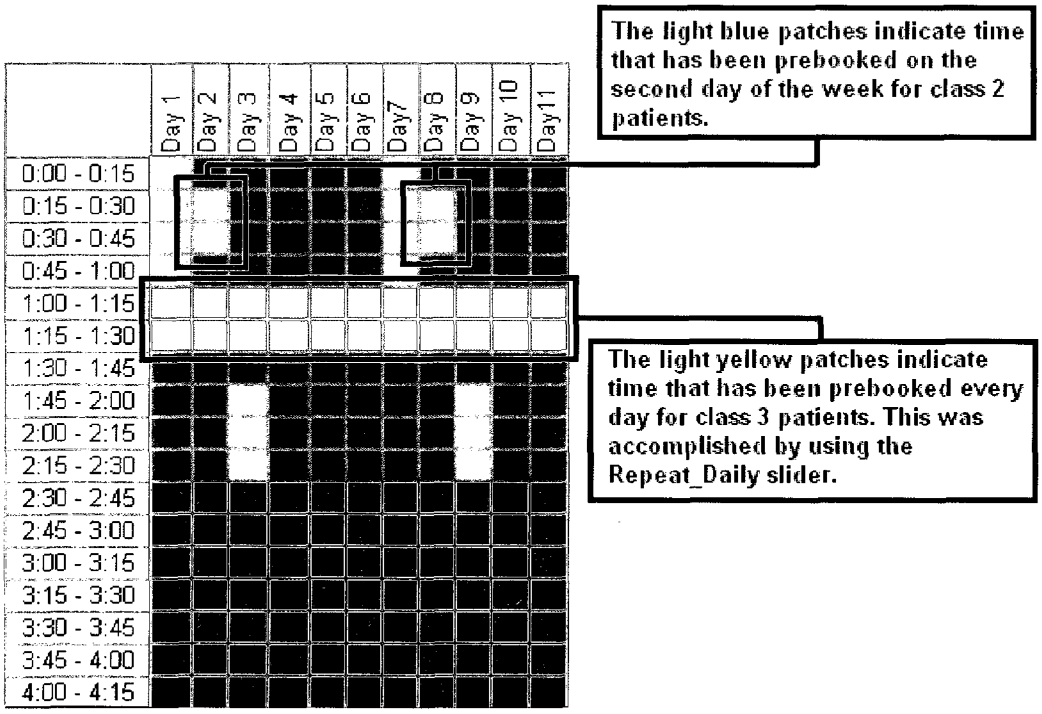
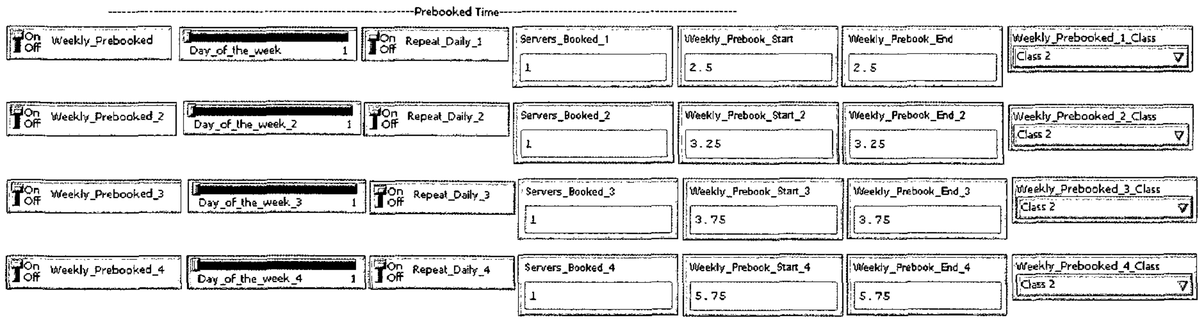

Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day7
Day 8
Day 9
Day 10
Day11
0:00 - 0:15
0:15 - 0:30
0:30 - 0:45
0:45 - 1:00
1:00 - 1:15
1:15 - 1:30
1:30 - 1:45
1:45 - 2:00
2:00 - 2:15
2:15 - 2:30
2:30 - 2:45
2:45 - 3:00
3:00 - 3:15
3:15 - 3:30
3:30 - 3:45
3:45 - 4:00
4:00 - 4:15
The light blue patches indicate time that has been prebooked on the second day of the week for class 2 patients.
The light yellow patches indicate time that has been prebooked every day for class 3 patients. This was accomplished by using the Repeat_Daily slider.

Figure 6 - Prebooked Time



Prebooked Time

On Off Weekly_Prebooked	Day_of_the_week 1	On Off Repeat_Daily_1	Servers_Booked_1 1	Weekly_Prebook_Start 2.5	Weekly_Prebook_End 2.5	Weekly_Prebooked_1_Class Class 2
On Off Weekly_Prebooked_2	Day_of_the_week_2 1	On Off Repeat_Daily_2	Servers_Booked_2 1	Weekly_Prebook_Start_2 3.25	Weekly_Prebook_End_2 3.25	Weekly_Prebooked_2_Class Class 2
On Off Weekly_Prebooked_3	Day_of_the_week_3 1	On Off Repeat_Daily_3	Servers_Booked_3 1	Weekly_Prebook_Start_3 3.75	Weekly_Prebook_End_3 3.75	Weekly_Prebooked_3_Class Class 2
On Off Weekly_Prebooked_4	Day_of_the_week_4 1	On Off Repeat_Daily_4	Servers_Booked_4 1	Weekly_Prebook_Start_4 5.75	Weekly_Prebook_End_4 5.75	Weekly_Prebooked_4_Class Class 2

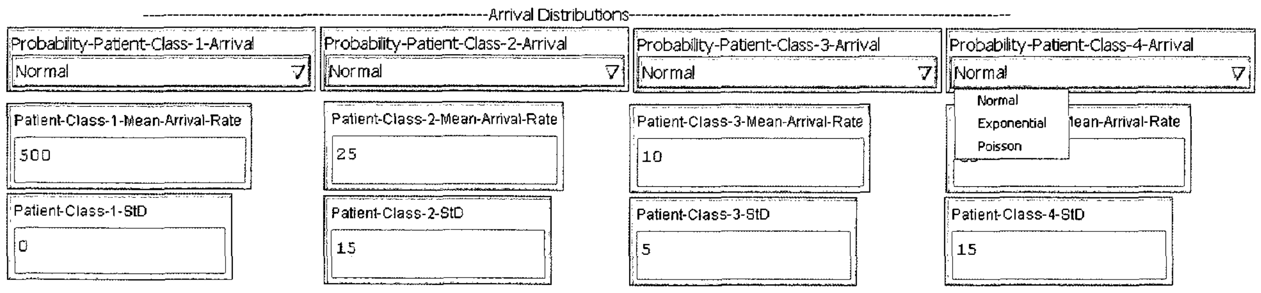
Figure 7 - Prebooked Time Control

3.2 Patients

Requests for patient service, also known simply as patients, are the driving force of the ABMT. The following subsections describe the different types of patients, the method with which they come to be in the system, and their interactions with each other and the simulation environment.

Patients, represented by turtles, come in 4 priority classes. These four classes are representations of the patient priority class 1 through 4 used in Canadian hospitals; each patient requesting service from the diagnostic imaging department is assigned a prior level by their physician. Class 1 patients require immediate attention while class 2, 3 and 4 patients are to be scheduled if possible within 2, 10 and 28 days respectively based on ministry of health guidelines.

Requests for patient service are received or 'arrive' according to probability distributions. The distributions govern the inter-arrival time between patients of the same class. The distributions available in NetLogo™ to describe the arrival rate are normal, exponential and Poisson. The user selects the distribution that most accurately describes their system from a drop down menu as seen below. Seen below in Figure 8 are the controls for the arrival rates of all 4 patient priority classes, example means and standard deviations can be seen in the input boxes. In this example we can see that Class 1 patients have a mean interarrival time of 500 minutes, thus Class 1 patients' arrivals are normally distributed with a mean of 500 minutes.


Arrival Distributions

Probability-Patient-Class-1-Arrival: Normal
Probability-Patient-Class-2-Arrival: Normal
Probability-Patient-Class-3-Arrival: Normal
Probability-Patient-Class-4-Arrival: Normal

Patient-Class-1-Mean-Arrival-Rate: 500
Patient-Class-2-Mean-Arrival-Rate: 25
Patient-Class-3-Mean-Arrival-Rate: 10
ean-Arrival-Rate:

Patient-Class-1-StD: 0
Patient-Class-2-StD: 15
Patient-Class-3-StD: 5
Patient-Class-4-StD: 15

Figure 8 - Arrival Distribution Control

The ABMT uses a deterministic service time of 15 minutes per patient. The assumption is made that all scans can be completed within 15 minutes and subsequent scans do not begin until 15 minutes has elapsed since the preceding scan started. This may not always be the reality but because the focus of this study is on access time not wait time and the resulting difference is considered negligible. In those instances where the scheduled length of the scan is 30 minutes, one patch is blocked off as prebooked for 'None.' That is to say that one of the patches, representing 15 minutes, is made unavailable for scheduling to account for the time lost to the 30 minute appointment.

3.3 Scheduling Discipline

Scheduling operates on a first come first serve basis with the exception of emergency patients and prebooked time. After a patient arrives in the system based on an arrival rate, the scheduling operation searches for an available appointment slot (red patch or appropriate prebooked time) by moving the patient down its current column patch by patch. If a patch is booked (not red or the appropriate prebooked time colour) the patient moves on to the next patch (the one directly below it). This continues until one of two things happens; if the patient comes to the end of scheduled time for a day it is moved to the top of the next column (next day) and it continues its search or alternatively if the patient finds an available appointment its search stops. Once the patient finds an available appointment it changes the colour of the free appointment patch to its patient priority class colour (blue patients make blue patches, brown patients make brown patches etc.). In this way patients are assigned to appointment slots. When scheduling reaches the end of the planning horizon it resumes at the beginning. This process is depicted below in Figure 9.

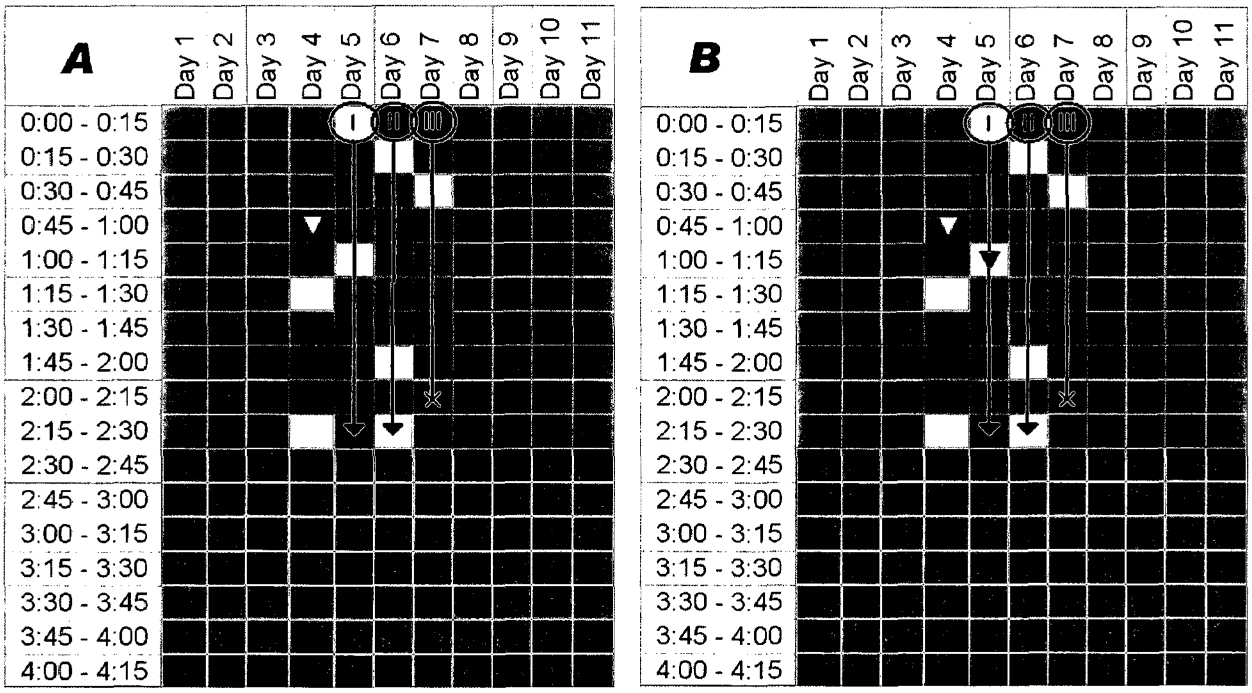
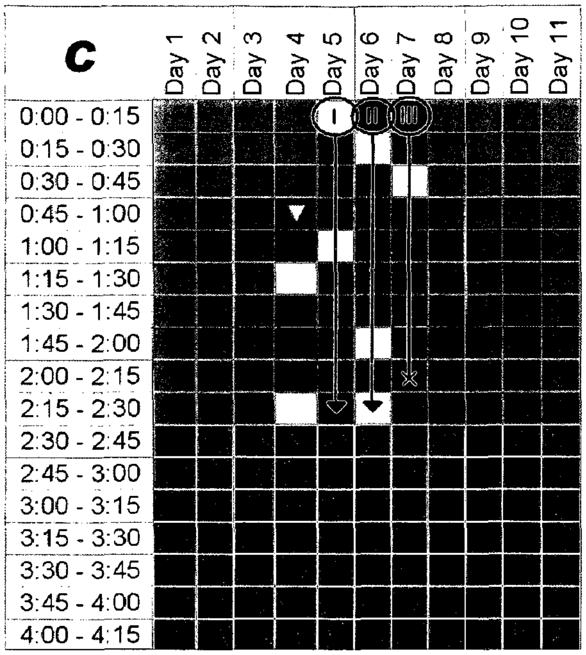


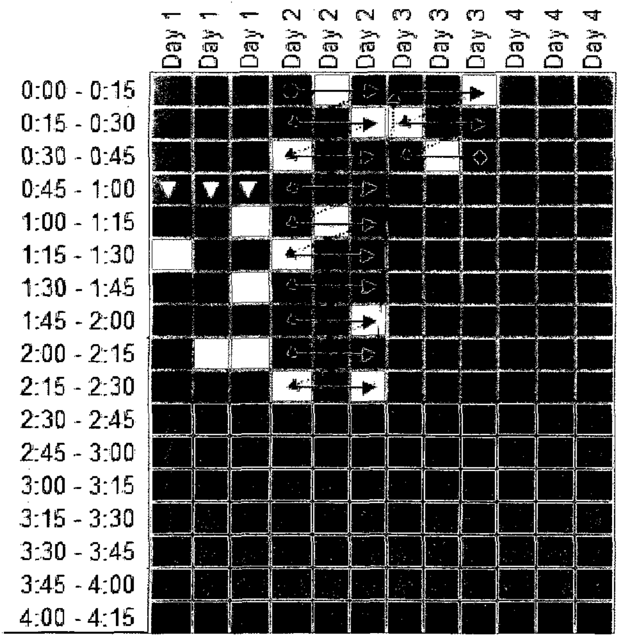
Figure 9 - Scheduling Process: Single Server



Example: Pictured above and adjacent is an example of patient scheduling for a single server. Section A shows the route the patient will take in search of an appointment (I then II then III). Section B shows us that it is a class 4 patient, as indicated by the brown triangle. Section C shows us the final result of the search and the subsequent appointment.

Figure 9 Continued - Scheduling Process: Single Server

Scheduling of patients occurs in much the same way for multiple servers as it does for a single server. The primary difference is that the scheduling operation attempts to schedule patients on each server at the earliest possible time before moving on to a later time. Figure 10 depicts scheduling in a multi-server scenario.



Example: Pictured adjacent is an example of patient scheduling in a multi-server scenario. In this case there are 3 servers and the planning horizon is 4 days long. The scheduling operation begins searching for an available appointment slot at the beginning of day 2 on the first server (furthest to the left in the horizon). This appointment is booked so the search continues by considering the availability of the 2nd server during that same period. The 2nd server is also unavailable so the search continues with the 3rd server. Because this server is also unavailable and there are no more servers the search begins again in the next time period (0:15 – 0:30) with the 1st server. The search continues in this way until an available appointment is found.

Figure 10 - Scheduling Process: Multi-Server

Class 1 patients require immediate attention; they pre-empt other patients, bumping them from their currently scheduled slot to the subsequent appointment slot. Bumping is the only action that takes precedence over prebooked time and the only action that can result in overtime for the hospital staff. The bumping process can be seen below in Figure 11. After the bump, all patients are moved forward in the same day. So, while the patient waits more time for service while in the clinic, it does not affect wait time as defined.

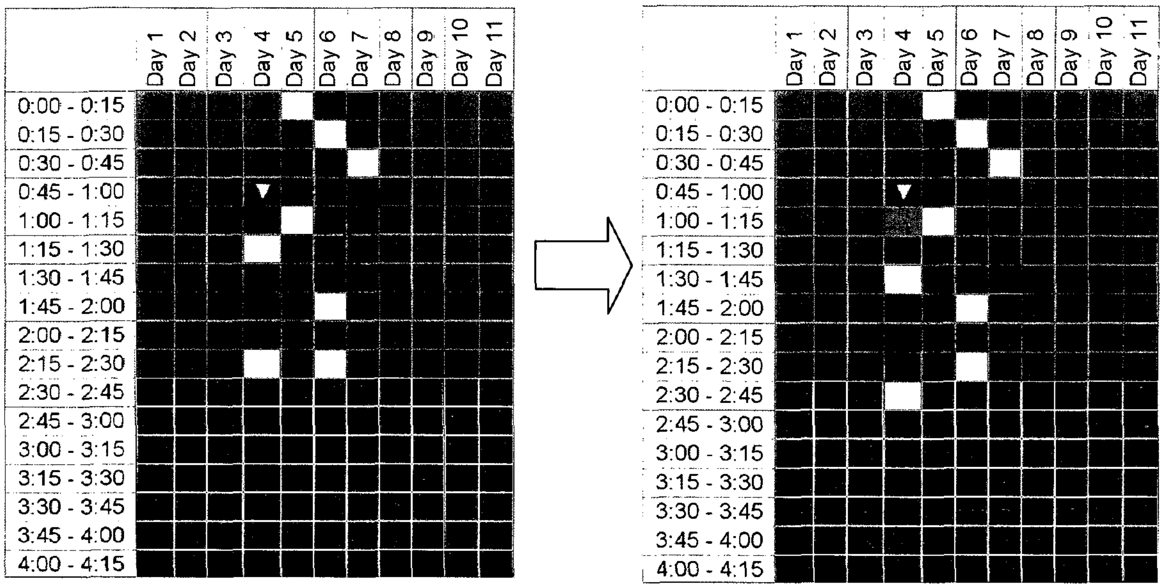


Figure 11 - Bumping: Before and After

The simulator works by scheduling patients in future appointment slots relative to a constantly updated 'current time'. Because the simulator uses a static number of days in its planning horizon it is necessary to reuse days (columns) to prevent the horizon from becoming full. Once scheduling reaches the end of the horizon (the far right column) it continues at the beginning of the horizon (the far left column).

Beginning from the first appointment slot on the first day of the horizon the current time 'updater' moves from appointment slot to subsequent appointment slot on each tick of the system. When the updater moves to an appointment it clears the patch of any previous appointments, returning the patch to its original (unscheduled) colour (red, black, or grey). In this way appointment slots are cleared for future appointments allowing for a stable queue of scheduled appointments to be simulated indefinitely. Additionally, the updater is used in the scheduling process to determine where the scheduling operation should begin its search for appointments. For example, patients are never scheduled on the day that they request an appointment (except emergencies), so the earliest a patient can be scheduled is the current day (as determined by the updater) plus one.

3.4 User Interface

3.4.1 Setup & Go

These controls update the main display area with the currently inputted prebooked times and initiate the simulation. Setup also clears the graphical outputs of the model as well as the average wait times and percentage of patients who exceed guidelines.

3.4.2 Data Recording

NetLogo™ allows the user to export data from simulations to external files. The ABMT has been configured to export the patient class and wait time data for each patient that is scheduled to a Microsoft Excel file. The GUI controls allow the user to choose whether or not they wish to record data, delete existing data or close the file the data is being recorded to.

3.4.3 Random Fill

The ABMT was designed to assist hospital decision makers in assessing changes to scheduling in diagnostic imaging systems. In order to accurately capture the current state of an existing system it is necessary to also simulate the existing queue of patients. The random fill functionality fills the planning horizon with class 4 patients up to a specified number of days. For example, if the user wished to model a system that at present has a 4 week wait time they would select a random fill of 28 days so that scheduling of patients would begin on the 29th day.

3.4.4 Simulation Run Time

The 'Days_to_run' input controls the duration of the simulation. The user enters the number of simulated days they wish the model to run for and the ABMT halts operation after that number of simulated days have passed.

3.4.5 Number of Servers

This control allows the user to select the number of servers that will be used in the system.

3.4.6 Scheduled Hours per Day

This input determines the division between available appointments during operating hours (red patches) and available appoints during non-operating hours (black patches).

3.4.7 Information Display

The simulator's graphical user interface has been designed to give the user as much relevant data as possible regarding the progress of a simulated model. At present there are several output figures, graphics and charts to help the user make an initial analysis of the model being simulated.

Model behaviour can be seen in the **main display window** where scheduling takes place; this display window offers insight into how patients are interacting with the schedule and the nature of the appointment usage; the planning horizon is displayed here.

Adjacent to the main display window are **plots of patient wait times** broken down by priority class. The plots operate by recording the time between arrival and services for each patient that enters the system. Additionally, in order to keep the plots chronologically synchronous they are updated on every system tick regardless of whether or not a patient of the type they are tracking is created.

The simulator also displays the **percentage of patients** who have exceeded their recommended wait time by class. This number is updated on every patient arrival. Additionally, the simulator tracks and displays the **average wait** for each patient type.

Figure 12 shows the ABMT's main display, the average wait time for each class, the percentage of each class that exceeds their wait time targets as well as the output plot windows for class 2, 3 and 4 patients.

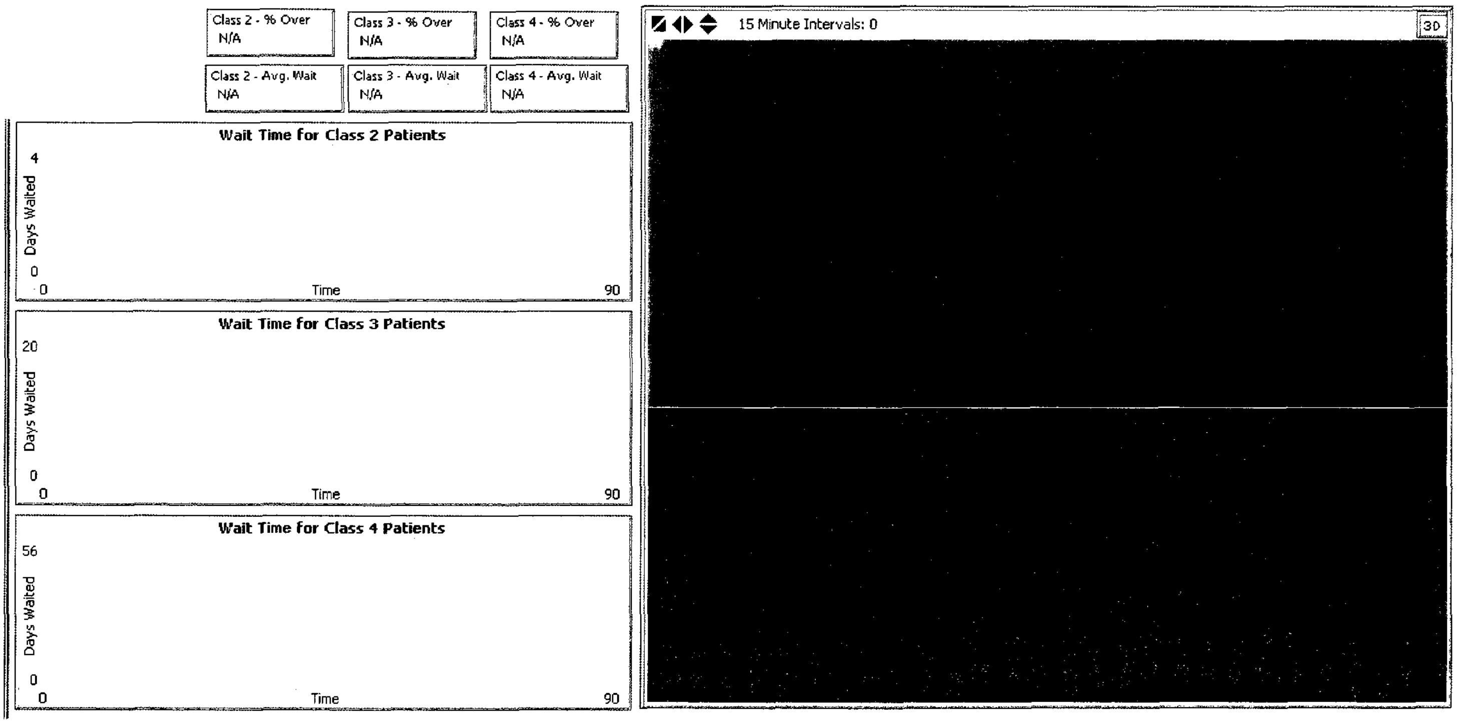

Class 2 - % Over N/A
Class 3 - % Over N/A
Class 4 - % Over N/A
Class 2 - Avg. Wait N/A
Class 3 - Avg. Wait N/A
Class 4 - Avg. Wait N/A
15 Minute Intervals: 0
3D
Wait Time for Class 2 Patients
4
Days Waited
0
0
Time
90
Wait Time for Class 3 Patients
20
Days Waited
0
0
Time
90
Wait Time for Class 4 Patients
56
Days Waited
0
0
Time
90

Figure 12 - Information Display

Configuring the ABMT to model a specific case is a relatively simple process but it does require the user to have pertinent historical data in order to establish a basis for comparison. In order for the ABMT to give the most accurate results the user should have access to or an approximation of the following data:

- The current number of days a patient served strictly on a FCFC basis can expect to wait (In most Canadian hospitals this information is available online)
- The number of servers the system uses
- The number of hours the system is operation per day
- The arrival rate of each class of patients
- Which times are prebooked and what classes they are prebooked for

With this data available the user is able to establish a model of the current system so that the effects of changes to the system can be gauged by comparison.

4. *Case Study: Hotel Dieu Grace Hospital*

To test the applicability of the ABMT and to assess its accuracy, historical data was used to model a recent change in the diagnostic imaging department of HDGH. In November 2007 an additional CT scanner was added to HDGH's diagnostic imaging department, bringing the total to two. The diagnostic imaging department provided access to historical scheduling data for the CT scanners as well as information regarding the scheduling process. For the ABMT to have successfully modelled the effects of the change it needed to predict the change in wait time trends for CT scans.

4.1 Scheduling Process

Scheduling of CT scans is the responsibility of a single CT booking clerk. The clerk receives requests for scans via telephone and fax throughout the day from both physicians' offices and patients themselves. The clerk takes the requests and books an appointment in the schedule. The clerk is also responsible for the confirmation of appointments as the scheduled scan date approaches.

4.2 ABMT Parameters

In order to model the scheduling process of the diagnostic imaging department at HDGH several key parameters needed to be determined. These parameters were:

- Arrival Rate of Class 1 Patients
- Arrival Rate of Class 2 Patients
- Arrival Rate of Class 3 Patients
- Arrival Rate of Class 4 Patients
- Operating hours
- # of operational scanners
- Prebooked periods

4.2.1 Arrival Rates

Ideally arrival rates and patterns (for requests for appointments) would be determined by fitting the number of arrivals per day and their arrival times to a mathematical distribution. Unfortunately HDGH does not record at what time during the day a request for service is made. They do however record how many requests were made per day. The assumption was made that the patient arrivals are governed by an exponential distribution, (Winston, 2004) cites (Devardo, 2003) in support of this decision, and a mean interarrival time value was calculated using the daily arrival totals. Values for each class can be seen in the next chapter.

4.2.2 Operating Hours & Number of Scanners

The CT scan unit at HDGH is operational for 13.25 hours per day with the exception of weekends during which it is only available for emergencies and inpatients in need of urgent scans. HDGH has 2 CT scanners, one having been added to the facility only a year ago.

4.2.3 Prebooked Periods

HDGH prebooks a significant number of appointment slots for class 2 priority patients. Of the 67 appointments available per day across both CT scanners, 13 are prebooked for class 2 patients; this represents nearly 20% of the total number of scans performed during a regular day. The class 2 patients that use these prebooked periods are typically inpatients but can also be lower acuity patients from the emergency room.

4.3 Acquired Data

Data was provided from HDGH's database of scheduling records. In years past data was only retained for 6 months after which point it was deleted. However, as a result of an increased interest in tracking performance, data has been retained from as far back as April 2007. The records used in this study were collected during the period of April 2007 to May 2009 by HDGH. It should be noted that data entry in the diagnostic imaging department of HDGH is a manual process. As such it is only as accurate as the person responsible for its entry. This is a limitation of the current data collection policy and procedure at the hospital.

The data that was used consisted of 17,689 medical records. Each record contained the following pertinent patient data:

- Medical Record Number
- Date that the request for service was made (scheduled)
- Date that the scan was performed
- The priority class of the patient

The wait / access time for each record was calculated by subtracting the date on which the appointment was scheduled from the date on which the scan was performed. Of the total data collected approximately 5% was unusable due to record keeping errors (failure to enter date of scheduling request) or difficulty in calculation (Microsoft Excel has difficulty accounting for leap years).

Figures 13 & 14 depict the arrival rate of requests and the average wait time for patients on a month by month, class by class basis.

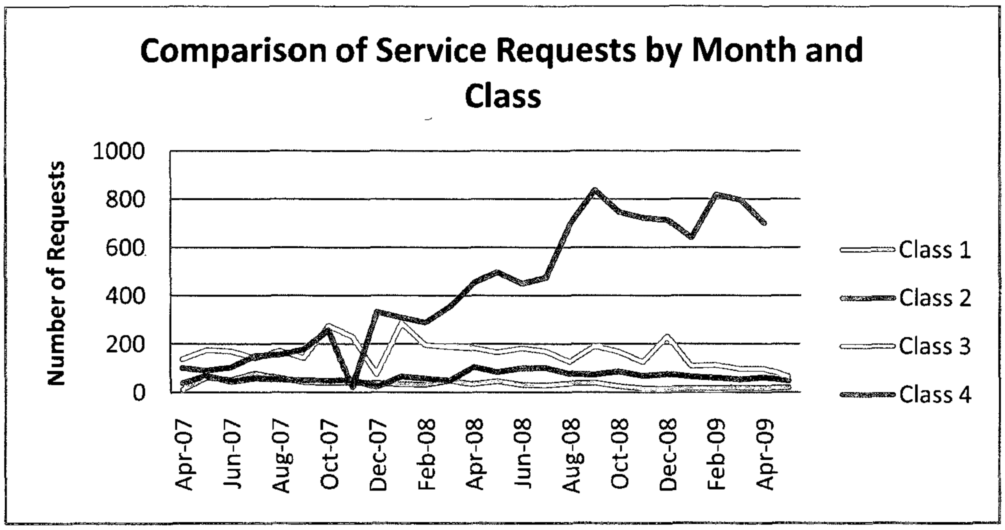


Figure 13 - Comparison of Service Request by Month and Class

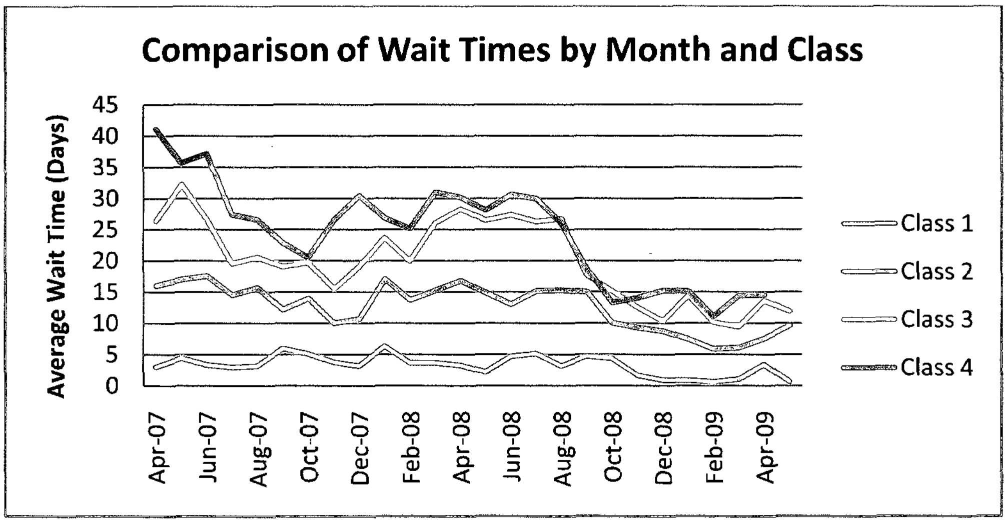


Figure 14 - Comparison of Wait Times by Month and Class

Figure 15 provides another comparison of the differences in volume between the patient classes.

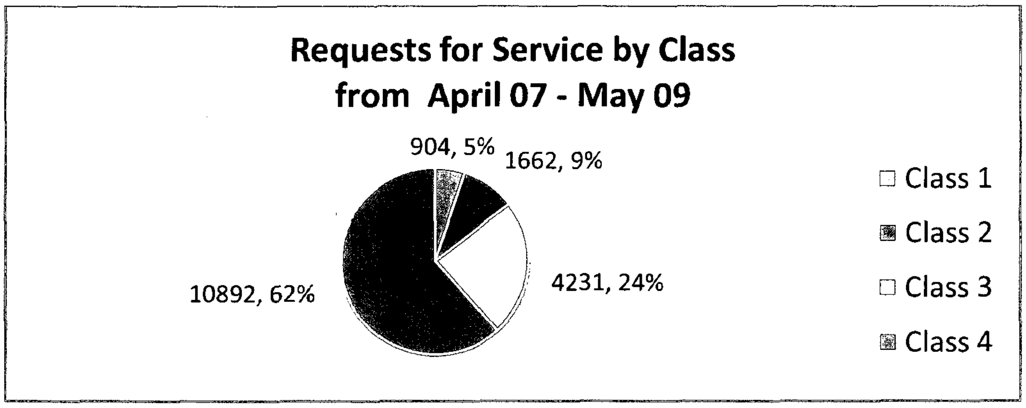


Figure 15 - Requests for Service by Class - April 07 to May 09

5. Verification and Validation

The verification and validation process has been described as one of the most important and difficult tasks in modelling (Banks et al, 2005). This chapter will describe the verification and validation process for the ABMT using data from HDGH.

Verification is described as building the model correctly, while validation ensures that the correct model is built. Verification asks: Is the model implemented correctly in the simulation software? And are the input parameters and logical structure of the model represented correctly? (Banks et al, 2005) In the case of the ABMT verification was achieved through modular development. Each of the ABMT's functionalities were created in different modules or sub segments of code; essentially each of the functions operate independently. This allowed for each function to be tested individually verifying that it was in fact behaving in the manner that the programmer intended. To ensure robustness each function was tested to its extremes; maximum and minimum arrival rates, extensive prebooked times etc. Additionally, the ABMT was constructed iteratively by a single programmer; this ensured that new work on the model was always based on previous work that had been verified to be correct.

Validation, according to Banks et al, attempts to confirm that a model is an accurate representation of a real system. This is accomplished in two ways; through consultation with those knowledgeable about the system being modelled and also through comparison of simulated and historical data. The ABMT was validated with the assistance of the CT scheduling clerk and historical data provided from HDGH. Ideally, the ABMT would be verified and validated using data from several different hospitals; unfortunately data was only available to the researchers from HDGH.

5.1 Simulation Parameters

In order to test the ABMT's ability to detect changing trends in the wait time it was configured to model a shift from a single server scenario to a dual server scenario. HDGH added a second CT scanner to their operation in November 2008, prior to that they had accumulated approximately 6 weeks of backlogged appointments.

To model this scenario a random fill value of 42 was used to fill the first 6 weeks of the schedule with booked appointments. Consultation with the CT scheduling clerk provided the information necessary to build a schedule that incorporated the prebooked time used at HDGH, see Figure 16 below.

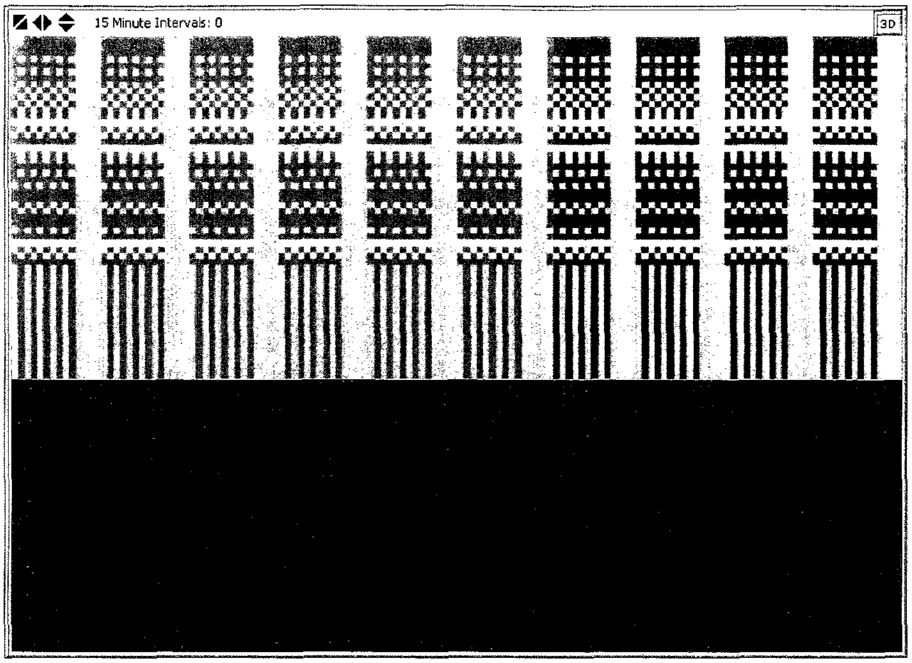


Figure 16 - HDGH Prebooked Schedule

The following parameters were used in the HDGH model and were derived from historical data and discussion with HDGH personnel:

Class 1 Patient Interarrival Time:	mean of 1440 min
Class 2 Patient Interarrival Time:	mean of 480 min
Class 3 Patient Interarrival Time:	mean of 240 min
Class 4 Patient Interarrival Time:	mean of 84 min
Scheduled Hours per Day:	13.25
Number of Servers:	2

As previously mentioned, exponential distributions were used to model the arrival rates. This was necessary because HDGH does not record the time at which a request for service is made; they only record the day that the request was made. The above means were used as the exponential parameters.

The ABMT was used to simulate 17 months of scheduling beginning at the point when the second CT scanner was added at HDGH to a point in the future when the system had reached a steady state. This point was determined by observing the output plots in the GUI. The simulation was run 25 times with nearly identical results each time. Each run took approximately 50 minutes. Figures 17-19 depict standard results for each of the patient classes from the simulation runs.

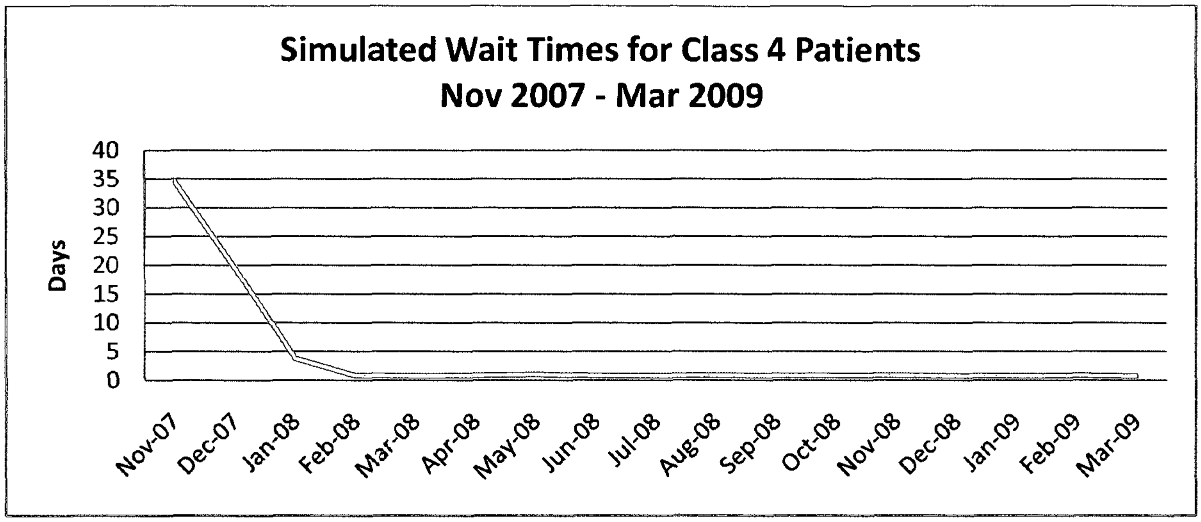


Figure 17 - Simulated Wait Times for Class 4 Patients

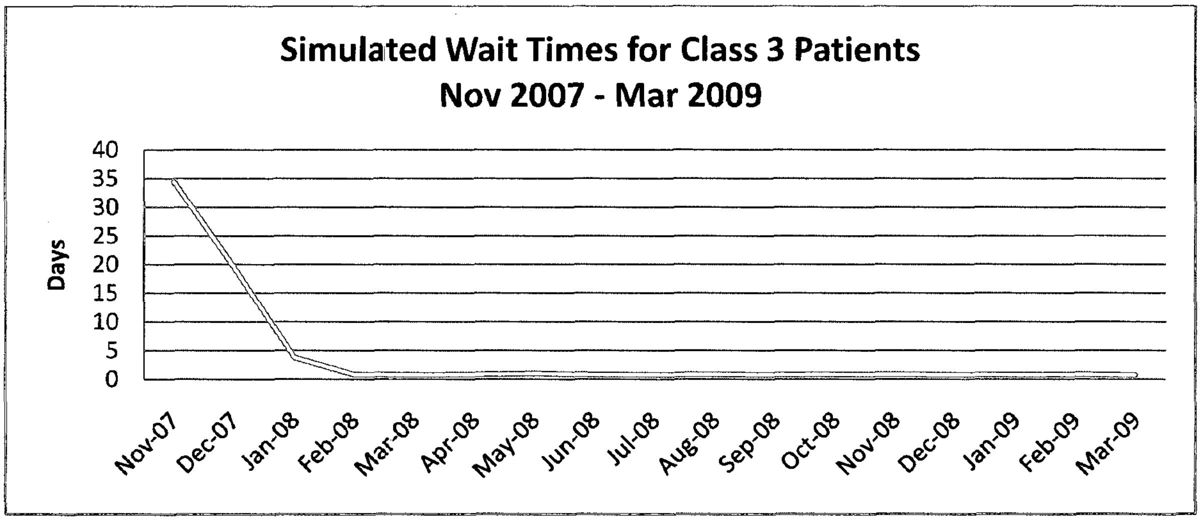


Figure 18 - Simulated Wait Times for Class 3 Patients

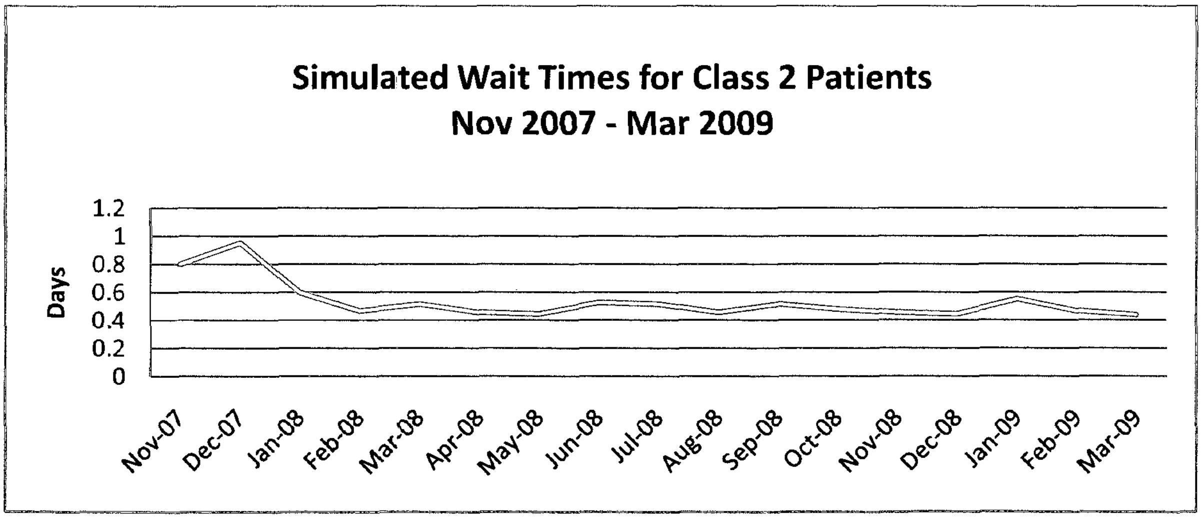


Figure 19 - Simulated Wait Times for Class 2 Patients

5.2 Discussion of Simulated vs. Historical Data

HDGH was a case study that served to highlight both the ABMT's strengths and weaknesses. The data that was available to validate the ABMT represented a time period during which the system transitioned from a one CT scanner unit to a two CT scanner unit, effectively doubling its capacity. This would appear to be an ideal situation in which to test the ability of the ABMT to predict the effect of the change on the scheduled patient queue. The ABMT was able to predict the decline in wait times however, it did not accurately predict the rate at which wait times would decrease. The reason for the discrepancy was unclear until the CT booking clerk at HDGH was consulted. She brought to light what was clear from the arrival data; that the number of requests for scans, especially amongst low priority patients, had increased dramatically after the addition of the second CT scanner. Her reasoning was that as physicians became aware that the wait time for a scan had decreased significantly they began to order scans for patients they may not have historically order them for. Additionally, it was the scheduler's suspicion that physicians were now also diverting patients from other area hospitals due to shorter wait times at HDGH and superior service. Finally, inpatients from other area hospitals were being transferred to HDGH to undergo scans and then being returned to their originating hospital.

This trend can clearly be seen when comparing the arrival rate of class 4 patients over time as the wait time for those patients decreased. Figure 20 shows the decreasing wait times for class 4 patients, while figure 21 shows the substantial increase in requests for scans for class 4 patients.

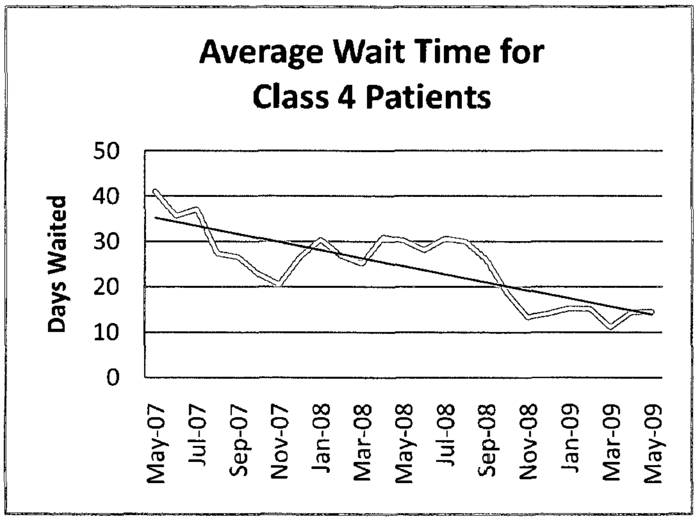


Figure 20 - Average Wait Time for Class 4 Patients

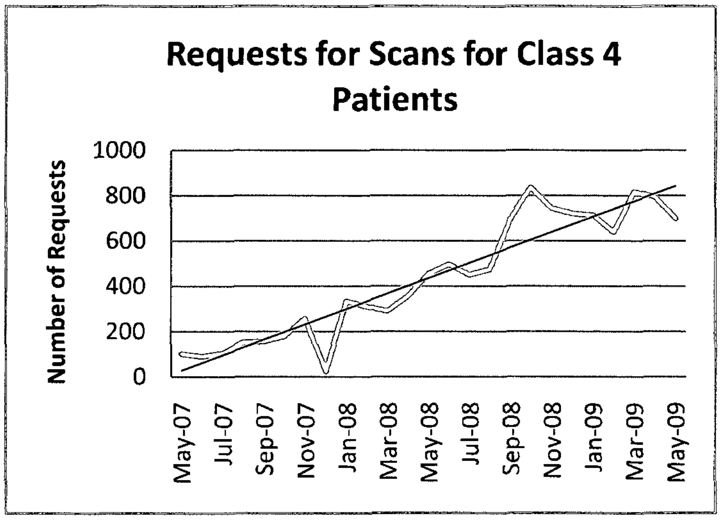


Figure 21 - Requests for Scans for Class 4 Patients

6. Discussion

The ABMT succeeded in achieving its primary objectives which were to provide hospital decision makers with insight into the effects that operational level changes would have to their systems via a graphical user interface. The ABMT was able to predict the new trend of decreasing wait times for patients at HDGH; information that would have been valuable to decision makers prior to the addition of the second server. The ABMT was not able however to accurately predict that rate at which wait times would decrease. While this may represent a weakness of the ABMT it does not represent a failure. The ABMT was designed to simulate the impact of changes to the system, not to the system's environment. It was not within the scope of the design to model the impact changes might have on the local network of CT scanners and their queues nor the psychological impact reduced wait times would have on the tendency of physicians to order CT scans. A potential extension of the ABMT may be to allow the user to dynamically increase or decrease the arrival rates based on the current wait times.

The ABMT usefulness lies in its ability to explore 'what-if' scenarios and provide insight into how changes might affect the wait time of patients. Questions one might consider using the ABMT to explore include:

- How will increasing or decreasing the number of servers impact the wait time for patients?
- How will extending or decreasing the number of operational hours per day impact the wait time of patients?
- What will be the impact on the schedule of increasing or decreasing the available prebooked times for each patient class?
- At what volume of patients will wait times begin to rise to unreasonable levels?
- Where should funds be invested to have the greatest impact on wait time? Increasing operating hours or adding servers?
- What impact will increasing the number of prebooked appointments for class W have on the wait times for classes X, Y and Z?

Essentially the number of scenarios that the ABMT can explore is limitless however it is most useful in the hands of a system expert who can use it to explore those scenarios that could potentially be of the most benefit to patients.

The decision to use simulation in the creation of the decision support tool appears to have been an acceptable choice. The user friendliness and customizability of the user interface and the simulation parameters proved invaluable in the presentation of the ABMT to hospital decision makers and helped to garner support for the project. The value of using agent-based modelling to create the decision support tool is debatable. NetLogo's™ two dimensional main display, which was used to show the schedule in the ABMT, was both a help and a hindrance. The main display was useful in that it provided users with real time insight into how the schedule was developing but using it to accurately describe a date, time and server proved unwieldy. Controlling the positioning of agents as they searched each column for available appointment slots was cumbersome as even slight variation in heading would result in agents assuming illogical appointment slots. Finally, the dynamic nature of the planning horizon made all positioning of agents and patches relative to the current date; keeping track of the current date and updating available appointment slots and prebooked time proved computationally intensive resulting in increasing slowdowns as the current time approached the end of the horizon. As a scheduling simulation tool NetLogo™ proved to be acceptable however the interaction between agents and agents and their environment, the core of agent-based modelling, were not used to their full potential by this application.

7. Conclusions and Future Work

This thesis resulted in the creation of an agent-based simulation tool with an easy to understand graphical user interface (GUI) that will allow hospital decision makers to assess the impact of potential operational level changes to the diagnostic imaging department on the department's schedule of patients. Additionally, this thesis served to expand the knowledge of the agent-based modelling in outpatient scheduling field.

The ABMT proved capable of detecting trends in patient wait times in a case study of HDGH's CT scanning unit and in the future could be used by the hospital to study its other diagnostic imaging services. While accomplishing its objective of providing hospital decision makers with a tool to assess the impact of internal changes the ABMT could be expanded, as a future endeavour, to consider the effect the changes might have on other diagnostic imaging providers in the local area. A tool modelling a network of diagnostic imaging centres in a given Local Health Integration Network (LHIN) may prove invaluable to decision makers responsible for the administration of services to hundreds of thousands of patients each year.

The ABMT has already been used to determine the point at which HDGH can expect wait times to begin increasing given the current trend in class 4 patient volume. The exploration of this question required minimal effort because of the flexibility of the ABMT; evaluating the effects of different patient volumes required only slight modification of the arrival rate of class 4 patients. In the future (if data becomes available from other area hospitals) it would be a worth while investigation to examine whether or not the addition of a second CT scanner at HDGH significantly impacted the wait times at other area hospitals. This would be a relatively simple task using the ABMT; one would only need to configure the ABMT to model another area hospital (approx. 15 minutes of setup) using data collected before the addition of the second CT scanner at HDGH. A difference between the simulated and historical data at hospitals not undergoing major changes to their diagnostic imaging departments could be in part attributed to the changes at HDGH. This information would be useful in determining the impact of adding a CT scanner to the LHIN as a whole.

The ABMT is a novel application of agent-based modelling to outpatient scheduling. The development of the ABMT served to highlight some of the challenges of using a tool designed for dynamic, evolutionary behaviour in an environment based on the precise coordination of thousands of individuals. While outpatient scheduling may not have taken full advantage of the ability of agents to interact with one another, a model of a schedule developed by patients (as opposed to one organized by a single human clerk) may prove useful in understanding the preferences of patients and assist in the allocation of resources to better serve them.

The ABMT has already generated interest from researchers in fields outside of diagnostic imaging. Researchers in the field of radiation therapy have expressed interest in the ABMT has a potential scheduling modelling tool as radiation therapy and diagnostic imaging services share many similar scheduling traits.

Working with healthcare professionals to develop the ABMT provided unique insight into the complex nature of healthcare systems. The shift in patient arrival rates as explained by the CT scheduling clerk brought to light the reality that healthcare is unlike any other industry and that the application of industrial engineering techniques here will require an understanding of the healthcare system as a whole, not just isolated elements. To that end this thesis will be used as a basis for a peer-reviewed journal article so that the knowledge gained here can be shared with other industrial engineering practitioners in healthcare to help to develop the whole system knowledge required to make a meaningful impact in the lives of healthcare professionals and patients.

Bibliography

Aktas, E., Ulengin, F., & Sahin, S. (2007). A decision support system to improve the efficiency of resource allocation in healthcare management. *Socio-Economic Planning Sciences , 41*, 130-146.

Axelrod, R. (2005). AGENT-BASED MODELING AS A BRIDGE BETWEEN DISCIPLINES. In R. Axelrod, K. Judd, & L. Tesfatsion (Eds.), *Handbook of Computational Economics,Vol. 2: Agent-Based Computational Economics.*

Bailey, N. (1952). A study of queues and appointment systems in hospital outpatient departments, with special reference to waiting times. *J Roy Stat Soc , A15*, 185-199.

Banks et al. (2005). *Discrete-Event System Simulation* (4th ed.). Prentice Hall.

Blake, J., Carter, M., & Richardson, S. (1996). AN ANALYSIS OF EMERGENCY ROOM WAIT TIME ISSUES VIA COMPUTER SIMULATION. *INFOR , 34* (4), 263-273.

Blanco White, M., & Pike, M. Appointment systems in Out-patients' Clinics and the Effect of Patients' Unpunctuality. *Medical Care , 2* (3), 133-145.

Borshchev, A., & Filippov, A. (2004). From System Dynamics and Discrete Event to Practical Agent Based Modelling: Reasons, Techniques, Tools. *22nd International Conference of the System Dynamics Society.* Keble College, Oxford.

Butler, T., Karwan, K., & Sweigart, J. (1992). Multi-Level Strategic Evaluation of Hospital Plans and Decisions. *The Journal of the Operational Research Society , 43* (7), 665-675.

Canadian Institue for Health Services. (2007). *National Survey of Selected Medical Imaging Equipment.*

Canadian Institute for Health Information. (2008). *Health Care in Canada 2008.* Ottawa, Ont.: CIHI.

Canadian Institute for Health Information. (2007). *Medical Imaging in Canada.* Ottawa, Ont: CIHI.

Canadian Institute for Health Information. (2004). *National Survey of Selected Medical Imaging Equipment.*

Canadian Institute for Health Services. (2007). *National Survey of Selected Medical Imaging Equipment.*

Cayirli, T., & Veral, E. (2003). Outpatient scheduling in health care: a review of literature. *Production and Operations Management Society , 12* (4), 519-549.

Cayirli, T., Veral, E., & Rosen, H. (2004). Assessment of patient classification in appointment systems. *1st Conference of the POMS College of Service Operations.* New York, NY, USA.

Cayirli, T., Veral, E., & Rosen, H. (2006). Designing appointment scheduling systems for ambulatory care services. *Health Care Management Science , 9* (1), 47-58.

Chen, R., & Robinson, L. (2005). *Scheduling doctor's appointments with unpunctual patient arrivals.* Davis Graduate School of Management, University of California, USA.: Working paper,.

Clague, J., Reed, P., Barlow, J., Rada, R., Clarke, M., & Edwards, R. (1997). Improving outpatient clinic efficiency using computer simulation. *International Journal of Health Care Quality Assurance , 10* (5), 197-201.

Coyle, R. G. (1996). *System dynamics modelling: a practical approach.* CRC Press.

Denton, B., & Gupta, D. (2003). A Sequential Bounding Approach for Optimal Appointment Scheduling. *IIE Transactions , 35* (11), 1003-1016.

Devardo, E. (2003). *Dynamic Programming: Models and Applications.* Prentice Hall.

Dumas, M. (1985). Hospital bed utilization: an implemented simulation approach to adjusting and maintaining appropriate levels. *Health Services Research , 20* (1), 43-61.

Everett, J. (2002). A Decision Support Simulation Model for the Management of an Elective Surgery Waiting Model. *Health Care Management Science , 5*, 89–95.

Fetter, R., & Thompson, J. (1966). Patients' waiting time and doctors' idle time in the outpatient setting. *Health Services Research , 1* (1), 66-90.

Fries, B., & Marathe, B. (1981). Determination of optimal variable-sized multiple-block appointment systems. *Operations Research , 29* (2), 324-345.

Garcia et al. (1995). REDUCING TIME IN AN EMERGENCY ROOM VIA A FAST-TRACK. *Proceedings of 1995 Winter Simulation Conference*, (pp. 1048-1053).

Harper, P., & Gamlin, H. (2003). Reduced outpatient waiting times with improved appointmentscheduling: a simulation modelling approach. *OR Spectrum , 5* (2), 207-222.

Ho, C., Lau, H., & Li, J. (1995). Introducing variable-interval appointment scheduling rules in service systems. *International Journal of Production & Operations Management , 15* (6), 59-68.

http://www.health.gov.on.ca. (n.d.). *Ontario Wait Times Strategy: Introduction.* Retrieved April 29, 2009, from http://www.health.gov.on.ca/transformation/wait_times/public/wt_public_mn.html

Hutzschenreuter, A. (2004). *Waiting Patiently: An analysis of the performance aspects of outpatient scheduling in health care institutes.* Vrije Universiteit, Amsterdam, The Netherlands.

Jun, J., Jacobson, S., & Swisher, J. (1999). Application of discrete-event simulation in healthcare clinics: A survey. *Journal of the Operational Research Society , 50* (2), 109-123.

Kaandorp, G., & Koole, G. (2007). Optimal outpatient appointment scheduling. *Health Care Management Science , 10* (3), 217-229.

Kendall, D. (1951). Some Problems in the Theory of Queues. *Journal of the Royal Statistical Society* , 151-185.

Klassen, K., & Rohleder, T. (2004). Outpatient appointment scheduling with urgent clients in a dynamic, multi-period environment. *International Journal of Service Industry Management* , *15* (2), 167-186.

Klassen, K., & Rohleder, T. (1996). Scheduling outpatient appointments in a dynamic environment. *Journal of Operations Management* , *14* (2), 83-101.

Lehaney, B., Clarke, S., & Paul, R. (1999). A case of intervention in an outpatient department. *Journal of the Operational Research Society* , *50* (9), 877-891.

Liu, L., & Liu, X. (1998). Block appointment systems for outpatient clinics with multiple doctors. *Journal of the Operational Research Society* , *49*, 1254-1259.

Lowery, J. (1992). Simulation of a hospital's surgical suite and critical care area. *Proceedings of the 24th conference on Winter simulation*, (pp. 1071-1078).

Macal, C. M., & North, M. J. (2007). AGENT-BASED MODELING AND SIMULATION: DESKTOP ABMS. *Proceedings of the 2007 Winter Simulation Conference*, (pp. 95-106).

McClean, S., & Millard, P. (1995). A decision support system for bed-occupancy management and planning hospitals. *Journal of Mathematics Applied in Medicine & Biology* , *12*, 249-257.

McGuire, F. (1994). USING SIMULATION TO REDUCE LENGTH OF STAY IN EMERGENCY DEPARTMENTS. *Proceedings of the 1994 Winter Simulation Conference*, (pp. 861-867).

Ministry of Health. (2005). *Canada's Healthcare System.* HC Pub.: 5912.

NetLogo Website. (n.d.). *NetLogo Manual: What is NetLogo.* Retrieved 06 01, 2009, from NetLogo Home Page: http://ccl.northwestern.edu/netlogo/docs/

Robinson, L., & Chen, R. (2003). Scheduling doctors' appointments: optimal and empirically-based heuristic policies. *IIE Transactions* , *35*, 298-307.

Rohleder, T., & Klassen, K. (2000). Using client-variance information to improve dynamic appointment scheduling performance. *Omega* , *28* (3), 293-302.

Schriber, T., & Brunner, D. (1997). INSIDE DISCRETE-EVENT SIMULATION SOFTWARE: HOW IT WORKS AND WHY. In S. Andradóttir, K. J. Healy, D. H. Withers, & B. L. Nelson (Ed.), *Proceedings of the 1997 Winter Simulation Conference.*

Shannon, R. (1992). Introduction to Simulation. In D. G. J. J. Swain (Ed.), *Proceedings of the 1992 Winter Simulation Conference* (pp. 65-73). ACM New York, NY, USA.

Smith, S., Schroer, B., & Shannon, R. (1979). Scheduling of patients and resources for ambulatory healthcare. *Proceedings of the 11th conference on Winter simulatio. 2*, pp. 553-561. IEEE Press Piscataway, NJ, USA.

Teweldemedhin, E., Marwala, T., & Mueller, C. (2004). Agent-based Modelling: A Case Study in HIV Epidemic. *Fourth International Conference on Hybrid Intelligent Systems* , (pp. 154-159).

Triola, M., & Holzman, R. (2003). Agent-Based Simulation of Nosocomial Transmission in the Medical Intensive Care Unit. *Computer-Based Medical Systems, Proceedings. 16th IEEE Symposium*, (pp. 284-288).

Turcotte, M., & Schellenberg, G. (2006). *A Portrait of Seniors in Canada.* Ottawa: Statistics Canada.

Vanden Bosch, P., Dietz, D., & Simeoni, J. (1999). Scheduling Customer Arrivals to a Stochastic Service System. *Naval Research Logistics , 46*, 549-559.

Vissers, J., & Wijngaard, J. (1979). The outpatient appointment system: design of a simulation study. *European Journal of Operations Research , 3* (6), 459-463.

Welch, J., & Bailey, N. (1952). Appointment systems in Hospital Outpatient Departments. *The Lancet , 259*, 1105-1108.

Westeneng, J. (2007). *Outpatient appointment scheduling: An evaluation of alternative appointment systems to reduce waiting times and underutilization in an ENT outpatient clinic.* University of Twente, Industrial Engineering and Management, Enschede, The Netherlands.

Winston, W. L. (2004). *Operations Research: Applications and Algorithms.* Thomson.

Appendix I: NetLogo™ Code for ABMT

```
;;;;;;;;;;;;;; -=GLOBAL VARIABLES=- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
globals [n count1 count2 count3 count4 count1over count2over count3over count4over class2total
class3total class4total MS D G F B
bump_counter_1 bump_counter_2 bump_counter_3 Arrival_Rate_Counter_1 Arrival_Rate_Counter_2
Arrival_Rate_Counter_3 Arrival_Rate_Counter_4
Normal_Holder_1 Exponential_Holder_1 Poisson_Holder_1 Normal_Holder_2 Exponential_Holder_2
Poisson_Holder_2 Normal_Holder_3 Exponential_Holder_3 Poisson_Holder_3
Normal_Holder_4 Exponential_Holder_4 Poisson_Holder_4 tick_counter]


;; Globals are variables passed throughout the program
breed [class-1-patients patient-1] ;;;green
breed [class-2-patients patient-2] ;;;blue
breed [class-3-patients patient-3] ;;;yellow
breed [class-4-patients patient-4] ;;;brown
breed [updaters update-1]          ;;;cyan
;; Breeds used to call patients by group
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;; -=SETUP=- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Setup clears all data from previous runs and initializes all variables.
to setup
 clear-all
 setup-patches
 create-time-updater

 set count1 0
 set count2 0
 set count3 0
 set count4 0
 set count1over 0
 set count2over 0
 set count3over 0
 set count4over 0
 set class2total 0
 set class3total 0
 set class4total 0
 set B 0
 set Arrival_Rate_Counter_1 0
 set Arrival_Rate_Counter_2 0
 set Arrival_Rate_Counter_3 0
 set Arrival_Rate_Counter_4 0
 if Record [file-open "output_data.xls"]
 carefully [file-print date-and-time]
 [print "Please close output file"]

 if Daily_Prebooked ;; Checks to see if the Daily_Prebooked switch is on and blocks off time accordingly.
 [ask patches
  [ if pycor <= Daily_Prebook_Start * -4 and pycor >=  Daily_Prebook_End * -4  [ set pcolor grey ] ]
 ]

if Weekly_Prebooked   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_1

   [
   if pycor <= Weekly_Prebook_Start * -4 and pycor >=  Weekly_Prebook_End * -4
       and pxcor = Servers_Booked_1 + n - 1
    [if Weekly_Prebooked_1_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_1_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_1_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_1_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start * -4 and pycor >=  Weekly_Prebook_End * -4
       and pxcor = (n + Servers_booked_1 - 1 + (Day_of_the_week - 1) * (number_of_servers) )
    [if Weekly_Prebooked_1_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_1_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_1_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_1_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
   ]
 ]
 ]

if Weekly_Prebooked_2   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_2

   [
   if pycor <= Weekly_Prebook_Start_2 * -4 and pycor >=  Weekly_Prebook_End_2 * -4
       and pxcor = Servers_Booked_2 + n - 1
    [if Weekly_Prebooked_2_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_2_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_2_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_2_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_2 * -4 and pycor >=  Weekly_Prebook_End_2 * -4
       and pxcor = (n + Servers_booked_2 - 1 + (Day_of_the_week_2 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_2_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_2_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_2_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_2_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]


if Weekly_Prebooked_3   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_3

   [
   if pycor <= Weekly_Prebook_Start_3 * -4 and pycor >=  Weekly_Prebook_End_3 * -4
       and pxcor = Servers_Booked_3 + n - 1
    [if Weekly_Prebooked_3_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_3_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_3_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_3_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]
    [
     if pycor <= Weekly_Prebook_Start_3 * -4 and pycor >=  Weekly_Prebook_End_3 * -4
       and pxcor = (n + Servers_booked_3 - 1 + (Day_of_the_week_3 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_3_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_3_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_3_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_3_Class = "Class 4" [set pcolor 37] ]
    set n n + number_of_servers * 7
    ]
  ]
 ]
 ]


if Weekly_Prebooked_4   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_4

   [
   if pycor <= Weekly_Prebook_Start_4 * -4 and pycor >=  Weekly_Prebook_End_4 * -4
       and pxcor = Servers_Booked_4 + n - 1
   [if Weekly_Prebooked_4_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_4_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_4_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_4_Class = "Class 4" [set pcolor 37] ]
    set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_4 * -4 and pycor >=  Weekly_Prebook_End_4 * -4
       and pxcor = (n + Servers_booked_4 - 1 + (Day_of_the_week_4 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_4_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_4_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_4_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_4_Class = "Class 4" [set pcolor 37] ]
    set n n + number_of_servers * 7
    ]
  ]
 ]
 ]
```

```
if Weekly_Prebooked_5  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_5

   [
   if pycor <= Weekly_Prebook_Start_5 * -4 and pycor >=  Weekly_Prebook_End_5 * -4
        and pxcor = Servers_Booked_5 + n - 1
    [if Weekly_Prebooked_5_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_5_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_5_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_5_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
     ]

     [
      if pycor <= Weekly_Prebook_Start_5 * -4 and pycor >=  Weekly_Prebook_End_5 * -4
        and pxcor = (n + Servers_booked_5 - 1 + (Day_of_the_week_5 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_5_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_5_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_5_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_5_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
     ]
   ]
 ]
 ]


if Weekly_Prebooked_6  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_6

   [
   if pycor <= Weekly_Prebook_Start_6 * -4 and pycor >=  Weekly_Prebook_End_6 * -4
        and pxcor = Servers_Booked_6 + n - 1
   [if Weekly_Prebooked_6_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_6_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_6_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_6_Class = "Class 4" [set pcolor 37] ]
    set n  n + Number_of_servers
   ]

   [
    if pycor <= Weekly_Prebook_Start_6 * -4 and pycor >=  Weekly_Prebook_End_6 * -4
      and pxcor = (n + Servers_booked_6 - 1 + (Day_of_the_week_6 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_6_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_6_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_6_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_6_Class = "Class 4" [set pcolor 37] ]
    set n n + number_of_servers * 7
   ]
  ]
 ]
 ]


if Weekly_Prebooked_7   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
  [

  ifelse repeat_daily_7

  [
  if pycor <= Weekly_Prebook_Start_7 * -4 and pycor >=  Weekly_Prebook_End_7 * -4
      and pxcor = Servers_Booked_7 + n - 1
   [if Weekly_Prebooked_7_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_7_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_7_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_7_Class = "Class 4" [set pcolor 37] ]
    set n  n + Number_of_servers
   ]

   [
    if pycor <= Weekly_Prebook_Start_7 * -4 and pycor >=  Weekly_Prebook_End_7 * -4
      and pxcor = (n + Servers_booked_7 - 1 + (Day_of_the_week_7 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_7_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_7_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_7_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_7_Class = "Class 4" [set pcolor 37] ]
```

```
      set n n + number_of_servers * 7
      ]
    ]
  ]
  ]

if Weekly_Prebooked_8   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_8

   [
   if pycor <= Weekly_Prebook_Start_8 * -4 and pycor >=  Weekly_Prebook_End_8 * -4
       and pxcor = Servers_Booked_8 + n - 1
    [if Weekly_Prebooked_8_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_8_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_8_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_8_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
     ]

     [
      if pycor <= Weekly_Prebook_Start_8 * -4 and pycor >=  Weekly_Prebook_End_8 * -4
        and pxcor = (n + Servers_booked_8 - 1 + (Day_of_the_week_8 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_8_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_8_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_8_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_8_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
     ]
   ]
  ]
  ]


if Weekly_Prebooked_9   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [
   ifelse repeat_daily_9

   [
   if pycor <= Weekly_Prebook_Start_9 * -4 and pycor >=  Weekly_Prebook_End_9 * -4
       and pxcor = Servers_Booked_9 + n - 1
    [if Weekly_Prebooked_9_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_9_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_9_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_9_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_9 * -4 and pycor >=  Weekly_Prebook_End_9 * -4
       and pxcor = (n + Servers_booked_9 - 1 + (Day_of_the_week_9 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_9_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_9_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_9_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_9_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]

if Weekly_Prebooked_10   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
  [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
  [set n 0
  while [n <= max-pxcor]
   [

   ifelse repeat_daily_10

   [
   if pycor <= Weekly_Prebook_Start_10 * -4 and pycor >=  Weekly_Prebook_End_10 * -4
       and pxcor = Servers_Booked_10 + n - 1
    [if Weekly_Prebooked_10_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_10_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_10_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_10_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_10 * -4 and pycor >=  Weekly_Prebook_End_10 * -4
       and pxcor = (n + Servers_booked_10 - 1 + (Day_of_the_week_10 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_10_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_10_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_10_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_10_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
   ]
  ]
  ]

if Weekly_Prebooked_11   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_11

   [
   if pycor <= Weekly_Prebook_Start_11 * -4 and pycor >=  Weekly_Prebook_End_11 * -4
       and pxcor = Servers_Booked_11 + n - 1
    [if Weekly_Prebooked_11_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_11_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_11_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_11_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_11 * -4 and pycor >=  Weekly_Prebook_End_11 * -4
       and pxcor = (n + Servers_booked_11 - 1 + (Day_of_the_week_11 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_11_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_11_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_11_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_11_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
   ]
  ]
  ]

if Weekly_Prebooked_12   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
```

```
while [n <= max-pxcor]
  [

  ifelse repeat_daily_12

  [
  if pycor <= Weekly_Prebook_Start_12 * -4 and pycor >=  Weekly_Prebook_End_12 * -4
       and pxcor = Servers_Booked_12 + n - 1
   [if Weekly_Prebooked_12_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_12_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_12_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_12_Class = "Class 4" [set pcolor 37] ]
    set n  n + Number_of_servers
   ]

   [
    if pycor <= Weekly_Prebook_Start_12 * -4 and pycor >=  Weekly_Prebook_End_12 * -4
       and pxcor = (n + Servers_booked_12 - 1 + (Day_of_the_week_12 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_12_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_12_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_12_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_12_Class = "Class 4" [set pcolor 37] ]
    set n n + number_of_servers * 7
   ]
 ]
]
]

if Weekly_Prebooked_13   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
  [

  ifelse repeat_daily_13

  [
  if pycor <= Weekly_Prebook_Start_13 * -4 and pycor >=  Weekly_Prebook_End_13 * -4
       and pxcor = Servers_Booked_13 + n - 1
   [if Weekly_Prebooked_13_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_13_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_13_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_13_Class = "Class 4" [set pcolor 37] ]
    set n  n + Number_of_servers
   ]
```

```
    [
     if pycor <= Weekly_Prebook_Start_13 * -4 and pycor >=  Weekly_Prebook_End_13 * -4
       and pxcor = (n + Servers_booked_13 - 1 + (Day_of_the_week_13 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_13_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_13_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_13_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_13_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]

if Weekly_Prebooked_14   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_14

   [
   if pycor <= Weekly_Prebook_Start_14 * -4 and pycor >=  Weekly_Prebook_End_14 * -4
       and pxcor = Servers_Booked_14 + n - 1
    [if Weekly_Prebooked_14_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_14_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_14_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_14_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_14 * -4 and pycor >=  Weekly_Prebook_End_14 * -4
       and pxcor = (n + Servers_booked_14 - 1 + (Day_of_the_week_14 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_14_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_14_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_14_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_14_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]

if Weekly_Prebooked_15   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
```

```
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_15

   [
   if pycor <= Weekly_Prebook_Start_15 * -4 and pycor >=  Weekly_Prebook_End_15 * -4
        and pxcor = Servers_Booked_15 + n - 1
    [if Weekly_Prebooked_15_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_15_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_15_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_15_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_15 * -4 and pycor >=  Weekly_Prebook_End_15 * -4
        and pxcor = (n + Servers_booked_15 - 1 + (Day_of_the_week_15 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_15_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_15_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_15_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_15_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]

if Weekly_Prebooked_16   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_16

   [
   if pycor <= Weekly_Prebook_Start_16 * -4 and pycor >=  Weekly_Prebook_End_16 * -4
        and pxcor = Servers_Booked_16 + n - 1
    [if Weekly_Prebooked_16_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_16_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_16_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_16_Class = "Class 4" [set pcolor 37] ]
```

```
      set n  n + Number_of_servers
      ]

      [
       if pycor <= Weekly_Prebook_Start_16 * -4 and pycor >=  Weekly_Prebook_End_16 * -4
          and pxcor = (n + Servers_booked_16 - 1 + (Day_of_the_week_16 - 1) * (number_of_servers) )
     [if Weekly_Prebooked_16_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_16_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_16_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_16_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
      ]
    ]
  ]
  ]
 if Weekly_Prebooked_17   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
  [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
  [set n 0
  while [n <= max-pxcor]
    [

    ifelse repeat_daily_17

    [
    if pycor <= Weekly_Prebook_Start_17 * -4 and pycor >=  Weekly_Prebook_End_17 * -4
          and pxcor = Servers_Booked_17 + n - 1
     [if Weekly_Prebooked_17_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_17_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_17_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_17_Class = "Class 4" [set pcolor 37] ]
      set n  n + Number_of_servers
      ]

      [
       if pycor <= Weekly_Prebook_Start_17 * -4 and pycor >=  Weekly_Prebook_End_17 * -4
          and pxcor = (n + Servers_booked_17 - 1 + (Day_of_the_week_17 - 1) * (number_of_servers) )
     [if Weekly_Prebooked_17_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_17_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_17_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_17_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
      ]
    ]
  ]
  ]
```

```
if Weekly_Prebooked_18   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_18

   [
   if pycor <= Weekly_Prebook_Start_18 * -4 and pycor >=  Weekly_Prebook_End_18 * -4
       and pxcor = Servers_Booked_18 + n - 1
    [if Weekly_Prebooked_18_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_18_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_18_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_18_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_18 * -4 and pycor >=  Weekly_Prebook_End_18 * -4
       and pxcor = (n + Servers_booked_18 - 1 + (Day_of_the_week_18 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_18_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_18_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_18_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_18_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]
 if Weekly_Prebooked_19   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_19

   [
   if pycor <= Weekly_Prebook_Start_19 * -4 and pycor >=  Weekly_Prebook_End_19 * -4
       and pxcor = Servers_Booked_19 + n - 1
    [if Weekly_Prebooked_19_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_19_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_19_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_19_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_19 * -4 and pycor >=  Weekly_Prebook_End_19 * -4
       and pxcor = (n + Servers_booked_19 - 1 + (Day_of_the_week_19 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_19_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_19_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_19_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_19_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]
 if Weekly_Prebooked_20   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_20

   [
   if pycor <= Weekly_Prebook_Start_20 * -4 and pycor >=  Weekly_Prebook_End_20 * -4
       and pxcor = Servers_Booked_20 + n - 1
    [if Weekly_Prebooked_20_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_20_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_20_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_20_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_20 * -4 and pycor >=  Weekly_Prebook_End_20 * -4
       and pxcor = (n + Servers_booked_20 - 1 + (Day_of_the_week_20 - 1) * (number_of_servers) )
   [if Weekly_Prebooked_20_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_20_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_20_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_20_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]
 if Weekly_Prebooked_21  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches       ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_21

   [
   if pycor <= Weekly_Prebook_Start_21 * -4 and pycor >=  Weekly_Prebook_End_21 * -4
       and pxcor = Servers_Booked_21 + n - 1
    [if Weekly_Prebooked_21_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_21_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_21_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_21_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
     ]

    [
     if pycor <= Weekly_Prebook_Start_21 * -4 and pycor >=  Weekly_Prebook_End_21 * -4
       and pxcor = (n + Servers_booked_21 - 1 + (Day_of_the_week_21 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_21_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_21_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_21_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_21_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
     ]
   ]
  ]
  ]

 if Weekly_Prebooked_22  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches       ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_22

   [
   if pycor <= Weekly_Prebook_Start_22 * -4 and pycor >=  Weekly_Prebook_End_22 * -4
       and pxcor = Servers_Booked_22 + n - 1
    [if Weekly_Prebooked_22_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_22_Class = "Class 2" [set pcolor 107]
```

```
      if Weekly_Prebooked_22_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_22_Class = "Class 4" [set pcolor 37] ]
      set n  n + Number_of_servers
     ]

     [
      if pycor <= Weekly_Prebook_Start_22 * -4 and pycor >=  Weekly_Prebook_End_22 * -4
        and pxcor = (n + Servers_booked_22 - 1 + (Day_of_the_week_22 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_22_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_22_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_22_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_22_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
     ]
    ]
  ]
  ]

  if Weekly_Prebooked_23   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
  [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
  [set n 0
  while [n <= max-pxcor]
    [

    ifelse repeat_daily_23

    [
    if pycor <= Weekly_Prebook_Start_23 * -4 and pycor >=  Weekly_Prebook_End_23 * -4
        and pxcor = Servers_Booked_23 + n - 1
     [if Weekly_Prebooked_23_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_23_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_23_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_23_Class = "Class 4" [set pcolor 37] ]
      set n  n + Number_of_servers
     ]

     [
      if pycor <= Weekly_Prebook_Start_23 * -4 and pycor >=  Weekly_Prebook_End_23 * -4
        and pxcor = (n + Servers_booked_23 - 1 + (Day_of_the_week_23 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_23_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_23_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_23_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_23_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
     ]
    ]
```

```
 ]
 ]

 if Weekly_Prebooked_24  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_24

   [
   if pycor <= Weekly_Prebook_Start_24 * -4 and pycor >=  Weekly_Prebook_End_24 * -4
       and pxcor = Servers_Booked_24 + n - 1
    [if Weekly_Prebooked_24_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_24_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_24_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_24_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
     ]

    [
     if pycor <= Weekly_Prebook_Start_24 * -4 and pycor >=  Weekly_Prebook_End_24 * -4
       and pxcor = (n + Servers_booked_24 - 1 + (Day_of_the_week_24 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_24_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_24_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_24_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_24_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
     ]
   ]
 ]
 ]

 if Weekly_Prebooked_25  ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_25

   [
   if pycor <= Weekly_Prebook_Start_25 * -4 and pycor >=  Weekly_Prebook_End_25 * -4
```

```
        and pxcor = Servers_Booked_25 + n - 1
    [if Weekly_Prebooked_25_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_25_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_25_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_25_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
      if pycor <= Weekly_Prebook_Start_25 * -4 and pycor >=  Weekly_Prebook_End_25 * -4
        and pxcor = (n + Servers_booked_25 - 1 + (Day_of_the_week_25 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_25_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_25_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_25_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_25_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
   ]
  ]
  ]

 if Weekly_Prebooked_26   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_26

   [
   if pycor <= Weekly_Prebook_Start_26 * -4 and pycor >=  Weekly_Prebook_End_26 * -4
        and pxcor = Servers_Booked_26 + n - 1
    [if Weekly_Prebooked_26_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_26_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_26_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_26_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
      if pycor <= Weekly_Prebook_Start_26 * -4 and pycor >=  Weekly_Prebook_End_26 * -4
        and pxcor = (n + Servers_booked_26 - 1 + (Day_of_the_week_26 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_26_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_26_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_26_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_26_Class = "Class 4" [set pcolor 37] ]
```

```
        set n n + number_of_servers * 7
        ]
      ]
    ]
    ]

 if Weekly_Prebooked_27   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_27

   [
   if pycor <= Weekly_Prebook_Start_27 * -4 and pycor >=  Weekly_Prebook_End_27 * -4
       and pxcor = Servers_Booked_27 + n - 1
    [if Weekly_Prebooked_27_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_27_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_27_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_27_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
     ]

    [
     if pycor <= Weekly_Prebook_Start_27 * -4 and pycor >=  Weekly_Prebook_End_27 * -4
       and pxcor = (n + Servers_booked_27 - 1 + (Day_of_the_week_27 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_27_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_27_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_27_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_27_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
     ]
   ]
 ]
 ]

 if Weekly_Prebooked_28   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_28
   [
   if pycor <= Weekly_Prebook_Start_28 * -4 and pycor >=  Weekly_Prebook_End_28 * -4
        and pxcor = Servers_Booked_28 + n - 1
    [if Weekly_Prebooked_28_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_28_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_28_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_28_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_28 * -4 and pycor >=  Weekly_Prebook_End_28 * -4
        and pxcor = (n + Servers_booked_28 - 1 + (Day_of_the_week_28 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_28_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_28_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_28_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_28_Class = "Class 4" [set pcolor 37] ]
     set n n + number_of_servers * 7
    ]
  ]
 ]
 ]

 if Weekly_Prebooked_29   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches         ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_29

   [
   if pycor <= Weekly_Prebook_Start_29 * -4 and pycor >=  Weekly_Prebook_End_29 * -4
        and pxcor = Servers_Booked_29 + n - 1
    [if Weekly_Prebooked_29_Class = "None" [set pcolor grey]
     if Weekly_Prebooked_29_Class = "Class 2" [set pcolor 107]
     if Weekly_Prebooked_29_Class = "Class 3" [set pcolor 47]
     if Weekly_Prebooked_29_Class = "Class 4" [set pcolor 37] ]
     set n  n + Number_of_servers
    ]

    [
     if pycor <= Weekly_Prebook_Start_29 * -4 and pycor >=  Weekly_Prebook_End_29 * -4
        and pxcor = (n + Servers_booked_29 - 1 + (Day_of_the_week_29 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_29_Class = "None" [set pcolor grey]
```

```
      if Weekly_Prebooked_29_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_29_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_29_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
     ]
   ]
  ]
  ]

 if Weekly_Prebooked_30   ;; Checks to see if the Weekly_Prebooked switch is on and blocks off time
accordingly.
 [ask patches        ;; This loops asks patches to check if they are between the prebook start and end and
if they are on the day of the week in question.
 [set n 0
 while [n <= max-pxcor]
   [

   ifelse repeat_daily_30

   [
   if pycor <= Weekly_Prebook_Start_30 * -4 and pycor >=  Weekly_Prebook_End_30 * -4
        and pxcor = Servers_Booked_30 + n - 1
    [if Weekly_Prebooked_30_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_30_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_30_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_30_Class = "Class 4" [set pcolor 37] ]
      set n  n + Number_of_servers
     ]

     [
      if pycor <= Weekly_Prebook_Start_30 * -4 and pycor >=  Weekly_Prebook_End_30 * -4
        and pxcor = (n + Servers_booked_30 - 1 + (Day_of_the_week_30 - 1) * (number_of_servers) )
    [if Weekly_Prebooked_30_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_30_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_30_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_30_Class = "Class 4" [set pcolor 37] ]
      set n n + number_of_servers * 7
     ]
   ]
  ]
  ]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;; -=SETUP - PATCHES=- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 ;; This block asks patches to assume a colour based on whether or not they represent time that is
bookable.
to setup-patches
ask patches[
 ifelse pycor >= (Scheduled-Hours-per-day) * -4
 [set pcolor red]
 [ set pcolor black]
 ]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;; -= RANDOM FILL =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


to random_fill

ask patches
[
if pcolor = red  or pcolor = 37 or pcolor = 47 and pxcor <= number_of_random_fill_days *
number_of_servers

[set pcolor brown]
]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;;; -= GO =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block initiates the simultor. The code used is based on the technique that has been selected.
;; For detailed explaination of each called function see below.
to go

 if Scheduling_Technique = 1
 [create-patient-class-1
 create-patient-class-2
 create-patient-class-3
 create-patient-class-4
 schedule-class-1
 set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 - 15
 set Arrival_Rate_Counter_2 Arrival_Rate_Counter_2 - 15
 set Arrival_Rate_Counter_3 Arrival_Rate_Counter_3 - 15
 set Arrival_Rate_Counter_4 Arrival_Rate_Counter_4 - 15 ]

 if Scheduling_Technique = 2 ; Disabled, future work will be continued here.
 [
]
 current-time-update
 set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 - 15
 tick
 set tick_counter tick_counter + 1
 if tick_counter = 96 * Days_to_run
 [ stop]
end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;; -= CREATING PATIENT CLASS 1 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block creates class one patients based on a probability which is evaluated on every tick of the
system.
;; The patients are created directly at the updater as they are emergency patients.


to create-patient-class-1

while [Arrival_Rate_Counter_1 <= 0]
[
if Probability-Patient-Class-1-Arrival = "Normal" and Patient-Class-1-Mean-Arrival-Rate > 0

 [set Normal_Holder_1 random-normal Patient-Class-1-Mean-Arrival-Rate Patient-Class-1-StD
  set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 + Normal_Holder_1
  create-class-1-patients 1 [set color green set size 1.5 move-to update-1 b set count1 count1 + 1 ]
  ]

  if Probability-Patient-Class-1-Arrival = "Exponential" and Patient-Class-1-Mean-Arrival-Rate > 0

 [set Exponential_Holder_1 random-Exponential Patient-Class-1-Mean-Arrival-Rate
  set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 + Exponential_Holder_1
  create-class-1-patients 1 [set color green set size 1.5 move-to update-1 b set count1 count1 + 1 ]
  ]

  if Probability-Patient-Class-1-Arrival = "Poisson" and Patient-Class-1-Mean-Arrival-Rate > 0
 [set Poisson_Holder_1 random-Poisson Patient-Class-1-Mean-Arrival-Rate
  set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 + Poisson_Holder_1
  create-class-1-patients 1 [set color green set size 1.5 move-to update-1 b set count1 count1 + 1 ]
  ]

   if Patient-Class-1-Mean-Arrival-Rate = 0
   [ set Arrival_Rate_Counter_1 Arrival_Rate_Counter_1 + 15]
]



;;This block of code allows this patient class to bump existing appointments to one slot later than the
one they currently occupy.
ask patches
[
 if any? class-1-patients[

 if pxcor = [pxcor] of update-1 b and pycor < [pycor] of update-1 b
   [ set pcolor [pcolor] of patch-at-heading-and-distance 0 1 ]

 set b b + 1
 if b > Number_of_Servers - 1 [set b 0]
 ]
 ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;; -= CREATING PATIENT CLASS 2 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block creates class two patients based on a probability which is evaluated on every tick of the
system.
;; The patients are created and moved to the day that follows the day that the updater is currently
processing. (ie. Tomorrow)
to create-patient-class-2

while [Arrival_Rate_Counter_2 <= 0]
[
if Probability-Patient-Class-2-Arrival = "Normal" and Patient-Class-2-Mean-Arrival-Rate > 0

 [set Normal_Holder_2 random-normal Patient-Class-2-Mean-Arrival-Rate Patient-Class-2-StD
  set Arrival_Rate_Counter_2 Arrival_Rate_Counter_2 + Normal_Holder_2
  create-class-2-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [ setxy xcor + 1 0 ] set count2 count2 + 1 ]
  ]
  if Probability-Patient-Class-2-Arrival = "Exponential" and Patient-Class-2-Mean-Arrival-Rate > 0

 [set Exponential_Holder_2 random-Exponential Patient-Class-2-Mean-Arrival-Rate
  set Arrival_Rate_Counter_2 Arrival_Rate_Counter_2 + Exponential_Holder_2
  create-class-2-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count2 count2 + 1 ]
  ]

  if Probability-Patient-Class-2-Arrival = "Poisson" and Patient-Class-2-Mean-Arrival-Rate > 0

 [set Poisson_Holder_2 random-Poisson Patient-Class-2-Mean-Arrival-Rate
  set Arrival_Rate_Counter_2 Arrival_Rate_Counter_2 + Poisson_Holder_2
  create-class-2-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count2 count2 + 1 ]

  ]
   if Patient-Class-2-Mean-Arrival-Rate = 0
   [ set Arrival_Rate_Counter_2 Arrival_Rate_Counter_2 + 15]


ask class-2-patients
[
 facexy 9999999999999999999999999999 max-pycor / 2
 set D 1
 while [xcor <= max-pxcor]
 [
   ifelse (pcolor = green) or (pcolor = black)or (pcolor = blue)or (pcolor = yellow)or (pcolor = brown) or
(pcolor = grey)or (pcolor = 37) or (pcolor = 47)   ;;; If the patch is occupied
   ;; Cut out xcor = ([xcor] of update-1 (number_of_Servers - 1)+ D * Number_of_Servers)  and    from
below
    [ifelse  ycor = (Scheduled-Hours-per-day) * -4 - 1   ;;;; If it is the end of the day

      [setxy [xcor] of update-1 (number_of_servers - 1) + D * number_of_servers + 1 0  if pcolor = red or
pcolor = 107 [if xcor > [xcor + 2 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set
count2over count2over + 1 ]set pcolor blue update-plot-class2 die set D 1]set D D + 1]

      [ if xcor = ([xcor] of update-1 (Number_of_Servers - 1) + D * Number_of_Servers)  or xcor + max-
pxcor - ([xcor]of update-1 (number_of_servers - 1) + D * Number_of_Servers) = -1
       ;; ^^ IF you are at the last server or
       [ setxy ([xcor] of update-1 (Number_of_Servers - 1)+ D * Number_of_Servers -
Number_of_servers) + 1 ycor - 1  if pcolor = red or pcolor = 107[if xcor > [xcor + 2 * Number_of_Servers]
of update-1 (Number_of_Servers - 1) [set count2over count2over + 1 ]set pcolor blue update-plot-class2
die set D 1]] if number_of_Servers > 1 [ forward 1]] ]

    [if xcor > [xcor + 2 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set count2over
count2over + 1 ]set pcolor blue update-plot-class2 die set D 1]
 ]
```

```
]
]

if Arrival_Rate_Counter_2 > 0
;; This block advances the plot pen for patient class two without making a mark. This is done if no class
two patients arrive during a given period.
[set-current-plot "Wait Time for Class 2 Patients"
 set-current-plot-pen "Class 2"
 plot-pen-up
 plot 1]

end
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;




;;;;;;;;;;;;;; -= CREATING PATIENT CLASS 3 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block creates class three patients based on a probability which is evaluated on every tick of the
system.
;; The patients are created and moved to the day that follows the day that the updater is currently
processing. (ie. Tomorrow)
to create-patient-class-3

while [Arrival_Rate_Counter_3 <= 0]
[
if Probability-Patient-Class-3-Arrival = "Normal" and Patient-Class-3-Mean-Arrival-Rate > 0

 [set Normal_Holder_3 random-normal Patient-Class-3-Mean-Arrival-Rate Patient-Class-3-StD
  set Arrival_Rate_Counter_3 Arrival_Rate_Counter_3 + Normal_Holder_3
  create-class-3-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count3 count3 + 1 ]
  ]

  if Probability-Patient-Class-3-Arrival = "Exponential" and Patient-Class-3-Mean-Arrival-Rate > 0

 [set Exponential_Holder_3 random-Exponential Patient-Class-3-Mean-Arrival-Rate
  set Arrival_Rate_Counter_3 Arrival_Rate_Counter_3 + Exponential_Holder_3
  create-class-3-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count3 count3 + 1 ]
  ]

  if Probability-Patient-Class-3-Arrival = "Poisson" and Patient-Class-3-Mean-Arrival-Rate > 0

 [set Poisson_Holder_3 random-Poisson Patient-Class-3-Mean-Arrival-Rate
  set Arrival_Rate_Counter_3 Arrival_Rate_Counter_3 + Poisson_Holder_3
  create-class-3-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count3 count3 + 1 ]
  ]
   if Patient-Class-3-Mean-Arrival-Rate = 0
   [ set Arrival_Rate_Counter_3 Arrival_Rate_Counter_3 + 15]
ask class-3-patients
[
 facexy 9999999999999999999999999999 max-pycor / 2
 set G 1
 while [xcor <= max-pxcor]
 [
  ifelse (pcolor = green) or (pcolor = black)or (pcolor = blue)or (pcolor = yellow)or (pcolor = brown) or
(pcolor = grey)or (pcolor = 37) or (pcolor = 107)   ;;; If the patch is occupied
   ;; Cut out xcor = ([xcor] of update-1 (number_of_Servers - 1)+ D * Number_of_Servers)  and    from
below
    [ifelse  ycor = (Scheduled-Hours-per-day) * -4 - 1    ;;;; If it is the end of the day

      [setxy [xcor] of update-1 (number_of_servers - 1) + G * number_of_servers + 1 0  if pcolor = red or
pcolor = 47 [ if xcor > [xcor + 10 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set
count3over count3over + 1 ]set pcolor yellow update-plot-class3 die set G 1]set G G + 1]

      [ if xcor = ([xcor] of update-1 (Number_of_Servers - 1) + G * Number_of_Servers)  or xcor + max-
pxcor - ([xcor]of update-1 (number_of_servers - 1) + G * Number_of_Servers) = -1

       [ setxy ([xcor] of update-1 (Number_of_Servers - 1)+ G * Number_of_Servers -
Number_of_servers) + 1 ycor - 1  if pcolor = red or pcolor = 47[if xcor > [xcor + 10 * Number_of_Servers]
of update-1 (Number_of_Servers - 1) [set count3over count3over + 1 ]set pcolor yellow update-plot-
class3 die set G 1]] if number_of_Servers > 1 [ forward 1]] ]

    [if xcor > [xcor + 10 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set count3over
count3over + 1 ]set pcolor yellow update-plot-class3 die set G 1]
 ]
]

]

;ifelse random 100 >= (100 - Probability-Patient-Class-3-Arrival) [set count3 count3 + 1 create-class-3-
patients 1
;[set color yellow set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse xcor = max-pxcor [setxy
min-pxcor 0] [  setxy xcor + 1 0]]]
if Arrival_Rate_Counter_3 > 0
;; This block advances the plot pen for patient class three without making a mark. This is done if no class
three patients arrive during a given period.
[set-current-plot "Wait Time for Class 3 Patients"
 set-current-plot-pen "Class 3"
 plot-pen-up
 plot 1]
end
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;; -= CREATING PATIENT CLASS 4 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block creates class four patients based on a probability which is evaluated on every tick of the
system.
;; The patients are created and moved to the day that follows the day that the updater is currently
processing. (ie. Tomorrow)
to create-patient-class-4

while [Arrival_Rate_Counter_4 <= 0]
[
if Probability-Patient-Class-4-Arrival = "Normal" and Patient-Class-4-Mean-Arrival-Rate > 0

 [set Normal_Holder_4 random-normal Patient-Class-4-Mean-Arrival-Rate Patient-Class-4-StD
  set Arrival_Rate_Counter_4 Arrival_Rate_Counter_4 + Normal_Holder_4
  create-class-4-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count4 count4 + 1 ]
  ]

  if Probability-Patient-Class-4-Arrival = "Exponential" and Patient-Class-4-Mean-Arrival-Rate > 0

 [set Exponential_Holder_4 random-Exponential Patient-Class-4-Mean-Arrival-Rate
  set Arrival_Rate_Counter_4 Arrival_Rate_Counter_4 + Exponential_Holder_4
  create-class-4-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count4 count4 + 1 ]
  ]

  if Probability-Patient-Class-4-Arrival = "Poisson" and Patient-Class-4-Mean-Arrival-Rate > 0

 [set Poisson_Holder_4 random-Poisson Patient-Class-4-Mean-Arrival-Rate
  set Arrival_Rate_Counter_4 Arrival_Rate_Counter_4 + Poisson_Holder_4
  create-class-4-patients 1 [set color green set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse
xcor = max-pxcor [setxy min-pxcor 0] [  setxy xcor + 1 0 ] set count4 count4 + 1 ]
  ]
   if Patient-Class-4-Mean-Arrival-Rate = 0
   [ set Arrival_Rate_Counter_4 Arrival_Rate_Counter_4 + 15]

ask class-4-patients
[
 facexy 9999999999999999999999999999 max-pycor / 2
 set F 1
 while [xcor <= max-pxcor]
 [
  ifelse (pcolor = green) or (pcolor = black)or (pcolor = blue)or (pcolor = yellow)or (pcolor = brown) or
(pcolor = grey)or (pcolor = 107) or (pcolor = 47)   ;;; If the patch is occupied
  ;; Cut out xcor = ([xcor] of update-1 (number_of_Servers - 1)+ D * Number_of_Servers)  and    from
below
    [ifelse  ycor = (Scheduled-Hours-per-day) * -4 - 1    ;;;; If it is the end of the day

      [setxy [xcor] of update-1 (number_of_servers - 1) + F * number_of_servers + 1 0  if pcolor = red or
pcolor = 37 [if xcor > [xcor + 28 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set
count4over count4over + 1 ]set pcolor brown update-plot-class4 die set F 1]set F F + 1]

      [ if xcor = ([xcor] of update-1 (Number_of_Servers - 1) + F * Number_of_Servers)  or xcor + max-
pxcor - ([xcor]of update-1 (number_of_servers - 1) + F * Number_of_Servers) = -1

        [ setxy ([xcor] of update-1 (Number_of_Servers - 1)+ F * Number_of_Servers - Number_of_servers)
+ 1 ycor - 1  if pcolor = red or pcolor = 37[if xcor > [xcor + 28 * Number_of_Servers] of update-1
(Number_of_Servers - 1) [set count4over count4over + 1 ]set pcolor brown update-plot-class4 die set F
1]] if number_of_Servers > 1 [ forward 1]] ]

    [if xcor > [xcor + 28 * Number_of_Servers] of update-1 (Number_of_Servers - 1) [set count4over
count4over + 1 ]set pcolor brown update-plot-class4 die set F 1]
  ]
]
]

;ifelse random 100 >= (100 - Probability-Patient-Class-4-Arrival) [set count4 count4 + 1 create-class-4-
patients 1
;[set color brown set size 1.5 move-to update-1 (Number_of_Servers - 1) ifelse xcor = max-pxcor [setxy
min-pxcor 0] [ setxy xcor + 1 0]]]

if Arrival_Rate_Counter_4 > 0
;; This block advances the plot pen for patient class four without making a mark. This is done if no class
four patients arrive during a given period.
[set-current-plot "Wait Time for Class 4 Patients"
 set-current-plot-pen "Class 4"
 plot-pen-up
 plot 1]

end


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;; -= SCEHDULING PATIENT CLASS 1 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; This block schedules class one patients that have already been created. It points the downward, moves
them forward one patch and kills them.
to schedule-class-1

ask class-1-patients
[
 facexy max-pxcor / 2 -9999999999999999999999999999
 forward 1
 set pcolor green
 die
]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;; -= CREATE TIME UPDATER =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block creates the turtle that acts as the 'current time.'

to create-time-updater
create-updaters Number_of_Servers [set color cyan set size 2.5 facexy max-pxcor -
9999999999999999999999999999 ]

Set MS 0

while [MS <= (Number_of_servers - 1)]
[
ask update-1 MS [setxy MS 0]
set MS MS + 1
]
end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;; -= CURRENT TIME UPDATE =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;This block updates the current day by clearing the previous schedule. It returns red patches to red,
black to black and grey to grey.                                                                  ;;

to current-time-update
ask updaters
[
 facexy max-pxcor / 2 -9999999999999999999999999999
```

```
;; This section does grey and red for the scheduled time.
   ifelse ycor >= (Scheduled-Hours-per-day) * -4
   [ifelse (ycor <= Daily_Prebook_Start * -4 and ycor >=  Daily_Prebook_End * -4 and Daily_Prebooked)
     [ set pcolor grey  ] [set pcolor red ] forward 1 ]

;; This section does grey and black for unscheduled time.
  [ifelse (ycor <= Daily_Prebook_Start * -4 and ycor >=  Daily_Prebook_End * -4 and Daily_Prebooked)
  [ set pcolor grey ] [set pcolor black]

;; This section handles moving the updater from the end of one day to the beginning of another and
resets the counter for bumping patients.
  if ycor = -96[ setxy xcor + Number_of_Servers 0 ifelse (ycor <= Daily_Prebook_Start * -4 and ycor >=
Daily_Prebook_End * -4 and Daily_Prebooked)
  [ set pcolor grey ] [set pcolor red] ]  forward 1]

]

 ;;;; Updates For Weekly Prebooks at the end of every day;;;;

  if [ycor] of update-1 0 = -96
   [
   if Weekly_Prebooked or Weekly_Prebooked_2 or Weekly_Prebooked_3 or Weekly_Prebooked_4 or
Weekly_Prebooked_5 or Weekly_Prebooked_6
   or Weekly_Prebooked_7 or Weekly_Prebooked_8 or Weekly_Prebooked_9 or Weekly_Prebooked_10
or Weekly_Prebooked_11 or
   Weekly_Prebooked_12 or Weekly_Prebooked_13 or Weekly_Prebooked_14 or Weekly_Prebooked_15
    or Weekly_Prebooked_16 or Weekly_Prebooked_17 or Weekly_Prebooked_18 or
Weekly_Prebooked_19 or Weekly_Prebooked_20 or Weekly_Prebooked_21
     or Weekly_Prebooked_22 or Weekly_Prebooked_23 or Weekly_Prebooked_24 or
Weekly_Prebooked_25 or Weekly_Prebooked_26 or Weekly_Prebooked_27
      or Weekly_Prebooked_28 or Weekly_Prebooked_29 or Weekly_Prebooked_30


    [ask patches
    [set n 0
    while [n <= [xcor] of update-1 0]
      [

      if ( pycor <= Weekly_Prebook_Start * -4 and pycor >=  Weekly_Prebook_End * -4 and
weekly_prebooked and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_1) - 1  + (Day_of_the_week - 1  ) * (number_of_servers ) and not
repeat_Daily_1  )
      [  if Weekly_Prebooked_1_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_1_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_1_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_1_Class = "Class 4" [set pcolor 37] ]
      if ( pycor <= Weekly_Prebook_Start_2 * -4 and pycor >=  Weekly_Prebook_End_2 * -4 and
weekly_prebooked_2 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_2) - 1   + (Day_of_the_week_2 - 1  ) * (number_of_servers ) and
not repeat_Daily_2  )
      [  if Weekly_Prebooked_2_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_2_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_2_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_2_Class = "Class 4" [set pcolor 37] ]


      if ( pycor <= Weekly_Prebook_Start_3 * -4 and pycor >=  Weekly_Prebook_End_3 * -4 and
weekly_prebooked_3 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_3) - 1   + (Day_of_the_week_3 - 1  ) * (number_of_servers ) and
not repeat_Daily_3  )
      [  if Weekly_Prebooked_3_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_3_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_3_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_3_Class = "Class 4" [set pcolor 37] ]


      if ( pycor <= Weekly_Prebook_Start_4 * -4 and pycor >=  Weekly_Prebook_End_4 * -4 and
weekly_prebooked_4 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_4) - 1   + (Day_of_the_week_4 - 1  ) * (number_of_servers ) and
not repeat_Daily_4  )
      [  if Weekly_Prebooked_4_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_4_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_4_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_4_Class = "Class 4" [set pcolor 37] ]


      if ( pycor <= Weekly_Prebook_Start_5 * -4 and pycor >=  Weekly_Prebook_End_5 * -4 and
weekly_prebooked_5 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_5) - 1   + (Day_of_the_week_5 - 1  ) * (number_of_servers ) and
not repeat_Daily_5  )
      [  if Weekly_Prebooked_5_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_5_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_5_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_5_Class = "Class 4" [set pcolor 37] ]


      if ( pycor <= Weekly_Prebook_Start_6 * -4 and pycor >=  Weekly_Prebook_End_6 * -4 and
weekly_prebooked_6 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_6) - 1   + (Day_of_the_week_6 - 1  ) * (number_of_servers ) and
not repeat_Daily_6  )
      [  if Weekly_Prebooked_6_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_6_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_6_Class = "Class 3" [set pcolor 47]
```

```
    if Weekly_Prebooked_6_Class = "Class 4" [set pcolor 37] ]


    if ( pycor <= Weekly_Prebook_Start_7 * -4 and pycor >=  Weekly_Prebook_End_7 * -4 and
weekly_prebooked_7 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_7) - 1   + (Day_of_the_week_7 - 1  ) * (number_of_servers ) and
not repeat_Daily_7  )
    [  if Weekly_Prebooked_7_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_7_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_7_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_7_Class = "Class 4" [set pcolor 37] ]


    if ( pycor <= Weekly_Prebook_Start_8 * -4 and pycor >=  Weekly_Prebook_End_8 * -4 and
weekly_prebooked_8 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_8) - 1   + (Day_of_the_week_8 - 1  ) * (number_of_servers ) and
not repeat_Daily_8  )
    [  if Weekly_Prebooked_8_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_8_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_8_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_8_Class = "Class 4" [set pcolor 37] ]


    if ( pycor <= Weekly_Prebook_Start_9 * -4 and pycor >=  Weekly_Prebook_End_9 * -4 and
weekly_prebooked_9 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_9) - 1   + (Day_of_the_week_9 - 1  ) * (number_of_servers ) and
not repeat_Daily_9  )
    [  if Weekly_Prebooked_9_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_9_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_9_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_9_Class = "Class 4" [set pcolor 37] ]


    if ( pycor <= Weekly_Prebook_Start_10 * -4 and pycor >=  Weekly_Prebook_End_10 * -4 and
weekly_prebooked_10 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_10) - 1   + (Day_of_the_week_10 - 1  ) * (number_of_servers )
and not repeat_Daily_10  )
    [  if Weekly_Prebooked_10_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_10_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_10_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_10_Class = "Class 4" [set pcolor 37] ]


    if ( pycor <= Weekly_Prebook_Start_11 * -4 and pycor >=  Weekly_Prebook_End_11 * -4 and
weekly_prebooked_11 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_11) - 1   + (Day_of_the_week_11 - 1  ) * (number_of_servers )
and not repeat_Daily_11  )
    [  if Weekly_Prebooked_11_Class = "None" [set pcolor grey]
```

```
        if Weekly_Prebooked_11_Class = "Class 2" [set pcolor 107]
        if Weekly_Prebooked_11_Class = "Class 3" [set pcolor 47]
        if Weekly_Prebooked_11_Class = "Class 4" [set pcolor 37] ]


        if ( pycor <= Weekly_Prebook_Start_12 * -4 and pycor >=  Weekly_Prebook_End_12 * -4 and
weekly_prebooked_12 and pxcor < [xcor] of update-1 0
        and pxcor  = (n + Servers_booked_12) - 1   + (Day_of_the_week_12 - 1  ) * (number_of_servers )
and not repeat_Daily_12  )
        [  if Weekly_Prebooked_12_Class = "None" [set pcolor grey]
        if Weekly_Prebooked_12_Class = "Class 2" [set pcolor 107]
        if Weekly_Prebooked_12_Class = "Class 3" [set pcolor 47]
        if Weekly_Prebooked_12_Class = "Class 4" [set pcolor 37] ]


        if ( pycor <= Weekly_Prebook_Start_13 * -4 and pycor >=  Weekly_Prebook_End_13 * -4 and
weekly_prebooked_13 and pxcor < [xcor] of update-1 0
        and pxcor  = (n + Servers_booked_13) - 1   + (Day_of_the_week_13 - 1  ) * (number_of_servers )
and not repeat_Daily_13  )
        [  if Weekly_Prebooked_13_Class = "None" [set pcolor grey]
        if Weekly_Prebooked_13_Class = "Class 2" [set pcolor 107]
        if Weekly_Prebooked_13_Class = "Class 3" [set pcolor 47]
        if Weekly_Prebooked_13_Class = "Class 4" [set pcolor 37] ]


        if ( pycor <= Weekly_Prebook_Start_14 * -4 and pycor >=  Weekly_Prebook_End_14 * -4 and
weekly_prebooked_14 and pxcor < [xcor] of update-1 0
        and pxcor  = (n + Servers_booked_14) - 1   + (Day_of_the_week_14 - 1  ) * (number_of_servers )
and not repeat_Daily_14  )
        [  if Weekly_Prebooked_14_Class = "None" [set pcolor grey]
        if Weekly_Prebooked_14_Class = "Class 2" [set pcolor 107]
        if Weekly_Prebooked_14_Class = "Class 3" [set pcolor 47]
        if Weekly_Prebooked_14_Class = "Class 4" [set pcolor 37] ]


        if ( pycor <= Weekly_Prebook_Start_15 * -4 and pycor >=  Weekly_Prebook_End_15 * -4 and
weekly_prebooked_15 and pxcor < [xcor] of update-1 0
        and pxcor  = (n + Servers_booked_15) - 1   + (Day_of_the_week_15 - 1  ) * (number_of_servers )
and not repeat_Daily_15  )
        [  if Weekly_Prebooked_15_Class = "None" [set pcolor grey]
        if Weekly_Prebooked_15_Class = "Class 2" [set pcolor 107]
        if Weekly_Prebooked_15_Class = "Class 3" [set pcolor 47]
        if Weekly_Prebooked_15_Class = "Class 4" [set pcolor 37] ]
```

```
    if ( pycor <= Weekly_Prebook_Start_16 * -4 and pycor >=  Weekly_Prebook_End_16 * -4 and
weekly_prebooked_16 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_16) - 1   + (Day_of_the_week_16 - 1  ) * (number_of_servers )
and not repeat_Daily_16  )
    [  if Weekly_Prebooked_16_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_16_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_16_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_16_Class = "Class 4" [set pcolor 37] ]

        if ( pycor <= Weekly_Prebook_Start_17 * -4 and pycor >=  Weekly_Prebook_End_17 * -4 and
weekly_prebooked_17 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_17) - 1   + (Day_of_the_week_17 - 1  ) * (number_of_servers )
and not repeat_Daily_17  )
    [  if Weekly_Prebooked_17_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_17_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_17_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_17_Class = "Class 4" [set pcolor 37] ]

        if ( pycor <= Weekly_Prebook_Start_18 * -4 and pycor >=  Weekly_Prebook_End_18 * -4 and
weekly_prebooked_18 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_18) - 1   + (Day_of_the_week_18 - 1  ) * (number_of_servers )
and not repeat_Daily_18  )
    [  if Weekly_Prebooked_18_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_18_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_18_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_18_Class = "Class 4" [set pcolor 37] ]

        if ( pycor <= Weekly_Prebook_Start_19 * -4 and pycor >=  Weekly_Prebook_End_19 * -4 and
weekly_prebooked_19 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_19) - 1   + (Day_of_the_week_19 - 1  ) * (number_of_servers )
and not repeat_Daily_19  )
    [  if Weekly_Prebooked_19_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_19_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_19_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_19_Class = "Class 4" [set pcolor 37] ]

        if ( pycor <= Weekly_Prebook_Start_20 * -4 and pycor >=  Weekly_Prebook_End_20 * -4 and
weekly_prebooked_20 and pxcor < [xcor] of update-1 0
    and pxcor  = (n + Servers_booked_20) - 1   + (Day_of_the_week_20 - 1  ) * (number_of_servers )
and not repeat_Daily_20  )
    [  if Weekly_Prebooked_20_Class = "None" [set pcolor grey]
    if Weekly_Prebooked_20_Class = "Class 2" [set pcolor 107]
    if Weekly_Prebooked_20_Class = "Class 3" [set pcolor 47]
    if Weekly_Prebooked_20_Class = "Class 4" [set pcolor 37] ]

        if ( pycor <= Weekly_Prebook_Start_21 * -4 and pycor >=  Weekly_Prebook_End_21 * -4 and
weekly_prebooked_21 and pxcor < [xcor] of update-1 0
```

```
      and pxcor  = (n + Servers_booked_21) - 1   + (Day_of_the_week_21 - 1  ) * (number_of_servers )
and not repeat_Daily_21  )
      [  if Weekly_Prebooked_21_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_21_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_21_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_21_Class = "Class 4" [set pcolor 37] ]

            if ( pycor <= Weekly_Prebook_Start_22 * -4 and pycor >=  Weekly_Prebook_End_22 * -4 and
weekly_prebooked_22 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_22) - 1   + (Day_of_the_week_22 - 1  ) * (number_of_servers )
and not repeat_Daily_22  )
      [  if Weekly_Prebooked_22_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_22_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_22_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_22_Class = "Class 4" [set pcolor 37] ]

            if ( pycor <= Weekly_Prebook_Start_23 * -4 and pycor >=  Weekly_Prebook_End_23 * -4 and
weekly_prebooked_23 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_23) - 1   + (Day_of_the_week_23 - 1  ) * (number_of_servers )
and not repeat_Daily_23  )
      [  if Weekly_Prebooked_23_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_23_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_23_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_23_Class = "Class 4" [set pcolor 37] ]

            if ( pycor <= Weekly_Prebook_Start_24 * -4 and pycor >=  Weekly_Prebook_End_24 * -4 and
weekly_prebooked_24 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_24) - 1   + (Day_of_the_week_24 - 1  ) * (number_of_servers )
and not repeat_Daily_24  )
      [  if Weekly_Prebooked_24_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_24_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_24_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_24_Class = "Class 4" [set pcolor 37] ]

            if ( pycor <= Weekly_Prebook_Start_25 * -4 and pycor >=  Weekly_Prebook_End_25 * -4 and
weekly_prebooked_25 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_25) - 1   + (Day_of_the_week_25 - 1  ) * (number_of_servers )
and not repeat_Daily_25  )
      [  if Weekly_Prebooked_25_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_25_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_25_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_25_Class = "Class 4" [set pcolor 37] ]

            if ( pycor <= Weekly_Prebook_Start_26 * -4 and pycor >=  Weekly_Prebook_End_26 * -4 and
weekly_prebooked_26 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_26) - 1   + (Day_of_the_week_26 - 1  ) * (number_of_servers )
and not repeat_Daily_26  )
      [  if Weekly_Prebooked_26_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_26_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_26_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_26_Class = "Class 4" [set pcolor 37] ]

          if ( pycor <= Weekly_Prebook_Start_27 * -4 and pycor >=  Weekly_Prebook_End_27 * -4 and
weekly_prebooked_27 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_27) - 1  + (Day_of_the_week_27 - 1  ) * (number_of_servers )
and not repeat_Daily_27  )
      [  if Weekly_Prebooked_27_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_27_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_27_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_27_Class = "Class 4" [set pcolor 37] ]

          if ( pycor <= Weekly_Prebook_Start_28 * -4 and pycor >=  Weekly_Prebook_End_28 * -4 and
weekly_prebooked_28 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_28) - 1  + (Day_of_the_week_28 - 1  ) * (number_of_servers )
and not repeat_Daily_28  )
      [  if Weekly_Prebooked_28_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_28_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_28_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_28_Class = "Class 4" [set pcolor 37] ]

          if ( pycor <= Weekly_Prebook_Start_29 * -4 and pycor >=  Weekly_Prebook_End_29 * -4 and
weekly_prebooked_29 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_29) - 1  + (Day_of_the_week_29 - 1  ) * (number_of_servers )
and not repeat_Daily_29  )
      [  if Weekly_Prebooked_29_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_29_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_29_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_29_Class = "Class 4" [set pcolor 37] ]

          if ( pycor <= Weekly_Prebook_Start_30 * -4 and pycor >=  Weekly_Prebook_End_30 * -4 and
weekly_prebooked_30 and pxcor < [xcor] of update-1 0
      and pxcor  = (n + Servers_booked_30) - 1  + (Day_of_the_week_30 - 1  ) * (number_of_servers )
and not repeat_Daily_30  )
      [  if Weekly_Prebooked_30_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_30_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_30_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_30_Class = "Class 4" [set pcolor 37] ]



      set n  n + 7 * (Number_of_servers)
      ]
    ]
   ]

   ]
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;UPDATING REPEAT DAILYS;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if [ycor] of update-1 0 = -96
  [
  if Weekly_Prebooked or Weekly_Prebooked_2 or Weekly_Prebooked_3 or Weekly_Prebooked_4 or
Weekly_Prebooked_5 or Weekly_Prebooked_6
  or Weekly_Prebooked_7 or Weekly_Prebooked_8 or Weekly_Prebooked_9 or Weekly_Prebooked_10
or Weekly_Prebooked_11 or
  Weekly_Prebooked_12 or Weekly_Prebooked_13 or Weekly_Prebooked_14 or Weekly_Prebooked_15
   or Weekly_Prebooked_16 or Weekly_Prebooked_17 or Weekly_Prebooked_18 or
Weekly_Prebooked_19 or Weekly_Prebooked_20 or Weekly_Prebooked_21
    or Weekly_Prebooked_22 or Weekly_Prebooked_23 or Weekly_Prebooked_24 or
Weekly_Prebooked_25 or Weekly_Prebooked_26 or Weekly_Prebooked_27
     or Weekly_Prebooked_28 or Weekly_Prebooked_29 or Weekly_Prebooked_30
   [ask patches
   [set n 0
   while [n <= [xcor] of update-1 0]
     [

    if pycor <= Weekly_Prebook_Start * -4 and pycor >=  Weekly_Prebook_End * -4
         and pxcor = Servers_Booked_1 + n - 1  and repeat_daily_1 and Weekly_Prebooked
     [if Weekly_Prebooked_1_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_1_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_1_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_1_Class = "Class 4" [set pcolor 37] ]


    if pycor <= Weekly_Prebook_Start_2 * -4 and pycor >=  Weekly_Prebook_End_2 * -4
         and pxcor = Servers_Booked_2 + n - 1  and repeat_daily_2 and Weekly_Prebooked_2
     [if Weekly_Prebooked_2_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_2_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_2_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_2_Class = "Class 4" [set pcolor 37] ]


    if pycor <= Weekly_Prebook_Start_3 * -4 and pycor >=  Weekly_Prebook_End_3 * -4
         and pxcor = Servers_Booked_3 + n - 1  and repeat_daily_3  and Weekly_Prebooked_3
     [if Weekly_Prebooked_3_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_3_Class = "Class 2" [set pcolor 107]
      if Weekly_Prebooked_3_Class = "Class 3" [set pcolor 47]
      if Weekly_Prebooked_3_Class = "Class 4" [set pcolor 37] ]


    if pycor <= Weekly_Prebook_Start_4 * -4 and pycor >=  Weekly_Prebook_End_4 * -4
         and pxcor = Servers_Booked_4 + n - 1  and repeat_daily_4 and Weekly_Prebooked_4
     [if Weekly_Prebooked_4_Class = "None" [set pcolor grey]
      if Weekly_Prebooked_4_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_4_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_4_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_5 * -4 and pycor >=  Weekly_Prebook_End_5 * -4
     and pxcor = Servers_Booked_5 + n - 1  and repeat_daily_5  and Weekly_Prebooked_5
 [if Weekly_Prebooked_5_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_5_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_5_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_5_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_6 * -4 and pycor >=  Weekly_Prebook_End_6 * -4
     and pxcor = Servers_Booked_6 + n - 1  and repeat_daily_6      and Weekly_Prebooked_6
 [if Weekly_Prebooked_6_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_6_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_6_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_6_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_7 * -4 and pycor >=  Weekly_Prebook_End_7 * -4
     and pxcor = Servers_Booked_7 + n - 1  and repeat_daily_7    and Weekly_Prebooked_7
 [if Weekly_Prebooked_7_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_7_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_7_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_7_Class = "Class 4" [set pcolor 37] ]




if pycor <= Weekly_Prebook_Start_8 * -4 and pycor >=  Weekly_Prebook_End_8 * -4
     and pxcor = Servers_Booked_8 + n - 1  and repeat_daily_8     and Weekly_Prebooked_8
 [if Weekly_Prebooked_8_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_8_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_8_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_8_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_9 * -4 and pycor >=  Weekly_Prebook_End_9 * -4
     and pxcor = Servers_Booked_9 + n - 1  and repeat_daily_9     and Weekly_Prebooked_9
 [if Weekly_Prebooked_9_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_9_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_9_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_9_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_10 * -4 and pycor >=  Weekly_Prebook_End_10 * -4
     and pxcor = Servers_Booked_10 + n - 1  and repeat_daily_10     and Weekly_Prebooked_10
 [if Weekly_Prebooked_10_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_10_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_10_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_10_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_11 * -4 and pycor >=  Weekly_Prebook_End_11 * -4
    and pxcor = Servers_Booked_11 + n - 1  and repeat_daily_11   and Weekly_Prebooked_11
 [if Weekly_Prebooked_11_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_11_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_11_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_11_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_12 * -4 and pycor >=  Weekly_Prebook_End_12 * -4
    and pxcor = Servers_Booked_12 + n - 1  and repeat_daily_12    and Weekly_Prebooked_12
 [if Weekly_Prebooked_12_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_12_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_12_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_12_Class = "Class 4" [set pcolor 37] ]



if pycor <= Weekly_Prebook_Start_13 * -4 and pycor >=  Weekly_Prebook_End_13 * -4
    and pxcor = Servers_Booked_13 + n - 1  and repeat_daily_13     and Weekly_Prebooked_13
 [if Weekly_Prebooked_13_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_13_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_13_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_13_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_14 * -4 and pycor >=  Weekly_Prebook_End_14 * -4
    and pxcor = Servers_Booked_14 + n - 1  and repeat_daily_14      and Weekly_Prebooked_14
 [if Weekly_Prebooked_14_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_14_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_14_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_14_Class = "Class 4" [set pcolor 37] ]


if pycor <= Weekly_Prebook_Start_15 * -4 and pycor >=  Weekly_Prebook_End_15 * -4
    and pxcor = Servers_Booked_15 + n - 1  and repeat_daily_15      and Weekly_Prebooked_15
 [if Weekly_Prebooked_15_Class = "None" [set pcolor grey]
  if Weekly_Prebooked_15_Class = "Class 2" [set pcolor 107]
  if Weekly_Prebooked_15_Class = "Class 3" [set pcolor 47]
  if Weekly_Prebooked_15_Class = "Class 4" [set pcolor 37] ]

     if pycor <= Weekly_Prebook_Start_16 * -4 and pycor >=  Weekly_Prebook_End_16 * -4
    and pxcor = Servers_Booked_16 + n - 1  and repeat_daily_16      and Weekly_Prebooked_16
```

```
[if Weekly_Prebooked_16_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_16_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_16_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_16_Class = "Class 4" [set pcolor 37] ]

    if pycor <= Weekly_Prebook_Start_17 * -4 and pycor >=  Weekly_Prebook_End_17 * -4
   and pxcor = Servers_Booked_17 + n - 1  and repeat_daily_17     and Weekly_Prebooked_17
[if Weekly_Prebooked_17_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_17_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_17_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_17_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_18 * -4 and pycor >=  Weekly_Prebook_End_18 * -4
   and pxcor = Servers_Booked_18 + n - 1  and repeat_daily_18     and Weekly_Prebooked_18
[if Weekly_Prebooked_18_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_18_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_18_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_18_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_19 * -4 and pycor >=  Weekly_Prebook_End_19 * -4
   and pxcor = Servers_Booked_19 + n - 1  and repeat_daily_19     and Weekly_Prebooked_19
[if Weekly_Prebooked_19_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_19_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_19_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_19_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_20 * -4 and pycor >=  Weekly_Prebook_End_20 * -4
   and pxcor = Servers_Booked_20 + n - 1  and repeat_daily_20     and Weekly_Prebooked_20
[if Weekly_Prebooked_20_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_20_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_20_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_20_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_21 * -4 and pycor >=  Weekly_Prebook_End_21 * -4
   and pxcor = Servers_Booked_21 + n - 1  and repeat_daily_21     and Weekly_Prebooked_21
[if Weekly_Prebooked_21_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_21_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_21_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_21_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_22 * -4 and pycor >=  Weekly_Prebook_End_22 * -4
   and pxcor = Servers_Booked_22 + n - 1  and repeat_daily_22     and Weekly_Prebooked_22
[if Weekly_Prebooked_22_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_22_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_22_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_22_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_23 * -4 and pycor >=  Weekly_Prebook_End_23 * -4
   and pxcor = Servers_Booked_23 + n - 1  and repeat_daily_23     and Weekly_Prebooked_23
[if Weekly_Prebooked_23_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_23_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_23_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_23_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_24 * -4 and pycor >=  Weekly_Prebook_End_24 * -4
   and pxcor = Servers_Booked_24 + n - 1  and repeat_daily_24     and Weekly_Prebooked_24
[if Weekly_Prebooked_24_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_24_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_24_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_24_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_25 * -4 and pycor >=  Weekly_Prebook_End_25 * -4
   and pxcor = Servers_Booked_25 + n - 1  and repeat_daily_25     and Weekly_Prebooked_25
[if Weekly_Prebooked_25_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_25_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_25_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_25_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_26 * -4 and pycor >=  Weekly_Prebook_End_26 * -4
   and pxcor = Servers_Booked_26 + n - 1  and repeat_daily_26     and Weekly_Prebooked_26
[if Weekly_Prebooked_26_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_26_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_26_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_26_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_27 * -4 and pycor >=  Weekly_Prebook_End_27 * -4
   and pxcor = Servers_Booked_27 + n - 1  and repeat_daily_27     and Weekly_Prebooked_27
[if Weekly_Prebooked_27_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_27_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_27_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_27_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_28 * -4 and pycor >=  Weekly_Prebook_End_28 * -4
   and pxcor = Servers_Booked_28 + n - 1  and repeat_daily_28     and Weekly_Prebooked_28
[if Weekly_Prebooked_28_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_28_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_28_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_28_Class = "Class 4" [set pcolor 37] ]

        if pycor <= Weekly_Prebook_Start_29 * -4 and pycor >=  Weekly_Prebook_End_29 * -4
   and pxcor = Servers_Booked_29 + n - 1  and repeat_daily_29     and Weekly_Prebooked_29
[if Weekly_Prebooked_29_Class = "None" [set pcolor grey]
 if Weekly_Prebooked_29_Class = "Class 2" [set pcolor 107]
 if Weekly_Prebooked_29_Class = "Class 3" [set pcolor 47]
 if Weekly_Prebooked_29_Class = "Class 4" [set pcolor 37] ]
```

```
              if pycor <= Weekly_Prebook_Start_30 * -4 and pycor >=  Weekly_Prebook_End_30 * -4
           and pxcor = Servers_Booked_30 + n - 1  and repeat_daily_30     and Weekly_Prebooked_30
      [if Weekly_Prebooked_30_Class = "None" [set pcolor grey]
       if Weekly_Prebooked_30_Class = "Class 2" [set pcolor 107]
       if Weekly_Prebooked_30_Class = "Class 3" [set pcolor 47]
       if Weekly_Prebooked_30_Class = "Class 4" [set pcolor 37] ]




       set n  n + Number_of_servers
       ]
   ]
  ]

  ]


end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;; -= UPDATE-PLOT-CLASS2 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block updates the plot of class two patient wait times.
;; The code counts the number of days (columns) between the current time (updater) and the
appointment patch to determine the days waited.
to update-plot-class2
 set-current-plot "Wait Time for Class 2 Patients"
 set-current-plot-pen "Class 2"
 plot-pen-down
 ifelse [xcor] of one-of class-2-patients >= [xcor] of update-1 (Number_of_Servers - 1)
    [
     plot floor (([xcor] of one-of class-2-patients - [xcor] of update-1 (Number_of_Servers - 1)) /
Number_of_Servers )
     set class2total class2total + floor(( ([xcor] of one-of class-2-patients - [xcor] of update-1
(Number_of_Servers - 1)) ) / Number_of_Servers )
    if Record [ file-write "Class 2" file-print floor (([xcor] of one-of class-2-patients - [xcor] of update-1
(Number_of_Servers - 1)) / Number_of_Servers )]
    ]
    [
    plot floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor] of one-of class-2-
patients) / Number_of_Servers)
    set class2total class2total + floor ((( max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor]
of one-of class-2-patients)) / Number_of_Servers)
```

```
    if Record [file-write "Class 2" file-print floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers - 1)
+ [xcor] of one-of class-2-patients) / Number_of_Servers)  ]
    ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;; -= UPDATE-PLOT-CLASS3 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block updates the plot of class three patient wait times.
;; The code counts the number of days (columns) between the current time (updater) and the
appointment patch to determine the days waited.
to update-plot-class3
  set-current-plot "Wait Time for Class 3 Patients"
  set-current-plot-pen "Class 3"
  plot-pen-down
  ifelse [xcor] of one-of class-3-patients >= [xcor] of update-1 (Number_of_Servers - 1)
    [
     plot floor (([xcor] of one-of class-3-patients - [xcor] of update-1 (Number_of_Servers - 1)) /
Number_of_Servers )
     set class3total class3total + floor(( ([xcor] of one-of class-3-patients - [xcor] of update-1
(Number_of_Servers - 1)) ) / Number_of_Servers )
    if Record [ file-write "Class 3" file-print floor (([xcor] of one-of class-3-patients - [xcor] of update-1
(Number_of_Servers - 1)) / Number_of_Servers ) ]
    ]
    [
    plot floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor] of one-of class-3-
patients) / Number_of_Servers)
     set class3total class3total + floor ((( max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor]
of one-of class-3-patients)) / Number_of_Servers)
   if Record [    file-write "Class 3" file-print floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers -
1) + [xcor] of one-of class-3-patients) / Number_of_Servers)   ]
    ]


end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;; -= UPDATE-PLOT-CLASS4 =- ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; This block updates the plot of class four patient wait times.
;; The code counts the number of days (columns) between the current time (updater) and the
appointment patch to determine the days waited.
to update-plot-class4
```

```
 set-current-plot "Wait Time for Class 4 Patients"
 set-current-plot-pen "Class 4"
 plot-pen-down
 ifelse [xcor] of one-of class-4-patients >= [xcor] of update-1 (Number_of_Servers - 1)
   [
    plot floor (([xcor] of one-of class-4-patients - [xcor] of update-1 (Number_of_Servers - 1)) /
Number_of_Servers )
    set class4total class4total + floor(( ([xcor] of one-of class-4-patients - [xcor] of update-1
(Number_of_Servers - 1)) ) / Number_of_Servers )
   if Record [  file-write "Class 4" file-print floor (([xcor] of one-of class-4-patients - [xcor] of update-1
(Number_of_Servers - 1)) / Number_of_Servers )]
   ]
   [
   plot floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor] of one-of class-4-
patients) / Number_of_Servers)
    set class4total class4total + floor ((( max-pxcor - [xcor] of update-1 (Number_of_Servers - 1) + [xcor]
of one-of class-4-patients)) / Number_of_Servers)
if Record [  file-write "Class 4" file-print floor floor ((max-pxcor - [xcor] of update-1 (Number_of_Servers
- 1) + [xcor] of one-of class-4-patients) / Number_of_Servers)   ]
   ]

end

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Vita Auctoris

Brendan Eagen was born in Windsor, Ontario. He graduated from Assumption High School in 2003. From there he went on to the University of Windsor where he obtained a BASc in Industrial and Manufacturing Systems Engineering in 2007. He is currently a candidate from for the Master's degree in Industrial and Manufacturing Systems Engineering at the University of Windsor and hopes to graduate in Fall 2009.