ISSN 1744-1986

Technical Report N° 2008/ 25

A multi-agent system approach to a sumulation study comparing the performance of aircraft boarding using pre-assigned seating and free-for-all strategies

R. Livermore

28 June, 2008

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk

A multi-agent system approach to a simulation study comparing the performance of aircraft boarding using pre-assigned seating and free-for-all strategies

A dissertation submitted in partial fulfilment
of the requirements for the Open University's
Master of Science Degree
in Software Development

Robert Anthony Livermore
[redacted]

6 March 2009

Word Count: **15,752**

Preface

Thanks go to my employer, Durham University, for its generosity in providing financial support for this work.

I am extremely grateful to the Open University staff, both academic and administrative, who have provided me with such professional and knowledgeable support. Special mention must be made of my supervisor, Dr Doug Leith, whose guidance has been invaluable throughout.

I owe a debt of gratitude to all those involved in the University of Calgary's TeleSim project, who have kindly made their SimKit Java package available for public use. I am similarly grateful to Dr Jason Steffen of the Fermilab Center for Particle Astrophysics, who kindly sent me pre-prints of his submitted papers.

Last, but by no means least, I am especially thankful for the encouragement shown by my family and in particular for the patience afforded me by my lovely wife, Emma.

Table of Contents

List of Figures

List of Tables

Abstract

Achieving true efficiency is an important commercial driver for airlines and can be of huge value in differentiating them in a competitive marketplace. The aircraft boarding process remains a relatively unstudied area in this regard and is perhaps one of the few remaining standard airline operations where significant improvements may still be delivered.

Studies to date have focused on improving the process by applying varying levels of control to passenger ordering as they enter the aircraft. However, passenger actions and interactions are, by their nature, governed by an element of chance and so the natural state of the boarding system tends towards randomness.

In acknowledgement of this fact, this simulation-based study investigates the performance of the boarding process when controls are relaxed to a greater or lesser degree. It investigates whether multi-agent systems are appropriate for simulating stochastic processes by comparison with baseline results and whether they allow real conclusions to be drawn on the relative merits of different boarding systems.

The results produced by this work cannot be statistically proven to be the same as the baseline and thus it cannot be said in this context that multi-agent systems are appropriate for simulating stochastic processes. However, in relative terms,

the findings of this work do appear to follow the patterns hypothesised in earlier studies – that is that boarding using pre-assigned seating but with no correlation between the order passengers enter the aircraft and the position of their seat is preferable over a range of different scenarios to Free-for-All boarding. This has allowed useful future work to be identified that will ensure that the results presented in this study are built upon in a more comprehensive manner to develop a fuller picture of the types of passenger interaction and interference that cause differential performance across boarding strategies.

Chapter 1 Introduction

The study of airplane boarding patterns is rooted in the drive for efficiency that has characterised the response of traditional airlines to the twin threats of changing market conditions and the significant success of ventures adhering to the alternative 'low-cost' model of operation (Alamdari & Fagan, 2005). Delays on the ground have been estimated to cost domestic carriers in the United States over $22/minute, equating to an expenditure of approximately $220 million across the entire national system (Funk 2003). A recent study has shown that a majority of low-cost airlines have considerably higher fleet utilisation rates, equating to quicker turnaround times, than British Airways (Alamdari & Fagan, 2005). The comparison is based on similar aircraft types. Such figures clearly establish the motivation for reducing ground turnaround times and ensuring that aircraft spend as much time as possible in the air, where they are not subject to costly financial penalties and may thus maximise their potential for generating revenue. In addition, the validity of comparison between the processes employed by traditional carriers and their non-traditional competitors is established.

1.1 Background to the Research

An early commercial study of aircraft boarding strategies (Marelli et al., 1998) identified the key elements of the turnaround process (see Figure 1 below). Broadly speaking, these fall into three main categories: Passenger Services, Luggage/Cargo Handling and Aircraft Servicing. Figure 1 shows that, of these

three main categories, Passenger Services unambiguously dictates the length of the turnaround process. Luggage/Cargo Handling can potentially take as long, depending on the volume of work required, but the inherent variability indicates that the adoption of efficient planning and loading strategies should be able to minimise the time required for completion. Within the Passenger Services process as a whole, the ‘Board Passengers’ activity is one of the only ones that does not run in parallel with one or more other activity for its entire duration, making it a critical activity in the overall process. Due to this, reductions in the overall time taken to complete the activity will translate to reductions in the time taken to complete the entire process. This, along with the quantifiable financial benefits that such improvements will realise, justifies the passenger boarding process as a valid area for study. Putting this in the context of the steady increase in passenger boarding times observed since the 1970s (Marelli et al., 1998) sees the situation become more relevant still.

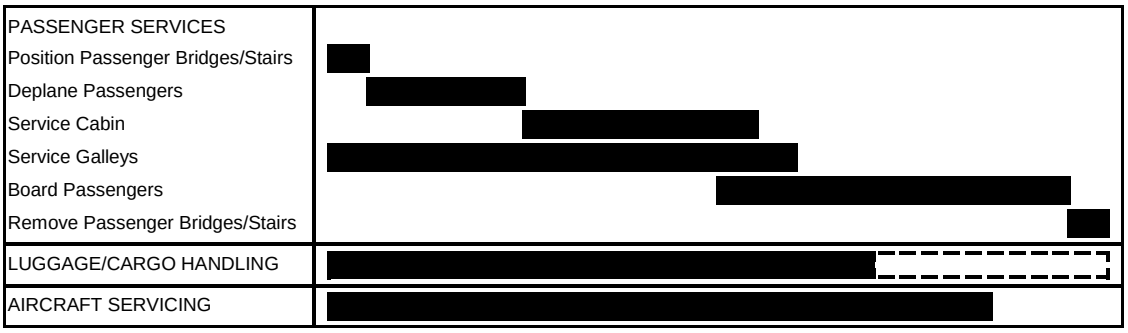

Figure 1 Sequence of Turnaround Activities.
Adapted from Marelli et al., 1998.

Later studies, carried out by academic researchers in conjunction with industrial collaborators, confirmed the passenger boarding activity as the bottleneck element in the whole sequence of activities that form the turnaround process, further reinforcing it as a valid target for improvement (van den Briel et al., 2005).

Subsequent academic studies have often been undertaken following a negative boarding experience on the part of the researcher themselves (Steffen, 2008a; Steffen, 2008b). Studies often make use of the problem domain as a useful real-world application for optimisation algorithms and statistical techniques (Bazargan, 2007; Steffen, 2008a; Steffen, 2008b), as well as for the novel application of established theoretical paradigms from other disciplines (Bachmat et al., 2006a).

Starting from the traditional and widespread model of back-to-front boarding, where passengers are assigned seats at or before check-in and then called to the aircraft in blocks of rows beginning at the back and working forwards, researchers have attempted to identify and test alternative strategies. Early work (Ferrari & Nagel, 2005; Marelli et al., 1998; Van Landeghem & Beuselinck, 2002) established baseline findings using simulation models which had been calibrated using empirically observed data on passenger movement times into and through the aircraft. These findings were then refined or expanded on by later work using either simulation, analytical modelling (Bachmat & Elkin, 2006; Bachmat et al.,

2006a; Bachmat et al., 2006b; Steffen, 2008a) or a combination of the two (Bazargan, 2007; van den Briel et al., 2005). Assuming that the physical dimensions of the aircraft are fixed, i.e. improvements such as wider entrances and aisles cannot be delivered, the next obvious focus of research is the arrangement of passengers prior to boarding, as this can most directly affect the efficient flow of passengers onto the aircraft. In general, elements of the traditional model are retained and researchers assume that all passengers have assigned seats prior to boarding. This premise has on occasion been questioned by commentators in the mainstream and business press (Elliot 2005), who feel that the demonstrable success of non-traditional low-cost carriers who do not assign seats to passengers should be investigated further. However, researchers often feel that, for their work to be of practical use, they are constrained by the operating culture of established airlines who have invested significantly in systems aimed at allowing customers freedom to choose their seats, often at the point of reservation (Millward & Highfield, 2005).

Whatever the perceived limitations of some of the work carried out, it is clear that many established airlines take it seriously and, in some cases, have even implemented the recommendations as a viable improvement on their previous practices (Yu, 2006; Zamiska, 2005). However, there is little consensus as to the optimum strategy in the commercial world and different airlines adopt variations on the theme in order to take into account their passengers' expectations and such other factors as luggage allowance and passengers boarding as part of a

group (Bear & Sostek, 2006). These two factors can be combined to form a proxy indication of the predominant type of passenger entering an aircraft (Lopez-Cainzos, 2006). A flight filled mainly with business travellers may be characterised by mostly single passengers, with little luggage. However, a holiday flight may be characterised by significant numbers of passengers travelling in groups, often with additional luggage (see section 3.2.4).

Despite the focus on boarding with assigned seats in the academic literature, recent years have seen several traditional carriers abandon the strategy in favour of free-for-all (FFA) boarding, i.e. an every man for himself approach to filling non-assigned seats (Reed & Yu, 2006). Interestingly, however, successful low-cost airlines, who have operated an efficient FFA boarding policy for many years, are at the same time beginning to consider alternative strategies in response to customer dissatisfaction with what many feel is ‘cattle-class’ treatment (Bigelow, 2006; Stoller, 2006).

Yet another view, taken by several commentators and industry professionals is that, due to the inherent unpredictability of a process which is heavily influenced by human behaviour, changing boarding strategies is ultimately fruitless. They feel that the variability that characterises human interaction makes too much control undesirable and possibly counter-productive. In other words, the defined parameters that influence how a hypothetical passenger moves through a

simulation model or mathematical construct break down in the face of reality (van den Briel et al., 2005). Such critics favour imposing stricter controls on more quantifiable factors such as hand luggage allowance. There is evidence to suggest that this approach may be justified, following tightening of baggage controls in response to recent security concerns (Finney, 2006). Observers note that industry sources claim these restrictions have notably improved boarding times in many cases.

It is clear from these conflicting views and observations that studies to date have failed to deliver a sector-wide panacea for what is often a financially inefficient process. Alternative strategies proposed in the academic literature tend to rely on pre-allocation of seating but have not been widely adopted by airlines operating such policies. At the same time, the rise of the budget airline has seen interest in their operating policies, which generally include non-assigned seating, increase. Whilst some researchers have now begun to address this area (Steffen, 2008b), there has not been a comprehensive response from academia. With the continuing financial pressures on airlines in today's competitive marketplace, it is clear that further investigation of the area is warranted in order to assess the possibility that it may confer an advantage on those who adopt it. It is also clear that other contributing factors, such as human interaction and hand luggage allowances, must give context to work to make it of practical relevance. Therefore, any study of this area must be holistic in nature.

1.2 Aims and Objectives of the Research Project

1.2.1 Aim of the Research

Whilst previous work has covered random boarding with assigned seats (Bachmat & Elkin, 2006; Ferrari & Nagel, 2005; Ferrari, 2005; Steffen, 2008a; Van Landeghem & Beuselinck, 2002) and noted its strong performance against more structured strategies, there has been significantly less emphasis placed on non-assigned strategies, commonly referred to as free-for-all (FFA) boarding (see section 2.6.6).

An earlier study included some very limited analysis of this area (Ferrari & Nagel, 2005) but the conclusions drawn were vague and the area has been identified as the immediate challenge for airplane boarding studies in later work (Bachmat et al., 2006b). Hypotheses relating to the success of this strategy, or lack of it, have also been advanced (Van Landeghem & Beuselinck, 2002). An as yet unpublished study applies principles of statistical mechanics to model FFA boarding and draws some limited conclusions regarding its performance against random boarding with assigned seating (Steffen, 2008b). However, the comparison assumes a very simple model of seat choice. An opportunity now exists to use simulation as a tool to further investigate this area and expand on the conclusions currently laid down. The most obvious point of comparison is

with the random assigned seats strategy, as it allows the question of whether non-assigned seating offers any advantage to be focused on.

The software aspect of this work will involve the use of multi-agent systems (MAS). Previous studies (Parker et al., 2003) have asserted that MAS systems allow emergent behaviour to be studied through the simulated interactions of agents and their environment rather than through explicit definition of interaction conditions. In other words, MAS is a useful and appropriate tool for studying complex systems. This claim will be assessed, with a key aim of the work being to investigate the appropriateness of MAS systems in providing a flexible solution for implementing simulation mechanisms able to deal with stochastic interactions between entities (e.g. differing passenger responses to stimuli, different rates of hand luggage) in this problem domain.

1.2.2 Research Question

The research question will be:

Can multi-agent systems adequately model stochastic uncertainty in a simulation study investigating the effect of pre-assigned seating on free-for-all aircraft boarding?

1.2.3 Contribution to Knowledge

This study will address an area of potential interest to several audiences, including those from the commercial sector as well as those concerned with academic research in this particular field and more generally.

Firstly, the work will be of interest to commercial airlines. The findings returned from the simulation may assist traditional airlines to inexpensively assess the impact of implementing an alternative boarding method on their flights. This could improve profitability and thus long-term sustainability in a competitive market and avoid the need for extensive human-based trials. Equally, low-cost airlines may find that their practices are validated through comparative analysis against more traditional models. The extension of this study to take into account a wide variety of aircraft types may also provide an insight into how the low-cost model could be extended to new operating environments and may be of interest to aircraft manufacturers in identifying possible new client areas as well as to traditional airlines seeking to diversify and increase their product offerings and market share.

Secondly, the work will also be of interest to other researchers interested in the area of airline boarding. As the work will build upon the findings of previous studies, researchers will be able to follow in the literature a clear development of currently published ideas. In particular, this work will allow questions and hypotheses regarding non-assigned seating strategies that

have been raised in previous studies to be addressed and answered. It will also provide data that can form the baseline for further investigations and comparisons, particularly in relation to multi-aisle aircraft.

Finally, the work may be of interest to researchers in the field of computing as it will demonstrate the suitability of multi-agent systems for implementing simulation studies containing elements of stochastic uncertainty, e.g. the decisions made by passengers and their emergent behaviour in response to changing system states. The applicability of more general simulation theory to a multi-agent environment will also be investigated, particularly with regard to distinct models of simulation such as discrete event simulation or agent-based simulation. It is hoped that the work will demonstrate that multi-agent systems provide a flexible and highly scalable solution for simulation modelling that will allow general parameters and rules to be applied across a variety of model scenarios. In addition, this work will demonstrate how human interactions and responses to stimuli can be modelled using software and how multi-agent systems make this far more possible and reliable than traditional single-agent or monolithic systems can. Another area of interest for computing research will be the development strategy used and how it incorporates reuse of software components for specialised purposes (see section 3.2.1).

1.3 Overview of the Dissertation

Chapter 2 evaluates the existing range of knowledge within the domain of interest, describes the research methods that have been used in previous work and identifies challenges that have yet to be fully addressed.

Chapter 3 describes and justifies the research methods used by this study, placing them in the context of the existing body of work within the problem domain. The data collection strategy is outlined and a discussion of methods of analysis presented.

Chapter 4 presents and discusses experimental results and provides a preliminary answer to the research question formulated in Chapter 1.

Chapter 5 outlines the conclusions reached and undertakes a critical review of the project approach and objectives. Recommendations for future work are made and their applicability to diverse fields of interest discussed.

Chapter 2 Literature Review

2.1 Overview

Industry figures show that boarding times have increased by approximately 50% since the mid-1970s (Marelli et al., 1998), leading to more unprofitable time spent on the ground. This work, which observed the boarding process on a Boeing 757 variant aircraft, found that passenger boarding using traditional strategies takes 26 minutes in total. The same study hypothesised that alternate strategies could yield up to a 65% improvement on this time – a saving of approximately 17 minutes. Later academic studies (Van Landeghem & Beuselinck, 2002) also identify a reduction in boarding time as desirable following a trend of decreasing prices and increasing pressure to gain a commercial advantage through improved quality and passenger service. Similar current boarding times as those put forward in the industrial study, ascertained through interviewing airline staff, are quoted with significant reductions in boarding time anticipated, reducing the process down to approximately 10 minutes. This is only slightly higher than the industry-proposed figure quoted previously but was put forward with the caveat that implementation may require an unrealistic degree of control over the boarding process.

2.2 Definition of the Problem Area

The scope of the problem area has been defined within the literature as starting from when passengers enter the aircraft, in a pre-determined order, to the time when they are all seated with no further impediments to take-off (Marelli et al.,

1998; Van Landeghem & Beuselinck, 2002). This definition, whether explicitly stated or not, appears to be common across all published studies and provides a firm basis for comparison of different work, ensuring that variable factors are relatively easy to identify and assess.

There are certain fixed elements in the majority of commercial boarding strategies, such as the preferential boarding group (e.g. passengers in 1st class). Customer expectations in this regard mean that it is unlikely that such patterns can be changed (Van Landeghem & Beuselinck, 2002) and so any improvements to boarding strategies will be limited to the main, economy section of the aircraft.

2.3 Primary Research Methods

Identifying solutions requires the ability to model and assess the impact of possible changes. Within the published literature, there are two clear and discrete approaches to this. The first makes use of simulations, whilst the second uses mathematical modelling to take a more analytical approach. A third approach combines both of these methods. This mix of methods often provides useful validation of work that has gone before, ensuring that later studies can build upon a robust baseline level of understanding.

2.3.1 Simulation Studies

The main output of the initial study was a proprietary discrete event simulation model, intended to evaluate strategies that would decrease

turnaround times and thus increase profitability (Marelli et al., 1998). A strategy, within the context of this work, is defined as the way in which passengers are ordered for the boarding process. A later study (Ferrari & Nagel, 2005) took the same approach. Simulation studies are used when the analysis focuses on a known strategy with variable elements. In other words, a simulation will not identify unknown strategies, but allows strategies identified elsewhere to be modelled and the full impact of various interacting factors assessed. An innovative study in this field (Steffen, 2008b) made use of the principles of statistical mechanics to model the free-for-all boarding process. Statistical mechanics are more commonly used by physicists to describe the motion of particles when under the influence of a force and its application to the domain of aircraft boarding is extremely novel. However, it must be noted that any simulation cannot completely capture the actual freedom of choice available to human beings (Kirchner et al., 2003).

Academic simulation studies can be implemented using commercially available simulation software, such as Witness (Lopez-Cainzos, 2006) and Arena (Van Landeghem & Beuselinck, 2002) – built around the specialised simulation language SIMUL8 - which reduces the need for technical development skills (Bapat & Sturrock, 2003; Takus & Profozich, 1997) but adds a requirement for training to become familiar with the interface and method of operation. Other studies utilise bespoke simulation applications

(Ferrari & Nagel, 2005). There is little discussion of the implementation of such applications but it is noted that they are cellular representations of the aircraft environment, where actions are carried out at discrete time points by independent agents representing passengers. The specific nature of this kind of development is likely to minimise the potential for reuse, although a well-planned development strategy should allow for future extension of the simulation model.

There are many ways to carry out a simulation study, but the method that most obviously matches the format of all the studies above is Discrete Event Simulation (DES). DES software combines theoretical paradigms of queuing behaviour with an analysis of random behaviour. This approach allows quantification of potential changes without the need to engage in expensive and disruptive in-service trials (Baines et al., 2004; Banks, 2000; Ferrari, 2005; Marelli et al., 1998; Maria, 1997).

In DES, activity is modelled in a linear series of time-steps, with a known start and end-point (Carson, 2005). The number of time-steps taken to move between these two points will vary, depending on the activities of other elements within the simulation (Ferrari & Nagel, 2005; Kirchner et al., 2003; Marelli et al., 1998; Schriber & Brunner, 2006). So, to put this description in context, an individual passenger, entering the aircraft through

the front door and with destination defined as seat *X*, will take longer than the optimal time to reach their destination if they encounter interferences along the way.

The strength of DES is that it provides visibility of passenger interactions and how they translate to the overall efficiency of any particular boarding strategy. Complex interaction phenomena such as self-organising lane formation and herding (Helbing et al., 2005) are ignored in favour of a cellular based environment, where each passenger occupies a defined cell and can only move from that cell if an unoccupied and adjacent cell is available (Kirchner et al., 2003; Van Landeghem & Beuselinck, 2002). It is believed that cellular representations of space perform very similarly to continuous space representation (Ferrari & Nagel, 2005).

2.3.2 Analytical Studies

Other work first established an analytical model – a mathematical construct able to identify efficient solutions in a more systematic, deterministic way than traditional simulation (Bachmat et al., 2006a; Bachmat et al., 2007). These studies focus on clearly identifying model parameters, such as hand baggage allowance, before investigating possible strategies within this framework. Characteristically, they do not produce a single optimal solution, but a group of related strategies with similar performance. They can be heavily constrained by assumptions that vastly simplify the actual physical

environment of aircraft boarding, e.g. that all passengers are infinitely thin (Bachmat et al., 2007), yet the results returned by such work are often very well aligned to the conclusions drawn through simulation alone.

There is no discussion of the practical implementation of analytical models in the literature. It is assumed that, due to the enormous range of possible output from this kind of investigation, high-performance computing facilities would be required to make them feasible. Work based partly on analytical optimisation algorithms assumes a minimum of 10,000 iterations of the model, with a realistic timescale for achieving a truly optimal solution being greater than the age of the universe itself (Steffen, 2008a).

2.3.3 Hybrid Studies

Several studies began from an analytical standpoint, in order to identify innovative strategies. Results were then further investigated and validated using a simulation model, to provide a more realistic reflection of the aircraft environment and the interaction of elements within it (Bazargan, 2007; Steffen, 2008a; van den Briel et al., 2005).

Implementation issues discussed above relate equally to this approach.

2.3.4 Comparison of Research Methods

Analytical studies are generally based upon a far simpler model than simulation studies (Ignall et al., 1978). Often, this means building in assumptions that do not translate well to a real-world simulation, reducing everyday complexity and variability into a linear system (Howrey & Klein, 1971; Shanthikumar & Sargent, 1983). More realistic models are generally too complex to solve, even with the significant processing power available to researchers (Steffen, 2008a). However, as a simple relation of inputs and outputs, analytical models can provide a useful investigative technique that may uncover previously unknown solutions. Simulation is a far more targeted strategy, incorporating more realistic parameters to provide insight into the performance of pre-defined scenarios under normal operating conditions characterised by a certain degree of uncertainty (Shanthikumar & Sargent, 1983).

As such, both are useful approaches. However, the full benefits of both techniques can be realised when they are used in conjunction. A problem domain can be fully investigated through analytical studies and results validated using simulation to demonstrate practical applicability. In this way, researchers can be confident in their investigative techniques and ensure that all novel solutions are explored.

2.3.5 Multi-Agent Systems within the Problem Domain

Existing simulation studies within the domain do not appear to have made used of multi-agent systems, opting instead to utilise the bespoke capabilities of specialist languages such as SIMUL8, usually in conjunction with commercially available platforms (see section 2.3.1). However, multi-agent systems have been used as a platform for discrete event simulations within closely related domains such as studies of crowd behaviour (Moulin et al., 2003) and pedestrian movements (Dijkstra, Jessurun & Timmermans, 2002). These studies have successfully utilised a multi-agent approach to simulation to demonstrate emergent behaviours arising from the interactions of multiple independent agents representing human beings. There is also evidence to suggest that discrete event simulations have been successfully implemented using multi-agent systems in a variety of other domains such as studies of predator and prey interaction (Logan & Theodoropoulos, 2001), the propagation of pathogens within the body (Luke et al., 2004), the response of distributed systems to infiltration attacks (Vincent et al., 1998) and the flow of items through manufacturing systems (Gianni, 2008). Although none of these examples utilise agents representing human beings, the underlying principles of the investigations are the same as those found within the domain of airline boarding – namely, that discrete agents operating under certain defined behavioural parameters interact within a system in a manner that is not possible to predict from the individual states of the agents themselves. In fact, it has been asserted that the majority of

simulation studies with elements of agent interaction are implemented as multi-agent systems with discrete event capabilities (Parunak, 1997). Bearing this in mind, it would seem that work carried out across a broad range of domains of interest indicates that the use of multi-agent systems within the area of aircraft boarding simulation is a viable strategy. So long as a multi-agent system is implemented with discrete event mechanisms, such as a simulation clock and centralised data collection – capabilities which are necessary to run and analyse a process based on discrete time-steps – it is generally accepted to be a useful tool in simulation studies (Gianni, 2008) (see section 3.2.2).

2.4 Factors Affecting Boarding Time

Researchers have identified several factors of interest that may impact on overall passenger boarding time. Some of these take the form of systematic obstructions, whilst others concentrate on performance in the presence of non-standard, unpredictable behaviour.

2.4.1 Hand Luggage Allowance

The initial work (Marelli et al., 1998) attributed increased boarding times to several factors, one of which in particular – an increase in hand luggage – has been concentrated on extensively in later studies (Bachmat et al., 2006a; Ferrari & Nagel, 2005; Lopez-Cainzos, 2006; Steffen, 2008a; Van Landeghem & Beuselinck, 2002; van den Briel et al., 2005, Steffen 2008b).

2.4.2 Human Interaction

Earlier work has been criticised for emphasising the importance of the interior layout of the aircraft, whilst underestimating the impact of passenger interaction (Van Landeghem & Beuselinck, 2002). Later work focuses on this perspective and investigates what are termed as ‘disturbances’ e.g. passengers boarding in the wrong order, reinforcing the importance of human behaviour on boarding times ((Ferrari & Nagel, 2005).

2.4.3 Aircraft Occupation Level

Several studies have touched upon the occupation level of the aircraft (Ferrari & Nagel, 2005; Van Landeghem & Beuselinck, 2002), although this is ultimately not considered to be significant as it is assumed that any level of occupancy below 100% will offer automatic advantages by reducing the number of possible obstructive interactions between passengers. In other words, lower levels of aircraft occupancy are more favourable for faster boarding times. Practically and financially, of course, this is not a desirable scenario.

2.5 Aircraft Geometry

Most studies have concentrated on single-aisle aircraft (Bazargan, 2007; Ferrari & Nagel, 2005; Marelli et al., 1998; Steffen, 2008a; Van Landeghem & Beuselinck, 2002; van den Briel et al., 2005). However, some work carried out

using analytical models produced results that could be applied to different aircraft types (Bachmat et al., 2006a; Bachmat et al., 2006b; Bazargan, 2007). Where this is investigated further, it is assumed that once passengers have been directed to the appropriate aisle for their seat, the problem decomposes into a series of single-aisle scenarios.

Most studies assume that passengers board through the front door of the aircraft only (Bachmat et al., 2006a; Bachmat et al., 2006b; Bazargan, 2007; Ferrari & Nagel, 2005; Kirchner et al., 2003; Van Landeghem & Beuselinck, 2002; van den Briel et al., 2005), although a limited assessment of the impact of using the back door as well has previously been carried out (Marelli et al., 1998). This early work established that using both doors to load the aircraft could reduce the time taken to complete the process by as much as five minutes.

2.6 Boarding Strategies

Several strategies have been proposed within the published literature, with each believed to be more efficient than the established back-to-front block boarding strategy favoured by many traditional carriers. This model of boarding assigns passengers into a number of groups, depending on their row number and then boards these groups in order from the back of the aircraft working forwards (van den Briel et al., 2005), as shown in Figure 2 below.

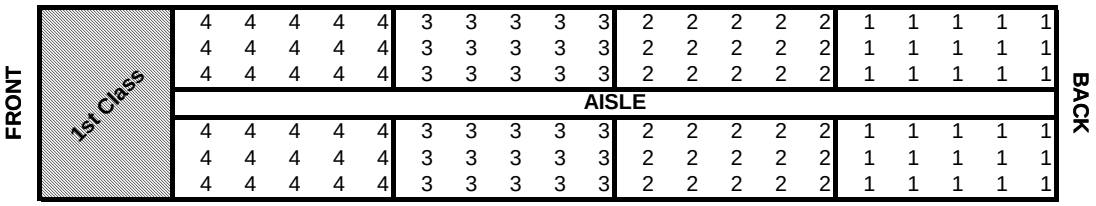

Figure 2 Back-to-Front 4-Block Boarding.
Adapted from van den Briel et al., 2005.

It has been found that this strategy is extremely inefficient as it concentrates conflict between passengers in reduced areas of the aircraft (Van Landeghem & Beuselinck, 2002). Two main types of conflict, or interference, have been identified (van den Briel et al., 2005): Seat Interference and Aisle Interference. Seat interferences are caused by passengers occupying seats close to the aisle, before those closest to the window are filled. In this situation, passengers with seats assigned close to the windows take longer to reach them as they must either wait for the obstructing passenger to move out of their way, or climb over them. The effect is, of course, amplified if two obstructing passengers are already in position. Aisle interferences occur when passengers pause to stow luggage in the overhead bins and thus block the progress of all passengers behind them. The time attributed to this type of interference is directly correlated with the mean amount of hand luggage allowed.

Alternative strategies attempt to either reduce the number of such interferences that occur, or mitigate their affect by distributing passengers more evenly throughout the aircraft.

2.6.1 Window-Middle-Aisle

The initial study (Marelli et al., 1998) stated that a non-traditional strategy referred to by them as 'outside-in' and later as Window-Middle-Aisle (Steffen, 2008a) (i.e. boarding all passengers occupying a window seat first, followed by those in the middle and lastly all aisle seats as shown in Figure 3 below) offers significant advantages over the traditional back-to-front strategy. This strategy distributes boarding passengers across the full length of the aircraft and so the probability of aisle interferences should reduce, with seat interferences being avoided altogether.

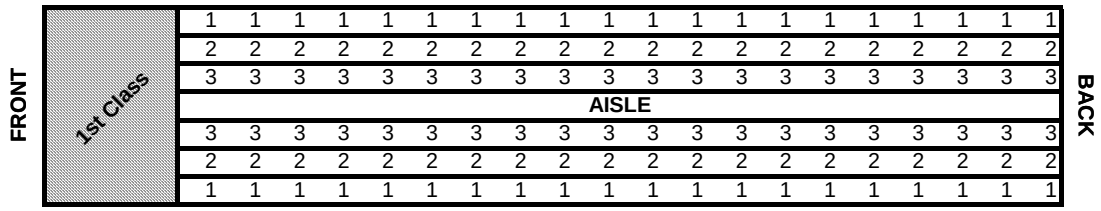

Figure 3 Window-Middle-Aisle Boarding.
Adapted from Ferrari & Nagel, 2005.

2.6.2 Seat Group

Seat Group boarding is a more refined Window-Middle-Aisle strategy, which incorporates the same principles but a greater number of groups (Ferrari & Nagel, 2005). In the same way as Window-Middle-Aisle, this strategy avoids seat interferences and, by reducing the group size yet further, also minimises the probability of aisle interferences.

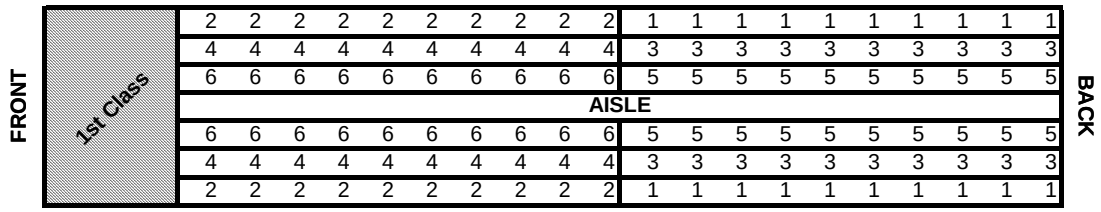

Figure 4 Seat Group Boarding.
Adapted from Ferrari & Nagel, 2005.

2.6.3 Reverse Pyramid

Reverse pyramid is a hybrid strategy, that combines elements of Window-Middle-Aisle with Back-to-Front (van den Briel et al., 2005). In common with the Seat Group strategy, it avoids seat interferences and minimises the probability of local conflicts in the aisle. Whilst it is marginally slower than Seat Group, it requires fewer groups and thus it can be argued that it is no less efficient (Ferrari & Nagel, 2005).

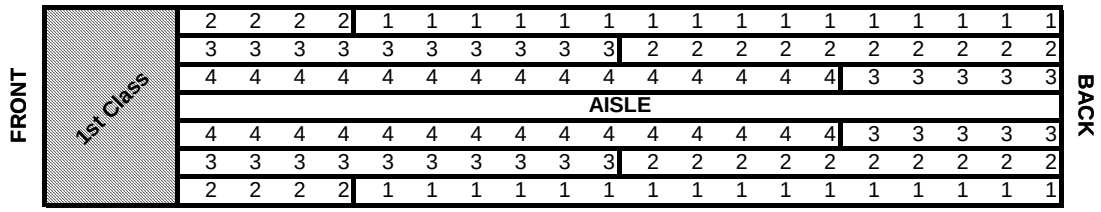

Figure 5 Reverse Pyramid Boarding.
Adapted from van den Briel et al., 2005.

It is believed that realistic efficiency can be achieved through these hybrid strategies by combining control over the size of each boarding group with the gap between groups (Ferrari & Nagel, 2005; Van Landeghem & Beuselinck, 2002).

2.6.4 Strict Ordering

The most efficient strategies demand a high level of control and essentially amount to calling passengers individually, as shown in Figure 6 below (Ferrari & Nagel, 2005; Van Landeghem & Beuselinck, 2002). It is commonly accepted in the literature that such strict ordering strategies are unrealistic (Ferrari & Nagel, 2005; van den Briel et al., 2005) due to the organisation required and the problems inherent in separating groups, e.g. families (Ferrari & Nagel, 2005). More recent work, however, does state that a strict ordering policy has been successfully implemented (Steffen, 2008a).

The advantage of this strategy is that it reduces the probability of any kind of interference to its lowest possible level by minimising control of the boarding group size to a single passenger and maximising control of the between-group interval.

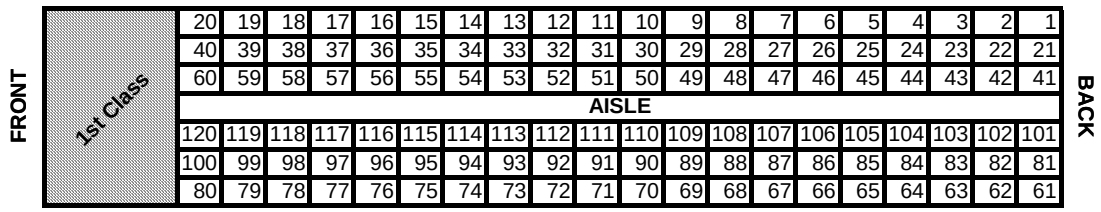

Figure 6 Strict Ordering Boarding.
Adapted from Ferrari & Nagel, 2005.

2.6.5 Random with Pre-Assigned Seats

A number of studies have found that random boarding outperforms several structured strategies, including but not limited to the widely used back-to-

front block strategy (Bachmat & Elkin, 2006; Ferrari & Nagel, 2005; Ferrari, 2005; Steffen, 2008a; Van Landeghem & Beuselinck, 2002). One study in particular, found random boarding to be close to optimal (Bachmat et al., 2006a).

Figure 7 Random Boarding with Pre-Assigned Seats

2.6.6 Free-for-All

FFA boarding has not received the attention that other strategies have within the academic literature. An early study hypothesised that FFA would initially perform similarly to random boarding with pre-assigned seats but would degrade as the process ran its course due to the final passengers attempting to locate the remaining free seats (Van Landeghem & Beuselinck, 2002). This hypothesis appeared to be confirmed by a later simulation study (Ferrari & Nagel, 2005), although the conclusions drawn were limited. Later work, however, notes that FFA boarding gives some of the fastest boarding times in the airline industry (Steffen, 2008b) and, in a simplified comparison, notes that it is almost equal in performance to random boarding with pre-assigned seats. The earlier hypothesis that the process would decrease in efficiency over time was confirmed by this study. However, with little degradation in performance in comparison to other

strategies, it is possible to infer that FFA is in fact more efficient and cost effective, as it does not require investment in seat allocation systems.

2.6.7 Summary

The key feature of the majority of the currently published work is the high level of agreement in the relative efficiencies of the various proposed boarding strategies, despite the use of different investigative techniques and varying levels of complexity in the models. Given this level of agreement, it can be assumed that further investigation using either analytical or simulation techniques will be unlikely to significantly improve our understanding of the scope for improvement on widely used assigned-seating strategies. With this in mind, FFA boarding, which is less well understood academically but has demonstrable success in the commercial world, is the next clear area of study – an assertion already laid down within the literature (Bachmat et al., 2006b).

2.7 Limitations of the Current Findings

The majority of the studies of aircraft boarding rely heavily on certain underlying assumptions or inferences drawn from earlier work (Bazargan, 2007; Ferrari & Nagel, 2005). Only two studies appear to have systematically collected data on boarding procedures from empirical observations in order to calibrate their models (Van Landeghem & Beuselinck, 2002; van den Briel et al., 2005) and, of these, only one has validated their conclusion through a practical implementation

of the proposed strategy (van den Briel et al., 2005). Earlier work was also partially validated using human-based simulation (Marelli et al., 1998) but includes assumptions relating to a passenger flow rate of 9 passengers/min (i.e. the rate that passengers arrive through the door) that have been criticised as being over-optimistic in a later study (Van Landeghem & Beuselinck, 2002), which found that even the most efficient strategies only equate to a flow rate of 6.6 passengers/min. The need for more accurate data relating to the flow rate of passengers into the boarding process has been identified (Lopez-Cainzos, 2006). Within the published literature, it is commonly accepted that more accurate and reliable results could be obtained following collection and comprehensive analysis of empirical data relating to all aspects of the boarding process, from luggage loading to passenger interaction (Steffen, 2008a; van den Briel et al., 2005).

2.8 Hypothesis

Analysis of the literature shows that simulation is a valid and accepted method for studying performance within the domain of aircraft boarding. Multi-agent systems, by their nature, appear well suited to model the interactions of the discrete agents and entities on which a simulation study is by necessity based. This leads me to the hypothesis that a multi-agent simulation environment will produce results that are appropriate for comparison with studies undertaken using proprietary simulation software. This will be tested against a defined baseline from the published literature.

The existing literature also clearly establishes the grounds for the secondary hypothesis: that FFA boarding will prove to perform less well than random boarding. This hypothesis will be tested under a number of different parameters, representing the stochastic variables that influence boarding times, in order to ensure that it can be addressed robustly across the range of operation specified for the domain.

Chapter 3 Research Methods

The research question posed for this project (see section 1.2.2) cannot be answered by existing studies within the problem domain. Due to this, an empirical approach must be taken, which requires that new evidence is provided to allow a meaningful conclusion to be drawn.

In order to achieve this, a software-based simulation study will be carried out. This approach is consistent with much of the work already carried out within the project domain (see section 2.3) and will ensure that the results and conclusions of this study provide an easy point of reference to earlier studies and meaningfully extend current knowledge.

Following the approach taken in several previous studies, the model will be implemented as a discrete event simulation (DES) (see section 2.3.1). Alternative methods, e.g. algorithmic investigative techniques (Steffen, 2008a) and Agent-Based Modelling (Macal & North, 2005) have been ruled out in favour of DES. The first is too costly in terms of processing and the second is too focused on agent interaction rather than the process itself (Banks, 2000). DES is seen as the most feasible approach given the limitations of other modelling techniques and the availability of baseline data to inform development of the model (see section 3.3.1).

3.1 Model Specification

DES allows a controlled experimental approach to be taken. The study conforms to the characteristics of controlled experiment as:

- The simulation model is a well-defined and simplified version of reality, with a high level of control possible (Banks, 2000; Maria, 1997; Robinson, 2005)
- Controlled manipulation of variables will be possible (e.g. luggage distribution) and the effects of such variables observable (Chick, 2006; Maria, 1997; Nance & Sargent, 2002)
- Entities within the model will interact according to defined rules, with elements of stochastic uncertainty included in order to ensure that the population is representative of populations utilised in prior work (Carson, 2005; Maria, 1997)

In order to take full advantage of this controlled environment, it must be fully specified and understood. The high-level concept diagram overleaf (
Figure 8) details the key elements described in the baseline study, which will also form the conceptual framework for this project.

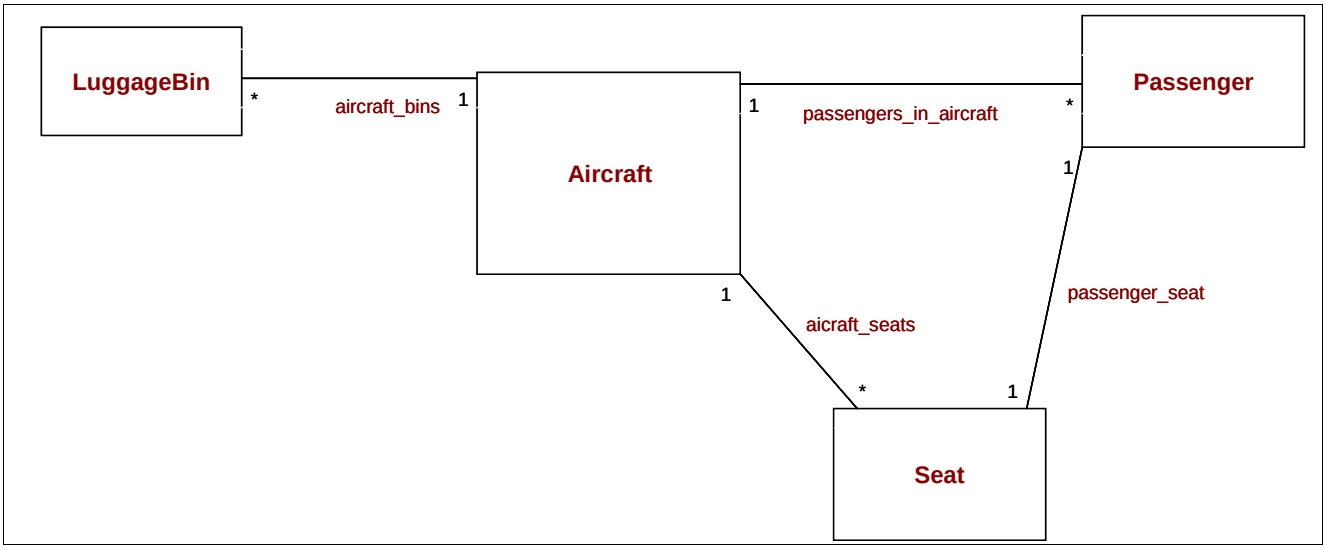

Figure 8 Key Elements for Simulation

These elements will form the basis for the implementation of agents within the DES environment. However, their interactions will be governed by a number of parameters, which will also be a direct replication of those used in the baseline study.

3.1.1 Flow Rate

The frequency with which passengers enter the aircraft for the simulation, also known as the flow rate, has been calculated at 1 every 6 seconds, or 10 per minute (Van Landeghem & Beuselinck, 2002). The rate was derived through observations of the aircraft boarding process combined with figures from an industry database and thus has a firm basis in reality. This means that, while passengers are entering the model, each time step will be at least 6 seconds long.

Measurement of the boarding process will begin from the moment at which the first passenger enters the aircraft and will end at the point at which the final passenger takes their seat, with individual measurements also being taken for each passenger as they complete their movements.

3.1.2 Movement Times

Passenger actions have been classified into three elementary, measurable movements: *passing one row*, *install in seat* and *exit from seat into* aisle

(Van Landeghem & Beuselinck, 2002). This classification, along with the associated timings (see Table 1), was again based on observation of boarding activities and data contained within an industry database.

Table 1 Movement times for triangular distribution. Adapted from van Landeghem & Beuselinck, 2002.

Movement Times	Min	Mode	Max
Passing one row	1.8	2.4	3.0
Install in seat	6.0	9.0	30.0
Exit from seat into aisle	3.0	3.6	4.2

Passenger movement times can be influenced by a wide variety of factors, such as age, fitness and height, which are beyond the scope of this study (see section 5.2 on proposed further work). Due to this, the times contain an inherent degree of variability, which can be modelled statistically as a triangular distribution, without the need to incorporate any understanding of the reasons for the variability. This technique takes as input the minimum, maximum and most commonly observed values (the mode) for each movement and can be used to produce a random time within this range (Weisstein). This approach and the movements times defined in Table 1 will be adopted for this study.

3.1.3 Hand Luggage Distribution

Hand luggage has been identified as a major cause of delay during the boarding process. It is therefore important to model its effect within the simulation. The distribution of hand luggage will be modelled according to two ratios, following the approach of the baseline study (Van Landeghem &

Beuselinck, 2002). These ratios relate to two loading states: normal and high (see Table 2), with individual passengers assigned a random number of items depending on the ratio under observation.

Table 2 Hand Luggage Distribution under different loading conditions. Adapted from van Landeghem & Beuselinck, 2002.

Hand Luggage Distribution	1 item	2 items	3 items
Normal Load	60%	30%	10%
High Load	20%	60%	20%

The use of two different ratios models both the expected performance under normal conditions, where the majority of passengers would be expected to carry on a single piece of hand luggage, along with performance under specialised conditions such as early morning business flights where passengers may carry on laptop computers and suit holders along with their normal luggage. Following the approach of the baseline study, bin occupancy rates will also be modelled in the simulation, which will ensure that the full effect of the two distribution ratios can be assessed. All passengers will carry at least one piece of hand luggage.

3.1.4 Extensions to the Model

The model used for this study will also be extended beyond the scope of that used for the baseline study (Van Landeghem & Beuselinck, 2002). This will increase the richness of the data generated, allowing the research question to be more fully addressed through investigation across a range of scenarios.

The baseline study does not explicitly model the effect of passengers travelling together as a group, although it does hypothesise that this will decrease the effectiveness of all boarding strategies (Van Landeghem & Beuselinck, 2002). However, it is realistic to assume that, in real-world situations, a certain proportion of passengers will travel together in a group. This is particularly true in situations such as charter holiday flights and so inclusion of the group effect in the model will increase its relevance to the airline industry and thus its contribution to knowledge. In terms of addressing the research question, inclusion of the group effect will be of assistance as it will allow another stochastic element to be modelled and the appropriateness of a multi-agent approach to implementing it assessed. A variable element within the model will specify the probability that passengers will enter the aircraft as part of a group, with a 0% probability equating to an aircraft full of solo travellers and a 90% probability equating to a situation where every passenger enters the aircraft as part of a group containing at least one other passenger. Thus the concept diagram shown in Figure 8, must be extended to incorporate the passenger-group relationship as shown in Figure 9, overleaf.

The baseline study, in common with the majority of studies in the domain, is focused on single aisle short-haul aircraft, holding approximately 130 passengers (Van Landeghem & Beuselinck, 2002; Ferrari & Nagel, 2005;

van den Briel et al., 2005). The model used for this study will be extended to cover a range of aircraft types, including multi-aisle long-haul aircraft. As with modelling the group effect, this will increase the relevancy of the model to a larger section of the airline industry as well as extending the contribution to knowledge made by this study. The model will allow for a maximum aircraft capacity of 240 economy class passengers, equating approximately in size to a Boeing 747 variant aircraft (SeatGuru).

One further addition to the model is required – a set of behavioural rules that govern free-for-all boarding. This is necessary in order to reflect the fact that free-for-all boarding is not simply a random process, but is driven by passenger perceptions of which seats are the most desirable, attention span and ease of access amongst other human-centred factors. As this kind of behaviour is not modelled in the baseline study, a set of rules proposed in a later study will be incorporated (Ferrari & Nagel, 2005). This study is itself a comparison with the baseline in use here and, as such, is a natural extension of the earlier work. The rules that will define passenger behaviour in the free-for-all boarding scenario are:

- window seats and seats next to the aisle will be preferred
- free rows will be preferred
- before sitting down, the passenger will ensure that there is no better place in the next three rows

- if passengers are queuing, there will be a possibility (P=0.3) that they will lose patience and accept a less desirable seat at their current location
- passengers will not change their direction of travel to find seats

It is assumed that for both free-for-all boarding and random boarding with assigned seats passengers within the simulation will not deviate from the rules governing that particular scenario. For example, passengers entering the aircraft with an assigned seat will never sit in any other seat. This is a departure from the approach taken in the baseline study, where this type of non-standard behaviour is included in the model (Van Landeghem & Beuselinck, 2002). However, no information is provided on how this behaviour has been modelled and, given that it is not systematic, it seems reasonable to assume that the effect of excluding it from the model may be less than that of including it at the wrong magnitude. Allowances will be made for the uncertainty this brings when comparing results against the baseline during the analysis phase (see section 3.4.2).

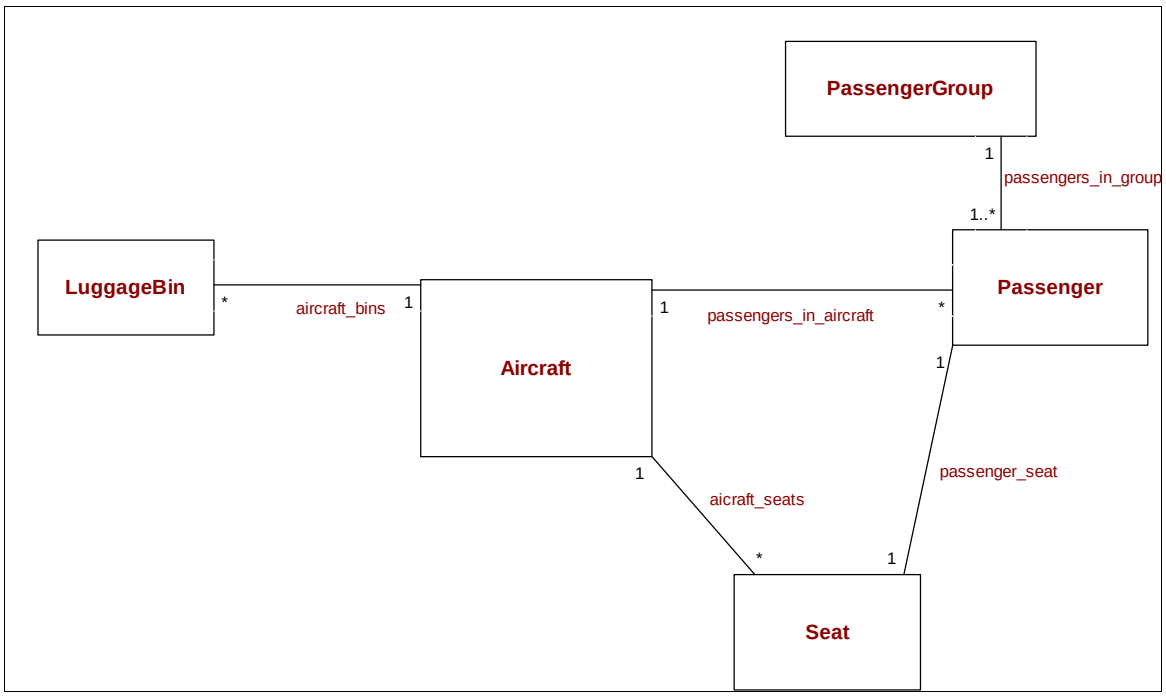

Figure 9 Key Elements for Simulation, Extended

3.1.5 Alternative Sources of Data

Initially, the strategy for this study was to populate and calibrate the simulation model using data gathered from airlines via a survey. However, before survey design took place, this approach was revised given the availability of unambiguous data in earlier studies. It was felt that it would be more beneficial to spend time addressing the computer science aspects of this project than in gathering data from industry sources that was already available elsewhere. Had the project been focussed more on process engineering, the reverse may well have held true.

3.2 Implementation

A multi-agent system will allow the different, discrete entities required for DES models to be implemented and their interactions monitored (see section 2.3.5). The system will be developed using the Java programming language, as its object-oriented principles will allow the necessary multi-agent environment to be supported (Robinson, 2005). This approach is consistent with other studies undertaken within different domains, such as studies of pathogen propagation and distributed security protocols (see section 2.3.5), which have successfully used Java to implement multi-agent discrete event simulations and recommend it for its portability and strict definitions of data types that allow easy replication of scenarios (Luke et al., 2004; Vincent et al., 1998). Its portability, in particular, can be seen as a key element in justifying its use over other object-oriented technologies such as the C language or one of its variants. A portable application

will ensure that the simulation environment is available to all interested parties with minimal deployment impact.

The majority of the elements specified within the conceptual framework will be implemented as bespoke classes using JBuilder, a Java IDE. However, there will be some supplementary components sourced from external bodies. A further discussion of key Java elements can be found below, followed by a high level diagram detailing the interaction of those elements and their relationship to the model entities specified in Figure 9, above (see Figure 10).

3.2.1 Component Reuse

The inheritance mechanism within Java facilitates a high level of reuse. This will aid considerably in the implementation of mathematical and statistical algorithms used to model stochastic variables. In particular the Java `math` package will be used to support random generation of numbers for use in events that are dependent to some degree on a probability distribution, e.g. passenger group generation.

Timings for passenger movements within the model will be generated as a random number within the range specified by the triangular distribution (see section 3.1.2). To do this, a java component, `SimKit`, developed by the University of Calgary's TeleSim project will be used (TeleSim Project). This requires the `SimKit` package to be imported into the java project in order

that a class designed to model the triangular distribution and generate random numbers based on this distribution, may be utilised.

Reuse of components is appropriate for this project, as it is an area of ongoing interest for the computing research community (Desouza et al., 2006). It will also ensure that specialised statistical techniques used in the baseline study can be incorporated into the work, ensuring that valid comparisons can be made. This may not have been the case with a bespoke implementation of these techniques due to a lack of expertise in the area.

3.2.2 Control

Although the simulation will involve numerous independently interacting elements, a single control element will be implemented (see Figure 10). This will act as the simulation clock, ensuring that elements are instantiated at the correct point, e.g. the entrance of individual passengers into the aircraft model according to the specified flow rate (see section 3.1.1). This follows general practice when running discrete event simulations (Schriber & Brunner, 2006).

The control element will also be the single point of contact for all communications from independent agents within the system. In this way, agents will communicate local information to a central point for storage,

allowing efficient collection of data pertaining to individual passenger's experience of the boarding process alongside global data for the process as a whole.

3.2.3 User Interface

A simple, cellular-based graphical user interface, built using the Java `awt` package, will form the model of the aircraft presented to the user (see Figure 10). Passengers will interact within this environment when the simulation is run, with positions being indicated graphically through cell colour changes.

The GUI will be generated dynamically according to user defined parameters governing the size of the aircraft model. These parameters will be entered on an initial menu screen, also developed using the `awt` package.

3.2.4 Model Inputs

The size of the aircraft model within the DES environment is dynamic and will be specified by the user as a particular aircraft type before the simulation is run (see Figure 10). Due to this, the size of the list of passenger details that will form the input for the model will also vary, requiring the list to be dynamically generated at the point of execution of the

model. A mechanism will exist to build in the luggage loading ratios as details appended to each passenger.

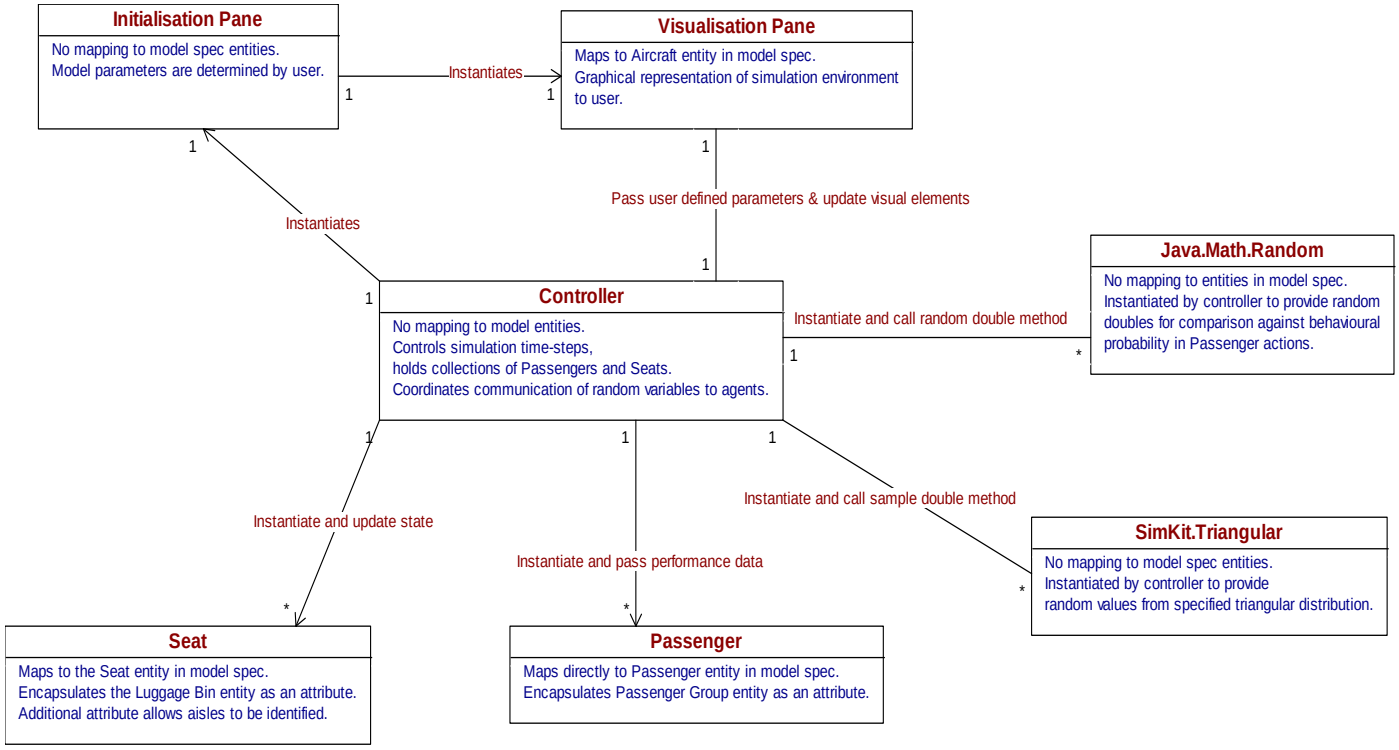

Figure 10 Java Elements of Model Implementation

3.3 Data Collection

In order to fully address the question that directs an empirical study, the data that is generated must allow for analysis that is of both sufficient breadth and detail. In other words, careful attention must be paid to both the richness of the data generated and its resolution, in order to ensure that it is fit for its intended purpose. This study in particular also requires a defined baseline in order for the question to be appropriately answered and this will inevitably dictate the richness and resolution of the new data that will be gathered.

3.3.1 Baseline

The literature review (see section 2) investigated previous work in the problem domain, identifying studies which have been used extensively as a basis for further work. One study in particular (Van Landeghem & Beuselinck, 2002) has been extensively quoted in later work (van den Briel et al., 2005; Bachmat et al., 2007) and has also been used previously as the baseline for a comparative study (Ferrari & Nagel, 2005). This study is especially useful as a source of baseline data, as it also contains a hypothesis relating to the effectiveness of free-for-all boarding, thus providing a clearly visible point of comparison. The study was simulation based, allowing for parameters to be clearly documented and replicated in this work, which takes the same approach. A full specification of the baseline is detailed in section 3.1.

3.3.2 Richness

The richness of the data relates to the breadth of analysis that will be possible. The baseline study captured data covering total boarding time, average individual boarding time and maximum individual boarding time (Van Landeghem & Beuselinck, 2002) and so clearly the data captured by this study must have an equal scope in order to allow appropriate comparisons to be drawn both at the process level and the passenger level. As this study is predicated upon comparison against the baseline, it is appropriate to maintain this level of richness.

3.3.3 Resolution

The resolution of the data must be at an appropriate level to allow the richness to be fully exploited. As the data to be collected covers not only the process as a whole, but aspects of the individual passenger's experience of it (see section 3.3.2 above), then the study must be designed to ensure that data can be collected at the level of that individual passenger. This resolution will also ensure that aggregate measures, such as the mean, can be calculated in order to smooth out the effects of statistical variability.

3.3.4 Process

In common with the baseline study, each simulation scenario will be run five times and the mean of these runs used for all further analysis (Van

Landeghem & Beuselinck, 2002). This approach is also consistent with the later comparative study on which the model of free-for-all passenger behaviour is based (Ferrari & Nagel, 2005). None of the studies discussed in the literature review advocate the use of discrete values in the analysis of simulation results, with the general consensus being that aggregate mean values reduce the inherent variability of data produced by stochastic models.

3.4 Analysis

In the baseline study, process times for the various boarding strategies under consideration are presented as 90% confidence intervals around the mean of five simulation model replications (Van Landeghem & Beuselinck, 2002). Whilst alternative strategies exist in the literature (see section 3.4.1 below) the baseline approach will be followed in this project in order to ensure overall consistency in the comparison of results.

3.4.1 Robustness of Results

Whilst the baseline study and later comparative work base analysis of results on five replications of the boarding model (Van Landeghem & Beuselinck, 2002; Ferrari & Nagel, 2005), a further study in the field which utilised a simulation model to validate earlier analytical findings, produced summary data that was based on 100 iterations of the model, with 95% confidence intervals of less than 60 seconds (van den Briel et al., 2005).

Later work, also based on a hybrid analytical and simulation approach, also implies that a greater number of model iterations were run (Steffen, 2008a). Whilst this approach, in principle, appears to be more statistically robust than that used in the baseline study, the work itself did not address either of the two boarding strategies under consideration in this project and only produced times for the process as a whole, rather than also capturing individual times for each passenger. In this context there is no point of comparison with which to assess the effectiveness of each approach against the other and so, for the purposes of this project, the baseline approach will be considered adequate.

3.4.2 Data for Comparison

In order to fully address the question posed for this research project - *Can multi-agent systems adequately model stochastic uncertainty in a simulation study investigating the effect of pre-assigned seating on free-for-all aircraft boarding?* – a definition of ‘adequate’ must be ascertained.

The context of ‘adequate’ in this case is the existing problem domain. So, stochastic uncertainty will be deemed to have been modelled adequately if the results generated by the simulation model are a reasonable approximation of those generated using the same parameters by the baseline study. A reasonable approximation here means that the results of this work can be shown, at a 90% confidence interval, to be the same as

those generated by the baseline study using a statistical t-test. This test is appropriate as it assesses the true difference between the means of two independent groups (independent means that data collected for each group was not influenced by the data collected for the other group). The 90% confidence interval used for the comparison reflects the degree of uncertainty introduced by the difference in modelling passenger conformance to boarding rules between this study and the baseline (see section 3.1.4) and is also consistent with the confidence intervals used in the baseline work (see section 3.4.1 above).

The mean values generated by the baseline study that will be used in this comparison are included in Table 3, below.

Table 3 Mean Process Times for Random Boarding with Assigned Seats. Adapted from van Landeghem & Beuselinck, 2002.

Random Boarding with Assigned Seats	Total	Average Individual	Max Individual
Time, mins	24.69	1.78	4.35

3.4.3 Additional Comparisons

The comparison described above will be used to address the research question. However, further comparisons will be made between random boarding process times generated through this work and free-for-all boarding, with results qualified by the findings of the initial comparison above. The same statistical methods will be used to analyse the difference between the two boarding strategies and will allow a conclusion to be

reached regarding their relative efficiencies. This approach will also hold true for an analysis of the stochastic variables that feed into the model and their relative effects on the two boarding strategies under consideration.

3.5 Summary

This chapter has discussed the different simulation paradigms available and justified the choice of discrete event simulation as a method with demonstrable success in other domains of study, which clearly supports the use of a multi-agent model implementation.

The fact that DES will focus results on the process level rather than the actions and interactions of individual agents has been discussed and shown to conform to the grain of investigation adopted for the baseline study.

A plan for a Java based implementation has been discussed and shown to fully incorporate the extended conceptual model developed from the baseline study. In support of this, a strategy for producing results has been put forward, that will ensure a direct comparison with the existing baseline data is possible. Opportunities to demonstrate the flexibility of the chosen method, whilst extending the current knowledge base within the literature, have been identified as a series of investigations into stochastic elements that may affect boarding performance.

Chapter 4 Results

Results are presented here around the context of the research question and the tolerances that have been defined in addressing it (see section 3.4.2). As described in section 3.4.3, the simulation environment allows further comparison of boarding strategies to be carried out, along with investigation of the effects of the stochastic variables that feed into the model.

4.1 Preliminary Analysis

Five iterations of the simulation model under the approximate conditions specified in the baseline study – Boeing 757 aircraft, Random boarding with pre-assigned seats, normal luggage load and no probability of groups forming (Van Landeghem & Beuselinck, 2002) – yields the following mean values for the process (Table 4).

Table 4 Mean Simulation Results for Random Boarding with Assigned Seats

Random Boarding with Pre-Assigned Seats	Total	Average Individual	Max Individual
Time, mins	37.29	1.12	2.48

Comparing the 90% confidence intervals of these mean process times with those generated for the same boarding strategy by the baseline study (Figure 11 and Figure 12, below) will allow the research question to be addressed. The exact confidence intervals for the baseline study have been calculated using the given mean (Table 3, above), quoted standard deviation and sample size.

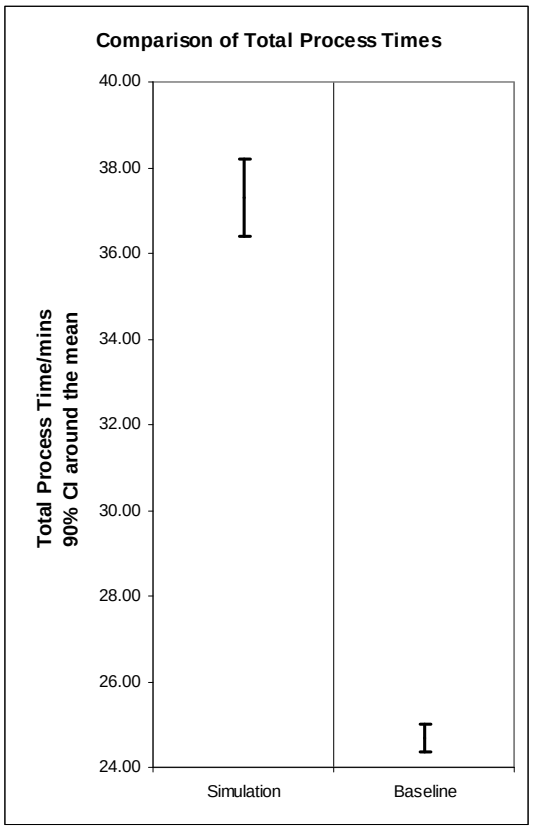

Figure 11 Comparison of Total Boarding Times generated by Simulation Model and Baseline Study (Van Landeghem & Beuselinck, 2002)

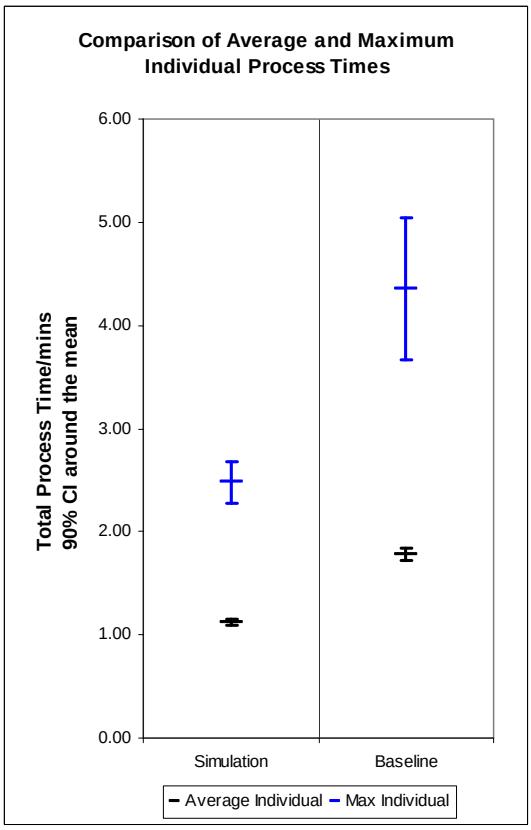

Figure 12 Comparison of Average and Maximum Individual Boarding Times generated by Simulation Model and Baseline Study (Van Landeghem & Beuselinck, 2002)

A brief visual analysis of both Figure 11 and Figure 12 , above, is enough to highlight several interesting points. Firstly, on none of the measures – whether they are of total boarding time (Figure 11) or either of the individual times (Figure 12) – do the 90% confidence intervals of the mean values generated by the simulation model overlap with those of the baseline study. In itself, this is a strong indication that the difference between the two sets of results is not down to chance and indeed this hypothesis is borne out statistically – a simple t-test, carried out at the 90% confidence level on each of the three measures shows that, in all cases, the possibility of the difference observed between the

simulation and baseline results cannot be said to be down to random fluctuations in the data. In fact, the test shows that, for this sample size, we can be over 99% certain for all three measures that the differences observed are real, systematic variations.

The second point of interest appears when comparing the relative positions of the overall process times and the individual passengers times across the simulation and baseline. Figure 11 shows that the total time taken for the boarding process in the simulation model is much higher than that returned in the baseline study – 37.29 minutes compared to 24.69 minutes. With this in mind, it appears to be a logical assumption that the average and maximum individual times presented in Figure 12 will also be considerably higher than those derived in the baseline study. On closer inspection however, the exact opposite of this assumption can be observed – each measure derived from the simulation model is lower than the same measure returned by the baseline study. In the case of the average individual boarding time, the simulation produces a result of 1.12 minutes, whilst the baseline result stands at 1.78, a difference of 0.66 minutes, or almost 40 seconds. A similar relationship is observed in the case of maximum individual boarding time, where the simulation result of 2.48 minutes is considerably lower than the baseline figure of 4.35 minutes.

This counter-intuitive pattern can also be observed in the final point of interest - the relationship between the confidence intervals of the baseline results and

those generated by the simulation model. At the level of total boarding time, the baseline confidence interval is narrower than that returned by the simulation. It might then be reasonable to assume that a similar relationship would be observed at the level of average and maximum individual processing times. However, figure 11 clearly shows that this is not the case – on both individual measures, but particularly maximum boarding time, it can be observed that the confidence intervals generated by the baseline study are wider than those returned in the simulation results. It is important to note that it cannot be said with any degree of certainty that this is a result of the simulation model results being more or less accurate than the baseline. We are interested in whether or not stochastic variation has been modelled accurately in the simulation model when compared against the yardstick of the baseline study. The fact that the simulation results for individual passenger boarding times are less widely spread about their mean than those of the baseline may provide an indication that the simulation has, in fact, not adequately modelled uncertainty. However, this must be set within the context of the overall process time, in which the simulation model results exhibited greater variability than the baseline and so, bearing that in mind, it is difficult to determine any coherent pattern that would allow meaningful commentary to be made regarding the success or otherwise of the strategies used to model stochastic elements in the two models.

So, why do these apparent discrepancies between the simulation and baseline results exist? As far as possible, the simulation environment was designed to

replicate the approach of the baseline study. However, differences between the two models can be observed and it is possible that these can be identified as the cause of the difference in results.

4.1.1 Additional Seats

One example is the aircraft specification on which the models have been built. The baseline study examined a ‘standard’ aircraft with 132 seats divided across 23 rows, with rows 1 and 23 having only three seats each and the remainder of the rows being comprised of six seats (Van Landeghem & Beuselinck, 2002). The commercial aircraft that appears to correspond most directly to this ‘standard’ is the Boeing 757 (SeatGuru). This project simplified the implementation of this standard model by including six seats across all 23 rows of the Boeing 757 scenario. Therefore the simulation model results are based on an aircraft model made up of 138 seats rather than 132. Could the additional six seats account for the differences in overall process time? The baseline study derives a process flow rate (passengers per minute) from the overall boarding time – this is distinct from the flow rate quoted in section 3.1.1, which is the rate at which passengers enter the model. Using this derived measure should reduce the impact of seat numbers on the comparison, by assessing the relative, rather than the absolute speed, of a passenger’s journey through the system. Even by this measure, the results from the simulation (3.7 pax/min) still differ markedly from the baseline (5.3 pax/min) and so it is reasonable to assume

that the additional seats did not play a significant role in the observed differences in performance.

For this to be demonstrated unequivocally, the simulation model would require extension to incorporate different defined aircraft types or a more flexible mechanism allowing the user to specify the exact configuration of the aircraft to be tested. Due to the extensible architecture of the system (see section 3.1.4) this development is entirely feasible and may form the basis of future work (see section 5.2).

4.1.2 Simplified Behaviour

Section 3.1.4 describes the situation implemented in the baseline study whereby an individual passenger may occupy the wrong seat alongside a discussion of why this feature would not be implemented for this project. Allowances in the confidence level of the comparison between the simulation and the baseline were made in order to allow for the effects of excluding this feature but it is possible that the effects are greater than anticipated. Whilst excluding this feature is unlikely to lengthen the overall process time – by its nature it would tend to deliver more interference – it is possible that it would affect individual boarding times. In fact, the relative width of the 90% confidence intervals and the higher standard deviation from the mean observed for the baseline measures of average and individual boarding times compared to the simulation results may in part be

accounted for by this feature skewing aggregated individual times to a greater degree than was anticipated. However, no information is provided in the baseline study on the probability of this situation occurring or its magnitude when it does. Thus, it is difficult to provide any definitive assessment of its effect as things stand and the best that can be done is to note it as a possible factor involved in the relative performances of the two models.

As with the effects of different aircraft configurations noted in section 4.1.1, the behavioural model underpinning the simulation is not fixed and could be altered in later work, which may draw on further empirical studies of passenger behaviour or theoretical models developed for other work. In this case, it would be possible to expand upon the results presented here.

4.1.3 The Black Box Effect

The baseline study describes the strategies used to simulate stochastic variability and provides the data used to characterise the distribution of model inputs (Van Landeghem & Beuselinck, 2002). However, no information is given on the mechanism of implementation other than that the commercially available Arena simulation software was used. Commercial descriptions of Arena mention that facilities exist for defining the distribution of model inputs and building statistical variability into a study, but again little or no information is provided on the exact method that this is achieved by

(Bapat & Sturrock, 2003; Takus & Profozich, 1997). For this reason, it is difficult to ascertain whether the implementations of statistical variability in this simulation study, for example through the Java `math` package and the bespoke `SimKit` component (TeleSim Project) used to return a random value limited by a defined triangular distribution, differ enough from the Arena implementation used in the baseline study to have influenced the discrepancy in results.

4.1.4 Agent Interaction

The simulation model is based upon multiple interacting agents (of which the most active are the passengers within the aircraft model) – the actions of any one of these agents can have an impact on the options available to any other agents who have entered the aircraft scenario after them. For example, a passenger who has halted to load their luggage will temporarily block other passengers from progressing down the aisle. However, no individual agent has any visibility of the internal state of any other agent (for a fuller discussion of why this is the case see Chapter 3). This leads to the possibility that there may be differences between the strategy adopted to determine passenger boarding times in the simulation study and that utilised by the baseline.

Process times in the simulation study are assigned to individual passengers, with only the longest time generated during each simulation step recorded

by the central simulation clock in order to work out the overall process time. This means that, where passengers experience interference due to an obstruction, the waiting times they generate will not match the process times generated by the obstructing agent as they can have no knowledge of the internal variables that store them. It is possible, therefore, that the mean process times generated in the simulation are kept artificially low by this strategy, although it is not possible to say whether this is truly the case without reference to further details of the approach taken in the baseline work.

The development of a more agent-based focus in the simulation model may well allow for a far greater understanding of the internal workings of the model and is discussed further in section 5.2.

4.2 Comparison of Strategies

Moving away from the relative performance of the simulation model against the baseline results, it will be useful to carry out further investigations solely within the simulation model in order to assess whether it is possible to gain any useful insight into its behaviour that may allow us to more accurately address the research question. The first such comparison will be of the random boarding with assigned seats strategy (random) against the free-for-all boarding strategy (FFA), characterised by the set of behavioural rules established in section 3.1.4. In order to focus solely on the effects of the strategy itself, all other model parameters will

remain in the configuration noted above, i.e. Boeing 757 aircraft, normal luggage load and no possibility of groups forming. The results for the random strategy will be as presented in Table 4, above.

Five further iterations of the simulation model, under the stated parameters, for the FFA strategy yields the following mean times, presented against those derived from the random strategy in Table 5, below.

Table 5 Mean Process Times (min) for Random and FFA Boarding

Mean Process Time, mins	Total	Average Individual	Max Individual
Random w/ Assigned Seats	37.29	1.12	2.48
FFA	38.58	1.03	3.31

There is one particularly striking aspect of this comparison – the considerable difference in maximum individual boarding time between the two strategies, against a background of broadly similar overall and average individual process times. This pattern can be observed further in the figures below.

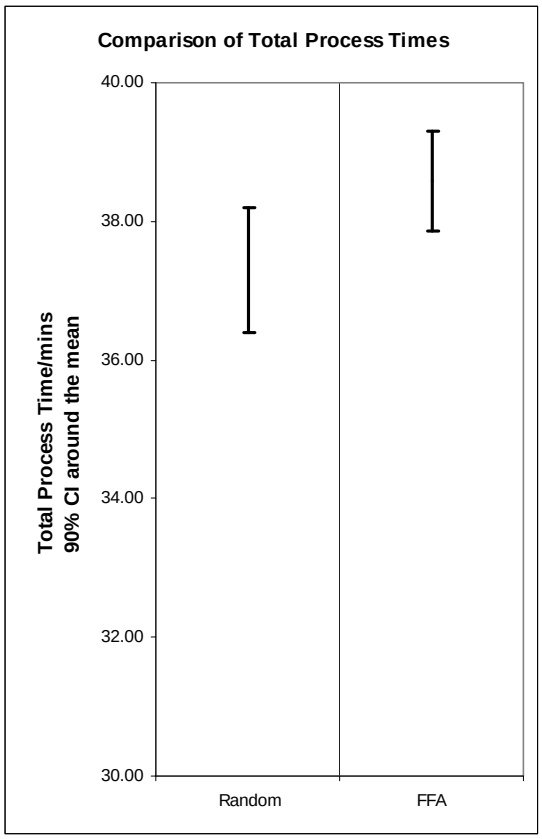

Figure 13 Comparison of Total Boarding Times generated for Random and FFA Boarding

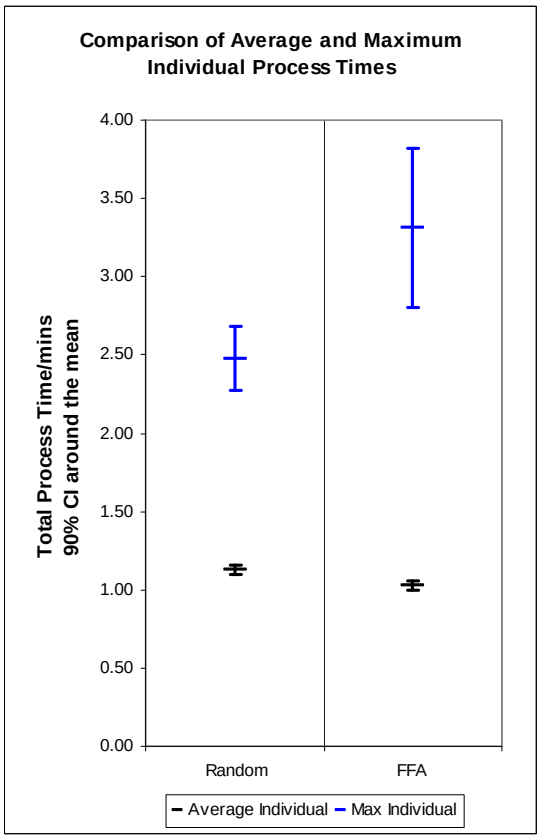

Figure 14 Comparison of Average and Maximum Passenger Boarding Times generated for Random and FFA Boarding

An examination of Figure 13 and Figure 14, above, confirms the initial observations. Whilst the 90% confidence intervals of the total process time for each strategy overlap within the same range and the average individual times appear to be close to one another and are characterised by a similarly narrow 90% confidence interval, the maximum individual time for FFA is clearly distinct from that of the random strategy and is derived from a far wider spread of results. Interestingly, FFA boarding appears to generate slightly longer process times than random boarding and it is reasonable to assume that this may be attributed to the greater incidence of long individual times. A t-test shows that the

differences between each measure for this limited sample size are statistically significant at the 90% confidence level.

Despite the results presented in section 4.1, above, in which it was established that comparisons made between the baseline and simulation models cannot be deemed reliable, it is useful nonetheless to revisit the hypothesis put forward in the baseline study for FFA boarding. This will allow us to determine, at least in relative terms, whether the results presented here follow the general pattern predicted in the baseline study – that FFA boarding is likely to increase overall boarding time when compared to random (Van Landeghem & Beuselinck, 2002) (see section 2.6.6). Clearly, in this scenario and speaking in statistical terms, this has been shown to be the case – a finding which also confirms the work of the later study on which the model of FFA behaviour was based (Ferrari & Nagel, 2005). However, even though the findings are statistically significant, it is worth noting once again that they are based on a very limited sample size. A more robust investigation, based upon a far greater number of model iterations may well return different results as the impact of any significant outliers in the data is likely to be reduced. In fact, even those studies which have found FFA to be close to optimum still state that it is very close to random in terms of overall performance (Steffen, 2008a), which may well justify additional investigation in order to establish full confidence in the results presented here (see section 5.2).

Without further data on passenger interferences – specifically number, duration, type and location of interference – it is difficult to fully understand the differences in performance, particularly with regard to the exact nature of the difference in maximum individual boarding times, as the process times collected give little insight into the mechanics of the strategy in operation. Only by assessing how passengers interact can the efficiency of each strategy in relation to the other be fully explained (see section 5.2).

4.3 Additional Analysis

Further analysis of the factors that may affect overall and individual boarding times will follow. This includes an assessment of the relative efficiency of each strategy depending on the size and complexity of the aircraft model, i.e. the number of rows and aisles. Additional investigations will focus on the effects of increased hand luggage and the impact of passengers travelling together in groups rather than as individuals.

4.3.1 Multi-Aisle Scenarios

It is logical to assume that an increase in the number of aisles (which is generally associated with an increase in the actual number of passengers) would increase overall boarding times. Figure 15 below, shows that this is indeed the case. However, counter-intuitively perhaps, the figure also clearly shows that individual passenger boarding times are reduced as the size of the aircraft increases.

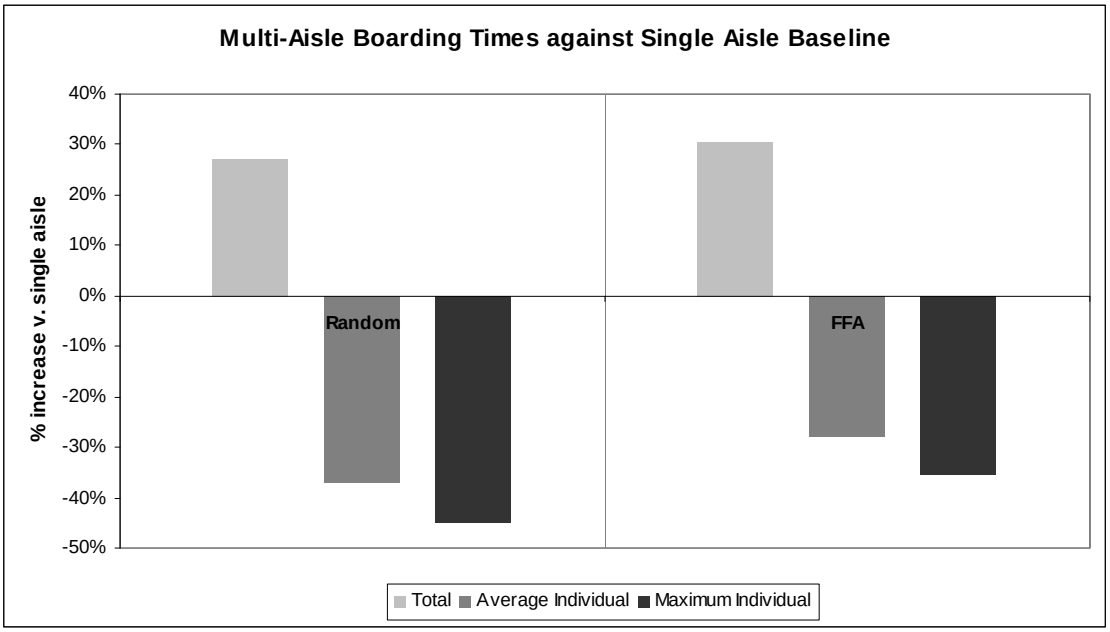

Figure 15 Percentage Increase in Boarding Times for Multi-Aisle Aircraft

Comparing against the baseline of a Boeing 757 aircraft, with 138 passengers and a single aisle, the total boarding time for a Boeing 747 aircraft, with 190 passengers and two aisles is 27% and 30% longer, for random and FFA strategies respectively. Both of these increases are statistically significant at the 90% confidence level.

We can assume that these increased process times are a function of the increased number of passengers passing through the system, rather than any additional complexities generated by the changes to the aircraft geometry. This is reasonable as the individual passenger times are in fact reduced in the multi-aisle scenario, as shown above. However, they continue to follow the pattern observed in the previous comparison of strategies (see section 4.2), with both strategies producing similar average

individual times but random producing noticeably lower maximum individual times than FFA. Each decrease recorded for the individual times is again statistically significant at the 90% confidence level. As mentioned in section 4.2, without further data on the nature of individual passengers' progress through the model, it is difficult to ascertain the exact reasons for the reductions in individual processing time. However, as the reductions appear to be closely linked to the increase in aircraft size, one possible cause is that, as activity becomes distributed across a greater area, the opportunities for interference between individual passengers are reduced and the time taken to reach the target seat is consequently faster.

Previous studies have focused almost exclusively on single aisle short-haul aircraft (see section 2.5). Where larger and more complex aircraft geometry has been discussed, it has been hypothesised that, once passengers have been assigned to an aisle, the boarding process will decompose into a series of single aisle scenarios (Bachmat et al., 2006a; Bazargan, 2007). If this is the case, it is reasonable to assume that boarding times would increase in a broadly linear trajectory as aircraft size (the number of rows) and complexity (the number of aisles) increases. Carrying out a regression analysis of the known data from this study to produce a trendline, including a limited projection of the trend forwards, shows that this is indeed the case (see Figure 16 and Figure 17, below). It should be noted that these results are based on a very limited number of data points and further analysis of

boarding times for different aircraft variants would serve to make them more accurate. The simplicity of this analysis must also be made clear – as aircraft size increases it is also likely that additional features will appear in the aircraft configuration, such as extra entry points and even levels of seating, which may well have effects on real-world boarding that have simply been excluded from this data. Thus, the trends presented in the figures below are useful predominantly as illustrations of the general principle that an increase in aircraft size and complexity will increase overall boarding times. Section 5.2 identifies future work directed at assessing the impact of the kinds of additional factors mentioned above.

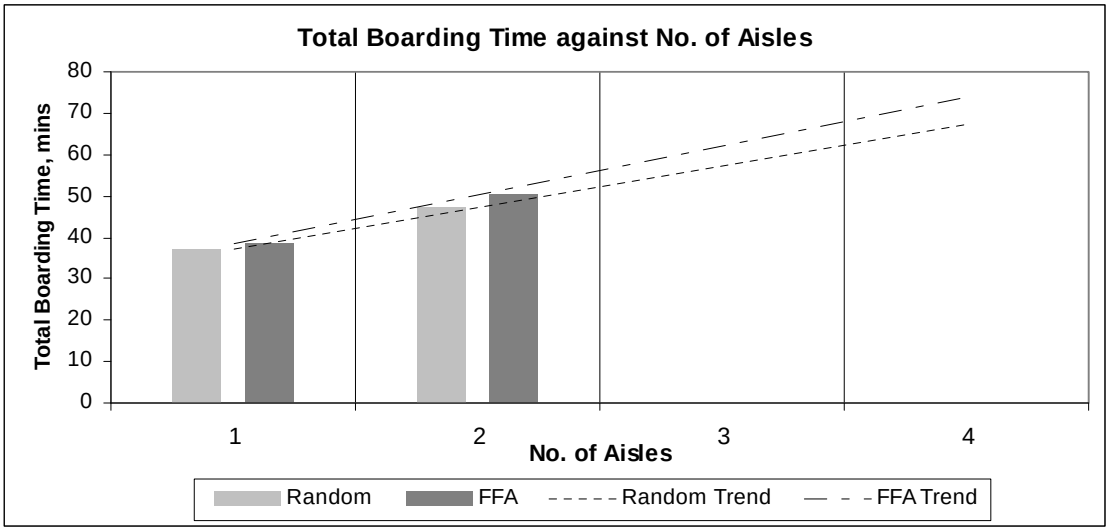

Figure 16 Total Boarding Time against No. of Aisles

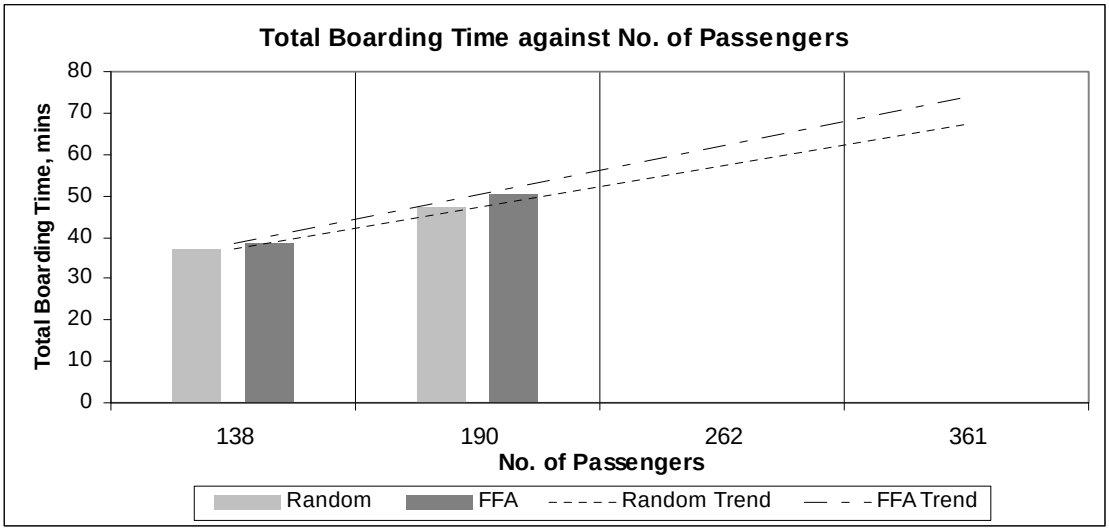

Figure 17 Total Boarding Time against No. of Passengers

4.3.2 Hand Luggage Capacity

Each piece of hand luggage carried onto the aircraft by a passenger is associated with a discrete action to store that piece of luggage in the overhead bin. Therefore, it appears reasonable to assume that any increase in the amount of hand luggage carried onto the aircraft by passengers would result in longer total boarding times and most probably longer individual times as well.

An initial analysis of this situation, presented in Figure 18 below, appears to corroborate this hypothesis, with the effects of an increased luggage load on boarding times being more marked for random boarding than for FFA. However, further analysis of these results shows that the increase in overall boarding times for both the random strategy and the FFA strategy is not statistically significant at the 90% confidence level. The same is true of the

maximum individual boarding times for each strategy, with neither returning a statistically significant difference. In terms of the average individual boarding time, however, both strategies exhibit increased times under high luggage loading which are statistically significant. Overall, then, it would appear that increased hand luggage is likely to degrade the boarding experience to some extent for all passengers, but not at a level that will bring significant additional inefficiency to the process as a whole.

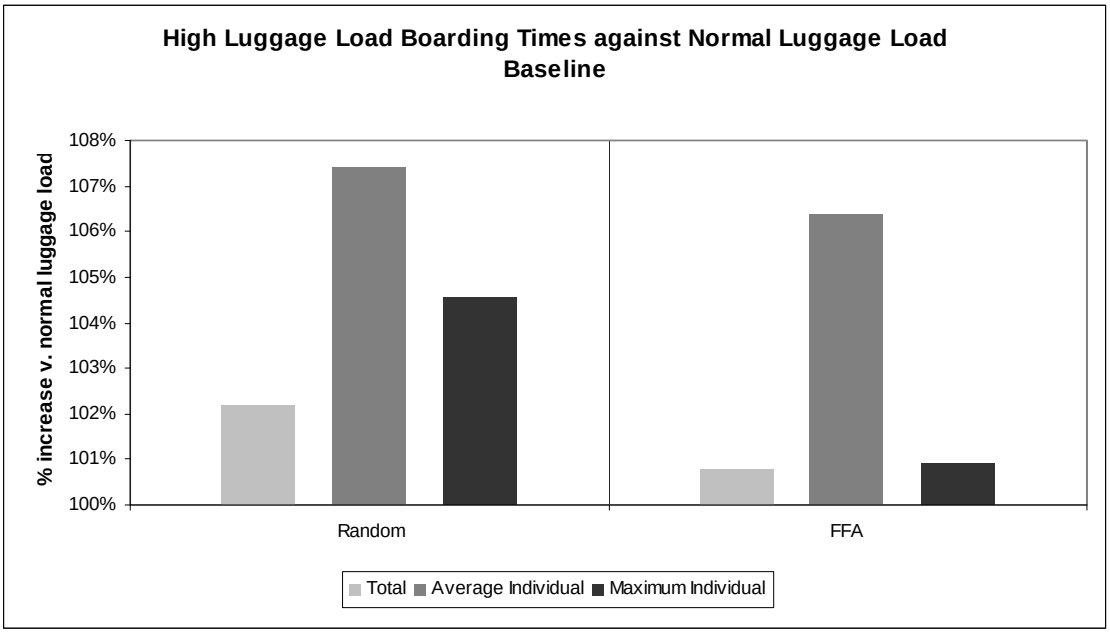

Figure 18 Increased Boarding Times under High Luggage Load

In broad terms, these findings mirror those of the baseline, which found that increased luggage loads under the random strategy have a more marked effect on individual boarding times than on the process overall (Van Landeghem & Beuselinck, 2002). It is also worth noting that, although all process times for both strategies have increased under high luggage

loading, the relative performance of the two strategies remains very similar to the patterns described in the initial comparison of strategies (see section 4.2). This may indicate that the internal performance and stochastic mechanisms of each strategy remains relatively robust under this kind of disturbance.

4.3.3 Passenger Group Formation

The probability that passengers may wish to travel as part of a group is a key distinguishing feature of different types of flight – those to holiday destinations and those on common business routes representing the opposing ends of the spectrum.

Running the simulation model under the constant scenario of a Boeing 757 single aisle aircraft with normal luggage loading and then increasing the probability of passenger groups forming along a scale ranging from 0.0 to 0.9, highlights several interesting findings (see Figure 19, Figure 20 and Figure 21, below). Moving averages, based upon two periods of the data, have been used to produce smoothed trendlines illustrating the overall pattern of performance at both process and passenger levels.

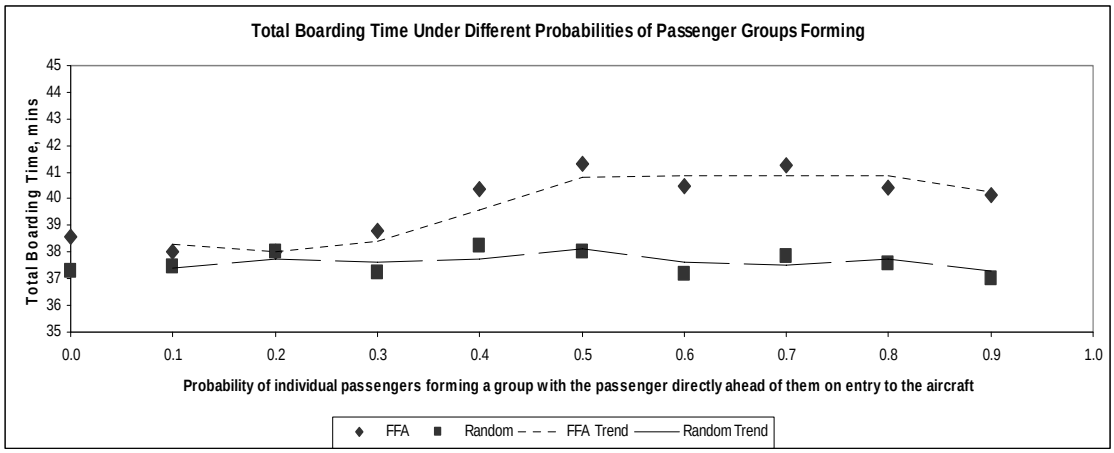

Figure 19 Total Boarding Time against Group Probability

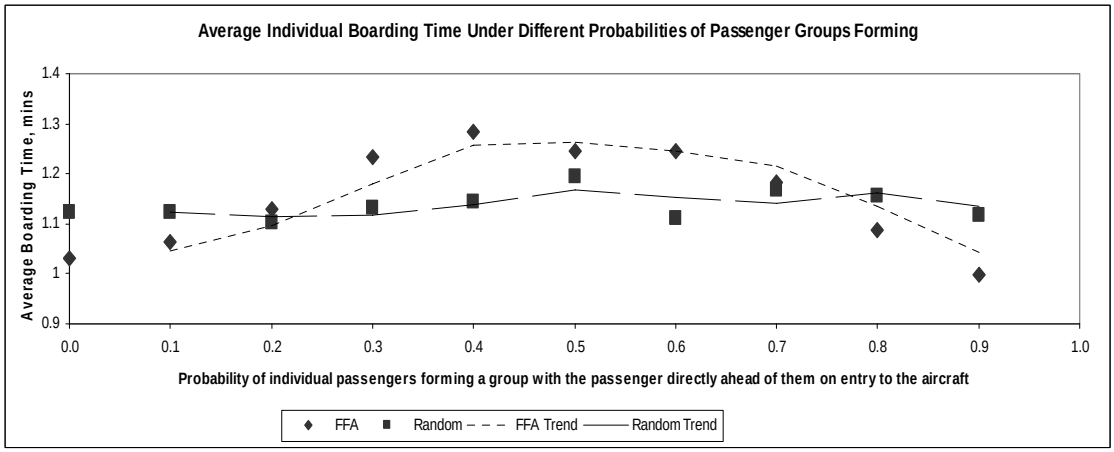

Figure 20 Average Individual Time against Group Probability

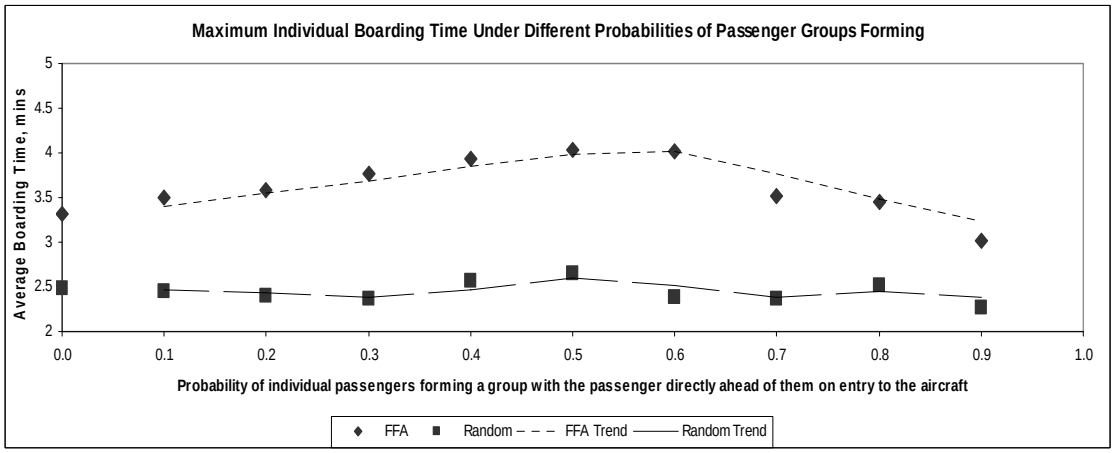

Figure 21 Maximum Individual Time against Group Probability

Generally speaking, there is significant differential between the manner in which random and FFA strategies perform under increased probability of passenger groups forming. Both strategies exhibit increases in the average individual boarding time which peak at approximately the mid-point of the x-axis (i.e. 0.5 probability of groups forming) and decrease thereafter, although this effect is far more marked in FFA boarding than in random. It is likely that this effect is due to the ratio of grouped passengers to individuals, with the effects being far greater in FFA boarding as individual passengers attempting to move forwards to occupy the best seats are blocked by grouped passengers congregating in sections close to one another. The effect is reduced in random boarding as seat choice is predetermined, with a greater chance that boarding will be spread across a larger proportion of the available aircraft space as even grouped passengers make for their pre-assigned areas. A further point of interest is the decrease in average individual time for FFA boarding as the probability of groups forming exceeds 0.5, with the performance at P=0.9 exceeding that at P=0.0. It seems logical that this is a function of the behavioural rules on which FFA boarding is based – a passenger who is a part of a group will place proximity to the rest of their group above all other considerations when choosing a seat and can thus be expected to occupy seats that they would have passed by had they been boarding as an individual. As group size increases, therefore, the number of passengers sitting down at an earlier

stage in the process will also increase and boarding time will subsequently reduce.

A similar pattern can be observed with regard to maximum individual boarding times. The effects of increased probability of passenger groups forming are again more marked in FFA boarding than in random. At the overall process level random boarding performs extremely robustly, with a difference of only 1.2 minutes between the fastest boarding time of 37.01 minutes (at P = 0.9) and the slowest time of 38.25 minutes (at P = 0.4). The impact on FFA boarding is far greater, with a difference of 3.29 minutes separating the fastest time of 38.00 minutes (at P = 0.1) from the slowest (41.30 at P = 0.5). Although FFA boarding does begin to exhibit some signs of an improvement in performance for higher values of P, it is clear that performance overall deteriorates as the preponderance of groups increases.

For this reason, random boarding may be more suited to flights where passenger groups may reasonably be expected, e.g. those serving popular holiday destinations, than FFA. The process appears to be robust in the presence of groups and thus performance should be relatively predictable. Passengers are unlikely to have to endure extremely long boarding times in the group-heavy scenarios and their expectations of boarding based on previous experiences in other scenarios are more likely to be met.

4.4 Validation

The current work has been able to address the original research question - *Can multi-agent systems adequately model stochastic uncertainty in a simulation study investigating the effect of pre-assigned seating on free-for-all aircraft boarding?* In short, the study has found that, within the defined parameters and scope, multi-agent systems cannot be shown to adequately model uncertainty in this context. This will be discussed further in the conclusions section which follows.

However, in itself this conclusion is not particularly interesting as it does not address the issue of *why* multi-agent systems have proved unsuccessful in the current work. To do that, more detailed data below the level of a complete boarding time is required – for instance, data on passenger interferences, such as the number, type and duration may well have allowed key differences in the process to be uncovered when making comparisons between the baseline model and the current work. This is not necessarily due to any limitations in the research question but is more to do with the specified richness of the data that has been collected (see section 3.3.2). If this specification were to be reconfigured to include a greater breadth of data then further and more meaningful analysis may have been possible. However, in the context of the current work, it is important to note that even if the richness of data captured had been extended, the conclusions that would potentially have been reached would

have still remained unclear in essence, as the same richness of data was not available from the baseline results.

Although we can see that the research question was properly formed and has thus been properly addressed there are improvements in *the way* in which it has been addressed that might be made. One clear example is the level of analysis that supports the conclusion that has been reached. Although defined, well-known and robust statistical techniques have been used to support the analysis, these have been based on very small sample sizes, generally five iterations of the simulation model. Whilst this approach was a necessity of making appropriate comparisons between the baseline results and the current work, other studies in the field have based their results on a far greater number of iterations, approximately 100 in one case (van den Briel et al., 2005). The study in question is the only one in the domain that has seen a real-world implementation of the recommendations made and it is important to note that, if findings from this kind of study are to be relevant to the area they seek to address, the conclusions must be based on an extremely sound base of evidence which, in the case of a simulation study, will often mean that far more exhaustive testing must be carried out (see section 5.2).

Chapter 5 Conclusions

This dissertation began by describing the problem domain – passenger aircraft boarding – and identifying areas where knowledge could be usefully extended in a manner that was relevant to both the airline and computer science research communities. The area of Free for All boarding was identified as an appropriate area for research by several existing studies (Van Landeghem & Beuselinck, 2002; Ferrari & Nagel, 2005; Steffen, 2008b; Bachmat et al., 2006b). Setting this within the wider context of computer science research required extension of the simulation model paradigm used by previous studies (Van Landeghem & Beuselinck, 2002; Ferrari & Nagel, 2005), which is based on proprietary software and specialised languages such as SIMUL8, to encompass a more general platform that would allow investigation of the suitability of a simulation project based on a multi-agent model for investigations within a domain which is characterised by several random variables.

This led to the establishment of the research question: *Can multi-agent systems adequately model stochastic uncertainty in a simulation study investigating the effect of pre-assigned seating on free-for-all aircraft boarding?* The investigations that were undertaken to address this question were hierarchical in nature – an initial comparison against an established baseline (Van Landeghem & Beuselinck, 2002) in order to establish the validity of further investigations and address the fundamental essence of the research question, followed by a series of domain specific analyses that may be of interest to commercial airlines in

assessing the effects of various stochastic factors upon defined boarding strategies. It follows, logically, that should the initial investigation fail to establish the suitability of multi-agent systems for the purposes of stochastic simulation, the conclusions that could be drawn from the later investigations would become limited in the scope of their applicability.

In fact, the initial investigation did find that, at the 90% confidence level, there was no similarity between the results returned from this project and those generated by the baseline study. Several differences between this project and the baseline approach were discussed and the possibility that they may have influenced the results explored. The most likely explanation appears to be a difference in the level of communication between agents in the simulation and those in the baseline study that could well have lead to inconsistency in the application of the mechanism generating passenger process times. It is also possible that slight modifications to the behavioural model used in the simulation, namely that passengers never occupy an incorrect seat, may account for differences in individual times. However, it was established that without additional data, covering a greater breadth of passenger activity, no firm conclusions could be reached to explain the difference in results (see section 5.2, below).

The answer to the research question then, as it stands, must be that: *No, multi-agent systems cannot adequately model stochastic uncertainty in the specified context*.

Despite these results, the domain specific investigations were still carried out in order to assess two factors of interest: i. Whether the general pattern of results fitted that of the baseline and other hypotheses put forward, in order to assess whether the results generated by the initial investigation may simply have been a problem of implementation and ii. Whether further results may give some insight into the model's behaviour that may allow inferences to be made back to the initial investigation.

A comparison was made between the random boarding strategy used in the initial investigation and the Free for All (FFA) strategy previously identified as an area of interest. This work found FFA boarding to be slower than random at the overall process level, although with slightly faster average individual passenger boarding times. The key factor in FFA's performance relative to random boarding was found to be its long maximum individual passenger boarding times. This suggests that, should FFA boarding be adopted as a real-world strategy, a proportion of passengers may experience some degradation in satisfaction as they take longer to reach their seat. It was found that these results verify the hypotheses put forward in earlier work (Van Landeghem & Beuselinck, 2002; Ferrari & Nagel, 2005) and could well be in line with the limited experimental data available from previous analytical work, which found that the major differences between random and FFA boarding were almost entirely down to the final unseated passengers searching the aircraft for the remaining free seats (Steffen,

2008b). Whilst this may be of interest and does appear to fit in logically with the observed increase in maximum individual times for FFA boarding, the results must be tempered by the initial investigation, which casts doubt on the validity of this project's simulation strategy.

In order to extend the study's applicability to a wider commercial context, the effect of larger aircraft variants on boarding times was also tested. The results continue the relative trend in performance already observed between random and FFA boarding and produce a linear trend that appears to confirm the assertions made in previous studies that multi-aisle boarding would decompose into a series of single-aisle scenarios as passengers entered the model (Bachmat et al., 2006a; Bazargan, 2007). However, these results were presented with the caveat that the analysis was based upon a very limited sample size and took no account of other features within the aircraft environment that may affect passenger progress through the system, e.g. additional levels or entrances.

The issue of increased carry-on hand luggage was investigated, with the results conforming to the general trend established by the baseline study: that an increase in the luggage load has a more marked affect on individual passenger boarding times than on the process as a whole. In this case, random showed a more marked effect than FFA. However, in absolute terms the maximum individual times for FFA remain significantly higher than those for random and this further reiterates the general conclusion that FFA may cause a certain level

of passenger dissatisfaction, despite its not dissimilar performance at the process level.

Finally, the appropriateness of each strategy was measured for different flight profiles, defined in this case by the probability of passengers travelling in groups of two or more. This investigation was intended to make the results of the project of more interest to not only those airlines operating a variety of aircraft types but also those operating in different sectors of the market, e.g. business v. leisure. In this context, the presence of groups was found to affect both strategies to quite different orders of magnitude, with random boarding far more robust under the presence of passenger groups than FFA. FFA performance was found to be at its worst at the midpoint of the group probability scale, with performance then improving as the proportion of groups increased. It was noted that the ratio of groups to individuals was crucial to this performance, with random boarding performing better as predetermined seating served to spread groups of passengers across wider sections of the aircraft and free up more space for individuals to utilise. With this in mind, random boarding once again emerges as the strategy most likely to deliver customer satisfaction due to the robust, relatively predictable performance under different probabilities of passenger groups forming. With far fewer perturbations in boarding time than FFA, customer expectations, based on their prior experience of boarding under different scenarios are far more likely to be met.

In summary, whilst these investigations do appear to follow the broad trend observed and hypothesised in previous work and put forward as a secondary hypotheses in section 2.8, at least in the areas of the number of aisles, hand luggage and passenger groups, it is difficult to make concrete recommendations for the commercial world due to the findings of the initial investigation. Until they are fully understood (see section 5.2), any further results must be treated with some caution.

5.1 Project Review

The objectives of the project were defined (see section 1.2.3) as i. to carry out work that would be of use to commercial airlines in informing decision making on boarding strategies, ii. to extend the knowledge currently available to researchers in this specific field by addressing the current hypotheses put forward for the efficacy of FFA boarding and iii. to produce results that would be of interest to researchers within the field of computer science who are interested in the application of simulation theory to a multi-agent environment.

In addressing these objectives, the project has been only partially successful. It has been established that, whilst results have been produced that show a general adherence to predicted and previously investigated trends, these data cannot be said to be fully reliable and robust as they have been derived from a model that has not been shown to be appropriate for the domain. Therefore, the results are likely to be of little use in the commercial sector, unless on an

anecdotal level. On reflection, this objective may well have been overly ambitious in scope given the level of detail of the data defined for collection. It is also reasonable to assume that commercial sources would wish to replicate the data produced and ensure accuracy through calibration with real-world systems.

In terms of extending the knowledge available to researchers in the field, the study has again been only partially successful. In part this is due to the results of the initial analysis. Had the project results matched those of the baseline, it would have been possible to produce much more definitive conclusions using the follow-on analyses of random variables that could have usefully extended the work carried out by the baseline study and validated or disproved further analytical work carried out on the FFA strategy. In this case, the objective was not too ambitious in scope but was unfortunately limited by the earlier findings. It is important to note that the scope of this objective was also limited to a great degree by the scope of the baseline study and any valid comparisons made with this work would, by necessity, have had to remain within those bounds.

Finally, the work will be of interest to researchers in the field of computer science as the study has thrown up interesting follow-on questions as to exactly why the multi-agent system failed in this situation. Future investigations might focus on the interaction between bespoke and reusable components and how these interacting agents perform against specialised simulation software. In this

respect, the objective has been met and this study paves the way for future research which may be usefully shaped by the lessons learned here.

5.2 Future Research

The incomplete nature of the findings derived from this study leaves plenty of scope for future work that may usefully build on those findings and extend our understanding of the effects that have been observed here.

During the discussion of results, it was noted that extensions to the simulation model would allow more flexibility in the investigations carried out. In particular, the inclusion of a flexible mechanism to replace the current hard-coded aircraft types and allow the user to specify the geometry of the aircraft space, especially the number of aisles and seating configuration, would ensure that the model was future-proofed for developing aircraft types and usage scenarios. This work would be a useful demonstration of the flexibility of multi-agent systems and prove their worth as an easily extensible platform within the simulation context.

A further, interesting, area for future development relates to the simulation approach. It was decided during the project planning phase that an agent-based approach to simulation was not suited to this work (see chapter 3), as it may distract focus from the process itself. However, following analysis of the results it was noted that a greater focus on agent communication may be desirable, particularly to ensure parity of process times generated by individuals through

greater visibility of the internal state of agents within the model. The main focus of this work would be to generate more robust results, rather than to switch attention away from the process itself.

To continue the theme of increased reliability of results, further work may usefully attempt to replicate the data produced here but across a far greater number of model iterations, in common with the approach taken by other studies within the domain. The work would be useful in assessing the validity of the original results through reduction of the impact of any unusually fast or slow runs. This would be useful as it would allow an investigator to be confident that the results truly reflected the underlying distributions of the stochastic model elements.

To complete the study it will be necessary to define additional data that could be collected via the simulation model and that would be useful in terms of more fully understanding how individual passengers move through the system. Although any additional data that is collected will have no point of comparison in the baseline results, it would nonetheless be the most logical method by which the detailed workings of the model might be examined and conclusions drawn about different behaviours under different parameters. The most obvious example relates to passenger interferences, which have been studied in detail in other work (van den Briel et al., 2005) and would be extremely useful in demonstrating how different strategies and other variables influence passengers to interact with

one another. Other data may be identified by surveying the literature from other fields such as the study of crowd movement and interaction.

Further work may also concentrate on further calibrating the distribution of the stochastic variables through empirical observation of aircraft boarding under differing conditions. An extension of this work would investigate and refine the behavioural model used in this project through incorporation of more sophisticated parameters based on studies in fields such as behavioural psychology and space design. This would be an extremely useful addition to this work and would make results more relevant to a wider audience of researchers.

References

Alamdari, F. & Fagan, S. (2005) 'Impact of the adherence to the original low-cost model on the profitability of low-cost airlines', *Transport Reviews,* vol. 25, no. 3, pp. 377-392.

Bachmat, E., Berend, D., Sapir, L. & Skiena, S. (2007) 'Optimal boarding policies for thin passengers', *Advances in Applied Probability,* vol. 39, pp. 1098-1114.

Bachmat, E., Berend, D., Sapir, L., Skiena, S. & Stolyarov, N. (2006a) 'Analysis of aeroplane boarding via spacetime geometry and random matrix theory', *Journal of Physics A-Mathematical and General,* vol. 39, no. 29, pp. L453-L459.

Bachmat, E., Berend, D., Sapir, L., Skiena, S. & Stolyarov, N. (2006b) 'Analysis of airplane boarding times', *Working Paper*.

Bachmat, E. & Elkin, M. (2006) 'Bounds on the performance of back-to-front airplane boarding policies', *Working Paper*.

Baines, T., Mason, S., Siebers, P.O. & Ladbrook, J. (2004) 'Humans: the missing link in manufacturing simulation?', *Simulation Modelling Practice and Theory,* vol. 12, no. 7-8, pp. 515-526.

Banks, J. (2000) 'Introduction to simulation', *Proceedings of the 2000 Winter Simulation Conference,* vol. 1 and 2, pp. 9-16.

Bapat, V. & Sturrock, D.T. (2003) 'The Arena product family: Enterprise modeling solutions', *Proceedings of the 2003 Winter Simulation Conference, Vols 1 and 2,* , pp. 210-217.

Bazargan, M. (2007) 'A linear programming approach for aircraft boarding strategy', *European Journal of Operational Research,* vol. 183, no. 1, pp. 394-411.

Bear, D. & Sostek, A. (2006) 'There's more than one way to fill a plane', *Pittsburgh Post-Gazette*, June 18.

Bigelow, B. (2006) 'No 'cattle car'? Southwest plans S.D. boarding tests', *The San Diego Union Tribune*, June 21.

Carson, J.S. (2005) 'Introduction to modeling and simulation', *Proceedings of the 2005 Winter Simulation Conference,* vol. 1-4, pp. 16-23.

Chick, S.E. (2006) 'Bayesian ideas and discrete event simulation: Why, what and how', *Proceedings of the 2006 Winter Simulation Conference,* vol. 1-5, pp. 96-106.

Desouza, K.C., Awazu, Y. & Tiwana, A. (2006) 'Four dynamics for bringing use back into software reuse', *Communications of the ACM,* vol. 49, no. 1, pp. 96-100.

Dijkstra, J., Jessurun, J. & Timmermans, H. 2002, ''A multi-agent cellular automata model of pedestrian movement'', *Pedestrian and Evacuation Dynamics; International Conference on Pedestrian and Evacuation Dynamics*, eds. M. Schreckenberg & S.D. Sharma, pp. 173.

Elliot, C. (2005) 'How to end airplane boarding bottlenecks', *The New York Times*, October 18.

Ferrari, P. (2005) 'Improving passenger boarding in airplanes using computer simulations', *International Airport Review*.

Ferrari, P. & Nagel, K. (2005) 'Robustness of efficient passenger boarding strategies for airplanes', *Airports, Airspace, and Passenger Management,* , no. 1915, pp. 44-54.

Finney, P. (2006) 'Loading an airliner is rocket science', *The New York Times*, November 14.

Funk, M. (2003) 'The visualization of the quantification of the commodification of air travel Or: why flying makes you feel like a rat in a lab cage', *Popular Science*, November.

Gianni, D. (2008) 'Bringing Discrete Event Simulation Concepts into Multi-Agent Systems', *Tenth International Conference on Computer Modeling and Simulation,* pp. 186 - 191.

Howrey, E.P. & Klein, L.R. (1971) 'Dynamic Properties of Nonlinear Econometric Models', *Econometrica,* vol. 39, no. 4, pp. 229-&.

Ignall, E.J., Kolesar, P. & Walker, W.E. (1978) 'Using Simulation to Develop and Validate Analytic Models - some Case Studies', *Operations research,* vol. 26, no. 2, pp. 237-253.

Kirchner, A., Klupfel, H., Nishinari, K., Schadschneider, A. & Schreckenberg, M. (2003) 'Simulation of competitive egress behavior: Comparison with aircraft evacuation data', *Physica A-Statistical Mechanics and its Applications,* vol. 324, no. 3-4, pp. 689-697.

Logan, B. & Theodoropoulos, G. (2001) 'The Distributed Simulation of Multiagent Systems', *Proceedings of the IEEE,* vol. 89, no. 2, pp. 174 - 185.

Lopez-Cainzos, S. 2006, *Discrete Event Simulation Model for Passenger Boarding*, MSc edn, Cranfield University.

Luke, S., Cioffi-Revilla, C., Panait, L. & Sullivan, K. (2004) 'MASON: A New Multi-Agent Simulation Toolkit', *Proceedings of the 2004 SwarmFest Workshop*.

Macal, C.M. & North, M.J. (2005) 'Tutorial on agent-based modeling and simulation', *Proceedings of the 2005 Winter Simulation Conference,* vol. 1-4, pp. 2-15.

Marelli, S., Mattocks, G. & Merry, R. (1998) 'The Role of Computer Simulation in Reducing Airplane Turn Time', *Aero Magazine*.

Maria, A. (1997) 'Introduction to modeling and simulation', *Proceedings of the 1997 Winter Simulation Conference,* vol. 1, pp. 7-13.

Millward, D. & Highfield, R. (2005) 'Travellers' scrum fills plane quickly', *Daily Telegraph*, December 22.

Moulin, B., Chaker, W., Perron, J., Pelletier, P., Hogan, J. & Gbei, E. 2003, "MAGS project: Multi-agent GeoSimulation and crowd simulation", *Spatial Information Theory, Proceedings - Foundations of Geographic Information Science; LECTURE NOTES IN COMPUTER SCIENCE; Conference on Spatial Information Theory (COSIT)*, eds. W. Kuhn, M. Worboys & S. Timpf, pp. 151.

Nance, R.E. & Sargent, R.G. (2002) 'Perspectives on the evolution of simulation', *Operations research,* vol. 50, no. 1, pp. 161-172.

Parker, D.C., Manson, S.M., Janssen, M.A., Hoffmann, M.J. & Deadman, P. (2003) 'Multi-agent systems for the simulation of land-use and land-cover change: A review', *Annals of the Association of American Geographers,* vol. 93, no. 2, pp. 314-337.

Parunak, H.V. (1997) '"Go to the ant": Engineering principles from natural multi-agent systems', *Annals of Operations Research,* vol. 75, pp. 69-101.

Reed, D. & Yu, R. (2006) 'Northwest tries first-in-line boarding', *USA TODAY*, June 20.

Robinson, S. (2005) 'Discrete-event simulation: from the pioneers to the present, what next?', *Journal of the Operational Research Society,* vol. 56, no. 6, pp. 619-629.

Schriber, T.J. & Brunner, D.T. (2006) 'Inside discrete-event simulation software: How it works and why it matters', *Proceedings of the 2006 Winter Simulation Conference,* vol. 1-5, pp. 118-128.

SeatGuru *, Airline Seating Charts, Best Airplane Seats* [Homepage of tripadvisor], [Online]. Available: http://www.seatguru.com [2008, 08/10]

Shanthikumar, J.G. & Sargent, R.G. (1983) 'A Unifying View of Hybrid Simulation Analytic Models and Modeling', *Operations research,* vol. 31, no. 6, pp. 1030-1052.

Steffen, J. (2008a) 'Optimal boarding method for airline passengers', *Submitted*.

Steffen, J. (2008b) 'A statistical mechanics model for free-for-all airplane passenger boarding', *Submitted*.

Stoller, G. (2006) 'Getting fliers on jets faster', *USA TODAY*, June 27.

Takus, D.A. & Profozich, D.M. (1997) 'Arena (R) software tutorial', *Proceedings of the 1997 Winter Simulation Conference,* , pp. 541-544.

TeleSim Project *, TeleSim - University of Calgary* [Homepage of University of Calgary], [Online]. Available: http://warp.cpsc.ucalgary.ca/ [2008, 08/10]

van den Briel, M.H.L., Villalobos, J.R., Hogg, G.L., Lindemann, T. & Mule, A.V. (2005) 'America West Airlines develops efficient boarding strategies', *Interfaces,* vol. 35, no. 3, pp. 191-201.

Van Landeghem, H. & Beuselinck, A. (2002) 'Reducing passenger boarding time in airplanes: A simulation based approach', *European Journal of Operational Research,* vol. 142, no. 2, pp. 294-308.

Vincent, R., Horling, B., Wagner, T. & Lesser, V. (1998) 'Survivability Simulator for Multi-Agent Adaptive Coordination', *Proceedings of the International Conference on Web-Based Modeling and Simulation*.

Weisstein, E.W. *, Triangular Distribution* [Homepage of MathWorld - A Wolfram Web Resource], [Online]. Available: http://mathworld.wolfram.com/TriangularDistribution.html [2008, 08/08] .

Yu, R. (2006) 'Airlines change how they herd us aboard', *USA TODAY*, January 9.

Zamiska, N. (2005) 'Plane Geometry: Scientists Help Speed Boarding of Aircraft', *The Wall Street Journal*, November 2.

Index

Glossary of Terms

Agent-Based Modelling:	A modelling paradigm that places agent-level interaction before holistic consideration of an overarching process.
Aircraft Turnaround:	The sequence of activities that must be carried out once an aircraft lands in order to ready it for take-off.
Analytical Studies:	An investigative method using algorithmic models to uncover near-optimal strategies.
Back-To-Front Boarding:	Traditional boarding strategy where passengers are assigned a group dependent on seat number and then boarded in order from the back of the aircraft working forwards.
Cellular Automata:	An array of cells, commonly used to represent space, where the state of each cell is determined by the state of those cells surrounding it.

Discrete Event Simulation:	A simulation paradigm where a real-world sequence of activity is represented by discrete, chronological time-steps.
Free-for-All Boarding:	A boarding strategy where passengers are not assigned seats and enter the aircraft in a random order.
Interference:	Any obstruction caused by one boarding passenger to another.
Multi-Agent System:	A system composed of multiple independent, interacting agents.
Random Boarding:	A boarding strategy where passengers are assigned seats prior to boarding but enter the aircraft in an order that is not correlated to their seat number.
Reverse Pyramid Boarding:	A refined version of the Window-Middle-Aisle boarding strategy, where passengers enter in staggered groups.

Seat Group Boarding:	A refined version of the Window-Middle-Aisle boarding strategy, where passengers enter in smaller groups.
Strict Ordering:	A boarding strategy where passengers are called to enter the aircraft individually, according to their seat number.
Triangular Distribution:	A continuous probability distribution characterised by an upper and lower limit, along with a mode.
Window-Middle-Aisle Boarding:	A boarding strategy where passengers enter the aircraft in groups determined by the proximity of their seat to the window position.

APPENDIX A: Results

This appendix contains the unanalysed results returned for each of the investigations discussed in Chapter 4.

Table A1 Results (mins) of Initial Investigation using Random Boarding for Comparison with Baseline

Run	Total	Average Individual	Maximum Individual
1	37.8299	1.1789	2.7170
2	37.0478	1.0908	2.5077
3	38.6286	1.1255	2.3397
4	36.2635	1.1334	2.6184
5	36.6745	1.0942	2.2100
Mean	37.2889	1.1246	2.4786
Parameters: Boeing 757, Assigned, Normal Luggage, No Groups			

Table A2 Results (mins) of Investigation of FFA Boarding for Comparison with Random

FFA Runs	Total	Average Individual	Maximum Individual
1	39.6870	1.0555	4.0416
2	37.6397	1.0155	3.5052
3	38.8414	1.0696	3.4398
4	38.5234	0.9810	2.7969
5	38.2090	1.0302	2.7593
Mean	38.5801	1.0304	3.3086
Parameters: Boeing 757, FFA, Normal Luggage, No Groups			

Table A3 Results (mins) of Investigation of Effect of No. of Aisles on FFA Boarding

FFA Runs	Total	Average Individual	Maximum Individual
1	51.4516	0.7453	2.3254
2	50.8565	0.7420	2.1575
3	49.6897	0.7293	1.7187
4	49.4104	0.7356	2.2156
5	50.1436	0.7543	2.2627
Mean	50.3104	0.7413	2.1360
Parameters: Boeing 747, FFA, Normal Luggage, No Groups			

Table A4 Results (mins) of Investigation of Effect of No. of Aisles on Random Boarding

Random	Total	Average Individual	Maximum Individual
1	47.3349	0.7092	1.4160
2	48.2950	0.7118	1.4407
3	46.1299	0.7080	1.3943
4	47.5048	0.7055	1.2855
5	47.4574	0.7121	1.2827
Mean	47.3444	0.70932	1.36384
Parameters: Boeing 747, Random, Normal Luggage, No Groups			

Table A5 Results (mins) of Investigation of Effect of High Luggage Load on FFA Boarding

FFA Runs	Total	Average Individual	Maximum Individual
1	39.403	1.0576	3.0771
2	38.667	1.0945	3.5877
3	39.5323	1.1517	3.5651
4	38.6142	1.0619	3.0879
5	38.2327	1.1158	3.3724
Mean	38.8898	1.0963	3.3380
Parameters: Boeing 757, FFA, High Luggage, No Groups			

Table A6 Results (mins) of Investigation of Effect of High Luggage Load on Random Boarding

Random	Total	Average Individual	Maximum Individual
1	39.9301	1.2038	2.6328
2	38.1314	1.1863	2.4368
3	38.4328	1.2189	2.6521
4	36.8091	1.2528	2.9281
5	37.2265	1.1780	2.3066
Mean	38.1060	1.2080	2.5913
Parameters: Boeing 757, Random, High Luggage, No Groups			

Table A7 Results (mins) of Investigation of Effect of Increasing P-Value of Group Formation on FFA Boarding

FFA Runs/P Value	0.0			0.1			0.2			0.3			0.4			0.5			0.6			0.7			0.8			0.9		
	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind
1	39.6870	1.0556	4.0416	39.6412	1.0370	3.4950	38.1345	1.1535	3.3907	37.9712	1.1931	4.0679	37.1103	1.2789	3.9008	40.9182	1.2493	3.9618	39.3539	1.3077	4.2550	40.2881	1.1866	3.4139	40.9874	1.0936	4.4471	43.4126	0.9838	2.9685
2	37.6397	1.0155	3.5052	38.5198	1.0504	3.7848	39.1263	1.0992	3.1082	39.2076	1.2286	3.4988	40.5893	1.2171	3.9798	40.2556	1.1232	2.8045	39.2840	1.0942	3.2265	41.6475	1.2101	3.7269	41.8829	1.0874	2.6394	38.8343	0.9515	2.7653
3	38.8414	1.0696	3.4398	38.2527	1.0834	3.6227	37.8204	1.0977	3.8028	38.7579	1.2867	3.8452	41.7309	1.3153	3.8670	40.7245	1.1933	4.3468	37.6707	1.2640	4.0968	41.7046	1.1684	3.3996	39.5995	1.0764	3.2495	40.3502	1.0141	2.9320
4	38.5234	0.9810	2.7969	36.9347	1.0874	3.4947	37.2076	1.1779	3.7116	37.7577	1.2131	3.2979	41.1369	1.3480	4.1203	41.8199	1.3590	4.7481	43.9177	1.3447	4.2478	41.0508	1.2542	3.7524	42.4665	1.1951	4.0816	40.0792	1.0278	3.0546
5	38.2090	1.0302	2.7593	36.6624	1.0603	3.1224	37.7943	1.1108	3.9118	40.3440	1.2410	4.1352	41.1155	1.2569	3.7816	42.7613	1.2959	4.2698	42.0445	1.2210	4.2582	41.5918	1.0928	3.2668	37.1979	0.9860	2.8728	37.8917	1.0068	3.4021
Mean	38.5801	1.03036	3.30856	38.00216	1.0637	3.50392	38.01662	1.12782	3.58502	38.80768	1.2325	3.769	40.33658	1.28324	3.9299	41.2959	1.24414	4.0262	40.45416	1.24632	4.01686	41.25656	1.18242	3.51192	40.42684	1.0877	3.45808	40.1136	0.9968	3.0245
Parameters: Boeing 757, FFA, Normal Luggage, Graduated Scale of Groups																														

Table A8 Results (mins) of Investigation of Effect of Increasing P-Value of Group Formation on Random Boarding

Random/ P Value	0.0			0.1			0.2			0.3			0.4			0.5			0.6			0.7			0.8			0.9		
	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind	Total	Ave Ind	Max Ind
1	37.8299	1.1789	2.7170	39.2945	1.1759	2.8700	38.1594	1.1235	2.5259	37.1325	1.1477	2.5155	38.9727	1.1948	2.5050	37.6450	1.1944	2.5382	37.7211	1.1203	2.2580	37.8870	1.1278	2.2846	37.4775	1.0845	2.1865	36.3880	1.1845	2.4945
2	37.0478	1.0908	2.5077	36.9295	1.1238	2.4308	36.7378	1.0947	2.3505	36.7254	1.1467	2.3066	38.7193	1.1145	2.5803	36.9139	1.1746	2.6950	35.8915	1.0418	2.0753	37.8317	1.1407	2.2664	38.6493	1.1923	2.5713	38.2197	1.0551	2.4497
3	38.6286	1.1255	2.3397	36.8526	1.1619	2.3151	38.8609	1.1625	2.6630	36.8501	1.1223	2.3820	36.6582	1.0181	2.2446	38.7357	1.1863	2.4179	36.4018	1.0459	1.9643	37.4455	1.1632	2.4826	35.5324	1.1056	2.5619	36.8148	1.0434	1.8673
4	36.2635	1.1334	2.6184	37.3409	1.1176	2.5410	39.7295	1.0876	2.3591	39.3666	1.1678	2.3906	38.4820	1.1360	2.4461	37.4382	1.1972	2.7118	38.1388	1.1611	2.8457	38.3332	1.2962	2.6464	37.5950	1.1448	2.4205	36.4421	1.2010	2.0348
5	36.6745	1.0942	2.2100	36.9564	1.0426	2.1292	36.6650	1.0461	2.1373	36.0658	1.0733	2.2076	38.4157	1.2548	3.0528	39.2672	1.2179	2.8480	37.8281	1.1913	2.7655	37.8701	1.1131	2.1894	38.5162	1.2475	2.8237	37.2048	1.0998	2.4661
Mean	37.28886	1.12456	2.47856	37.47478	1.12436	2.45722	38.03052	1.10288	2.40716	37.22808	1.13156	2.36046	38.24958	1.14364	2.56576	38	1.19408	2.64218	37.19626	1.11208	2.38176	37.8735	1.1682	2.37388	37.55408	1.15494	2.51278	37.01388	1.11676	2.26248
Parameters: Boeing 757, Random, Normal Luggage, Graduated Scale of Groups																														

APPENDIX B: Source Code

This appendix contains the Java source code used in the implementation of the discrete event simulation model, with the exception of the `Math` package which is available through the standard Java API.

The relationship to the high level diagram of Java entities is noted beside the heading of each Class.

Controller.java (See Controller on Figure 10)

Created with JBuilder

```
package boardingsimulation;

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.util.*;
import java.util.Collections;
import java.lang.Math;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2008
 * Company:
 * @author
 * @version 1.0
 */

public class Controller {

  simulationGUI GUI;
  Seat[] seats;
  simulationGUIPane GUIPane;
  Passenger[] passengers;
  Hashtable groupedPax;
  Vector groupNos;
  java.util.List pax;
  double totalBoardingTime;
  int rows;
  int aisles;
  int paxPerRow;
  int aislePos1;
  int aislePos2;
  Triangular passingRow = new Triangular(1.8, 2.4, 3.0);
  Triangular installInSeat = new Triangular(6.0, 9.0, 30.0);
  Triangular exitSeat = new Triangular(3.0, 3.6, 4.2);
  Double[] boardingTimes;

  public Controller() {
```

```
    GUI = new simulationGUI(this);
  }

  public static void main(String[] args) {
    Controller controller1 = new Controller();
  }

  public void buildSeats(int r, int a, int p, int a1, int a2,
simulationGUIPane sgp){
    rows = r;
    aisles = a;
    paxPerRow = p;
    aislePos1 = a1;
    aislePos2 = a2;
    GUIPane = sgp;
    seats = new Seat[(rows * (aisles + paxPerRow))];
    for(int i = 0; i<(rows * (aisles + paxPerRow)); i++)
      seats[i] = new Seat(i + 1);
    this.assignAisles(aislePos1, aislePos2, (paxPerRow + aisles));
    this.assignBinCapacity(aisles, paxPerRow);
    this.buildPax(rows, paxPerRow);
    for(int i = 0; i < seats.length; i++)
      if(seats[i].getIsAisle()){
        sgp.frame.getComponent(i).setBackground(Color.lightGray);
      }
      else{
        sgp.frame.getComponent(i).setBackground(Color.black);
      }
    this.assignGroups();
    this.passengerLuggage();
    boardingTimes = new Double[seats.length];
    for(int i = 0; i < boardingTimes.length; i++)
      boardingTimes[i] = new Double(0.0);
    groupNos = new Vector(1,1);
    for(int i = 0; i < passengers.length; i++)
      this.groupVector(passengers[i]);
    Object[] groupNos2 = this.groupNos.toArray();
    java.util.List groupNos3 = Arrays.asList(groupNos2);
    Collections.shuffle(groupNos3);
    Object[] groupNos4 = groupNos3.toArray();
    this.populateModel(groupNos4);
  }

  public void groupVector(Passenger p){
    if(!groupNos.contains(groupedPax.get(p))){
      groupNos.add(groupedPax.get(p));
    }
  }

  public void assignAisles(int aisle1, int aisle2, int rowSize){
    int a1 = aisle1;
    int a2 = aisle2;
    int rs = rowSize;
    for(int i = a1 -1; i<seats.length; i = i + rs)
      seats[i].isAisle = true;
    if(a2 > 0){
```

```
      for(int i = a2 -1; i<seats.length; i = i + rs)
        seats[i].isAisle = true;
    }
  }

  public void assignBinCapacity(int aisles, int paxPerRow){
    for(int i = 0; i < seats.length; i++)
      if(seats[i].getIsAisle()){
        if(aisles == 1){
          seats[i].setLuggage(paxPerRow-1);
        }
        else{
          seats[i].setLuggage((paxPerRow/2)-1);
        }
      }
  }

  public void buildPax(int r, int p){
    passengers = new Passenger[r*p];
    int i1 = 0;
    for(int i = 0; i < passengers.length; i++)
      if (seats[i1].getIsAisle()==true){
        passengers[i] = new Passenger(seats[i1+1].getSeatNo());
        i1 = i1+2;
      }
      else{
        passengers[i] = new Passenger(seats[i1].getSeatNo());
        i1++;
      }
  }

  public void assignGroups(){
    double prob2 = GUI.getGroupProb()/10;
    double prob = prob2/10;
    groupedPax = new Hashtable(passengers.length);
    int groupNo = 1;
    for(int i = 0; i<passengers.length; i++)
      if((Math.random()) < prob || Math.random() == prob){
        groupedPax.put(passengers[i], new Integer(groupNo));
      }
      else{
        groupNo++;
        groupedPax.put(passengers[i], new Integer(groupNo));
      }

  }

  public void passengerLuggage(){
    String id = new String();
    pax = Arrays.asList(passengers);
    Collections.shuffle(pax);
    StringTokenizer st = new
StringTokenizer(GUI.getLuggage(),",",false);
    for(int i = 1; i < 5; i++)
      st.nextToken();
```

```
    StringTokenizer st2 = new
StringTokenizer(st.nextToken(),"=",false);
    st2.nextToken();
    String load = st2.nextToken();
    if(load.equals("Normal")){
      int onePiece = (passengers.length * 60)/100;
      int twoPiece = (passengers.length * 30)/100;
      int threePiece = (passengers.length * 10)/100;
      int remainder = (passengers.length) - (onePiece + twoPiece +
threePiece);
      onePiece = onePiece + remainder;
      for(int i = 0; i < passengers.length; i++)
        if(onePiece > 0){
          id = pax.get(i).toString();
          for(int i1 = 0; i1 < passengers.length; i1++)
            if(passengers[i1].toString().equals(id)){
              passengers[i1].setLuggage(1);
            }
          onePiece--;
        }
        else{
          if(twoPiece > 0) {
            id = pax.get(i).toString();
            for(int i2 = 0; i2 < passengers.length; i2++)
              if(passengers[i2].toString().equals(id)){
                passengers[i2].setLuggage(2);
              }
            twoPiece--;
          }
          else{
            if(threePiece > 0) {
              id = pax.get(i).toString();
              for(int i3 = 0; i3 < passengers.length; i3++)
                if(passengers[i3].toString().equals(id)){
                  passengers[i3].setLuggage(3);
                }
              threePiece--;
          }
          }
        }
    }
    else{
      int onePiece = (passengers.length * 20)/100;
      int twoPiece = (passengers.length * 60)/100;
      int threePiece = (passengers.length * 20)/100;
      int remainder = (passengers.length) - (onePiece + twoPiece +
threePiece);
      onePiece = onePiece + remainder;
      for(int i = 0; i < passengers.length; i++)
        if(onePiece > 0){
          id = pax.get(i).toString();
          for(int i1 = 0; i1 < passengers.length; i1++)
            if(passengers[i1].toString().equals(id)){
              passengers[i1].setLuggage(1);
            }
          onePiece--;
```

```
        }
        else{
          if(twoPiece > 0) {
            id = pax.get(i).toString();
            for(int i2 = 0; i2 < passengers.length; i2++)
              if(passengers[i2].toString().equals(id)){
                passengers[i2].setLuggage(2);
              }
            twoPiece--;
          }
          else{
            if(threePiece > 0) {
              id = pax.get(i).toString();
              for(int i3 = 0; i3 < passengers.length; i3++)
                if(passengers[i3].toString().equals(id)){
                  passengers[i3].setLuggage(3);
                }
              threePiece--;
          }
          }
        }
    }

  }

  public void populateModel(Object[] groupNos){
    StringTokenizer st = new
StringTokenizer(GUI.getStrategy(),",",false);
    for(int i = 1; i < 5; i++)
      st.nextToken();
    StringTokenizer st2 = new
StringTokenizer(st.nextToken(),"=",false);
    st2.nextToken();
    String strategy = st2.nextToken();
    if(strategy.equals("FFA")){
      this.ffaProcess(groupNos);
    }
    else{
      if(strategy.equals("Assigned")){
        this.assignedProcess(groupNos);
      }
    }
  }

  public void ffaProcess(Object[] groupNos){
    Passenger targetPax;
    if(aisles == 1){
      for(int i = 0; i < groupNos.length; i++){
        targetPax = null;
        for(int i1 = 0, counter = 1; i1 < passengers.length; i1++)
          if(groupedPax.get(passengers[i1]).equals(groupNos[i])){
            double direction = Math.random();
            if(counter==1){
              this.ffaFreeRow(direction);
              counter++;
              targetPax = passengers[i1];
```

```
            }
            else{
              passengers[i1].setInGroup(true);
              passengers[i1].setTarget(targetPax);
              this.ffaFreeRow(direction);
            }
            GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
            passengers[i1].setCurrentPos(aislePos1);
            passengers[i].setEntryPoint(aislePos1);
            this.processTimes(true);
          }
      }
      for(int count = 0; count < 100; count++){
        //this loops an arbitrary 100 times. If the simulation does not
complete
        //it is likely to be because this number should be higher
        for(int i = 0; i < passengers.length; i++){
          if(!passengers[i].getSeated()){
            this.ffaFreeRow(Math.random());
            this.processTimes(false);
          }
        }
      }
      this.results();
    }
    else{
      //2 aisles
      int target = -1;
      for(int i = 0; i < groupNos.length; i++){
        targetPax = null;
        for(int i1 = 0, counter = 1; i1 < passengers.length; i1++)
          if(groupedPax.get(passengers[i1]).equals(groupNos[i])){
            //target = 0;
            double direction = Math.random();
            if(counter==1){
              this.ffaFreeRow(direction);
              counter++;
              targetPax = passengers[i1];
              if(direction < 0.5 || direction == 0.5){
                GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
                passengers[i1].setCurrentPos(aislePos1);
                passengers[i1].setEntryPoint(aislePos1);
                target = aislePos1;
              }
              else{
                GUIPane.frame.getComponent(aislePos2-
1).setBackground(Color.red);
                passengers[i1].setCurrentPos(aislePos2);
                passengers[i1].setEntryPoint(aislePos2);
                target = aislePos2;
              }
            }
            else{
              passengers[i1].setInGroup(true);
```

```
              passengers[i1].setTarget(targetPax);
              this.ffaFreeRow(direction);
              GUIPane.frame.getComponent(target-
1).setBackground(Color.red);
              passengers[i1].setCurrentPos(target);
              passengers[i1].setEntryPoint(target);
            }
            this.processTimes(true);
          }
      }
      for(int count = 0; count < 100; count++){
        //this loops an arbitrary 100 times. If the simulation does not
complete
        //it is likely to be because this number should be higher
        for(int i = 0; i < passengers.length; i++){
          if(!passengers[i].getSeated()){
            this.ffaFreeRow(Math.random());
            this.processTimes(false);
          }
        }
      }
      this.results();
    }
  }

  public void ffaFreeRow(double d){
    double direction = d;
    for(int i = 0; i < passengers.length; i++){
      if((passengers[i].getCurrentPos() + (paxPerRow + aisles)) >
seats.length){
        passengers[i].setAtEnd(true);
        passengers[i].setInGroup(false);
      }
      if(passengers[i].getAtEnd()==true & !passengers[i].getSeated() &
passengers[i].getSittingDown()==0 & !passengers[i].getInGroup()){
        this.ffaEnd(passengers[i], direction);
      }
      if(passengers[i].getInGroup() == true &
!passengers[i].getSeated() & passengers[i].getCurrentPos()>-1 &
passengers[i].getSittingDown()==0){
        this.ffaGroup(passengers[i], direction);
      }
      if(passengers[i].getCurrentPos()>-1 & !passengers[i].getSeated()
& passengers[i].getSittingDown()==1 & !passengers[i].getInGroup()){
        passengers[i].setWaiting(true);
        this.stowLuggage(passengers[i], false,
passengers[i].getSeatPos(), passengers[i].getCurrentPos());
      }
      if(passengers[i].getCurrentPos()>-1 & !passengers[i].getSeated()
& passengers[i].getSittingDown()==2 & !passengers[i].getInGroup()){
        passengers[i].setWaiting(true);
        this.stowLuggage(passengers[i], false,
passengers[i].getSeatPos(), passengers[i].getCurrentPos());
      }
```

```
      if(passengers[i].getCurrentPos() > -1 &
!passengers[i].getSeated() & passengers[i].getSittingDown()==0 &
!passengers[i].getInGroup()){
        if(seats[passengers[i].getCurrentPos()].getIsAisle()){
          boolean free = true;
          if(direction < 0.5 || direction == 0.5){
            for(int i1 = 1; i1 < aislePos1; i1++)

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()-
i1).getBackground().equals(Color.red)){
                free = false;
              }
            if(free == true && (!passengers[i].getSeated())){
              if(aisles ==2){
                if(passengers[i].getEntryPoint()==aislePos1){
                  this.stowLuggage(passengers[i], free, (-(aislePos1-
1)), passengers[i].getCurrentPos());
                }
              }
              else{
                this.stowLuggage(passengers[i], free, -(paxPerRow/2),
passengers[i].getCurrentPos());
                }
            }
            else{
              free = true;
              for(int i1 = 1; i1 < aislePos1; i1++)

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()+i1).getBack
ground().equals(Color.red)){
                  free = false;
                }
              if(free == true && (!passengers[i].getSeated())){
                if(aisles == 2){
                  if(passengers[i].getEntryPoint()==aislePos2){
                    this.stowLuggage(passengers[i], free, (aislePos1-
1), passengers[i].getCurrentPos());
                  }
                }
                else{
                  this.stowLuggage(passengers[i], free, (paxPerRow/2),
passengers[i].getCurrentPos());
                  }
              }
              else{
              if((passengers[i].getCurrentPos() +
(paxPerRow+aisles))>(seats.length)){
                passengers[i].setAtEnd(true);
                this.ffaEnd(passengers[i], direction);
              }
              else{
                if(!passengers[i].getSeated()){
                  this.ffaCheckRows(passengers[i]);
                }
              }
            }
```

```
            }
          }
          else{
            for(int i1 = 1; i1 < aislePos1; i1++)

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()+i1).getBack
ground().equals(Color.red)){
                free = false;
              }
            if(free == true && (!passengers[i].getSeated())){
              if(aisles == 2){
                if(passengers[i].getEntryPoint()==aislePos2){
                  this.stowLuggage(passengers[i], free, (aislePos1-1),
passengers[i].getCurrentPos());
                }
              }
              else{
                this.stowLuggage(passengers[i], free, (paxPerRow/2),
passengers[i].getCurrentPos());
              }
            }
            else{
              free = true;
              for(int i1 = 1; i1 < aislePos1; i1++)

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()-
i1).getBackground().equals(Color.red)){
                  free = false;
                }
              if(free == true && (!passengers[i].getSeated())){
                if(aisles == 2){
                  if(passengers[i].getEntryPoint()==aislePos1){
                    this.stowLuggage(passengers[i], free, (-(aislePos1-
1)), passengers[i].getCurrentPos());
                  }
                }
                else{
                  this.stowLuggage(passengers[i], free, -(paxPerRow/2),
passengers[i].getCurrentPos() );
                }
              }
              if((passengers[i].getCurrentPos() +
(paxPerRow+aisles))>seats.length){
                passengers[i].setAtEnd(true);
                this.ffaEnd(passengers[i], direction);
              }
              else{
                if(!passengers[i].getSeated()){
                  this.ffaCheckRows(passengers[i]);
                }
              }
            }
          }
        }
        if(!passengers[i].getSeated() && passengers[i].getCurrentPos()
>-1 && passengers[i].getSittingDown()==0 && !passengers[i].getAtEnd()){
```

```
          if((passengers[i].getCurrentPos() +
(paxPerRow+aisles))<seats.length){
            this.ffaCheckRows(passengers[i]);
          }
        }
      }
    }
  }

  public void ffaCheckRows(Passenger p){
    boolean free1 = false;
    boolean free2 = false;
    boolean f;
    double direction = Math.random();
    int counter = 0;
    f = true;
    checkLoop:{
    for(int i = p.getCurrentPos() + (paxPerRow + aisles); i <
seats.length & counter<3; i = i + (paxPerRow + aisles), counter++){
      f = true;
      if(counter < 3){
        //checkLoop:{
          for(int i1 = 1; i1 < aislePos1; i1++){
            if(GUIPane.frame.getComponent(i -
i1).getBackground().equals(Color.red)){
              f = false;
            }
          }
          if(f == true){
            break checkLoop;
          }
        //}
      }
    }
    }
    if(f == true){
      free1 = true;
    }
    else{
      free1 = false;
    }
    f = true;
    counter = 0;
    checkLoop:{
    for(int i = p.getCurrentPos() + (paxPerRow + aisles); i <
(seats.length) & counter<3; i = i + (paxPerRow + aisles), counter++){
      f = true;
      if(counter < 3){
        //checkLoop:{
          for(int i1 = 1; i1 < aislePos1; i1++){
            if(GUIPane.frame.getComponent(i +
i1).getBackground().equals(Color.red)){
              f = false;
            }
            if(f == true){
              break checkLoop;
```

```
            }
          }
        //}
      }
    }
    }
    if(f == true){
      free2 = true;
    }
    else{
      free2 = false;
    }
    if(!free1 && !free2){
      if(direction < 0.5 || direction == 0.5){
        if(!GUIPane.frame.getComponent(p.getCurrentPos()-
1).getBackground().equals(Color.red)){
          this.stowLuggage(p, true, -1, p.getCurrentPos());
        }
        else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+1).getBackground().equ
als(Color.red)){
            this.stowLuggage(p, true, 1, p.getCurrentPos());
          }
          else{
            this.ffaAisleSeats(p);
          }
        }
      }
      else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+1).getBackground().equ
als(Color.red)){
          this.stowLuggage(p, true, 1, p.getCurrentPos());
        }
        else{
          if(!GUIPane.frame.getComponent(p.getCurrentPos()-
1).getBackground().equals(Color.red)){
            this.stowLuggage(p, true, -1, p.getCurrentPos());
          }
          else{
            this.ffaAisleSeats(p);
          }
        }
      }
    }
    else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
          this.moveForward(p, (paxPerRow+aisles));
        }
        else{
          double bored = Math.random();
          if(bored < 0.3){
            //shows that they are not in the final row yet
```

```
            p.setAtEnd(false);
            this.ffaEnd(p, direction);
          }
          else{
            //queuing
            boardingTimes[p.getSeatNo()-1] = new Double(0.0);
            double a = 0.0;
            a = passingRow.sampleDouble();
            p.setBoardingTime(a);
            boardingTimes[p.getSeatNo()-1] = new Double(a);
          }
        }
    }
  }

  public void ffaAisleSeats(Passenger p){
    boolean free1 = false;
    boolean free2 = false;
    boolean f = true;
    double direction = Math.random();
    int counter = 0;
    for(int i = p.getCurrentPos() + (paxPerRow + aisles); i <
seats.length & counter < 3; i = i + (paxPerRow + aisles), counter++){
      if(counter < 3){
        if(GUIPane.frame.getComponent(i -
1).getBackground().equals(Color.red)){
          f = false;
        }
      }
      if(f == true){
        free1 = true;
        break;
      }
      else{
        free1 = false;
      }
    }
    counter = 0;
    f = true;
    for(int i = p.getCurrentPos() + (paxPerRow + aisles); i <
seats.length & counter < 3; i = i + (paxPerRow + aisles), counter++){
      if(counter < 3){
        if(GUIPane.frame.getComponent(i +
1).getBackground().equals(Color.red)){
          f = false;
        }
      }
      if(f == true){
        free2 = true;
        break;
      }
      else{
        free2 = false;
      }
    }
    if(!free1 && !free2 && !p.getSeated() && !p.getAtEnd()){
```

```
      if(direction < 0.5 || direction == 0.5){
        if(!GUIPane.frame.getComponent(p.getCurrentPos()-
2).getBackground().equals(Color.red)){
          this.stowLuggage(p, false, -2, p.getCurrentPos());
        }
        else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+2).getBackground().equ
als(Color.red)){
            this.stowLuggage(p, false, 2, p.getCurrentPos());
          }
          else{
            if(p.getCurrentPos()+(paxPerRow+aisles)<seats.length){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
                this.moveForward(p, (paxPerRow+aisles));
              }
              else{
                //queuing
                boardingTimes[p.getSeatNo()-1] = new Double(0.0);
                double a = 0.0;
                a = passingRow.sampleDouble();
                p.setBoardingTime(a);
                boardingTimes[p.getSeatNo()-1] = new Double(a);
              }
            }
          }
        }
      }
      else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+2).getBackground().equ
als(Color.red)){
          this.stowLuggage(p, false, 2, p.getCurrentPos());
        }
        else{
          if(!GUIPane.frame.getComponent(p.getCurrentPos()-
2).getBackground().equals(Color.red)){
            this.stowLuggage(p, false, -2, p.getCurrentPos());
          }
          else{
            if(p.getCurrentPos()+(paxPerRow+aisles) < seats.length){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
                this.moveForward(p, paxPerRow+aisles);
              }
              else{
                //queuing
                boardingTimes[p.getSeatNo()-1] = new Double(0.0);
                double a = 0.0;
                a = passingRow.sampleDouble();
                p.setBoardingTime(a);
                boardingTimes[p.getSeatNo()-1] = new Double(a);
              }
```

```
            }
          }
        }
      }
    }
    else{
      if(!p.getSeated() && !p.getAtEnd()){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
          this.moveForward(p, paxPerRow+aisles);
        }
        else{
          double bored = Math.random();
          if(bored < 0.3){
            //shows that they are not in the final row yet
            p.setAtEnd(false);
            this.ffaEnd(p, direction);
          }
          else{
            //queuing
            boardingTimes[p.getSeatNo()-1] = new Double(0.0);
            double a = 0.0;
            a = passingRow.sampleDouble();
            p.setBoardingTime(a);
            boardingTimes[p.getSeatNo()-1] = new Double(a);
          }
        }
      }
    }
  }

  public void ffaEnd(Passenger p, double d){
    double direction = d;
    if(aisles == 2){
      if(p.getEntryPoint() == aislePos1){
        if(direction < 0.5 || direction == 0.5){
          for(int check = 1; check < aislePos1-1; check++){
            if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, -check, p.getCurrentPos());
            }
          }
          for(int check = 1; check < (paxPerRow-((aislePos1-1)*2));
check++){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, check, p.getCurrentPos());
            }
          }
        }
        else{
          for(int check = 1; check < (paxPerRow-((aislePos1-1)*2));
check++){
```

```
if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, check, p.getCurrentPos());
            }
          }
          for(int check = 1; check < aislePos1-1; check++){
            if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, -check, p.getCurrentPos());
            }
          }
        }
      }
      else{
        //aislePos2
        if(direction < 0.5 || direction == 0.5){
          for(int check = 1; check < (paxPerRow-((aislePos1-1)*2));
check++){
            if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, -check, p.getCurrentPos());
            }
          }
          for(int check = 1; check < aislePos1-1; check++){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, check, p.getCurrentPos());
            }
          }
        }
        else{
          for(int check = 1; check < aislePos1-1; check++){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, check, p.getCurrentPos());
            }
          }
          for(int check = 1; check < (paxPerRow-((aislePos1-1)*2));
check++){
            if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
              this.stowLuggage(p, false, -check, p.getCurrentPos());
            }
          }
        }
      }
    }
    else{
      if(direction < 0.5 || direction == 0.5){
        for(int check = 1; check < aislePos1-1; check++){
          if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
            this.stowLuggage(p, false, -check, p.getCurrentPos());
```

```
          }

if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
            this.stowLuggage(p, false, check, p.getCurrentPos());
          }
        }
      }
      else{
        for(int check = 1; check < aislePos1-1; check++){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+check).getBackground()
.equals(Color.red) && !p.getSeated()){
            this.stowLuggage(p, false, check, p.getCurrentPos());
          }
          if(!GUIPane.frame.getComponent(p.getCurrentPos()-
check).getBackground().equals(Color.red) && !p.getSeated()){
            this.stowLuggage(p, false, -check, p.getCurrentPos());
          }
        }
      }
    }
    //this is for those passengers who have come through the 'bored'
route
    if(!p.getAtEnd() & !p.getSeated() & p.getSittingDown()==0){

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
        this.moveForward(p, (paxPerRow + aisles));
      }
      else{
        //queuing
        boardingTimes[p.getSeatNo()-1] = new Double(0.0);
        double a = 0.0;
        a = passingRow.sampleDouble();
        p.setBoardingTime(a);
        boardingTimes[p.getSeatNo()-1] = new Double(a);
      }
    }
    if(p.getAtEnd()==true & !p.getSeated() & p.getSittingDown()==0){
      boardingTimes[p.getSeatNo()-1] = new Double(0.0);
      double a = 0.0;
      endLoop:{
        for(int i = p.getCurrentPos(); i > 0 & !p.getSeated() &
p.getSittingDown() == 0; i = i - (paxPerRow+aisles)){
          if(!p.getSeated() & p.getSittingDown()==0){
            if(aisles == 2){
              for(int check = 1; check < (((paxPerRow/2)-aisles)+1);
check++){
                if(!GUIPane.frame.getComponent(i-
check).getBackground().equals(Color.red)){
                  this.stowLuggage(p, false, -check, i);
                  break endLoop;
                }
              }
            }
```

```
            else{
              for(int check = 1; check < paxPerRow/2+1; check++){
                if(GUIPane.frame.getComponent(i-
check).getBackground().equals(Color.black)){
                  this.stowLuggage(p, false, -check, i);
                  break endLoop;
                }
              }
            }
          }
          if(!p.getSeated() & p.getSittingDown()==0){
            if(aisles == 2){
              for(int check = 1; check < (((paxPerRow/2)-aisles)+1);
check++){

if(!GUIPane.frame.getComponent(i+check).getBackground().equals(Color.re
d)){
                  this.stowLuggage(p, false, check, i);
                  break endLoop;
                }
              }
            }
            else{
              for(int check = 1; check < paxPerRow/2+1; check++){

if(GUIPane.frame.getComponent(i+check).getBackground().equals(Color.bla
ck)){
                  this.stowLuggage(p, false, check, i);
                  break endLoop;
                }
              }
            }
          }
        }
      }
      if(!p.getSeated() && p.getSittingDown()==0){
        if(aisles == 2){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
          if(p.getEntryPoint()==aislePos1){
            p.setEntryPoint(aislePos2);
            p.setCurrentPos((paxPerRow-((aislePos1-1)*2))+1);

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
            a = a + installInSeat.sampleDouble();
            p.setBoardingTime(a);
            boardingTimes[p.getSeatNo()-1] = new Double(a);
          }
          else{
            p.setEntryPoint(aislePos1);
            p.setCurrentPos(-((paxPerRow-((aislePos1-1)*2))+1));

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
```

```
            a = a + installInSeat.sampleDouble();
            p.setBoardingTime(a);
            boardingTimes[p.getSeatNo()-1] = new Double(a);
          }
        }
      }
    }
  }

  public void ffaGroup(Passenger p, double d){
    //when passengers sit down they should set the aisle as their
target position. group members go to this.
    //so if their current position equals the target position for the
person before they try to sit down in the same row
    //or as close as possible (use ffaEnd). Otherwise they move
forward.
    if(p.getCurrentPos()>-1){
      if(p.getCurrentPos()==p.getTarget().getTargetPos()){
        //setting inGroup to false allows them to pass through the
normal processing from this point in.
        p.setInGroup(false);
        p.setAtEnd(false);
        this.ffaEnd(p,d);
      }
      else{
        if((p.getCurrentPos() + (paxPerRow+aisles))>(seats.length)){
          p.setInGroup(false);
          p.setAtEnd(true);
          this.ffaEnd(p, d);
        }
        else{

if(!GUIPane.frame.getComponent(p.getCurrentPos()+(paxPerRow+aisles)).ge
tBackground().equals(Color.red)){
            this.moveForward(p, (paxPerRow+aisles));
          }
          else{
            //queuing
            boardingTimes[p.getSeatNo()-1] = new Double(0.0);
            double a = 0.0;
            a = passingRow.sampleDouble();
            p.setBoardingTime(a);
            boardingTimes[p.getSeatNo()-1] = new Double(a);
          }
        }
      }
    }
  }


  public void assignedProcess(Object[] groupNos){
    if(aisles==1){
      for(int i = 0; i < groupNos.length; i++)
        for(int i1 = 0; i1 < passengers.length; i1++)
          if(groupedPax.get(passengers[i1]).equals(groupNos[i])){
            this.assignedMovements();
```

```
            GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
            passengers[i1].setCurrentPos(aislePos1);
            this.processTimes(true);
          }
      for(int count = 0; count < 100; count++)
      //this loops an arbitrary 100 times. If the simulation does not
complete
      //it is likely to be because this number should be higher
        for(int i = 0; i < passengers.length; i++){
          if(!passengers[i].getSeated()){
            this.assignedMovements();
            this.processTimes(false);
          }
        }
      this.results();
    }
    else{
      for(int i = 0; i < groupNos.length; i++){
        for(int i1 = 0; i1 < passengers.length; i1++){
          if(groupedPax.get(passengers[i1]).equals(groupNos[i])){
            this.assignedMovements();
            if(passengers[i1].getCurrentPos() == -1){
              for(int check = aislePos1; check < seats.length; check =
check + (paxPerRow+aisles)){
                for(int seatCheck = 1; seatCheck < aislePos1;
seatCheck++){
                  if(check - seatCheck == passengers[i1].getSeatNo()){
                    GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
                    passengers[i1].setCurrentPos(aislePos1);
                  }
                }
                if(check + 1 == passengers[i1].getSeatNo()){
                  GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
                  passengers[i1].setCurrentPos(aislePos1);
                }
              }
              if(passengers[i1].getCurrentPos() == -1){
                for(int check = aislePos2; check < seats.length; check
= check + (paxPerRow+aisles)){
                  for(int seatCheck = 1; seatCheck < aislePos1;
seatCheck++){
                    if(check + seatCheck ==
passengers[i1].getSeatNo()){
                      GUIPane.frame.getComponent(aislePos2-
1).setBackground(Color.red);
                      passengers[i1].setCurrentPos(aislePos2);
                    }
                  }
                  if(check - 1 == passengers[i1].getSeatNo()){
                    GUIPane.frame.getComponent(aislePos2-
1).setBackground(Color.red);
                    passengers[i1].setCurrentPos(aislePos2);
                  }
```

```
                }
                if(passengers[i1].getCurrentPos() == -1){
                  double direction = Math.random();
                  if(direction == 0.5 || direction < 0.5){
                    GUIPane.frame.getComponent(aislePos1-
1).setBackground(Color.red);
                    passengers[i1].setCurrentPos(aislePos1);
                  }
                  else{
                    GUIPane.frame.getComponent(aislePos2-
1).setBackground(Color.red);
                    passengers[i1].setCurrentPos(aislePos2);
                  }
                }
              }
            }
          this.processTimes(true);
          }
        }
      }
      for(int count = 0; count < 100; count++)
      //this loops an arbitrary 100 times. If the simulation does not
complete
      //it is likely to be because this number should be higher
        for(int i = 0; i < passengers.length; i++){
          if(!passengers[i].getSeated()){
            this.assignedMovements();
            this.processTimes(false);
          }
        }
      this.results();
    }
  }

  public void assignedMovements(){
    for(int i = 0; i < passengers.length; i++){
      if(passengers[i].getSittingDown()==1 &&
!passengers[i].getSeated()){
        passengers[i].setWaiting(true);
        this.stowLuggage(passengers[i], false,
passengers[i].getSeatPos(), passengers[i].getCurrentPos());
      }
      if(passengers[i].getSittingDown()==2 &&
!passengers[i].getSeated()){
        passengers[i].setWaiting(true);
        this.stowLuggage(passengers[i], false,
passengers[i].getSeatPos(), passengers[i].getCurrentPos());
      }
      if(passengers[i].getCurrentPos() > -1 &&
!passengers[i].getSeated() && passengers[i].getSittingDown()==0){
        //these boolean variables indicate whether seats are occupied
or not
        boolean window = false;
        boolean middle = false;
        boolean aisle = false;
```

```
        if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()-
3).getBackground().equals(Color.red)){
          window = true;
        }
        if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()-
2).getBackground().equals(Color.red)){
          middle = true;
        }
        if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()-
1).getBackground().equals(Color.red)){
          aisle = true;
        }
        for(int i1 = 1; i1 < (paxPerRow/2)+1; i1++)
          if((passengers[i].getCurrentPos()-
i1)==(passengers[i].getSeatNo()-1)){
            if(window == false && middle == false && aisle == false){
              this.stowLuggage(passengers[i], true, -i1,
passengers[i].getCurrentPos());
            }
            else{
              this.stowLuggage(passengers[i], false, -i1,
passengers[i].getCurrentPos());
            }
          }
        //these boolean variables indicate whether seats are occupied
or not
        window = false;
        middle = false;
        aisle = false;

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()+3).getBackg
round().equals(Color.red)){
          window = true;
        }

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()+2).getBackg
round().equals(Color.red)){
          middle = true;
        }

if(GUIPane.frame.getComponent(passengers[i].getCurrentPos()+1).getBackg
round().equals(Color.red)){
          aisle = true;
        }
        for(int i1 = 1; i1 < (paxPerRow/2)+1; i1++)

if((passengers[i].getCurrentPos()+i1)==(passengers[i].getSeatNo()-1)){
            if(window == false && middle == false && aisle == false){
              this.stowLuggage(passengers[i], true, i1,
passengers[i].getCurrentPos());
            }
            else{
              this.stowLuggage(passengers[i], false, i1,
passengers[i].getCurrentPos());
            }
          }
```

```
        if(!passengers[i].getSeated() &&
passengers[i].getSittingDown()==0){

if(!GUIPane.frame.getComponent(passengers[i].getCurrentPos()+(paxPerRow
+aisles)).getBackground().equals(Color.red)){
            this.moveForward(passengers[i], (paxPerRow+aisles));
          }
          else{
            //queuing
            boardingTimes[passengers[i].getSeatNo()-1] = new
Double(0.0);
            double a = 0.0;
            a = passingRow.sampleDouble();
            passengers[i].setBoardingTime(a);
            boardingTimes[passengers[i].getSeatNo()-1] = new Double(a);
          }
        }
      }
    }
  }

  public void setTotalBoardingTime(double time){
    totalBoardingTime = totalBoardingTime + time;
  }

  public double getTotalBoardingTime(){
    return totalBoardingTime;
  }

  public void processTimes(boolean incoming){
    double time = 0.0;
    for(int i = 0; i < boardingTimes.length; i++)
      if(time < boardingTimes[i].doubleValue()){
        time = boardingTimes[i].doubleValue();
      }
    if(incoming == true){
      if(time > 6){
        this.setTotalBoardingTime(time);
      }
      else{
        this.setTotalBoardingTime(6);
      }
    }
    else{
      this.setTotalBoardingTime(time);
    }
    for(int i = 0; i < boardingTimes.length; i++){
      boardingTimes[i] = new Double(0.0);
    }
  }

  public void results(){
    Double[] paxTimes = new Double[passengers.length];
    for(int i = 0; i < paxTimes.length; i++){
      paxTimes[i] = new Double(passengers[i].getBoardingTime());
    }
```

```
    double totalPaxTimes = 0.0;
    for(int i = 0; i <paxTimes.length; i++)
      totalPaxTimes = totalPaxTimes + paxTimes[i].doubleValue();
    double averagePaxTime = totalPaxTimes/paxTimes.length;
    double maxPaxTime = 0.0;
    for(int i = 0; i < paxTimes.length; i++)
      if(maxPaxTime < paxTimes[i].doubleValue())
        maxPaxTime = paxTimes[i].doubleValue();
    System.out.println(averagePaxTime/60);
    System.out.println(maxPaxTime/60);
    System.out.println(this.getTotalBoardingTime()/60);
    for(int i = 0; i < boardingTimes.length; i++)
      boardingTimes[i] = new Double(0.0);
    this.setTotalBoardingTime(0.0-this.getTotalBoardingTime());
  }

  public void stowLuggage(Passenger p, boolean free, int side, int
aislePlacer){
    double a = 0.0;
    double rowsPassed = 0.0;
    if(aislePlacer < p.getCurrentPos()){
      if(p.getEntryPoint()==aislePos1){
        rowsPassed = (p.getCurrentPos()-
aislePlacer)/(paxPerRow+aisles);
      }
      else{
        rowsPassed = (p.getCurrentPos()-
aislePlacer)/(paxPerRow+aisles);
      }

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

if(!GUIPane.frame.getComponent(aislePlacer).getBackground().equals(Colo
r.red)){
        p.setCurrentPos(-(p.getCurrentPos()-aislePlacer));

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.red);
        a = passingRow.sampleDouble() * rowsPassed;
        p.setBoardingTime(a);
        boardingTimes[p.getSeatNo()-1] = new Double(a);
        this.stowLuggageProcess(p, free, side, aislePlacer);
      }
      else{

      }
    }
    else{
      this.stowLuggageProcess(p, free, side, aislePlacer);
    }
  }

  public void stowLuggageProcess(Passenger p, boolean free, int side,
int aislePlacer){
    //free indicates whether the row is free or not
    p.setSeatPos(side);
```

```
    boardingTimes[p.getSeatNo()-1] = new Double(0.0);
    double a = 0.0;
    if(p.getLuggage()>0){
      for(int l = p.getLuggage(); l > 0; l--)
        if(seats[aislePlacer].getLuggage() > 0){

seats[aislePlacer].setLuggage(seats[aislePlacer].getLuggage()-1);
        }
        else{
          a = passingRow.sampleDouble();
          p.setBoardingTime(a);
          boardingTimes[p.getSeatNo()-1] = new
Double(boardingTimes[p.getSeatNo()-1].doubleValue() + a);
          p.setSeatPos(side);
          if(!p.getAtEnd()){
            p.setSittingDown(1);
            p.setSeatPos(side);
          }
        }
      p.setLuggage(0);
    }
    //if luggage all stowed successfully then sit down, otherwise wait.
    if(p.getAtEnd()==true & p.getCurrentPos()==aislePlacer &
!p.getSeated() & p.getSittingDown()==0){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
      p.setTargetPos(aislePlacer);
      p.setCurrentPos((aislePlacer+side)-aislePlacer);
      p.setSeated(true);
      double b = installInSeat.sampleDouble();
      double c = exitSeat.sampleDouble();
      double d = exitSeat.sampleDouble();
      //double e is additional time because people have to manouveur
round in a tight space
      double e = passingRow.sampleDouble();
      /*double f is an arbitrary timing representing the fact that
these pax have had to move
      against the flow of people to find a seat.*/
      double f = installInSeat.sampleDouble();
      if(side == 2 || side == -2){
        p.setBoardingTime(b+c+e+f);
        boardingTimes[p.getSeatNo()-1] = new Double(b+c+e+f);
      }
      else{
        if(side == 3 || side == -3){
          p.setBoardingTime(b+c+d+e+f);
          boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e+f);
        }
        else{
          p.setBoardingTime(b+e+f);
          boardingTimes[p.getSeatNo()-1] = new Double(b+e+f);
        }
      }
```

```
    }
    if(p.getSittingDown() == 0 & !p.getSeated() & !p.getAtEnd()){
      //check whether this is a free row
      if(free == true || side == 1 || side == -1){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
        //occupy window seat

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
        p.setTargetPos(aislePlacer);
        p.setCurrentPos((aislePlacer+side)-aislePlacer);
        p.setSeated(true);
        double b = installInSeat.sampleDouble();
        p.setBoardingTime(b);
        boardingTimes[p.getSeatNo()-1] = new Double(b);
      }
      else{
        //check whether they're in the final row
        if((p.getCurrentPos() + (paxPerRow+aisles))>(seats.length)){
          if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
            p.setTargetPos(aislePlacer);
            p.setCurrentPos((aislePlacer+side)-aislePlacer);
            p.setSeated(true);
            double b = installInSeat.sampleDouble();
            double c = exitSeat.sampleDouble();
            double d = exitSeat.sampleDouble();
            if(side == 2 || side == -2){
              p.setBoardingTime(b+c);
              boardingTimes[p.getSeatNo()-1] = new Double(b+c);
            }
            else{
              if(side == 3 || side == -3){
                p.setBoardingTime(b+c+d);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
              }
            }
          }
          else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
            p.setTargetPos(aislePlacer);
            p.setCurrentPos((aislePlacer+side)-aislePlacer);
            p.setSeated(true);
            double b = installInSeat.sampleDouble();
            double c = exitSeat.sampleDouble();
            double d = exitSeat.sampleDouble();
```

```
            //double e is additional time because people have to
manouveur round in a tight space
            double e = passingRow.sampleDouble();
            if(side == 2 || side == -2){
              p.setBoardingTime(b+c+e);
              boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
            }
            else{
              if(side == 3 || side == -3){
                p.setBoardingTime(b+c+d+e);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e);
              }
            }
          }
        }
        else{
          if((p.getCurrentPos() - (paxPerRow+aisles))<0){

if(!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackg
round().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                }
              }
            }
            else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              //double e is additional time because people have to
manouveur round in a tight space
```

```
              double e = passingRow.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c+e);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d+e);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e);
                }
              }
            }
          }
          else{
            if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red) ||
!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackgrou
nd().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                }
              }
            }
            else{
              p.setSittingDown(2);
              p.setSeatPos(side);
            }
          }
        }
      }
    }
    else{
      if(p.getSittingDown()==1 & p.getWaiting()==true){
        if((p.getCurrentPos() + (paxPerRow+aisles))>(seats.length)){
          if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
```

```
GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
            p.setTargetPos(aislePlacer);
            p.setCurrentPos((aislePlacer+side)-aislePlacer);
            p.setSeated(true);
            double b = installInSeat.sampleDouble();
            double c = exitSeat.sampleDouble();
            double d = exitSeat.sampleDouble();
            if(side == 2 || side == -2){
              p.setBoardingTime(b+c);
              boardingTimes[p.getSeatNo()-1] = new Double(b+c);
            }
            else{
              if(side == 3 || side == -3){
                p.setBoardingTime(b+c+d);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
              }
              else{
                p.setBoardingTime(b);
                boardingTimes[p.getSeatNo()-1] = new Double(b);
              }
            }
          }
          else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
            p.setTargetPos(aislePlacer);
            p.setCurrentPos((aislePlacer+side)-aislePlacer);
            p.setSeated(true);
            double b = installInSeat.sampleDouble();
            double c = exitSeat.sampleDouble();
            double d = exitSeat.sampleDouble();
            //double e is additional time because people have to
manouveur round in a tight space
            double e = passingRow.sampleDouble();
            if(side == 2 || side == -2){
              p.setBoardingTime(b+c+e);
              boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
            }
            else{
              if(side == 3 || side == -3){
                p.setBoardingTime(b+c+d+e);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e);
              }
              else{
                p.setBoardingTime(b+e);
                boardingTimes[p.getSeatNo()-1] = new Double(b+e);
              }
            }
          }
        }
        else{
          if((p.getCurrentPos() - (paxPerRow+aisles))<0){
```

```
if(!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackg
round().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                }
                else{
                  p.setBoardingTime(b);
                  boardingTimes[p.getSeatNo()-1] = new Double(b);
                }
              }
            }
            else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              //double e is additional time because people have to
manouveur round in a tight space
              double e = passingRow.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c+e);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d+e);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e);
                }
                else{
                  p.setBoardingTime(b+e);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+e);
```

```
                }
              }
            }
          }
          else{
            if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red) ||
!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackgrou
nd().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
              double c = exitSeat.sampleDouble();
              double d = exitSeat.sampleDouble();
              if(side == 2 || side == -2){
                p.setBoardingTime(b+c);
                boardingTimes[p.getSeatNo()-1] = new Double(b+c);
              }
              else{
                if(side == 3 || side == -3){
                  p.setBoardingTime(b+c+d);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                }
                else{
                  p.setBoardingTime(b);
                  boardingTimes[p.getSeatNo()-1] = new Double(b);
                }
              }
            }
            else{
              p.setSittingDown(2);
              p.setSeatPos(side);
            }
          }
        }
      }
      else{
        if(p.getSittingDown()==2 & p.getWaiting()==true){
          if((aislePlacer + (paxPerRow+aisles))>(seats.length)){
            if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
              p.setTargetPos(aislePlacer);
              p.setCurrentPos((aislePlacer+side)-aislePlacer);
              p.setSeated(true);
              double b = installInSeat.sampleDouble();
```

```
                double c = exitSeat.sampleDouble();
                double d = exitSeat.sampleDouble();
                if(side == 2 || side == -2){
                  p.setBoardingTime(b+c);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c);
                }
                else{
                  if(side == 3 || side == -3){
                    p.setBoardingTime(b+c+d);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                  }
                  else{
                    p.setBoardingTime(b);
                    boardingTimes[p.getSeatNo()-1] = new Double(b);
                  }
                }
              }
              else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
                p.setTargetPos(aislePlacer);
                p.setCurrentPos((aislePlacer+side)-aislePlacer);
                p.setSeated(true);
                double b = installInSeat.sampleDouble();
                double c = exitSeat.sampleDouble();
                double d = exitSeat.sampleDouble();
                //double e is additional time because people have to
manouveur round in a tight space
                double e = passingRow.sampleDouble();
                if(side == 2 || side == -2){
                  p.setBoardingTime(b+c+e);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
                }
                else{
                  if(side == 3 || side == -3){
                    p.setBoardingTime(b+c+d+e);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+c+d+e);
                  }
                  else{
                    p.setBoardingTime(b+e);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+e);
                  }
                }
              }
            }
            else{
              if((aislePlacer - (paxPerRow+aisles))<0){

if(!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackg
round().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);
```

```
GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
                p.setTargetPos(aislePlacer);
                p.setCurrentPos((aislePlacer+side)-aislePlacer);
                p.setSeated(true);
                double b = installInSeat.sampleDouble();
                double c = exitSeat.sampleDouble();
                double d = exitSeat.sampleDouble();
                if(side == 2 || side == -2){
                  p.setBoardingTime(b+c);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c);
                }
                else{
                  if(side == 3 || side == -3){
                    p.setBoardingTime(b+c+d);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                  }
                  else{
                    p.setBoardingTime(b);
                    boardingTimes[p.getSeatNo()-1] = new Double(b);
                  }
                }
              }
              else{

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
                p.setTargetPos(aislePlacer);
                p.setCurrentPos((aislePlacer+side)-aislePlacer);
                p.setSeated(true);
                double b = installInSeat.sampleDouble();
                double c = exitSeat.sampleDouble();
                double d = exitSeat.sampleDouble();
                //double e is additional time because people have to
manouveur round in a tight space
                double e = passingRow.sampleDouble();
                if(side == 2 || side == -2){
                  p.setBoardingTime(b+c+e);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c+e);
                }
                else{
                  if(side == 3 || side == -3){
                    p.setBoardingTime(b+c+d+e);
                    boardingTimes[p.getSeatNo()-1] = new
Double(b+c+d+e);
                  }
                  else{
                    p.setBoardingTime(b+e);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+e);
                  }
                }
              }
            }
            else{
```

```
            if(!GUIPane.frame.getComponent(aislePlacer-
(paxPerRow+aisles)).getBackground().equals(Color.red) ||
!GUIPane.frame.getComponent(aislePlacer+(paxPerRow+aisles)).getBackgrou
nd().equals(Color.red)){

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(aislePlacer+side).setBackground(Color.red);
                p.setTargetPos(aislePlacer);
                p.setCurrentPos((aislePlacer+side)-aislePlacer);
                p.setSeated(true);
                double b = installInSeat.sampleDouble();
                double c = exitSeat.sampleDouble();
                double d = exitSeat.sampleDouble();
                if(side == 2 || side == -2){
                  p.setBoardingTime(b+c);
                  boardingTimes[p.getSeatNo()-1] = new Double(b+c);
                }
                else{
                  if(side == 3 || side == -3){
                    p.setBoardingTime(b+c+d);
                    boardingTimes[p.getSeatNo()-1] = new Double(b+c+d);
                  }
                  else{
                    p.setBoardingTime(b);
                    boardingTimes[p.getSeatNo()-1] = new Double(b);
                  }
                }
              }
              else{
                p.setSittingDown(2);
                p.setSeatPos(side);
              }
            }
          }
        }
      }
    }
  }

  public void moveForward(Passenger p, int step){
    boardingTimes[p.getSeatNo()-1] = new Double(0.0);
    double a = 0.0;

GUIPane.frame.getComponent(p.getCurrentPos()).setBackground(Color.light
Gray);

GUIPane.frame.getComponent(p.getCurrentPos()+step).setBackground(Color.
red);
    p.setCurrentPos(step);
    a = passingRow.sampleDouble();
    p.setBoardingTime(a);
    boardingTimes[p.getSeatNo()-1] = new Double(a);
  }
```

```
}
```

Controller.java

Created with JBuilder

simulationGUI.java (See Initialisation Pane on Figure 10)

Created with JBuilder

```
package boardingsimulation;

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2008
 * Company:
 * @author
 * @version 1.0
 */

public class simulationGUI extends Frame implements ActionListener,
  WindowListener, ItemListener {

  Frame frame;
  Panel p1, p2, p3, p4, p5, userInputPanel, choicePanel, runPanel,
infoPanel;
  Panel resultsPanel, strategyPanel, luggagePanel, groupPanel;
  Panel[] rows;
  Label userPrompt, infoBox, resultsBox, groupLabel;
  TextArea infoArea, resultsArea;
  Choice aircraftConfig;
  Button runSimulation;
  Checkbox ffa, assigned, normalLuggage, highLuggage;
  CheckboxGroup luggage = new CheckboxGroup();
  CheckboxGroup strategy = new CheckboxGroup();
  Scrollbar groups;
  GridBagConstraints gc = new GridBagConstraints();
  simulationGUIPane GUIPane;
  Controller control;


  public simulationGUI(Controller c) {

    control = c;
```

```
    frame = new Frame("Boarding Simulation");
    frame.setResizable(false);
    frame.setLayout(new GridLayout(0,1));
    frame.addWindowListener(this);
    p1 = new Panel();
    p2 = new Panel();
    //frame.add(p1);
    frame.add(p2);
    this.buildInfoPanel();
    frame.pack();
    frame.setVisible(true);
  }

  /*this will initialise the drop down for aircraft configuration, the
info box
  and the results box*/
  public void buildInfoPanel() {
    p1.setLayout(new BorderLayout());
    p2.setLayout(new BorderLayout());
    gc.insets = new Insets(2,2,2,2);
    p3 = new Panel();
    p2.add(BorderLayout.NORTH, p3);
    p3.setLayout(new GridBagLayout());
    p3.setBackground(Color.black);
    userInputPanel = new Panel();
    userInputPanel.setBackground(Color.white);
    p3.add(userInputPanel, gc);
    userInputPanel.setLayout(new GridLayout(6,1));
    userPrompt = new Label("Select aircraft configuration:");
    userPrompt.setFont(new Font("Arial", Font.BOLD, 12));
    userInputPanel.add(userPrompt);
    choicePanel = new Panel();
    choicePanel.setLayout(new FlowLayout(FlowLayout.RIGHT));
    userInputPanel.add(choicePanel);
    aircraftConfig = new Choice();
    this.aircraftTypes();
    choicePanel.add(aircraftConfig);
    aircraftConfig.addItemListener(this);
    strategyPanel = new Panel();
    strategyPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
    userInputPanel.add(strategyPanel);
    Checkbox ffa = new Checkbox("FFA",strategy,true);
    Checkbox assigned = new Checkbox ("Assigned",strategy,false);
    strategyPanel.add(ffa);
    strategyPanel.add(assigned);
    luggagePanel = new Panel();
    luggagePanel.setLayout(new FlowLayout(FlowLayout.LEFT));
    userInputPanel.add(luggagePanel);
    Checkbox normalLuggage = new Checkbox("Normal",luggage, true);
    Checkbox highLuggage = new Checkbox("High", luggage, false);
    luggagePanel.add(normalLuggage);
    luggagePanel.add(highLuggage);
    groupPanel = new Panel();
    groupPanel.setLayout(new GridLayout(2,1));
    userInputPanel.add(groupPanel);
    groups = new Scrollbar(Scrollbar.HORIZONTAL,50,10,0,100);
```

```
    groupLabel = new Label("Group Probability: ");
    groupPanel.add(groupLabel);
    groupPanel.add(groups);
    runPanel = new Panel();
    runPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));
    userInputPanel.add(runPanel);
    runSimulation = new Button("Run Simulation");
    runPanel.add(runSimulation);
    runSimulation.addActionListener(this);
    p4 = new Panel();
    p2.add(p4);
    p4.setLayout(new GridBagLayout());
    p4.setBackground(Color.black);
    infoPanel = new Panel();
    infoPanel.setLayout(new BorderLayout());
    infoPanel.setBackground(Color.white);
    p4.add(infoPanel, gc);
    infoBox = new Label("Simulation Information:       ");
    infoBox.setFont(new Font("Arial", Font.BOLD, 12));
    infoPanel.add(BorderLayout.NORTH, infoBox);
    infoArea = new TextArea(" ", 4, 22, 3);
    infoPanel.add(BorderLayout.SOUTH, infoArea);
    p5 = new Panel();
    p2.add(BorderLayout.SOUTH, p5);
    p5.setLayout(new GridBagLayout());
    p5.setBackground(Color.black);
    resultsPanel = new Panel();
    resultsPanel.setLayout(new BorderLayout());
    resultsPanel.setBackground(Color.white);
    p5.add(resultsPanel, gc);
    resultsBox = new Label("Simulation Results:       ");
    resultsBox.setFont(new Font("Arial", Font.BOLD, 12));
    resultsPanel.add(BorderLayout.NORTH, resultsBox);
    resultsArea = new TextArea(" ", 10, 22, 3);
    resultsPanel.add(BorderLayout.SOUTH, resultsArea);
  }

  //this will be dependent on the user selecting the aircraft
configuration
  public void buildSimulationPanel(String config) {
    int rowNo =0, aisles = 0, paxPerRow = 0;
    if(config.equals("Boeing 747"))
      rowNo = 19;
      aisles = 2;
      paxPerRow = 10;
      this.assignRows(rowNo);
      for(int i = 0; i<rowNo; i++)
      p1.add(rows[i]);
  }

  public void assignRows(int r) {
    rows = new Panel[r];
    for(int i = 0; i<r; i++)
      rows[i] = new Panel(new FlowLayout(FlowLayout.LEFT));
  }
```

```
  //aircraft types for inclusion in user choice list
  public void aircraftTypes() {
    aircraftConfig.addItem("Boeing 747");
    aircraftConfig.addItem("Boeing 777");
    aircraftConfig.addItem("Boeing 767");
    aircraftConfig.addItem("Boeing 757");
    aircraftConfig.addItem("Boeing 737");
    aircraftConfig.addItem("Airbus A320");
  }

  public void itemStateChanged(ItemEvent item) {
    String aircraft = item.getItem().toString();
    this.updateInfoArea(aircraft);
    //this.buildSimulationPanel(aircraft);
  }

  public void updateInfoArea(String details) {
    infoArea.setText("");
    if(details.equals("Boeing 747"))
      infoArea.setText("Long-haul, 2 aisles.\n10 passengers per row, 19
rows.");
    if(details.equals("Boeing 777"))
      infoArea.setText("Long-haul, 2 aisles.\n9 passengers per row, 15
rows.");
    if(details.equals("Boeing 767"))
      infoArea.setText("Long-haul, 2 aisles.\n7 passengers per row, 22
rows.");
    if(details.equals("Boeing 757"))
      infoArea.setText("Short-haul or Long-haul, 1 aisle.\n6 passengers
per row, 23 rows.");
    if(details.equals("Boeing 737"))
      infoArea.setText("Short-haul, 1 aisle.\n6 passengers per row, 11
rows");
    if(details.equals("Airbus A320"))
      infoArea.setText("Short-haul, 1 aisle.\n6 passengers per row, 12
rows");
  }

  public void actionPerformed(ActionEvent a) {
    if(a.getActionCommand().equals("Run Simulation"))
      if(aircraftConfig.getSelectedItem().equals("Boeing 747"))
        GUIPane = new simulationGUIPane(19, 2, 10, 4, 9, this);
      if(aircraftConfig.getSelectedItem().equals("Boeing 777"))
        GUIPane = new simulationGUIPane(15, 2, 9, 4, 8, this);
      if(aircraftConfig.getSelectedItem().equals("Boeing 767"))
        GUIPane = new simulationGUIPane(22, 2, 7, 3, 7, this);
      if(aircraftConfig.getSelectedItem().equals("Boeing 757"))
        GUIPane = new simulationGUIPane(23, 1, 6, 4, 0, this);
      if(aircraftConfig.getSelectedItem().equals("Boeing 737"))
        GUIPane = new simulationGUIPane(11, 1, 6, 4, 0, this);
      if(aircraftConfig.getSelectedItem().equals("Airbus A320"))
        GUIPane = new simulationGUIPane(12, 1, 6, 4, 0, this);
  }

  public void windowClosing(WindowEvent closing) {
    frame.dispose();
```

```
    System.exit(0);
  }

  public void windowActivated(WindowEvent e) {
  }

  public void windowClosed(WindowEvent e) {
  }

  public void windowDeactivated(WindowEvent e) {
  }

  public void windowDeiconified(WindowEvent e) {
  }

  public void windowIconified(WindowEvent e) {
  }

  public void windowOpened(WindowEvent e) {
  }

  public String getLuggage(){
    return this.luggage.toString();
  }

  public int getGroupProb(){
    return groups.getValue();
  }

  public String getStrategy(){
    return this.strategy.toString();
  }

}
```

simulationGUI.java

Created with JBuilder

simulationGUIPane.java (See Visualisation Pane on Figure 10)

Created with JBuilder

```
package boardingsimulation;

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.Component.*;
```

```
import java.awt.color.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2008
 * Company:
 * @author
 * @version 1.0
 */

public class simulationGUIPane extends Frame {

  Frame frame;
  int rows, aisles, paxPerRow, aislePos1, aislePos2;
  //Panel[] rowSet;
  Panel[] structure;
  simulationGUI GUI;

  public simulationGUIPane(int r, int a, int p, int a1, int a2,
simulationGUI sg) {
    rows = r;
    aisles = a;
    paxPerRow = p;
    GUI = sg;
    aislePos1 = a1;
    aislePos2 = a2;
    frame = new Frame("Aircraft");
    frame.setLayout(new GridLayout(rows,(aisles + paxPerRow)));
    //this.assignRows();
    //for(int i = 0; i<rows; i++)
      //frame.add(rowSet[i]);
    //this.assignSeats();
    this.assignStructure();
    for(int i = 0; i<(rows * (aisles + paxPerRow)); i++)
      frame.add(structure[i]);
    frame.pack();
    frame.setVisible(true);
    sg.control.buildSeats(rows, aisles, paxPerRow, aislePos1,
aislePos2, this);
  }

  public Component getSeat(int i){
    return frame.getComponent(i);
  }

  public void assignStructure(){
    structure = new Panel[rows * (aisles + paxPerRow)];
    for(int i = 0; i<(rows * (aisles + paxPerRow)); i++)
      structure[i] = new Panel();
  }

  public void assignRows(){
    //rowSet = new Panel[rows];
    //for(int i = 0; i<rows; i++)
      //rowSet[i] = new Panel(new FlowLayout(FlowLayout.LEFT));
```

```
  }

  public void assignSeats(){

  }
}
```

simulationGUIPane.java

Created with JBuilder

Passenger.java (See Passenger in Figure 10)

Created with JBuilder

```
package boardingsimulation;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2008
 * Company:
 * @author
 * @version 1.0
 */

public class Passenger {

  int seatNo;
  int luggage;
  double boardingTime = 0;
  int currentPos = -1;
  boolean seated;
  boolean inGroup;
  int targetPos;
  int sittingDown = 0;
  boolean atEnd;
  int seatPos;
  boolean waiting = false;
  int entryPoint;
  Passenger target;

  public Passenger(int no) {
    seatNo = no;
  }

  public int getSeatNo(){
    return seatNo;
  }
```

```
public int getLuggage(){
  return luggage;
}

public void setLuggage(int pieces){
  luggage = pieces;
}

public void setBoardingTime(double time){
  boardingTime = boardingTime + time;
}

public double getBoardingTime(){
  return boardingTime;
}

public void setCurrentPos(int pos){
  currentPos = currentPos + pos;
}

public int getCurrentPos(){
  return currentPos;
}

public void setSeated(boolean finished){
  seated = finished;
}

public boolean getSeated(){
  return seated;
}

public void setInGroup(boolean group){
  inGroup = group;
}

public boolean getInGroup(){
  return inGroup;
}

public void setTargetPos(int i){
  targetPos = i;
}

public int getTargetPos(){
  return targetPos;
}

public void setSittingDown(int i){
  //0 = processing as normal
  //1 = luggage stowed
  //2 = other stuff
  sittingDown = i;
}
```

```
  public int getSittingDown(){
    return sittingDown;
  }

  public void setAtEnd(boolean b){
    atEnd = b;
  }

  public boolean getAtEnd(){
    return atEnd;
  }

  public void setSeatPos(int i){
    seatPos = i;
  }

  public int getSeatPos(){
    return seatPos;
  }

  public void setWaiting(boolean w){
    waiting = w;
  }

  public boolean getWaiting(){
    return waiting;
  }

  public void setEntryPoint(int i){
    entryPoint = i;
  }

  public int getEntryPoint(){
    return entryPoint;
  }

  public void setTarget(Passenger p){
    target = p;
  }

  public Passenger getTarget(){
    return target;
  }

}
```

Passenger.java

Created with JBuilder

Seat.java (See Seat in Figure 10)

Created with JBuilder

```
package boardingsimulation;

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;

/**
 * Title:
 * Description:
 * Copyright:    Copyright (c) 2008
 * Company:
 * @author
 * @version 1.0
 */

public class Seat {

  int seatNo;
  boolean isAisle = false;
  int luggage;

  public Seat(int no) {
    seatNo = no;
  }

  public boolean getIsAisle(){
    return isAisle;
  }

  public int getSeatNo(){
    return seatNo;
  }

  public void setLuggage(int l){
    luggage = l;
  }

  public int getLuggage(){
    return luggage;
  }
}
```

Seat.java

Created with JBuilder

Triangular.java (See SimKit.Triangular on Figure 10)

Thanks go to the TeleSim Project, University of Calgary for their generosity in making their SimKit modules available to all (TeleSim Project , *TeleSim - University of Calgary* [Homepage of University of Calgary], [Online]. Available: http://warp.cpsc.ucalgary.ca/ [2008, 08/10])

```
package boardingsimulation;

import java.lang.Math;

/**********************************************************
 * Triangular is used to generate random variables from
 * the triangular distribution.
 *


 * This is a continuous distribution.
 *


 * Tests Performed

 * 1000 samples were generated and the means and variances
 * were examined.  Subjectively, they seemed correct.
 * A goodness of fit test was performed with 100 samples
 * and 10 intervals.  It succeeded about 19/20 times.
 *

 * @version 1.96
 * @author Juraj Pivovarov
 **********************************************************/

public class Triangular extends RandomNumber {
  double a, b, c;  // left, center, right

    /**********************************************************
     * Triangular constructor.  The left, right and center points
     * along the x-axis characterize the triangular distribution.
     *
     * @param left The leftmost endpoint
     *
     * @param center The center point of the triangle (x-axis pos
     * not height.
     *
     * @param right The rightmost endpoint
     **********************************************************/

    public Triangular(double left, double center, double right) {
        a = left;
        b = center;
        c = right;

        if( !(left<=center && center <=right) )
```

```
            throw new RuntimeException();
    }

    // Random number sampling functions

    /************************************************************
     * Generate a random double from the triangular distribution.
     *
     * @return A double from the triangular distribution.  The
     * output is gauranteed to be between the left and right
     * endpoints.
     *
     * @see Triangular#Triangular
     ************************************************************/

    public double sampleDouble() {
        double x = sample01();
        if ( x < (b-a)/(c-a))
          x = Math.sqrt(x*(b-a)*(c-a)) + a;
        else
          x = c- Math.sqrt((1-x)*(c-a)*(c-b));
        return x;
    }

    /************************************************************
     * The sampleInt function should not be called for
     * this continuous distribution.
     ************************************************************/

    public int sampleInt() {
        throw new RuntimeException();
    }
}
```

Triangular.java

Created with JBuilder