Agent-based Approaches To Pedestrian Modelling

Nicole Amy Ronald

Submitted in partial fulfillment of the requirements of the degree
Master of Engineering Science
in the Department of Computer Science and Software Engineering
of The University of Melbourne
February 2007

Produced on acid-free paper
Abstract

This thesis investigates the early stages of the software development process for agent-based models of pedestrian behaviour. Planning for pedestrians is becoming more important as planners and engineers become more aware of the sustainability and environmental aspects of transport and infrastructure. It is also necessary for the planning and management of pedestrian areas and events. Pedestrian behaviour is more difficult to model than other transport modes as it is not as constrained and operates at a finer scale.

Many approaches have been developed for modelling pedestrian behaviour. The simplest involve a single mathematical equation taking into account area and attractiveness of an area to calculate the maximum capacity. More complicated mathematical models involving differential equations have also been used.

Agent-based modelling is a recent development in modelling and simulation. These simulations contain agents who interact with each other and the environment in which they are situated. Their similarity to human societies has led to their use for many social applications. Many modellers are unsure of what agents are and how to develop models using them. In some cases, agents may be useful. In other cases, the model outputs and realism may not offset the learning curve, development time, and increased complexity of an agent-based model.

The thesis uses the overall framework of three nested processes – the planning process, the modelling process, and the software development process – to explore pedestrian modelling.

A framework to assist in selecting an appropriate approach for modelling pedestrian behaviour is developed, using the domain and requirements developed by the client in the planning process as inputs. Given the range of approaches and research- or industry-based models now available, this framework is necessary to increase confidence in the pedestrian modelling undertaken for the planning process. This framework forms the first contribution of this thesis and assists in understanding who is involved in the modelling process and the different approaches involved.
The development of two prototypes of agent-based pedestrian behaviour is described. The first prototype uses the popular agent-based cellular automata (CA) approach to model movement in a corridor. The second prototype uses intelligent agents using the belief-desire-intention (BDI) architecture. The two prototypes are the second contribution of this thesis and provide an insight into the software development process required for agent-based models and the applicability of these models to pedestrian behaviour.

A set of evaluation criteria is developed, based on software engineering quality principles, to evaluate the two prototypes. The CA prototype is particularly suited to modelling movement because of the explicit spatial representation. However, the lack of scalability causes problems for larger simulations and high-level behaviour, such as planning, requires additional non-CA modules. The BDI prototype was more difficult to design and implement, as the spatial location of the agents must be defined by the user. It is useful for modelling human behaviour, but lack of knowledge about how pedestrians move at a low level means that it is limited to high-level movement and planning.
Declaration

I certify that

1. other than as specified in the preface, this thesis comprises my original work;

2. due acknowledgment has been made in the text to all other material used;

3. the thesis consists of approximately 30,000 words, exclusive of tables, maps, bibliographies and footnotes.

Nicole Ronald
I am grateful to the Transport group in the Melbourne office of Sinclair Knight Merz for allowing me to undertake this degree and work more flexible hours at the beginning of this project. The involvement in the pedestrian modelling for the Melbourne 2006 Commonwealth Games was a notable inspiration for this project – thanks to James Larratt and Craig McPherson (amongst others) for winning the tender.

I am also grateful to both my supervisors. Leon Sterling was responsible for setting me up with some teaching experience and making suggestions that only made sense three months later. Michael Kirley and his never-ending questions were appreciated despite the grumpy looks.

My current employer, the Department of Computer Science and Software Engineering at the University of Melbourne, provided me with a lovely new staff machine (solitaire), which has coped with model runs much better than my old postgrad machine and also my laptop (oddjob). My vintage ’99 desktop (octopussy) also helped with typing papers and this thesis.

My proofreader and sounding board Julien Fischer was also a great help, despite his many complaints about agents. My mother Jill Ronald pointed out that I should use people’s full names in the acknowledgements. I thank both of them for their assistance.
Contents

Preface xvii

1 Introduction 1
  1.1 Aim and scope ......................................................... 3
  1.2 Overview .............................................................. 4

2 The complexity of pedestrianism 5
  2.1 Transport planning and modelling ...................................... 6
  2.1.1 The planning process ............................................... 6
  2.1.2 Classical transport modelling ..................................... 7
  2.1.3 Data collection for modelling .................................... 9
  2.2 Complex systems ...................................................... 9
  2.3 Pedestrian behaviour .................................................. 10
  2.3.1 Planning for pedestrians ......................................... 11
  2.3.2 Crowding ........................................................... 12
  2.3.3 Urban behaviour .................................................. 13
  2.3.4 Cognitive maps ..................................................... 14
  2.3.5 Wayfinding ........................................................ 15
  2.4 Summary ............................................................... 16

3 Pedestrian modelling approaches 19
  3.1 Environment representations .......................................... 19
  3.2 Pedestrian modelling .................................................. 21
  3.3 Microsimulation ........................................................ 22
  3.4 Physical and mathematical approaches ................................ 23
3.5 Agent-based approaches .................................................. 25
  3.5.1 Agent-based simulation .............................................. 27
  3.5.2 Geosimulation .......................................................... 28
  3.5.3 Agent-based cellular automata approaches ....................... 28
  3.5.4 Agent-based approaches ............................................ 32
  3.5.5 Hybrid models ...................................................... 32
3.6 Summary .................................................................... 33

4 Approach framework ......................................................... 35
  4.1 Methodology ............................................................... 36
  4.2 Stakeholders .............................................................. 36
    4.2.1 Roles ................................................................ 38
  4.3 Framework elements .................................................... 40
    4.3.1 Categorisation of domains ...................................... 40
    4.3.2 Categorisation of behaviours ................................... 43
    4.3.3 Categorisation of approaches ................................... 44
    4.3.4 Framework .......................................................... 45
  4.4 Summary .................................................................... 47

5 Model prototypes ............................................................. 49
  5.1 The software development process .................................... 50
  5.2 Agent-based cellular automata ......................................... 51
    5.2.1 Methodology .......................................................... 52
    5.2.2 Model development ................................................ 53
    5.2.3 Scenarios ............................................................. 57
    5.2.4 Experimental results and discussion ......................... 59
  5.3 BDI agents ................................................................. 66
    5.3.1 Methodology .......................................................... 67
    5.3.2 Model development ................................................ 71
    5.3.3 Operational assessment ........................................... 78

6 Model evaluation and discussion ........................................ 81
  6.1 Evaluation criteria ....................................................... 81
    6.1.1 Functionality .......................................................... 82
CONTENTS

6.1.2 Usability ................................................... 82
6.1.3 Efficiency .................................................. 83
6.1.4 Maintainability ............................................. 83
6.1.5 Portability .................................................. 83
6.1.6 Discussion of criteria ...................................... 83

6.2 Functionality ................................................ 84
6.2.1 CA model ................................................... 84
6.2.2 BDI model ................................................... 85

6.3 Usability ....................................................... 86
6.3.1 CA model ................................................... 86
6.3.2 BDI model ................................................... 86

6.4 Efficiency ...................................................... 87
6.4.1 CA model ................................................... 87
6.4.2 BDI model ................................................... 87

6.5 Maintainability ............................................... 89

6.6 Portability ...................................................... 90

6.7 Other simulation languages ................................. 90
6.7.1 Repast ....................................................... 90
6.7.2 Swarm ....................................................... 91
6.7.3 Comparison to JACK Sim ................................ 91

6.8 Usefulness ...................................................... 92

7 Conclusions and further work .............................. 95
7.1 Further work .................................................. 96
7.1.1 Framework .................................................. 97
7.1.2 Agent architectures ....................................... 97
7.1.3 Model development ....................................... 97

References ......................................................... 99
List of Tables

2.1 A summary of pedestrian trip variables, from Fruin [34].................. 11

4.1 A summary of framework criteria for evaluating agent-based approaches to logistics
management, from Davidsson et al. [28]........................................ 37

4.2 A summary of the recommended approaches for a particular domain........ 48
List of Figures

1.1 The relationship between the planning, modelling, and software development processes, and the relevant chapters in this thesis. ................. 2

2.1 The planning process as described by Ortúzar and Willumsen [72] ............ 7

3.1 The modelling process as described by Law and Kelton [55]. .................. 21
3.2 Snapshot of Paxport’s visual output of a model of the Melbourne Sports and Entertainment Precinct. The block colour or shading represents the current level of service in the block at the current time. ................................. 24
3.3 Some of the mathematical equations used in the social force models of Helbing et al. [43]. .......................................................... 25
3.4 Two instantiations of the number 184 (10111000) cellular automata, 200 cells wide and for 200 timesteps (with time increasing from top to bottom). The first environment contains 88 objects that organise into free-flowing traffic quickly, as shown by the lack of black bands. The second environment contains 103 objects and some traffic jams continue to the end. Some jams in the upper left hand corner dissipate. 30

4.1 An example of a small-scale enclosed space: a corridor in a building. ........ 41
4.2 An example of an open-plan enclosed space: an airport. ......................... 41
4.3 An example of a mixed mode space: a pedestrian shopping mall. ............ 42
4.4 An example of an open space: a beach. ............................................ 42

5.1 The waterfall model, showing the main phases in the software development process. 50
5.2 A UML overview of the classes in the CA model. .............................. 55
5.3 Some of the distributions available in the CA model: Uniform, Normal, and Rise and Fall. ......................................................... 56
5.4 The probabilities of moving to the neighbouring cells for the random walk behaviour. 57
5.5 The order of trying to move to the neighbouring cells for the deterministic behaviour. 57
5.6 The probabilities of moving to the neighbouring cells for some configurations of the lookahead behaviour. 58
5.7 An example of a static floor field for an unblocked and a blocked environment for the upwards-moving pedestrians. In the blocked environment, the gradient breaks from the regular line pattern and curves around the blockage, leading people to walk around it. 60
5.8 The possible outcomes of a collision and their probabilities. 60
5.9 Time and distance travelled divided by the length $L$ of the environment. 61
5.10 The average number of times a pedestrian stops due to a collision. 62
5.11 Modelled crowds at the same time ($t=250$) for different behaviour and densities. The red pedestrians are moving upwards, the green downwards. From left: random-30%, floor fields-30%, deterministic-30%, lookahead-30%, lookahead-10%, lookahead-50%, lookahead-70%. 63
5.12 Entropy plots for different behaviours with density 30%. 64
5.13 Entropy plots for the lookahead behaviour at different densities. 65
5.14 Proximity plots for the different behaviours. 66
5.15 A goal diagram in Prometheus. 72
5.16 A coupling diagram in Prometheus. 72
5.17 An agent diagram in Prometheus. 73
5.18 A data diagram in Prometheus. 74
5.19 A flowchart of a JACK Sim model. System events are shown as rounded rectangles. 76
5.20 The relationship between the space available to a pedestrian with average walking speed and their speed. 77
5.21 A example of the environmental output from the BDI prototype. Each path is represented as a link between two numbered nodes. Square nodes are entries/ exits. Thicker lines and different colours mean that the level of service (density) is lower (higher) on those links. 79
6.1 The execution times and linear fits for different environment sizes. 88
6.2 The execution times and linear fits for different densities. 88
6.3 The different execution times and the fits for each of the BDI model stages. 89
Preface

This thesis is all my own work.

Several papers were published as a result of this research. Chapter 4 was published as Ronald, Kirley, and Sterling [84] and has undergone minor edits for this thesis. Parts of Chapter 5 were published as Ronald and Sterling [82, 83] (BDI model) and Ronald and Kirley [81] (CA model). Ronald [80] was primarily about my consulting work for Sinclair Knight Merz, however included a small discussion on topics covered in this thesis. Figure 3.2 was first published in that paper. Ronald, Sterling, and Kirley [85] included the reimplementation in JACK Sim and the evaluation from Chapter 6 and was published after the thesis was initially submitted for examination.

The snapshots in Figure 3.4 were taken from the Monash University’s Artificial Life Virtual Lab (VLAB) project (http://www.complexity.org.au/vlab/). The CA Asynchronous applet was used with rule 184 and synchronous updates.

Figure 4.3 was sourced from http://commons.wikimedia.org/wiki/Image:Wfm_john_lewis_glasgow.jpg and was taken by Finlay McWalter.

The code for both the CA and BDI models is available on request from the author. The user will require a current version of JACK Intelligent Agents (available from http://www.agentoriented.com/) to use the BDI model.
Chapter 1

Introduction

Understanding and forecasting the behaviour of pedestrians is necessary for the planning and management of pedestrian areas and events. Event planners require estimates of patrons and their movements so that plans can be developed to manage expected queues and crowding. Engineers of pedestrian infrastructure require forecasts of the number of people using the new infrastructure so that they can determine whether it is worth building and if so, what the dimensions should be.

Planning for pedestrians is becoming more important as planners and engineers become more aware of the sustainability and environmental aspects of transport and infrastructure. In the past, walking was associated with low living standards, however now there is a recognition of the importance of walking for the environmental benefits as well as personal health benefits. The recently-published *Melbourne 2030* planning policy includes objectives such as providing sustainable, liveable, equitable and efficient environments [98].

Computer simulation is commonly used in the planning process to investigate a wide range of scenarios without altering the “real world”. The effect of the scenario over longer time periods, such as years and decades, can also be explored [55]. However simulation can be time-consuming and provides only an approximation of the real world.

In general, creating simulations requires two processes to be followed. The *modelling process* involves gathering the requirements of the model, developing the model, followed by validation. The *software development process* involves gathering of non-model and system requirements from the client, developing the software following good practice, and testing the software before presenting the final product to the client. As most model developers are experts in the field they are modelling and are not software engineers, the first process often takes precedence over the second.
Figure 1.1 shows the relationship between the planning, modelling, and software development processes.

Creating computational models of pedestrian behaviour is difficult [14]. Walking behaviour is largely unconscious and unpredictable. Sometimes we don’t know why we took a particular path. It is also difficult to collect accurate data.

Many approaches have been developed for modelling pedestrian behaviour. The simplest involve a single mathematical equation taking into account area and attractiveness of an area to calculate the maximum capacity [41]. More complicated mathematical models involving differential equations have also been used (see Helbing and Mohar [42] and Teknomo [95] as examples). It has been recognised that there are two types of pedestrian models: the origin-destination models that are similar to traditional transport models of vehicles, and crowding models focusing on evacuation and panic behaviour [13]. Both types of model are distinct but also influence each other.

Agent-based modelling is a recent development in modelling and simulation. These simulations contain agents who interact with each other and the environment in which they are situated.
Their similarity to human societies has led to their use for many social applications, in particular transport applications, as well as stock markets and organisations [18]. Fields such as civil engineering, geography, and mathematics have all employed agent-based simulation for pedestrian behaviour, for both understanding elements of pedestrian behaviour and forecasting movement in an environment.

However, many modellers are unsure of what agents are and how to develop models using them. In some cases, agents may be useful. In other cases, the model outputs and realism may not offset the learning curve, development time, and increased complexity of an agent-based model.

1.1 Aim and scope

The aim of this thesis is to investigate the early stages of the software development process for agent-based models of pedestrian behaviour. In particular, the usefulness and development of agent-based approaches for models of pedestrian behaviour will be explored. Usefulness is defined as “the quality of having utility and especially practical worth or applicability”¹.

The project is limited in scope mainly by resources. No data were available and no resources were available to collect data for calibration and validation purposes. Therefore this project explores the early stages of modelling from requirements gathering to design and implementation. The project is not using pedestrian modelling as a domain for demonstrating computational advances.

The contributions of this project are intended to be accessible to modellers, who are usually planners, mathematicians, or engineers and do not necessarily have a computer science or software engineering background. Modellers are responsible for carrying out the modelling process and sometimes, despite in some cases not having a large amount of experience in software engineering, the software development process shown in Figure 1.1.

The disciplines that influence pedestrian modelling are many. Therefore this project has taken a broad view of the area as a necessity for understanding the different influences, however I am mostly influenced by my background in civil engineering and experience in industry as a transport modeller. Using Batty’s division between origin-destination and crowding models, this thesis is focused on origin-destination models, however crowding also has a role in these models.

¹Merriam-Webster Online Dictionary (http://www.m-w.com).
1.2 Overview

Chapter 2 presents background material on transport modelling and planning and psychological aspects of pedestrian behaviour. This highlights the wide range of opinions and studies on pedestrian behaviour and also discusses the decision-making and planning process that precedes the need for transport models.

Chapter 3 introduces the modelling process in general. Different approaches to modelling pedestrian behaviour are reviewed and agents and agent-based modelling are discussed.

Using the approaches presented in Chapter 3, a framework to assist in selecting an appropriate approach for modelling pedestrian behaviour is developed in Chapter 4. This framework forms the first contribution of this thesis and assists the stakeholders in understanding who is involved in the modelling process and the different approaches involved. The framework uses the type of environment and requirements from the client as inputs to recommend an approach. Given the range of approaches and research- or industry-based models now available, this framework is necessary for stakeholders to increase confidence in the pedestrian modelling undertaken for the planning process.

Following the selection of an appropriate approach for a particular model, the model must be created. The software development of two prototypes of pedestrian models is described in Chapter 5, using the two agent-based approaches from Chapter 4. The first prototype uses the popular agent-based cellular automata approach to model movement in a corridor. The second prototype uses intelligent agents using the belief-desire-intention (BDI) architecture. The JACK Intelligent Agents platform was used to implement the BDI agents, along with the Prometheus agent-based design methodology. The software development process involved in each prototype is also discussed. The two prototypes provide an insight into the software development process required for agent-based models and the applicability of these models to pedestrian behaviour, which is the second contribution of this thesis.

In Chapter 6, a set of evaluation criteria is developed, based on software engineering quality principles. The criteria is used to evaluate the two prototypes against the objectives of exploring the applicability and development of agent-based models for pedestrian behaviour. As little research has been undertaken into evaluating tools that can be used for a specific activity, such as creating pedestrian models, the evaluation criteria were developed using software quality standards and sets of desirable features for models from the agent-based modelling literature.

Finally, a discussion of extensions and further work concludes the thesis.
Chapter 2

The complexity of pedestrianism

Transport systems are a necessary part of urban infrastructure. Moving from one location to another is required for work, education, shopping, leisure, and travel. Demand for transport services is referred to as a derived demand, meaning that it is a by-product of demand for other services like those listed above.

Pedestrianism is the most common transport mode and is accessible by almost everyone [6]. All transport trips contain some walking, usually at the beginning or end of the trip [7]. In some locations, such as buildings, walking is the only mode of transport available.

There is current interest in improving and developing pedestrian environments, in order to improve our personal health and the environmental quality and liveliness of towns and cities [98]. These projects will involve planners and engineers analysing current situations and evaluating future scenarios, and the outputs of models of pedestrian behaviour will be required for the latter. However, before modelling begins, it is necessary to understand the role of the planner, techniques used for other transport models in the past, and the theoretical behaviour of pedestrians.

This chapter explores transport planning and modelling. The four-step model commonly used for vehicle traffic model and techniques for data collection are discussed. Complex systems are introduced, with reference to the complex nature of transport systems. Finally, a discussion of the history and current research into pedestrian behaviour is presented. Although not a comprehensive survey of all the influences on transport and pedestrian planning, a background for the remainder of the thesis is provided.
2.1 Transport planning and modelling

Preceding the need for movement is the provision of services. The design of a city has a strong influence on how people move and navigate around it and will also influence the type and amount of transport services required. A measure of city design is known as legibility: “the ease with which its parts can be recognized and can be organized into a coherent pattern” [61].

Lynch [61] describes the common elements found in cities as:

- paths: the main channels of movement;
- edges: boundaries and/or barriers;
- districts: large areas with a common character;
- nodes: junctions, crossings, and concentrated areas;
- landmarks: unique “points of reference” [34] that can be either distant or local, and also areas that cannot be entered.

This provides a basis for discussing the layout of environments.

The provision, operation, and management of transport facilities in cities is usually the responsibility of government planners and engineers. The main properties of the facilities are that they are “safe, rapid, comfortable, convenient, economical, and environmentally compatible” [71].

Transport systems are constrained, albeit sometimes weakly. For instance, people cannot cross the road whenever they feel like it. They should find a suitable place (such as an intersection) and wait until it is safe. They should also travel on the pedestrian network (e.g., designated paths) at all times, however if it becomes too congested, pedestrians may overflow onto the road or surrounding parkland. A stricter constraint is that pedestrians cannot walk through solid objects or on water.

2.1.1 The planning process

Transport planning is a decision-making process in which the problem is identified, solutions are developed, modelled and evaluated, and the preferred solution is recommended for implementation (Figure 2.1). Evaluating solutions requires examining the effects upon stakeholders and the environment for different scenarios and can be undertaken in many ways. Several solutions could be selected for a trial run, however physical tests are not always feasible. For example, it is impractical (not to mention expensive) to build several versions of a pedestrian bridge in order
to determine the option with the most benefit. In these cases, computers are used to set up an “artificial reality” – a computer model or simulation – which is used to test different strategies.

The inputs to transport models usually include demographic data (age, sex, place of residence, type of work), land use data (which assists in determining attractability of certain locations), and demand drivers (which locations are popular, what times are people travelling). The data are sourced from public data, such as census and land-use data, and data collected specifically for the model, such as the results of an observation or a questionnaire survey. The outputs of transport models can include economic, environmental and social data.

### 2.1.2 Classical transport modelling

The most well-known approach for modelling transport is the “four step” model. This model was developed in the 1950s for looking at travel behaviour at an aggregate level. The network usually
consists of zones, which can be based on defined zones (e.g., postcodes) or zones created by the modeller.

The first step of the model is trip generation, where the number of trip productions and attractions at the zonal level is determined. In the second step of the model, trip distribution, the matrix from the first step is used to create a trip matrix between zones. Each cell in the matrix \( T_{ij} \) is a function of the number of productions \( P_i \) and attractions \( A_j \) and network attributes such as travel times. This is commonly referred to as the gravity model, as it is related to the physics gravity model in which objects closer to each other are more attracted to each other. The third step is to assign transport modes to each trip. Finally, the route choice is calculated. There are many methods for calculating this and the process is usually iterative until some stability is reached. A common starting point to use All Or Nothing (AON) assignment, where all trips are assigned to the shortest path. This may cause some links to be overloaded, so some trips are then moved to other links.

A strength of the four step model is the logical steps [10]. However, many aspects of the model have been criticised. According to Banister [10], the process is too rigid and the model is more concerned with reducing travel time instead of other travel measurements. It was created to be simple, however this fails to recognise that transport is complex [20]. The model is based on trip-based methods, rather than activity-based methods [63], which contradicts what is commonly understood about travel behaviour in that is derived from activity. It also does not take into account individual choice [37].

McNally [63] also states that the four-step model is described as state-of-the art as it has been the only approach available. In the 1980s, disaggregate modelling started to become popular, with the advent of more advanced computing technology and more confidence in computational models [72].

From my experience, trip-based models are still being used in Australia. In the USA, where their transport problems are larger, they predominantly use activity-based models. However, as little software is available to support activity-based models, consultants are required to build their own software tools\(^1\).

\(^1\)From a personal conversation with Rick Donnelly, a principal with Parsons Brinckerhoff.
2.1.3 Data collection for modelling

In order to evaluate scenarios and calibrate and validate models, data about the current state of the environments are required. For vehicle models, common techniques for collecting objective data are road counters for volumes at certain points and number plate surveys to explore paths and timing. For subjective data, individual questionnaires are often employed, however these are difficult to oversee and process.

As pedestrians are more flexible in their movement, traditionally it has been difficult to collect accurate data. Fruin [34] describes the use of cordon counts and time-lapse photography to collect volume counts. Recent advances in location technology mean that it is now easier to collect detailed travel data.

Methods for collecting microscopic objective travel data of pedestrians include GPS tracking, timing systems, camera-base systems, mobile phone tracking, and PDA tracking [108]. Of these, all but the mobile phone tracking are quite expensive to collect. Mobile phone tracking, however, suffers from low resolution as it can only pinpoint the nearest signal tower. Questionnaires are still employed for subjective data.

2.2 Complex systems

Complex systems consist of many interconnected parts [12]. These parts combine together to form the behaviour of the whole, leading to the common short definition of “The whole is more than the sum of its parts”, originally attributed to Aristotle. The behaviour of the whole emerges from the individual behaviour of the parts.

Complex systems have three properties [17]:

1. a large number of interacting agents;
2. emergence; and
3. the emergence “does not result from the existence of a central controller”.

According to Boccara [17], emergence is the “single most distinguishing feature” of a complex system. He defines emergent properties as “large-scale effects of locally interacting agents that are often surprising and hard to predict even in the case of simple interaction”. However, the author admits that what may be surprising to one may not be surprising to another. Benenson and Torrens [14] believe local rules generate global behaviour in that “the actions of the parts do
not simply sum to the activity of the whole” and Bar-Yam [12] states that “the behaviour of the system cannot be simply inferred from the behaviour of its components”. The key word in all of these is *simple*, which is not concrete enough to be applied to a system objectively.

Measures of emergence and complexity have been developed. Entropy is a measurement used to determine the complexity of a complex system as ordered, complex, or chaotic [107].

Using Boccara’s properties, transport systems are clearly complex. They consist of a large number of agents (pedestrians or drivers for example), who interact with each other. The main interaction occurs when giving way or changing lanes, for example. In transport systems, there is also significant interaction with the environment. Aggregate behaviours emerge, such as lane formation in pedestrian traffic. Finally, the emergence is not the result of any central control as all agents in the model make their own decisions.

Another feature of complex systems is self-organisation. This is often confused for emergence as it often appears with emergence [104]. Self-organisation occurs when the system organises itself into a more-ordered state, with no central control. This feature also occurs in transport systems as the actors in the system are autonomous and make their own decisions about what to do next. Transport systems can increase their order as users become more familiar and also decrease order if a change occurs.

### 2.3 Pedestrian behaviour

Commuters scurry; shoppers meander; bushwalkers trek; power-walkers stride; lovers stroll; tourists promenade; protestors march ... But we all walk. [6].

Walking has played different roles in society throughout time. Our ancestors walked while hunting and travelling. Early societies created footpaths. Pilgrims walked to holy places. Eventually, as cities developed, paved paths for pedestrians were created [4].

Promenading and strolling became social activities, where one could display their clothes and posture and also observe others. Window shopping became popular in the 18th century. Scientists such as Goethe and Humboldt made discoveries about nature while walking around the countryside.

Eventually walkers began following rules, as kerbs and sidewalks were developed to ensure mode separation from vehicles. People would attend festivals, where police would keep control. In the 20th century, walking became the preferred form of transport for protests and marches [4].
2.3. PEDESTRIAN BEHAVIOUR

Table 2.1: A summary of pedestrian trip variables, from Fruin [34].

Pedestrian behaviour is usually individual-based and autonomous. In most cases, we decide where we want to go and how to get there without being told explicitly. In the system, there are cooperative elements (letting someone go through a door first, moving out of the way for a faster person) and competitive elements (pushing to get out of a stadium quickly).

Pedestrians often make unconscious decisions that are difficult to explain or measure. They move at a much smaller scale and in a less constrained manner than vehicles, meaning that techniques developed for modelling other modes of transport cannot easily be translated to pedestrians. Computational modelling of pedestrians is therefore difficult due to the complex and random nature of their movement. Table 2.1 shows some of the variables involved in a pedestrian trip.

Environmental psychology is the study of the “interrelationship between environments and human behavior” [109]. Four areas of relevance to pedestrians in environmental psychology are crowding, urban behaviour, cognitive maps and wayfinding.

2.3.1 Planning for pedestrians

Fruin [34] develops a set of level-of-service recommendations for pedestrian areas, based on existing highway engineering standards. There are six levels, ranging from free-flowing (level A) to shuffling (level F), primarily calculated using pedestrian flow rate and density. The objectives of planning for pedestrians are identified as safety, security, convenience, continuity, comfort, coherence, and
attractiveness. He also discusses the types of infrastructure design that lead to the satisfaction of these objectives, such as landscaping, adequate lighting, and grade separation of pedestrians and vehicles.

Alfonzo [3] develops a hierarchy of pedestrian needs, based on Maslow’s hierarchy of needs. The hierarchy consists of feasibility, accessibility, safety, comfort, and pleasurability and appears to move from more objective concepts (is walking possible?) to subjective concepts (is the route attractive?). She also notes that pedestrians “may or may not be able to articulate these needs as conscious considerations”. This hierarchy is a pedestrian view of motivations and is intended for use in policy-making. Tolley [98] also lists thirteen factors and settings for walking, both for use in policy-making and for infrastructure design.

### 2.3.2 Crowding

Crowding differs from the engineering or urban planning view of crowds in that psychologists are interested in the individual’s experience, rather than the formation, structure and movement of crowds [35]. This is a subjective approach rather than an objective approach. Some people may be comfortable in very crowded situations, but others may be uncomfortable at low densities. People are impacted on by density at small scales (rooms, communities) however this is influenced by the world’s population growth.

Crowding experts differentiate between indoor and outdoor density, social density (varying the amount of people in a fixed space) and spatial density (varying the amount of space available to a fixed number of people), and density (the number of people in a area divided by the area; this assumes the people are evenly spread) and proximity (assumes the people are grouped together). Studies into proximity led to the Population Density Index, which is the average distance between all pairs of people relative to their collective area. Another approach uses the gravity approach that is also used in the four step model. Knowles proposed a law of social interaction that states “the effects of others on an individual will increase with the square root of the number and decrease with the square root of their distance” [35, p176].

There are many influences on crowding, including physical setting, personal characteristics, and the sociocultural situation. These influences combine to produce emotional, stressful, and behavioural responses.

Personal influences are varied. Some people feel crowded as they feel they have “lost control” over their actions, and people who believe they have influence over the lives are more susceptible
2.3. PEDESTRIAN BEHAVIOUR

2.3.1 Physical influences

Physical influences include the scale (some people may feel crowded at home, but not in the town), architecture (high-rise vs. low-rise, long vs. short corridors, views and sunlight all have an effect), and temperature (crowds of the same density feel more crowded in high temperatures).

The consequences of crowding can include illness, walking faster, attempting to escape the situation, helplessness, and anti-social behaviour such as aggression or withdrawal. These reasons ensure that it is a common area of study for modelling experts, especially for evacuation scenarios.

2.3.3 Urban behaviour

Urban behaviour includes watching and walking, amongst other behaviours. Little is known about the relationship between walking and the environment. Gifford [35] describes studies that have produced some interesting qualitative and quantitative facts. A mathematical formula for average velocity with respect to town population was developed following studies in eight countries. Walking speeds vary though between different categories:

- groups walk slower than individuals;
- uphill walkers are slower than downhill;
- dry weather walkers are slower than those in wet weather;
- males walk faster than females.

We also make many decisions when we walk, such as destination, where to cross the road, whether to jaywalk, or whether to stop and look at a shop window. Most of these decisions are unconscious. Walkers generally choose the shortest route, but children and women often took more complex routes. Reasons for choosing particular routes were attractiveness or a building on the way.
Studies (described by Gifford [35]) indicate that walkers in urban areas generally use the same principle: “minimise involvement with others and maximise social order”. Noncontact and cooperation were evident in the “rules” walkers use. This also reinforces the opinion of some that transport systems are self-organising.

2.3.4 Cognitive maps

Cognitive maps are “conceptual manifestations of place-based experience and reasoning that allow one to determine where one is at any moment and what place-related objects occur in that vicinity” [37]. When we need to form a route from one place to another, our cognitive maps store our knowledge of that area and allow us to mentally form a complete or partial route. We can also attempt to explain a route to others, in words or with pictures. When moving through an area, we can store new information or if we have been there before, we can update our information [103]. The link between travel and cognitive maps is circular: travelling informs our cognitive maps and our cognitive maps inform our travel.

Routes consist of links and nodes, where the nodes are decision or reference points. The nodes can be common to everyone, such as the statue on the main street. They can also be personal, such as my old work building or the place my car broke down.

Using our cognitive maps is a four step process. The element in the environment must be consciously noticed. A characteristic (physical and/or personal) must then be assigned to it, before it is stored in our cognitive map. When required, it must be recalled.

Different types of knowledge are stored in a cognitive map [38]. Declarative or landmark knowledge contains the objects, events, and places that the person is aware of. The person can recognise the object when they see it and communicate with others about its location and appearance. Procedural knowledge is used to construct a path between two locations and reason about distance and also obstacles that are encountered. This knowledge can be hierarchical. For example, taxi drivers appear to have primary and secondary network structures in their cognitive maps [38].

The knowledge can be made up of personal experience in a area and also information communicated by other people and traditional maps. Weston and Handy [103] asked four people to draw a map of Austin, Texas, and the four maps all contained differing landmarks and paths. The personal influence means that cognitive maps are individual and therefore travel is individual.
2.3.5 Wayfinding

Wayfinding is the process of finding a path. This differs from navigation, which is following a predefined path [103]. It is difficult to define a generic wayfinding process as it depends on the purpose of the trip. Trips that are repeated frequently, such as to work or to school, are refined quickly. These trips usually follow the shortest path and minimise the amount of en-route decision making. Weston and Handy [103] refer to these as commuting trips: regular travel between two points. The wayfinding for leisure or recreation trips consists of search and exploration.

Another definition of wayfinding behaviour is knowing where you are, knowing your destination, knowing and following the best (or a serviceable) route to your destination, recognising your destination upon arrival, and then finding your way out again [26]. Alternative definitions look at the hierarchical series of decisions made by wayfinders and investigate the cognitive and perceptual processes. The literature is divided into perception and understanding of the environment and the processes people follow to move to their destination.

There are three interconnected elements in wayfinding systems: behavioural, design, and operational. Behavioural elements encompass the cognitive and perceptual activities undertaken by wayfinders and also their abilities and experiences. They require wayfinding skills including asking directions and reading signs and maps. Several design features contribute to wayfinding ease. These include intuitive layouts, differentiating between different buildings architecturally, landmarks such as artworks, signs (particularly at decision points), clear and consistent maps, and lighting of information and decision points.

Several strategies have been developed for locating destinations:

- seeing and moving towards the destination;
- following a marked path to the destination (e.g., coloured paths);
- obtaining information from signs and landmarks;
- using cognitive maps.

Bovy and Stern [19] present another view of route choice. They describe the route choice process as a black box, where the traveller and the network are inputs and the selected route is the output. The black box is a mental model with the traveller’s attitudes, emotions, perceptions, cognition and learning as variables.

The factors that influence travel behaviours and route choice are the physical, socio-demographic,
normative (e.g., norms and social values) and personal environments. The latter acts subjectively on the previous three to make a decision.

The Choice Situation is the set of all possible routes between an origin and destination, whereas the Choice Set is the set of routes that the traveller knows about and are suitable for his trip.

There are three basic methods of forming a route. *Simultaneous planning* involves selecting a complete path from the choice set of all complete paths from origin to destination. In this situation the traveller does not change his path during the trip. *Sequential* and *hierarchical* choice involve decision points during the trip. Sequential involves selecting the next link independently of the last link followed, but hierarchical takes into account the past. These are the extreme choice situations, however they are common in reality [19].

If a traveller experiences inferior conditions during the trip than what they expected, they may change their route. This is called *adaptive route choice* and is usually experienced when the traveller is under time pressure [19].

A traveller chooses a route based on many different attributes: fewest turns, avoiding congestion, maximising aesthetics [37]. Several studies of route choice factors in pedestrians have shown that the trip length is the most important attribute [19].

### 2.4 Summary

In this chapter, several different disciplines with an interest in or influence on pedestrian behaviour were discussed.

Town planners create the areas and locations that the pedestrians want to visit. Transport engineers then supply the transport networks that allow people to travel to these locations. Transport systems are influenced by dynamic and static views of the world: transport is particularly dynamic as people move around a system, however the buildings and the other services they require are static in terms of movement. Batty [13] argues that modelling at finer scales means modellers need to create more dynamic models.

Modellers assist the planners and the engineers by making predictions about the demand for locations. These models require objective data and produce objective data. The environmental psychologists' definitions and theories are based on subjective feeling. Pedestrians sometimes move unconsciously, which makes it difficult to collect detailed information. Due to the flexibility of pedestrian behaviour, it is difficult to objectively define how pedestrians move.

Wayfinding works on a hierarchy of knowledge, adding to our cognitive maps as we move...
around. However, wayfinding is very high-level in dealing with environment knowledge and route choice. Little is known about the cognitive processes involved in traversing a route, such as when people change lanes or accelerate. These behaviours are normally observed and collected on a project-specific basis. Emergence in complex systems also works on different levels: the individual influences the whole and vice versa. This is prevalent in transport systems, where small changes to the system can result in large change in aggregate behaviour.

In order to model pedestrian behaviour realistically, the themes of static vs. dynamic, objective vs. subjective, and micro vs. macro must be taken into account. The next chapter investigates current methods of modelling pedestrian behaviour.
Chapter 3

Pedestrian modelling approaches

During a discussion on the mathematical modeling of pedestrian trips, one analyst jokingly stated that the difficulty with the pedestrian is that at any instant he may sit down and become a non-pedestrian, or leap up suddenly and either walk or run toward any point of the compass. (Fruin [34, p132])

Models are frequently used in transport planning, primarily as a tool for forecasting future conditions of a system. The four-step model has been popular for several decades, however it has been criticised as being a too simple representation of a complex system. As a result, it is not at the forefront of current pedestrian modelling techniques. For pedestrian modelling, engineers, geographers, planners, and computer scientists have concentrated on an individual-based approach (rather than an aggregate approach) to modelling common pedestrian behaviours such as wayfinding and crowding.

This chapter begins with an overview of computational representations of the environment. The environment plays an important part in transport models, as it influences how people move by guiding and constraining them. Following on from the planning process described in the previous chapter, the modelling process is introduced. This is followed by discussions of different pedestrian models, in particular centralised models, mathematical models, and agent-based models.

3.1 Environment representations

Representations and abstractions of the environment are not only important for transport models, but also for robotics research, where physical robots need to move around an environment. Several
representations have been proposed for this latter purpose which are also used in pedestrian models. The requirements for an environment representation for robots is that it can be derived from sensor data, incremental, maintainable, economical, allows path planning, and place recognition.

The possible map types are:

- metric, which uses an absolute coordinate system;
- topological, which uses a graph representation;
- hybrid, which combines both.

Both metric and topological maps have their strengths, but the trend is towards hybrid approaches [101].

Metric maps can take two forms. *Cellular* maps usually have an uniform cellular decomposition, where the cells are either free or blocked. An alternative is to use occupation grids where the probability of a cell being occupied is calculated. Grids are easy to update, but are space-intensive and are inefficient for path planning. *Geometric* maps use geometric primitives, such as squares and rectangles, to represent different area. The primitives are overlaid on a continuously-valued world. The space efficiency depends on the approach and some pre-processing is required for path planning to limit the number of choices.

Topological maps are a graph representation of the environment. Nodes designate decision points and edges designate connections and paths. The most common form of topological map is a Voronoi diagram which corresponds to the possible paths through the environment. Topological maps have many advantages including no need for metric coordinates, efficient path planning (although paths may not be optimal), and space-efficiency. However, they are difficult to construct and they do not adapt well to changes in the environment.

Hybrid maps combine both metric and topological approaches. The topological map is derived from the metric map and is used for path planning, however the other disadvantages of metric maps (e.g., space inefficiency) are still present. An example of a hybrid approach for sparse and static environments is a global topological map with metric maps at the nodes.

Geographic Information Systems (GIS) are frequently used as a source for model data. The two traditional roles of GIS are storing and preparing data for models and also output visualisation [90]. The visualisation of the data can be in either raster form, where the environment is divided in equal cells and each cell is assigned a value, or vector form, where the data is represented as lines and points (for example, roads and postcode boundaries). Progress has been made combining
transport models and GIS into integrated modelling systems. Initially there were issues with integration such as concepts from one system not able to be represented in the other (GIS roads, transport modelling matrices) and also a lack of standard programming languages. Now integrated systems with a subset of GIS functionality are common. The GIS functionality allows the package to be linked to a standard GIS package and introduces the benefits of accurate networks and a database.

### 3.2 Pedestrian modelling

The steps in the modelling process are problem definition, data collection, development of a computational model, validation, experimentation or scenario design and runs, analysis of outputs, and reporting of results (Figure 3.1). The software development process is contained within this process and is discussed in Chapter 5.

There are many approaches to modelling everyday pedestrian behaviour. These can be divided into two schools. The first school is the “civil engineering” approach. This is concerned with forecasting demand so that decisions can be made about provision of new infrastructure. The
main outputs of these models are numbers of people travelling along various routes and objective crowding measurements. They are generally spatially macroscopic models, where the smallest detail of a pedestrian’s movement is the locations they visited and the paths they used to get there. The outputs of these models are usually aggregate, such as arrival and departure profiles at certain nodes and link flows and volumes. Another useful output is popular paths through the environment.

The second school is the “architecture/urban geography” approach. This group is interested in how people move around areas, in particular how design and location of certain attractions influence their movements. These models are usually spatially microscopic, in that they model a pedestrian’s path in more detail, usually in terms of steps or small grid squares. They are usually developed for small areas only, however some have been expanded to cover entire cities.

Some models for entertainment purposes have influenced or borrowed from the field, in particular Boids and Massive. Boids [79] was developed for animation in computer games and is the genesis of the separation, cohesion and alignment model sometimes used for groups of pedestrians. Motion is presented as a hierarchy of action selection, steering or path determination, and locomotion. Massive [54] is proprietary software for creating large computer-generated crowd scenes for films such as The Lord Of The Rings trilogy. The available literature emphasises the body movements and appearance of the individuals in the crowd, however the individuals are able to move around as they perceive the environment and other individuals using a set of cues.

3.3 Microsimulation

Microsimulation is an individual modelling technique, which is commonly used for transport modelling. Each individual is provided with a set of properties and transition probabilities are defined to determine their next action at each timestep. Cellular automata models are often referred to as microsimulations and will be discussed separately in Section 3.5.3.

A microsimulation model that is used frequently in industry is Paxport [40]. Paxport is a pedestrian modelling software package developed by Halcrow, a global consulting company based in the United Kingdom. The software was developed initially for modelling train stations and airport terminals, but has also been used for stadiums and their surrounds [80]. The environment is drawn using blocks, which is then processed into a graph representation, and is static for the duration of the simulation. It uses traditional traffic assignment methods to calculate the paths of each pedestrian and then uses microsimulation to move the pedestrian along the chosen path.
Figure 3.2 shows a typical output image showing the density at a particular time, however 3D animations are included in current versions of the software. Some of the limitations of the software are:

- Ideally pedestrians can only have one main purpose (i.e., catch a plane, alight from a train) which is sufficient for airports and stations, but is unrealistic for larger simulations such as the Commonwealth Games held in Melbourne in March 2006, where pedestrians may attend more than one event in a day, or town centre simulations, where people usually have several purposes.

- It cannot cope with “base load” (or no activity) pedestrians. In the Commonwealth Games model for example, these pedestrians are not attending a Commonwealth Games activity in the precinct. They may be walking through the precinct to get to Richmond Station, or walking across Princes Bridge at lunchtime. The model can simulate this by reducing the capacity of certain areas, however this is static for the duration of the simulation and therefore cannot be adjusted for peak periods.

- In terms of temporal demand, a profile is provided for the arrival and departure of trains and planes. This profile takes the form: 10% of passengers are to arrive 60 minutes before the flight leaves, 50% should have arrived 30 minutes prior, and 100% should have arrived 10 minutes prior. These values are provided to the passengers as a target, so although all passengers should have arrived 10 minute prior to take-off, some may be held up by congestion and could possibly miss the flight. The issue with this type of profile is that for planes and trains it is advisable to arrive before it leaves. However, for other events such as sporting events, it is unnecessary for people to arrive by the advertised start time, and their enjoyment of the event is not likely to be affected by arriving slightly late. It is also not uncommon for spectators to leave sporting events early, however Paxport cannot handle this as it forces pedestrians to stay for the whole event.

### 3.4 Physical and mathematical approaches

The main contributor to the use of physical and mathematical principles for pedestrian movement is Helbing. He developed a “social force” model, based on attraction and repulsion between pedestrians [43]. Other forces taken into account include environment boundaries and preferred
speed. Differential equations are used to determine where the pedestrian should move to and their speed. Figure 3.3 shows some sample equations from the social force model.

Teknomo’s research is similar to Helbing, however the physical forces in his model are slightly different. Forces relating to moving forward and avoiding collisions with others are modelled. This model was validated using video footage of a pedestrian crossing [95].

Hoogendoorn and Bovy [45] develop a mathematical model, consisting of three layers: activity choice, wayfinding, and walking. The walking layer was based on optimal control theory, where the agent has a state (position and velocity) and control (forward acceleration or sidestepping) and attempts to minimise its expected cost. This model was based on the physical nature of position, velocity, and acceleration, however acceleration also includes a component controlled by the agent. The uncontrolled acceleration component deals with the repulsion and friction causes when pedestrians collide. Calculation of the controlled component involves estimating the movements of others and my associated running costs and then selecting the optimal acceleration.

As mentioned in Chapter 2, transport is complex and complex systems are not able to be modelled analytically. However, an analytical approach that has been used for pedestrians is Space Syntax, which “investigates the relationship between human societies and space” [9]. The environment is analysed using graphs and maps to represent the accessibility of the building.
3.5. AGENT-BASED APPROACHES

Current location: $\tilde{r}$

Equation of motion: $\frac{d\tilde{r}(t)}{dt} = \tilde{v}(t)$

Equation of acceleration: $\frac{d\tilde{v}(t)}{dt} = \tilde{f}(t) + \xi(t)$

Behavioural force: $\tilde{f}(t) = \tilde{f}_0(v) + \tilde{f}_B(r) + \sum_{\beta \neq s} \tilde{f}_\beta(\tilde{r}_s, \tilde{v}_s, \tilde{r}_\beta, \tilde{v}_\beta) + \sum \tilde{f}_i(\tilde{r}_s, \tilde{r}_i, t) + \xi(t)$

where:
- $\tilde{f}_0(v)$: desired speed
- $\tilde{f}_B(r)$: repulsive force from borders
- $\tilde{f}_\beta(\tilde{r}_s, \tilde{v}_s, \tilde{r}_\beta, \tilde{v}_\beta)$: repulsive force from another pedestrian $\beta$
- $\tilde{f}_i(\tilde{r}_s, \tilde{r}_i, t)$: attractive forces to attraction $i$

Figure 3.3: Some of the mathematical equations used in the social force models of Helbing et al. [43].

3.5 Agent-based approaches

The definition of an agent is a contentious issue. Many different ideas have been put forward regarding the definition of an agent. A frequently used definition is given by Wooldridge [106, p5]:

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

The core concepts of situation, environment and action are mentioned in this definition, however it does not capture other desirable features of agents.

A more recent definition that builds on Wooldridge is the following from Padgham and Winikoff [74, p3]:

An Intelligent Agent is a piece of software that is

- Situated - exists in an environment
- Autonomous - independent, not controlled externally
- Reactive - responds (in a timely manner!) to changes in its environment
- Proactive - persistently pursues goals
- Flexible - has multiple ways of achieving goals
- Robust - recovers from failure
- Social - interacts with other agents

This defines the concept of an intelligent agent and adds the properties of reactive/proactive behaviour and social behaviour, which also takes into account communication.
The models used for transport systems are likely to contain many (possibly heterogeneous) agents. A definition of a multi-agent system (MAS) from Sycara [94] is:

The characteristics of MASs are that

1. each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;
2. there is no system global control;
3. data are decentralised; and
4. computation is asynchronous.

An alternative definition is given by Ferber [33]. He describes MAS as a system with the following elements:

1. An environment
2. A set of objects, situated in the environment
3. An assembly of agents, which are a subset of the objects in the environment
4. An assembly of relations, linking objects to each other
5. An assembly of operations, allowing agents to interact with objects
6. Laws of the universe (operators)

This definition provides a more detailed view of the concepts and relationships which is a basis for the agent-based simulation definitions in Section 3.5.1 and the model prototypes in Chapter 5.

The AgentLink Roadmap is published every few years with a summary of current agent technology and future trends. According to the 2005 Roadmap [60], agents are used as a design metaphor. This was also noted by Drougoul et al.: as the agents are usually developed in object-oriented or logic languages, the computational agents themselves are not proactive or autonomous [31]. Other usages of agents identified by the Roadmap are as a source of technologies (for example, understanding coalitions and cooperation) and for simulation and decision support systems:

Agent technologies provide a way to conceptualise [complex] systems as comprising interacting autonomous entities, each acting, learning or evolving separately in response to interactions in their local environments [60].
3.5.1 Agent-based simulation

As with agents, a definition for agent-based simulation has not been agreed on, however several papers have attempted a definition.

Davidsson [27] defines multi-agent based simulation (MABS) as “simulated entities modelled and implemented in terms of agents”, and that MABS is mostly used for microsimulation applications, as opposed to macro or aggregate models.

Edmonds [32] concentrates on MABS as a science, which is defined as a “tool for understanding observed systems”. MABS itself is defined as an “attempt to model a multi actor system with a multi agent system”, where the modelled actors can be anything: people, objects, concepts, institutions etc.

O’Sullivan and Haklay’s definition is “agent-based modelling removes agents from real world and places them in artificial reality” [73].

The three definitions are all different. Davidsson’s definition does not acknowledge the real world, whereas those of Edmonds and O’Sullivan and Haklay do. These two also acknowledge the close relationship between the real entities and the simulated entities, although O’Sullivan and Haklay imply that no abstraction or simplification of the agents in the problem occurs, which is overly ambitious. For transport modelling, Edmonds is the most accurate and this thesis will assume his definition.

Bonabeau [18] believes that “agent-based modelling is a mindset more than a technology” and that microsimulation and differential equation modelling are also agent-based. While this is valid, as these approaches are individual-based, I have chosen to treat them differently as they do not fully conform with Padgham and Winikoff’s definition of an agent. Davidsson [27] claims the transition probabilities used in microsimulation do not directly translate to personal beliefs and plans. The differential equation modelling has more in common with particle modelling. Sometimes Helbing refers to agents in his work, but makes no attempt to explain the concept or justify his choice of words. In Helbing and Molnàr [42], the key article on his social force model, the word “agent” is not mentioned at all.

Little research has been undertaken into methodologies specifically for agent-based simulation. Drougoul et al. [31] note that there are many methodologies associated with agent-based simulation that are not agent-specific. Campos et al. [25] present a methodology for agent-based simulation, however at a high level the phases (requirements, modelling, design, implementation, verification/validation) are very similar to software development. The inclusion of a modelling phase is
still not agent-specific. However it is noted that more agent-based concepts will be included in future versions.

3.5.2 Geosimulation

Geosimulation is a new term to relate the areas of urban geography, analysis and modelling [14]. It aims to describe the gap between theory and modelling and is based on the trend for disaggregate modelling.

An architecture for geosimulation models has been developed based on object-oriented design principles [14]. In essence, the architecture consists of automata (e.g., agents, houses), states that the automata can be in, transition rules between states, locations, movement rules, neighbourhoods and neighbourhood rules. Each model has a city with some populations of automata. Each population has its own domain. Each automata has relationships with other automata. Only fixed automata (e.g., houses) can have relationships with each other, and non-fixed automata (e.g., pedestrians) can have relationships with fixed automata. Non-fixed automata cannot have relationships as the neighbourhoods are always changing. Each automata has some attributes, which are classified as vector, fixed, or continuous. The simulation can run in either synchronous or asynchronous mode.

The geosimulation framework covers two approaches: cellular automata (CA) and multi-agent systems (MAS). In the CA version, the automata are stationary as in traditional CA models. In the MAS, there is no built-in representation of the environment. This division is also echoed in Batty [13] (both authors used to work with the same group), although the latter book is more academic and not aimed towards a practical audience.

Benenson and Torrens argue that this framework is preferred to microsimulation and Artificial Life. Microsimulation is not spatially microscopic, as it may use zones as the geographic location of each entity. Geosimulation is also preferable to Artificial Life as the latter is too long term for geographical applications.

3.5.3 Agent-based cellular automata approaches

Cellular automata were originally developed as a representation of self-replicating systems, and have been demonstrated as such by Conway’s Game of Life in the 1970s and by Wolfram in the 1980-90s [105]. In these systems, the world is represent as a grid and a set of local rules are defined for the system. At each time step, the state (usually “on” or “off”) of each cell
is updated synchronously using these rules. Both Conway and Wolfram demonstrated global pattern emergence which belies the simple rules in the system.

The use of CA in pedestrian modelling is a variant of the original approach, as the concept of cell states needs to be adapted. The update rule needs to be able to move pedestrians from one cell to another, which requires more complex rules to check for conflicts. The notion of using CA for pedestrian modelling was influenced by its use for vehicle behaviour in the 1990s [67]. This model was based on the number 184 CA [17]. This CA rule, as shown in Figure 3.4, is number-preserving. That is, it has the same number of on-cells at each time step. It is a simple representation of one-way, single lane behaviour.

Unlike the simple model developed by Nagel and Schreckenberg [67], in other models of pedestrian behaviour the automata physically move around the environment. This approach is often referred to as agent-based CA, to distinguish it from the traditional CA approach.

Cellular automata have primarily been used for investigating micro-level behaviours, such as crowd behaviour and single link flows. Such models use very simple rule-based agents that perform actions based on the world immediately around them. However, the results of the models mirror real-life behaviour. In particular, Blue and Adler [16] model passageways and demonstrates that the aggregate flows are comparable to reality.

Blue and Adler claim that an emergent property of the model was that agents form corridors of movement, similar to two-way passageways in reality. However, their lane changing rule set gives precedence to lanes where the person in front is heading in the same direction. It is debatable whether this property is emergent since it is built into the rules.

The AMANDA project [29], currently underway at the Faculty of Architecture, Building and Planning at the Eindhoven University of Technology, examines how design influences behaviour and focuses on shopping malls. It also uses a CA approach with a 3D environment. Some cells are designated as “decision” cells, where the pedestrian can decide to change directions based on its long term goals. A layer of decision making was built on top of the CA [30]. Each pedestrian’s behaviour consists of selecting an action, “steering”, and movement between cells in the environment. This echoes the hierarchy used in Brooks’ Creatures Brooks [22] and also in Boids by Reynolds [79].

The research undertaken at Napier uses an agent-based model to investigate pedestrian behaviour [49]. Each agent occupies a cell and at each time step, decides where to move next based on where they are going and what they can see, so both micro- and macro-level behaviours are
Figure 3.4: Two instantiations of the number 184 (10111000) cellular automata, 200 cells wide and for 200 timesteps (with time increasing from top to bottom). The first environment contains 88 objects that organise into free-flowing traffic quickly, as shown by the lack of black bands. The second environment contains 103 objects and some traffic jams continue to the end. Some jams in the upper left hand corner dissipate.
modelled. Some advantages of this model are that it is based on a small set of parameters, a decision system which makes simulations easy to set up, group behaviour can be modelled, and the outputs include both individual statistics as well as traditional model outputs. The model was validated using video data.

Schadschneider [88] introduces the use of floor fields. The model uses two fields: a static field signifying fixed information about the environment such as the distance to the exit, and a dynamic field which is created by the pedestrians as they move around. This has the effect of storing global or long distance knowledge locally. The pedestrians can have varying sensitivity to these fields. They can have a higher factor for the static field, which corresponds to someone familiar with the environment. Those with higher factors for the dynamic field are more likely to follow other pedestrians. The fields diffuse over time, so older trails left in the dynamic path will have less influence. The software PedGo, used primarily for evacuation analysis and crowd behaviour, also uses a cellular automata approach with the floor fields [99].

Henein and White [44] adapt Schadschneider’s model and build a model of egress from a sporting arena, in particular looking at the effect of pushing forces due to urgency or injury. Each person chooses where to move next based on the cells around them and also the overall force field of the crowd. Those who become injured as a result of being crushed stop where they are and create a “block”.

The most difficult concept in CA involving movement is avoiding conflicts or collision between pedestrians. A more recent technique involves determining the collision rules during the simulation [69]. The bidirectional model is run for 2000 time steps after which collision patterns are collated and the probabilities of collisions occurring are calculated and communicated to all pedestrians. Each pedestrian then continues on walking, preferring collision patterns with lower probabilities of resulting in conflict.

Kirchner et al. [52] again use Schadschneider’s model to experiment with collision behaviours. Traditionally, collisions are either broken evenly between the parties or weighted towards the pedestrian with the highest probability of moving to that cell. In both cases, one of the parties moves into the disputed cell. Kirchner et al. add in a friction parameter $\mu$, which means that with probability $\mu$ no party moves into the cell. This occurs in real life as “you go”, “no, you go” behaviour. Experiments in an evacuation situation found the friction parameter had the most effect in higher density situations. Another situation in which interesting behaviour occurred was when an obstacle was placed just in front of the exit. In situations where pedestrians were mainly
influenced by the static field, the obstacle had the effect of dividing the pedestrians and lowering evacuation times.

Finally, the TRANSIMS model is an agent-based cellular automata model that can model all modes of behaviours [24]. It consists of several steps to analyse data to create a population, activities, and movement. It has been used in the USA to model large urban areas, such as Portland, Oregon.

Although the cell-based nature of CAs produces good results for small models, in reality people move in a vector-like fashion, i.e., in many different directions and at different speeds. On first impressions this approach is rather limited. However, several of the models reviewed allow pedestrians to move more than one cell at a time which assists in realistic output. Given the current range of projects, it is a very popular approach to modelling pedestrian behaviour.

### 3.5.4 Agent-based approaches

Most models make use of cell-based environments, mostly due to the implicit inclusion of spatial location. However some use agents in non-cell-based environments.

Legion [57] is proprietary software specialising in modelling crowd behaviour. It treats each agent as a “virtual person” who senses their environment and make decisions about where to move accordingly. Train stations, sporting events and evacuation scenarios have been modelled using the software. Legion was first developed as part of Keith Still’s PhD thesis [91] as a model of ingress and egress from events.

An interesting aspect of Legion is the incorporation of worst-case analysis. Worst-case analysis is used in structural design, where several different extreme load combinations are tested and the combination that produces the worst result is used in the design process [102]. With Legion, three evacuation choices are tested: choosing the nearest exit, choosing a random exit, and all pedestrians choosing the same exit [91]. If all of these situations produce satisfactory results, then the design is satisfactory. If one situation performs badly, then a redesign is required.

### 3.5.5 Hybrid models

Hybrid models use several environment representations.

The AlpSim project [36] involves a large scale model that will be used to evaluate future scenarios in a tourist landscape. The techniques used include vision, arbitrary movement, and learning (several pre-runs to build up knowledge). Scalability issues are also discussed due to the
large area covered by the model. This group has also investigated a hybrid approach of modelling using both CA and graph-based environments.

The research team at the Centre for Advanced Spatial Analysis at University College London are also interested in pedestrian movement, but from a geographical and urban planning perspective. This group created a model combining the agent modelling language SWARM with GIS, concentrating on movement in urban spaces [73]. The model involves two stages: a pre-model stage where all the parameters are set, and the actual model stage. The pedestrians move around the cells created by the raster layer in GIS. It makes use of socio-economic data to set up each person’s characteristics and determine which locations they would be most likely to visit. Several areas are identified as requiring further research including attractability, group behaviour, and navigation.

3.6 Summary

This chapter has focused on approaches to modelling pedestrian behaviour. Many models have been created by computer scientists and mathematicians looking for applications of theory and also engineers and planners requiring valid models for industry.

At the end of Chapter 2, several themes were identified that appeared throughout the models described in Chapter 3. All the models (except for Space Syntax) involve dynamic movement around an environment and these environments contain both static (fixed) and dynamic (non-fixed) elements. The models produce objective measurements as outputs. Finally, some models considered small-scale movements at the microscopic scale, some considered broader movements, and some could produce several levels of planning and movement. Some models discussed the emergent properties of the micro-movements.

It is clear that modellers have a lot of choice when they are deciding on a type of model to use for their project. In the next chapter, a framework to help make this choice is developed.
Chapter 4

Approach framework

The ability to choose and adapt models for particular contexts is one of the most important elements in the complete planner’s tool-kit (Ortuzá and Willumsen [72]).

In Chapter 3, a range of pedestrian models were reviewed. All had two concepts in common: pedestrians moving around an environment. Some captured different behaviours to others and some worked in certain environments only.

This chapter is inspired by the issues involved in determining the scope of a pedestrian model, given the wide choice of approaches presented in the last chapter and the experience and needs of the people involved in the modelling project. This activity occurs at the beginning of the modelling process, after the planners have identified the problem and now want to evaluate scenarios. Undertaking scope definition well can lead to better estimates of time and cost and well-defined work responsibilities [89], and eventually to a model that provides the data required by the planners for their decision-making. This requires the modellers and planners to communicate using the same concepts to avoid misunderstanding.

A review of processes for selecting an approach for a model is presented. An important part of scope definition is defining the different roles of the stakeholders. This is discussed with an analysis of the requirements for pedestrian models from the viewpoints of the stakeholders. The key factors in designing pedestrian models are identified and a framework for selecting a particular modelling technique based on the type of model desired is developed.
4.1 Methodology

Little work has been undertaken in understanding the selection of the technical approach to a problem. An obvious parallel is that of selecting a programming language or platform for a software project. McConnell [62] states that programmers are more productive when working with high-level languages and languages with which they are familiar, however he does not discuss the choice of language for a particular project. Law and Kelton [55] claim “the most important decision a modeler or analyst must make in performing a simulation study is the choice of language”.

Ortúzar and Willumsen [72] suggest selecting a transport model should be based on the scope of the project, the accuracy required, the data available or required, the resources available, and the training or skill required. Their criteria are all relevant to pedestrian modelling and the main criterion is the context of the model required. Some criteria apply to the approach overall rather than to the domain, such as resources and training. These will be discussed under the relevant approach. The training and skill required is one of the criteria used in Chapter 6 where the prototypes presented in Chapter 5 will be evaluated.

Davidsson et al. [28] develop a framework for evaluating agent-based approaches to logistics management, in particular the movement of freight. A summary of the criteria is shown in Table 4.1. These criteria are model-specific, whereas our framework will deal only with generic approaches.

The methodology for this framework is to identify the environment domains, the behaviours, and the approaches, using the existing models and theory. These elements are similar to Davidsson et al.’s domain, mode of transport, and approach, whereas the first two are referred to by Ortúzar and Willumsen’s context. The domains each have their own behaviours, so an approach will be recommended for a domain taking into account the behaviours required.

4.2 Stakeholders

All projects have stakeholders, who are involved in or affected by the project [89]. They may have different needs and expectations of the project. As an example, transport models are intended to be an aid in the overall planning process, not the final arbitrator nor explicit solution generators. I was once involved in developing a model for a client which was exploring the congestion effects of placing services in different locations. During a presentation, we were asked about the limitations of the model, in particular could it suggest appropriate positions for services.
Table 4.1: A summary of framework criteria for evaluating agent-based approaches to logistics management, from Davidsson et al. [28].

An example of different needs within the project team is demonstrated in the following anecdote from a social scientist about an encounter with modellers.

We had identified a case study of a prominent local catchment, highly valued and used by the local communities, but with increasingly serious environmental problems. We decided we wanted to build a model to help sustainably manage the catchment. So we [a group of modellers and social scientists] all sat around a table to begin this.

A modeller immediately launched into a list of “things” she needed for the model - data requirements, and modules, and other “things” that modellers would be familiar with. I listened for a while and then I said, “Who are we doing this for? What are the issues
we need to address? Who needs to be involved? Who decides what should happen in the catchment?” There was genuine confusion and questioning on both sides.

So this was the first enlightenment. I wanted to start with a list of people to talk to. She wanted to start with the physical data and model specifications. The approaches were similar in that they both involved data collection, but for quite different purposes. One had assumed the problem, the other wanted to identify and define the problem. 

(Nancarrow [68])

4.2.1 Roles

There are several stakeholders in the development of a pedestrian model. The roles of people involved in the development of a model have not been researched in depth, but are of interest to agent-based modelling [31].

The client has a need for forecasts and is likely to be a planner or event organiser. They are likely to have or have access to the most information about the domain, including the problem and the environment. They will have observed the environment and can provide information about the current situation in the form of current usage or environment layout. They may sometimes provide their opinion on the cause of particular behaviours. They will also usually provide a set of scenarios, which will assist them in making their final decision. However, clients have their own notion of what will happen for certain scenarios and are often sceptical if the model suggests differently.

The practitioner is likely to be an engineer or planner and needs to provide a service to the client, including provision of forecasts and analysis of scenarios. They interface between the client and the developer and therefore require some knowledge of both those roles.

The developer creates a model that is representative of the reality. They may develop a model from scratch using a general purpose language (e.g., in Visual Basic for Applications or .NET) or use an existing simulation language or package (e.g., Paxport)\(^1\). They need to have good understanding of the features of the package or language, so that they can suggest modelling methods to the practitioner. The model needs to meet the requirements of the practitioner and the client.

Murray and Cuddy [65] identify three roles in their work: model developer, model deliverer and model user. Their model user uses the model to run scenarios and then interprets the results,\(^1\)

\(^1\)The different types of languages and packages will be explained in Chapter 5.
4.2. STAKEHOLDERS

making it a combination of our practitioner and client. The model developer is the go-between who has knowledge of the domain and modelling, and the model deliverer is identical to our developer. Drougoul et al. [31] use the roles of thematician, modeler and computer scientist which correspond roughly to our roles. Finally, Campos et al. [25] define the end-user (our client); the domain expert and modeler (similar to Murray and Cuddy’s model developer); and the software architect, designer, developer, and tester who are equivalent to our developer.

A person may perform one or more roles, e.g., the practitioner and developer may be the same person. The client and the practitioner have different requirements for the model.

The client’s requirements are:

- an understanding of the model scope: they need to understand (at a high level) the behaviours the model can create, in order to make a judgement about the validity of the model. They also need an understanding of the environment constraints. If they are interested and/or experienced in modelling, they may also like an understanding of the model parameters.

- results in a variety of formats: these include charts and maps for reports. 2D and 3D animations are also useful for presentations to senior stakeholders or the community, as it provides a more realistic feel for the effects of a scenario.

The practitioner’s requirements are:

- ease of use: the package should follow current software engineering principles and should be straightforward to use. Graphical user interfaces should be intuitive and input and output data should be read and written to appropriate locations and in an easy-to-manipulate format.

- a clear understanding of parameters: the parameter should have some resemblance to the real world. It should also be clear what the intended effects of each parameter are, if changed in isolation. This may not always be possible with emergent models.

- the ability to modify parameters quickly and easily: again, the GUI should enable parameters to be changed easily. It should not take long to make changes to several parameters to set up a new scenario.

- the ability to read in environment data from various sources, including temporal constraints: environment data is extensive and there are various representations that can be used.
• flexibility with output: the practitioner should be able to select different outputs to suit the project and the client’s needs. This involves selecting different calculations and at different environmental scales (e.g., block, street, area, model) and at different times (peak/offpeak).

• a reasonable running time: often a large number of scenarios need to be developed and analysed, so model running time is a key factor. Often practitioners will not have access to a dedicated model-running machine.

4.3 Framework elements

The elements in the framework are the environment domains, behaviours and approaches.

4.3.1 Categorisation of domains

Generally, the location to be modelled will be known at the start of the project. The following domains are typical examples of pedestrian areas.

Small-scale enclosed spaces

Small-scale enclosed spaces consist of small rooms connected by corridors and exits. For example, buildings often have many enclosed spaces (e.g., offices, meeting rooms). Multistorey buildings will have lifts, stairs, or escalators to facilitate movement between floors. The type of walking trips in this environment are short and purposeful, with little chance of distraction. The number of pedestrians is variable depending on the location and the map representation would be at a small scale. Figure 4.1 shows an example of this domain.

Open-plan enclosed spaces

These are generally larger buildings that are open-plan. For example, sports arenas consist of an area filled with seats, aisles, and exits. Cafes and souvenir stands could also be found within the arena, so queues are likely to occur outside these shops and also at exits at the end of the event. The trips in this environment are likely to be short and purposeful (e.g., from entrance to seat, from seat to cafe), but some will be familiar with the environment, others not. Another example is an airport (Figure 4.2) or a train station, where the main purpose of walking is to change between pedestrian and public transport mode. In these environments, temporal constraints are present, e.g., one needs to catch a train/plane at a particular time (hard constraint) or one should aim to
arrive for the start of a sports match (soft constraint). These environments often contain a large number of pedestrians in a small area.

**Mixed mode spaces**

A mixed mode environment is an area, possibly shared with cars or public transport, that connects the pedestrian to building entrances and other streets (Figure 4.3). The pedestrian has static objects (e.g., public seating, rubbish bins, garden areas) to navigate around. Another element of this environment is a queue, which can be either an ordered queue of people waiting to get into a busy shop or an unordered queue of people waiting to cross the street or waiting for a bus. Trips in this environment are likely to be a mix of familiar trips with a purpose (e.g., those who are walking to work), purposeful but unfamiliar trips (e.g., I need a pharmacy but I don’t know where
one is), and purposeless trips (e.g., a shopping trip to the city).

**Open spaces**

These consist of open areas, possibly with some designated pathways. The purpose is most likely to be leisurely, so the behaviour will consist of meandering, frequent stopping, and possibly longer stops for picnics or sightseeing (Figure 4.4).

However some open spaces, such as national parks, are quite constrained. These areas consist of set paths that walkers must keep to. These trips are likely to be purposeful, but with occasional distractions. Decision points are likely to be well-signed for wayfinding purposes.
Hybrid spaces

This category includes generally pedestrian areas or low-traffic areas containing several attractions, such as sports precincts or universities. It will consist of a combination of behaviours from the open space environment (e.g., meandering, afternoons on the lawn), the mixed-mode environment (e.g., avoiding vehicles, queueing for public transport), and the enclosed spaces environments (e.g., moving around lecture theatres).

4.3.2 Categorisation of behaviours

Pedestrians exhibit many behaviours and some of the more prevalent are described in this framework.

Purposeful and familiar

The pedestrian knows where they are going, how to get there, and has a low probability of being distracted on the way. This is described by Weston and Handy [103] as commuting.

Purposeful and unfamiliar

The pedestrian knows where they want to go, but is not sure how to get there and as a result may get distracted or lost on the way. This is described by Weston and Handy [103] as a quest.

Purposeless

The pedestrian is in wandering mode. Where the starting and end point is the same, this is described as a circuit by Weston and Handy [103].

Evacuation and/or panic

In an extreme situation, the pedestrian may panic and behave differently to normal. If this behaviour is required, it will be the main focus of the model and other behaviours will probably not be included.

Forced waiting

This behaviour occurs in environments where pedestrians have to wait for an action to occur before they can continue. For example, they may wait in a queue to buy a train ticket, but they have no control over this queue. Another example is an unordered queue at a traffic light where
pedestrians must wait for the green light. Fruin [34] describes these as *ordered* and *bulk* queues respectively.

**Temporal constraints**

These occur in train stations and airports (so-called hard constraints: the train leaves at a certain time and if you miss it, you miss it completely) and also at sporting events (soft constraints: you can arrive slightly late for the game, but you can still be admitted and see most of it). Temporal constraints have not always been considered in pedestrian models, but in my experience they are a requirement for accurate outputs for some environments.

### 4.3.3 Categorisation of approaches

The different approaches or techniques are based on the review of models presented in Chapter 3.

One method for approaching this is to look to the categorisation of agent architectures. For example, Wooldridge [106] lists the different architectures as logic, reactive, belief-desire-intention, and layer-based. Ferber [33] describes modular architectures, blackboard-based architectures, production rules, classifiers, and subsumption-based architectures amongst others. These categorisations deal only with the internal structure of the agent. The environment is of no relevance. However with transport modelling, the environment is just as important as the representation of the human or vehicle.

The categorisation presented as part of the framework is more detailed than those for agent architectures. This is necessary as the environment is also an important factor in transport modelling. The historical background and influences of and knowledge required for the approach are also taken into account.

**Mathematical approaches**

These approaches were discussed in Section 3.4. To use this approach, advanced knowledge of applied mathematics and physics is required. The environment is usually continuous and differential equations are used to determine where to step next. The models of Helbing and Molnár [42] and Hoogendoorn and Bovy [46] are good examples of this approach.
Microsimulation

These techniques were discussed in Section 3.3. The individuals in the model use a set of transition probabilities to determine their next state, however an environmental representation is not specified as part of the approach. Paxport [40] is an example of this approach and an understanding of traditional transport modelling techniques and basic simulation principles is required.

Cell-based agent approaches (CA)

Several projects using this technique were discussed in Section 3.5.3. Individuals are situated in a discretised environment and move zero or more cells in each timestep. A prototype developed using this approach is described in the next chapter and the training required will be discussed in Chapter 6.

Agent-based simulation (ABS)

These approaches were discussed in Section 3.5.4. The agent architecture and environment are not explicitly specified as part of the approach. As with the CA approach, a prototype developed using this approach is described in the next chapter and the training required will be discussed in Chapter 6.

4.3.4 Framework

In software engineering, the language or package to be used for building a system is dependent on the requirements of the system. In practice, the development environment is usually chosen before the requirements are set out. This could be because the developers have extensive experience in a particular package, or the client prefers a particular package.

With modelling, the same sometimes occurs: the approach chosen is the one that the stakeholders are the most familiar with. The client may want a model they can reuse or adapt in the future and this puts a constraint on the selected approach. The practitioner may also be more familiar with one approach over another. This may not necessarily be the best approach for the job. Ideally, the approach should be chosen in a more formal manner based on the requirements of the model.

For pedestrian models, there are several key factors that can point to an approach being more suitable than another.
Earlier, five domains (Section 4.3.1) and six behaviours (Section 4.3.2) were identified that may be modelled. Not all behaviours are present in the chosen environment or are of significant interest to clients. Some approaches suit one environment or behaviour better than others. For example, an agent approach may be more suitable when there are complex decisions to be made about activities and moving through the environment.

Scale is also a key factor in choosing an approach. This includes both the number of pedestrians to appear in the model as well as the size of the environment and the detail required. The choice of the scale of the model is also related to the outputs. If approximate volume counts are required, then an approach that models the exact steps of each pedestrian is unnecessary.

As the first item to be decided usually is the location (or type of location) to be simulated, this should dictate the approach chosen. From my experience, it is common for clients to decide later the exact outputs required.

**Small-scale enclosed spaces**

*Behaviours: direct, some wandering, possibly evacuation*

Office buildings consist of mostly direct trips, whereas shops and other “leisure” buildings have a combination of direct and distractable trips. A CA approach would suit the former, as there would not be enough traffic to require a mathematical approach and the microsimulation and agent approaches would be too complicated. For “leisure” buildings, an agent approach would be preferred, in order to see the individual choices that people make when moving through the area.

**Large-scale enclosed spaces**

*Behaviours: direct, some congestion and queueing, hard and soft temporal constraints, possibly evacuation*

A microsimulation approach would be good as this could model the queues and level of service at exits easily. For more detail of the crowding, a mathematical or CA model could be used. As there is minimal decision making involved in the environment and the pedestrian count may be high, an agent approach may not be appropriate, however Legion has also been used for arenas and train stations [57]. Paxport was designed for the airport and train station environments.
Mixed mode

*Behaviours: direct, wandering, queues, congestion*

All approaches are suitable for a mixed mode environment, however if the model is too large in area a CA model would probably be unsuitable, due to the large amount of detail required for the environment. Agent-based simulation and microsimulation could also model the vehicles in the environment, especially public transport vehicles that are effectively an exit from the pedestrian model.

Open space

*Behaviours: wandering, bushwalking (leisure), picnicking*

Depending on the level of choice involved in the model, the best approaches are a microsimulation approach or an agent approach. There is not enough interaction or congestion to warrant a mathematical approach or a CA approach. Ideally the model should scale between a large block representation and smaller units. Itami and Gimblett [47] have successfully used agents to model the decision-making behaviour of people in recreational settings.

Hybrid environments

Hybrid environments are complex and it is difficult to recommend an approach. It may be necessary to create more than one model to retrieve the required outputs.

4.4 Summary

This chapter presented an analysis of the requirements for pedestrian models from both the end-user’s and developer’s viewpoints by reviewing models developed in research and in industry. The key factors in designing pedestrian models were explored and a framework for selecting a particular modelling technique based on the type of model desired was developed and presented. An summary of the framework outcomes is shown in Table 4.2, where ticks signify appropriate approaches and tildes signify approaches that are not so appropriate.

Davidsson et al.’s related project on logistics approaches noted that although logistics problems have characteristics that align with agent technology and many approaches have been suggested and implemented, the maturity of the research was low and systems were not being evaluated against traditional systems.
Batty [13] claims that there has “never been enough applications to generalise this field into distinct types, for there are elements of each approach in every other”. This is true, but the number of applications is growing as the interest in pedestrian modelling increases. The framework can also serve as a tool for understanding the different elements of pedestrian models.

This framework will be of use to clients, practitioners, and developers. It will play a strong role in the usefulness and reliability of pedestrian modelling in the decision-making process for planning and design of pedestrian-frequented areas. The practitioner and client can use it to select the best approach to solving the client’s problem. The client, depending on their knowledge and interest, can also verify that a defined process has been followed for the specification of the model. Using common concepts to discuss the scope of the model early in the model development process may lead to less misunderstanding of the model’s capabilities later in the project.

As seen in Section 4.2.1 regarding the different stakeholders in modelling projects, there can be more than one organisation or categorisation of objects or concepts. The framework described in this chapter is one viewpoint. The framework could be adapted as necessary and also developed further as noted in Chapter 7.

Once the approach has been chosen, the next step is to implement the model. The next chapter develops two prototypes using agent-based cellular automata and belief-desire-intention agents.

<table>
<thead>
<tr>
<th></th>
<th>Mathematical</th>
<th>Cellular Automata</th>
<th>Microsimulation</th>
<th>ABS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small-scale: office</td>
<td>✓</td>
<td>✓✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Small-scale: leisure</td>
<td>✓</td>
<td>✓✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Large-scale</td>
<td>✓</td>
<td>✓✓</td>
<td>✓</td>
<td>~</td>
</tr>
<tr>
<td>Mixed mode</td>
<td>✓</td>
<td>~</td>
<td>✓✓</td>
<td>✓✓</td>
</tr>
<tr>
<td>Open space</td>
<td>~</td>
<td>~</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4.2: A summary of the recommended approaches for a particular domain.
Chapter 5

Model prototypes

Once an approach has been selected, the model must be created. For traditional transport modelling, point-and-click modelling packages exist for creating models, such as TRIPS. Many transport modellers have been trained in the theory of the four-step model and training is available for modelling packages. Law and Kelton [55] distinguish between simulators that require no programming skill, simulation languages that provide additional features for modelling but are general in nature, and general-purpose languages that provide no special features for modelling.

With a few exceptions, packages do not exist for agent-based modelling of pedestrians. Few modellers have received formal training in agent-based simulation. However, many wish to take advantage of “new technology”.

In this chapter, the software development step of the modelling process is investigated. Two prototypes of pedestrian models are developed using common software tools and methods. The overall objective is to explore the software development process required for agent-based models and the applicability of these models to pedestrian behaviour. An evaluation of the prototypes using software engineering quality principles follows in Chapter 6.

The first approach, agent-based cellular automata, was chosen as it was one of the more popular approaches, based on the literature survey in Chapter 3. This is an example of using a general-purpose language (Java) and object-oriented design to implement a simulation model. Several behaviours will be experimented with to determine differences between behaviours.

The second approach was the belief-desire-intention (BDI) agent architecture. The agent programming language JACK has been successfully used for defence simulation projects. The objective of this project was to see if it was as successful for other domains. The cultural influence of
BDI programming was a factor in selecting this technique, as this project was based in Melbourne near the home of Agent-Oriented Software and the research being undertaken at RMIT. JACK, and in particular the JACK Sim add-on which is utilised for this prototype, is an example of a simulation language. This prototype will also make use of an agent-oriented design methodology.

As discussed in the framework in Chapter 4, the two approaches used for the prototypes are appropriate for different applications. As a result, the same application cannot be used for both models and therefore identical experiments cannot be run with both prototypes.

### 5.1 The software development process

As discussed in earlier chapters, the modelling process contains a software development step. The process contains five steps, and their order is defined by a software process model. Figure 5.1 shows the waterfall model, where the phases of requirements engineering, design, implementation, testing, and maintenance are all followed in that order with little or no backtracking. Other models include the spiral model and the incremental model, which are more commonly used due to their adaptability.
The different steps in the process are described by Pressman [76]:

- The software engineer begins by gathering and analysing the requirements and preparing a specification for the software. This includes information about the domain as well as the software’s function, performance, and interfaces.

- From the specification, the software is designed. The resulting design considers many layers of a program from the high-level architectural overview to algorithms and data structures used.

- During the implementation step, the design is translated into code.

- The code is then tested to ensure that it meets the requirements and that it works as expected.

- Finally, the product is maintained during its lifetime. During this step, errors discovered during usage may need to be fixed or new features may need to be added.

For both prototypes, a lightweight spiral methodology was used. The spiral model combines the traditional waterfall model, where each software development step is executed in order, with the quick implementation time from the prototype model [76]. Effectively, one iteration of the spiral was made, resulting in a prototype. If desired, further iterations around the spiral could be undertaken to deliver a final product.

5.2 Agent-based cellular automata

In this section, the development of a agent-based cellular automata model is presented and the model is used to compare different behaviours.

The purpose of the application is to investigate the usefulness of modelling pedestrians using a cellular automata approach and the issues involved in using a general-purpose programming language for modelling. Pedestrians will move around cells in the environment. The objective of the model is to explore:

- the effect of different environment representations;

- the effect of different neighbourhood sizes;

- the effect of different behaviours, in particular conflict behaviour and also taking into account the number of pedestrians in the model.
The hypothesis is that more complicated behaviours will produce more realistic behaviour than very simple behaviours.

5.2.1 Methodology

For this model, a “start from scratch” model was developed using a general-purpose language, rather than using pre-existing modelling packages or libraries. In some cases, libraries are inappropriate or unable to be used for licensing reasons. It was designed using the Unified Modelling Language (UML) and implemented in Java.

UML

UML [86] is a standard technique for visualising a software system. Different views of a system can be described, including the structure of the system (objects and packages), the behaviour (use case diagrams), and interactions between objects in the system. UML is based on objects and is commonly used for object-oriented development.

Java

Java has been described as a good language for experienced programmers coding models [8]. The social simulation community also recommend Java as it “has a guaranteed future, it is a widely used powerful language, and it appears to be developing into a standard” [97].

Java was developed in the mid 1990s as a “portable, interpreted, high-performance, simple, object-oriented” platform [39]. The syntax is derived from C/C++, making it easy to learn for those familiar with those languages, but removes many low-level details such as explicit memory management.

The object-oriented programming paradigm is based on the concept of objects with attributes and methods and is seen to be a close abstraction of the real world [76]. A transport modelling example would be a vehicle object. This object would have attributes such as number of seats, number of wheels and maximum speed. It would also have methods such as accelerate. Each type of vehicle would then have an object that inherits or specialises the vehicle object. A car object might add the number of passengers, as some lanes may have restrictions on the minimum number of passengers allowed. Depending on the level of detail required, drivers and passengers may also be objects, containing socio-economic information which has been calculated from a database. In the model, instances of these objects are created, and their variables assigned a value. The
instances will act when one of their methods is called from another object instance. The simulation controller may call a car’s accelerate method, and the car may call the driver’s aggression method to find out how much to accelerate.

The Java source files are compiled into Java bytecodes, which is the language understood by the Java Virtual Machine (JVM). Most operating systems have their own implementation of the JVM that executes the bytecode output. Because the source is compiled to an intermediate language, rather than to the native language of the operating system, the developer does not need to create different versions of the program for different operating systems.

The Java Platform is distributed with the Java API, which provides a range of classes such as different Collections (containers for storing and organising objects for efficient access [5]) and several flavours of input and output streams for different types of data. It also provides user interface toolkits, which are important for modellers. These allow the developer to create forms and graphical output easily, without having to cater for different graphics systems on different operating systems.

5.2.2 Model development

The simulation model has been developed in Java using object-oriented principles. It is essentially a framework, which permits the developer to create and manipulate their own environment and pedestrian properties. It is similar to the geosimulation framework developed for geographic automata systems in Benenson and Torrens [14].

Requirements

The concepts required in the model are different environment representations, different neighbourhood sizes, and different behaviours. Other requirements are that the environment, cell and agent properties are easy to manipulate and it is easy to run simulations. User interfaces are required for data visualisation.

Design and implementation

The main components in the model are the environment, which consists of cells, and the pedestrians (agents). This segregation between environment and agents provides the necessary flexibility to model individual pedestrian behaviour independently and capture complex interactions.

1 The code for this prototype is available on request from the author.
In this study, we are dealing in abstract time and space, however, real-world sizes could be assigned to the cell dimensions and time steps based on current pedestrian design theory.

Agents occupy cells in the CA grid. They attempt to move towards their destination by selecting an unoccupied neighbouring site. They require a direction (either up or down) and the developer can create properties as required (such as speed, familiarity, current cell). These properties are manipulated when the agent moves.

There are six packages in the simulation program:

- data: contains objects for reading input and writing output;
- dist: random distributions for setting up the environment;
- env: contains representations of environments;
- ped: contains pedestrian information;
- sim: contains the simulation controller;
- update: contains operations for update pedestrians and the environment.

Some of the classes in the env, ped, and sim packages are shown in Figure 5.2. Additionally, two other applications were developed:

- Blocking: creates input data for an environment with obstacles;
- PathOutput: creates a visualiser for path data.

A package of random distributions is included in the model, such as Uniform and Triangular (Figure 5.3). The distributions are used to generate the entry cell for the pedestrian and the time they enter the environment. Each Distribution also has a second constructor with the same arguments plus a seed. In order for the starting configuration (i.e., entry point and time) to be “fixed”, the seeds for the upwards-moving entry cell distribution, the downwards-moving entry cell distribution, and the entry time distribution need to be set.

Each of the main objects in the model (environment, cell, and pedestrian) has properties. The environment properties store aggregate properties of the system. The cell properties constitute an information layer over the environment. The Moore-neighbourhood cells (the immediate surrounding cells to the top, left, bottom and right) and the agents can have alter the property value of a cell. The neighbourhood width is usually 1. Some of the pedestrian properties are compulsory e.g., the entry time and location, however all other properties are defined by the developer.
Figure 5.2: A UML overview of the classes in the CA model.
Each pedestrian has access to their current vision, the size of which is defined by the width and length of the vision. Cells that do not exist (e.g., outside the environment boundaries) and cells that are blocked are differentiated.

Three Update classes have been created to simplify updating properties during the simulation. These extend an Update class that contains generic operations on the Properties objects.

The input file contains the type of distribution for the up-moving pedestrians, the type of distribution for the down-moving pedestrians, the distribution for the entry times, and the list of pedestrian types and their initial parameters. A separate application has been developed to edit the blocked areas of environments. For cells, environments, and pedestrians, the model can output property data to a file.

**Deployment**

The Simulation object controls the simulation. It must be extended by the developer to define the environment and pedestrian initialisation, what happens during each time step, and the outputs required.

The command line requires three arguments: the simulation duration, the number of pedestrians, and the location of the pedestrian types file. This means simulations can be run with the same properties and it is simple to change the environment and number of pedestrians. The command line arguments are passed to several of the methods in the Simulation instance, so that extra arguments can be used e.g., environment size, output files/folders.
5.2. AGENT-BASED CELLULAR AUTOMATA

Figure 5.4: The probabilities of moving to the neighbouring cells for the random walk behaviour.

Figure 5.5: The order of trying to move to the neighbouring cells for the deterministic behaviour.

5.2.3 Scenarios

The investigation is limited to bi-directional agent movement in different environments (corridors of different widths with permanent blocks such as walls), and examining the effects of pedestrian traffic density and resulting macroscopic properties of the model. Four alternative behaviours were investigated.

Random walk

The simplest behaviour is a random walk. This involves randomly selecting a neighbouring existing cell (i.e., one that is not blocked or outside the environment), as shown in Figure 5.4. This behaviour was included as a baseline with which to compare other behaviours.

Deterministic

The next behaviour uses a deterministic rule-based agent. This agent tries to move forward. If the cell is occupied, it tries forward right or forward left. If neither of these exist, it tries immediate right or immediate left. This behaviour is shown in Figure 5.5.

Lookahead

The third behaviour uses a “lookahead” procedure adapted from Blue and Adler [16] to determine whether to continue in the same lane or change lanes. In short, the pedestrians look ahead a
Figure 5.6: The probabilities of moving to the neighbouring cells for some configurations of the lookahead behaviour.

number of cells in their current lane and the lanes to their left and right. If there is someone ahead of them moving in the same direction, then the gap distance is set at the actual distance to that pedestrian. If there is someone ahead of them who is moving in the opposite direction, then the gap distance is set to half the actual distance. The pedestrian then chooses the maximal gap and ties between gap distance are broken randomly. Figure 5.6 shows some examples of this behaviour.

**Floor fields**

The fourth behaviour [88] uses the concept of static and dynamic floor fields. The static floor field consists of the distance to a specific activity or exit. An example is shown in Figure 5.7. The dynamic floor field consists of the number of pedestrians who have passed through the cell
on the way to a specific activity or exit. Each agent has a parameter that dictates whether they follow the static floor field more (representing those who are familiar with the environment) or the dynamic floor field more (representing those unfamiliar with the environment who will follow another pedestrian going to the same destination). This has also been developed by Henein and White [44]. A variation from Henein and White was partially adopted because in crowded situations, the agents are surrounded by other agents, therefore all cells returned a probability of 0. This meant the agent did not attempt to move, leading to total gridlock. This system shows an example of two-way link between the parts and the whole, as the pedestrians influence the dynamic field and are then themselves influenced by it. This property is an example of emergence [104].

In this behaviour, the cell neighbourhood width is set to 3. The crux of the behaviour is shown in Equation 5.1, which shows the calculation of the transition probability from the current cell $i$ to a neighbouring cell $j$. Contributions to the probability are the static floor field $S_{ij}$, the dynamic floor field $D_{ij}$, the pedestrian’s preference for moving in that direction $M_{ij}$, whether the cell is occupied or not $n_j$ (which is always 1 in this model) and whether the cell exists or not $e_j$. The probabilities are normalised by the factor $N$.

$$p_{ij} = N M_{ij} D_{ij} S_{ij} n_j e_j$$ (5.1)

Collision behaviours

In each time step, each agent chooses their next cell. If two or more agents have chosen the same cell, the collision rules decide who moves to that cell and who stays in their current cell. The rules are based on the principles in Kirchner et al. [52], however, for this set of results the friction parameter $\mu$ was set to 0. The possible outcomes for a collision is shown in Figure 5.8.

5.2.4 Experimental results and discussion

Model parameters

Two different environment sizes were used in this study. The environment was 200 cells long ($L$) and the width either 15 or 20 cells ($W$). For some runs, a permanent blockage was introduced in the form of two doors at $L/2$.

Four density values ($D$) were used to vary to the volume of agents ($V$): $D = \{0.1, 0.3, 0.5, 0.7\}$
Figure 5.7: An example of a static floor field for an unblocked and a blocked environment for the upwards-moving pedestrians. In the blocked environment, the gradient breaks from the regular line pattern and curves around the blockage, leading people to walk around it.

Figure 5.8: The possible outcomes of a collision and their probabilities.
5.2. AGENT-BASED CELLULAR AUTOMATA

and \( V = D \times W \times L \). These densities provide conditions ranging from free-flowing to very congested.

Half of the agents started at the top of the environment and half at the bottom. The agents were randomly assigned entry times between 0 and \( L/2 \) and a starting cell in the first row.

Movement and collision resolution was synchronous and all agents moved at most one cell per timestep. A simulation trial was run until all pedestrians reached their destination or time reached \( L \times 100 \), whichever occurred first. The latter cutoff was chosen as a result of informal experimentation – usually by this time it was apparent a deadlock had occurred. Each scenario (environment-agent behaviour) was run nine times with different random seeds.

Results

A range of simulation model outputs were recorded during each trial, including the distance covered by each agent, the time they were in the model, the number of collisions, entropy and proximity.

The time and distance plots (Figure 5.9) show that the agents using the random and floor fields behaviours take longer to move through the environment. The deterministic and lookahead behaviours are more efficient in moving from the entrance to the exit.

The stops due to collisions plots (Figure 5.10) show that the agents using the random and floor fields behaviours are held up due to collisions a similar amount. The deterministic behaviour performs well at low densities. However, it degrades quickly at higher densities. The lookahead

Figure 5.9: Time and distance travelled divided by the length \( L \) of the environment.
CHAPTER 5. MODEL PROTOTYPES

Figure 5.10: The average number of times a pedestrian stops due to a collision.

behaviour appears to be approaching a maximum.

The snapshots (Figure 5.11) show the differences between the behaviours at the same point in time. The random and floor fields behaviours are clustering in the centre, whilst the deterministic agents have all moved to the right and out of each other’s way. Given their ruleset, which is to move right before left, this is expected. The lookahead behaviour is also clustered, but in one large cluster. The differences between densities for the same behaviour (lookahead) are also clear. At the time for the lowest density, the agents have almost reached their destinations. At the higher densities, a cluster has formed, but agents are still moving.

In order to investigate the population dynamics in more detail, the Shannon entropy of agent movement throughout a trial were recorded. The equation for Shannon entropy is shown in Equation 5.2. The variables measured were the number of pedestrians moving ($Q_0$) or not moving ($Q_1$) for all pedestrians currently in the model environment ($n$). The entropy was used to compare the similarity of behaviours.

$$e = -\sum_i \frac{Q_i}{n} \log_2 \frac{Q_i}{n}$$  \hspace{1cm} (5.2)

The peaks of the entropy plots (Figures 5.12 and 5.13) occur when the types of movement are evenly split i.e., half the agent are moving and half are stopped. The troughs correspond to the
Figure 5.11: Modelled crowds at the same time \((t=250)\) for different behaviour and densities. The red pedestrians are moving upwards, the green downwards. From left: random-30%, floor fields-30%, deterministic-30%, lookahead-30%, lookahead-10%, lookahead-50%, lookahead-70%. 
situation where the agents are mostly doing the same thing i.e., all moving or all stopped. The results for deterministic behaviour and the lookahead behaviour are similar, while the random and the floor fields behaviours are similar to each other. In this simple environment, the floor field behaviour is essentially an “intelligent” random choice. The entropy plots also show that the behaviours go through similar phases throughout their run. Model runs with a larger number of pedestrians have similar peaks and troughs, but translated along the time axis.

For densities 30% and above, the deterministic and lookahead behaviours have a trough around $t=100$ and a peak around $t=150$. At $t=100$, the agents who entered the model first have reached the centre. Up until this point, they could move freely as they were surrounded by pedestrians moving in their direction. However, at the centre they encounter pedestrians moving in the other direction and therefore collisions start occurring. At $t=150$, the two middle quarters are full of pedestrians, with very few pedestrians in the end quarters. The agents who entered the model first are encountering the last of the agents heading in the other direction. After this time, agents start to move freely again.

For the random and floor fields behaviours, there is a peak at $t=250$. As these behaviours are more likely to move sideways than the other behaviours, they take longer to reach the centre of the environment.

There are other peaks later on for some model runs. They usually occur when a crowd has
5.2. AGENT-BASED CELLULAR AUTOMATA

The slight increase in width had minimal effect on the behaviour.

The final metric used to help describe agent macroscopic behaviour is a proximity value $D'$. Traditionally density is measured as the number of pedestrians in a space divided by the area of the space. This does not give an indication though of whether the pedestrians are all spread out evenly or huddled together. An alternative method to measure proximity was developed: each cell has a density $d_c$ (the count of the pedestrians in its Moore neighbourhood) and the proximity at a certain time is the variance of the cell densities at that time.

$$D' = \frac{1}{L \times W} \sum_{c=L \times W}^0 (d_c - \overline{d_c})^2$$ (5.3)

The proximity plots (Figure 5.14) show that the number of pedestrians moving decreased as the proximity increased, as expected. However, the decrease in movement and subsequent increase as the proximity decreased were not along a similar gradient. The initial gradient is more concave and the later gradient more convex.

From the simulation results, it is clear that in this environment the floor field behaviour provides...
little advantage over the random behaviour. The floor field behaviour is more suited to a more
complex situation with many choices for exits and/or activities.

The deterministic and lookahead behaviours were similar, however the lookahead behaviour
performed better for collisions at higher densities and the differences between distance and time
travelled were negligible.

5.3 BDI agents

The second prototype used JACK Intelligent Agents, which is based on the belief-desire-intention
architecture. As discussed in Chapter 3, few models use agents without the cellular automata
environment. However, BDI agents have been used to model human behaviour and decision-
making, which is a large part of pedestrian movement.

The purpose of this prototype is to explore the development of an agent-based model of pedes-
trian behaviour, using agent-oriented design methodologies and an agent-based/simulation lan-
guage. The focus of the analysis of this model will be more on performance and usability rather
than behaviours. Unlike the other prototype described in this chapter, a library of behaviours to
experiment with does not exist.
5.3.1 Methodology

Agent-oriented software engineering is similar to object-oriented software engineering, however some of the concepts are different. As with the previous prototype, the design and implementation steps will be discussed.

Object-oriented programming operates at a higher level of abstraction than procedural techniques. Agent-oriented programming is further abstracted from the real world.

Belief, desire and intention

The belief-desire-intention (BDI) architecture is based on philosophy, in particular the work of Bratman [21]. The philosophical component of BDI is based upon practical reasoning. Practical reasoning is defined as reasoning toward actions, as opposed to theoretical reasoning, which is reasoning about beliefs. Practical reasoning can be broken down further into two activities: deliberation (deciding what goals to achieve) and means-end reasoning (how to achieve a goal) [106]. A feature of the BDI architecture is the ability to act in both a reactive and proactive manner, however there is a danger of being too reactive or too proactive. BDI agents are usually more proactive than reactive.

The key concepts in the BDI architecture are:

- beliefs: what I know or don’t know about the world;
- desires: what I want to do;
- intentions: how I plan to do what I want to do.

Desires are defined as a “potential influencer of conduct”, and other desires need to be taken into account before anything is committed to i.e., once I have decided to try and achieve something, it becomes an intention and I make a reasonable attempt to achieve that intention. However, the semantics of the concepts within the BDI architecture vary between philosophers and also between computer scientists.

A basic architecture consists of separate data structures for beliefs, desires and intentions. This separation is not necessary, however it permits clearer identification of what is happening within the system during implementation and verification [77].

A simple loop of execution for a BDI agent is as follows [77]:

- generate options from event queue;
• deliberate over options;
• update the intentions stack with the selected options;
• execute intentions;
• get new external events;
• drop successful attitudes;
• drop impossible attitudes.

This loop encapsulates Bratman’s notion of practical reasoning [21]: option generation, deliberation, execution, and intention handling. However, it is unclear how the first two steps can be made sufficiently fast to run in real-time. The agent is also operating on a closed set of beliefs, desires and intentions.

BDI fits the proposed problem well because:

• People have beliefs about the environment that affect their decisions (e.g., “The main street is always crowded at lunchtime - I will take another route”);

• People have desires to do something or to visit somewhere. This is more obvious with vehicle travel as people do not drive around the city just for something to do, whereas people will sometimes walk somewhere just because it is there. If people are wandering “just because”, then that is still a desire;

• People have plans or procedures of deciding where to go first, how to get there, and how to create a path to follow;

• If a route is blocked due to congestion or temporary infrastructure, a new plan can be formulated and a new path taken to reach a location.

Prometheus

Prometheus is a methodology developed for specifying agent-oriented software systems. Although there are several design methodologies that could be used for this system [15], Prometheus was chosen because of its maturity (a book [74] was recently published) and because the concepts used in the methodology tie in with the concepts used in JACK Intelligent Agents, the chosen implementation language.
Object-oriented design methodologies are often used as a starting point for agent-oriented design methodologies. The former are not useful for agent-based systems in their current form as they cannot capture many agent behaviours. For example, mental attitudes are absent from OO methodologies as objects do not have attitudes, and communication is usually represented as an arc label. In Prometheus, messages are treated as an object so that they can be reused in different communications.

Prometheus consists of three design phases. The system specification phase contains activities relating to identifying functionalities, inputs, outputs and shared data sources. The agents required and their interaction is determined in the architectural design phase. The internals of agents are designed in the detailed design phase. The resulting design is a combination of forms and diagrams, which clearly describe the percepts, action, environment, agents, capabilities and plans in the system.

Although not tied to any implementation, Prometheus fits nicely with JACK Intelligent Agents. It would not be unfamiliar to those familiar with UML for object-oriented design. By reusing elements of OO methodologies, it is expected that those familiar with UML will not have difficulty in adapting to agent-oriented design [51].

Prometheus is “intended to be useful and usable by industry developers” [74]. The creators quantify this as being well-described and complete so it can be used effectively. Providing and supporting effective tools – such as the Prometheus Design Tool – is also a criteria.

Other methodologies are also at a similar level of maturity as Prometheus, for example Tropos. Tropos has also been developed for BDI agents [59], however it does not appear to be as closely connected to JACK as Prometheus is. The ROADMAP methodology [48], which is based on GAIA [59], includes similar concepts to Prometheus, but also includes a environment model. Although ROADMAP is intended for open systems [48], this model would be useful in defining the environment for the agents to move around in. Again, no guidelines are available on how to translate the environment model into JACK.

**JACK**

JACK is based on the BDI architecture and was purpose-built for simulations, in particular defence simulations. The aim of the package was to develop a stable, lightweight and practical agent-based programming language that would not be superseded quickly and would facilitate further research. It is based on Java with a few syntactic extensions, and compiles to Java code [23]. Java was chosen
due to its widespread availability and acceptance. As the JACK files are compiled into Java before execution, normal Java statements can be embedded in JACK files.

JACK has been used for several applications, mainly within defence, however it has a strong reputation worldwide both in research and industry [1]. The audience for JACK is people with knowledge of agent-based applications, concurrent object-oriented programming and software engineering [23].

JACK supports the concepts in the BDI architecture: agents, events, beliefs (data), capabilities, and plans.

- Agents react to events and execute plans of action. They can communicate with each other and also send themselves events.

- Events are either sent by an agent (to themselves or to another agent) or can be set up to trigger when a belief changes.

- Beliefs store the current knowledge of the agent. They can be accessed logically (similar to Prolog). An agent can store both true and false beliefs.

- Capabilities are the equivalent of packages in object-oriented programming. They permit events, plans and belief definitions to be defined as related to each other.

- Plans are the actions of the agents. The JACK model defines several methods for invoking plans, in particular on failure of a plan.

Many implementation tools exist for multi-agent systems, such as JADE and Zeus. JADE [96] is particularly popular as a toolkit for open agent systems as it is FIPA-compliant. It contains some built-in debugging tools and an agent directory, which makes it easy to find the appropriate agents for a task (similar to a Yellow Pages directory). However no development GUI exists [59]. As simulations are usually closed applications [59] (that is, agents are not joining the system from foreign sources, unlike in a trading or marketplace situation), modellers are not as concerned with making their simulations compliant with standards and do not require features for open applications.
5.3.2 Model development

Overall architecture

Transport systems can be broken down into three main concepts: user, vehicle and environment. The user has a perception of attributes of the environment and their vehicle, and needs to guide their vehicle through the environment. The vehicle interacts with and changes the environment. The environment is constantly updated with the new locations of vehicles and provides perceptions to vehicles and users.

For a driver, the user is the driver in the car, the vehicle is the car, and the environment is the road network and the other vehicles. For pedestrians however, the user and vehicle are essentially the same object: a human. However, most of our walking is done subconsciously and therefore it is permissible to separate these two concepts. The user is defined as the human’s brain and the vehicle as the human’s legs.

Given the high-level decision making role of the user, the belief-desire-intention architecture would be useful for modelling user behaviour in transport systems.

The environment is represented as a graph, rather than the metric representation used in the previous prototype. It uses three of Lynch’s city elements described in Chapter 2: paths, nodes, and landmarks. It is a spatially macroscopic model: the smallest detail of the pedestrian’s path is a link, rather than their steps.

Designing with Prometheus

System specification. The system specification involves identifying system goals and functionalities, developing the interface between system and environment, and developing use case scenarios.

Firstly, the system goals and possible subgoals need to be established. An example of this is shown in Figure 5.15, which shows three goals (visit attractions, arrive at the stadium at a reasonable time, move through the environment) and their subgoals using an oval shape. One of the goals is to move through the environment, with the subgoals of satisfying network constraints and taking a reasonable path.

Goals can then be grouped together to create functionalities. Scenarios can also be developed. These consist of steps such as percepts, messages, goals and actions. The system interface involves determining the actions and percepts of the interface to the environment.

Architectural design phase. The architectural design phase involves grouping the function-
CHAPTER 5. MODEL PROTOTYPES

Figure 5.15: A goal diagram in Prometheus.

Figure 5.16: A coupling diagram in Prometheus.

alities into similar areas, developing agents to control each area and specifying the interactions between agents.

The functionalities can be grouped into areas by investigating the data that is present in the system and which data is required by each functionality. This is then used to determine the agents required and what functionalities they should have. Figure 5.16 shows an example of a coupling diagram. Each data source is represented by a yellow cylinder and the functionalities by rectangles. The arrows signify whether the functionalities read (arrowhead at the functionality end) or write (arrowhead at the data end) data.

As part of this phase, communication between agents can also be specified at a high level.

**Detailed design phase.** In the detailed design phase, the agent’s capabilities, events, plans and data structures are developed in more detail. An example of a detailed design for the Pedestrian agent is shown in Figure 5.17. This shows the events (percepts: stars shapes; actions: arrow
5.3. BDI AGENTS

Figure 5.17: An agent diagram in Prometheus.

shapes; messages: envelope shapes), capabilities (rounded rectangles), and plans that the agent requires. The arrows again signify the incoming and outgoing nature of events.

Capabilities are similar to modules in that related plans, events, and data can be combined together in a coherent manner. In this model the pedestrian has a capability for each of its main activities.

Events can be actions (affecting the environment in some way), percepts (knowledge coming from the environment), and messages (to and from other agents). For each of these concepts, several parameters need to be designed including the information carried by the percept/message, the effect of the action, and what to do in case of failure.

Descriptions of data usage are also required. Plans need to specify whether they are reading and/or writing data. Figure 5.18 shows two plans (p_Startup and p_UpdateEnvironment) that use three data sources (Links, Nodes, and Attractions). p_Startup has write-only access, whereas p_UpdateEnvironment has both read and write access to the data sources.

In this stage, plans are described at a high level, including their name, the percepts that trigger them (as shown in Figure 5.17 by the star shapes) and the actions that occur during the plan. They will be designed in more detail depending on the implementation platform.

The main issue with using an agent-oriented methodology is that it designs the simulation elements only i.e., the agents and what they are doing. It cannot design the core of the simulation i.e., how the clock will tick over, the graphical user interfaces required to set up the simula-
CHAPTER 5. MODEL PROTOTYPES

Figure 5.18: A data diagram in Prometheus.

tion, the methods to collect outputs. Therefore Prometheus needs to be combined with another methodology to design the entire simulation.

Implementation

A prototype model was constructed using JACK Intelligent Agents which involved agents entering a sports precinct and moving towards a stadium. Several “distractions” were located on the way to the stadium, such as food stands and street performers. For this prototype, the entire model architecture (user-vehicle-environment) was implemented in JACK to avoid complex interfacing.

It is straightforward to implement goal-directed behaviour, such as moving towards the stadium. The belief system, however, is similar to facts in a logic programming language such as Prolog and does not handle complex beliefs well. For example, it is difficult to represent an environment in detail using JACK beliefsets. Ideally an interface to the environment should be developed and then any environment format (e.g., graph, cells, shapes) could be used behind that.

The decision-making used in BDI and in JACK cannot elegantly handle continuous events, such as stepping. It is also difficult to define the subconscious decisions behind walking. Therefore the vehicle model in our architecture would be better suited to an object representation rather than an agent one.

An advantage of agent-based technology is that agents are capable of doing several things concurrently without trouble e.g., walking and looking about. So, if you are walking to the post office and you see a shop that has a sale, you continue walking as you make a decision whether to detour or not. If you decide to keep going, you continue with your existing walking plan. However, if you decide to detour, you stop your current walking plan and construct another to get to the sale. JACK had problems interrupting one plan before starting another and in this model, agents
sometimes found themselves in two environmental locations at once. A solution is to increase the lookahead of the pedestrian agent, so that they make decisions and construct new plans earlier.

Reimplementation

JACK supplies several add-on packages, such as JACK Teams which is popular for developing teams of agents. Another package is JACK Sim [2], which adds more simulation functionality to JACK, in the form of discrete-event simulation.

Discrete-event simulation is used for models where the events in the system drive the model. Queues are used to keep track of the events that will happen in the near future. There are commonly three world views of a simulation [11]:

- event, which focuses on the events and their effects on a system;
- process, which models the entities/objects in a system and their lifecycles; and
- activity, which focuses on the activities in a system and when they begin and end.

JACK Sim provides a new world view: the BDI world view [2], where behaviours are defined within agents and the simulation is driven by the JACK model. It appears to be a combination of the process model (agents have a defined lifecycle) and the event model (the agents perceive internal and external events, which cause them to execute plans).

Applications built using JACK Sim use a single clock for the entire application, which is controlled by a TimeSource agent. The agents in the application are allowed to execute until they are all blocked. Each agent is managed by a TimeDispatcher agent. The TimeDispatcher’s role is to let each agent execute and then let the TimeSource know the next time one of its agents is ready to execute again. The TimeSource advances the clock to the earliest time it receives from the TimeDispatchers and another round of execution begins. The clock stops when there is no more activities to be scheduled. Not all agents in the simulation have to be controlled by a TimeDispatcher.

There are three phases to a JACK Sim program. During Creation, the agents are created, initialised, and registered with a TimeSource agent. Setup allows for developer-designed setup to take place. Finally, the Execution phase steps through time until there are no events left to execute. The start of the Setup phase and the start and the end of the execution phase are signalled with an event. The process is shown in Figure 5.19.

\footnote{The code for this prototype is available on request from the author. The user will require a current version of JACK Intelligent Agents (available from \url{http://www.agentoriented.com/}).}
The JACK Sim packages solve the plan interrupting problem from using JACK alone. In JACK Sim the execution time of the plan is irrelevant – the clock is stopped until a block is reached in the plan.

For this version, a prototype was constructed of pedestrians moving around a network, represented as a directed graph. Entry and exit points to the environment and the number of people using them was specified, as well as locations within the model that a certain proportion of the model population needed to visit. No time constraints on the pedestrians’ planning were implemented.

Two pedestrian flavours were implemented based on route choice planning theory described in Section 2.3.5. Familiar pedestrians knew their route through the environment and are based on the simultaneous planning described by Bovy and Stern [19]. It was most likely the shortest and they were unlikely to deviate from it. For this model, each pedestrian retrieved their route from a central database containing the shortest routes between two points based on distance. This could be enhanced by randomly altering the path returned. Some adaptive planning ability was also added, so if the pedestrian found they were going too slowly, they would replan their route based on real-time time estimates between their current location and their destination.

Unfamiliar pedestrians have no knowledge of the environment and are based on the sequential
route planning described by Bovy and Stern [19]. They are given an origin, a destination, and possibly locations in-between to visit. They select a link to travel on, and at the end of that link select the next link. The next link is selected either randomly or as provided by the central database.

Reaching a node is an event for the TimeSource agent. Every time an agent reaches a node, they calculate their next route and inform the Environment agent. The Environment agent calculates how long they will take to travel along the link based on the speed and the current density of the link. The calculation is based on the speed-density curves in Austroads’ *Guide to Traffic Engineering Practice* [7]. The curve is based on the space per pedestrian or inverse density, as opposed to the number of pedestrians in an area. The curve was originally developed by Fruin [34], who stated that the density is the most significant factor in determining walking speed. The equation for this curve was estimated in Equation 5.4 and is shown in Figure 5.20. In order to keep everyone moving, the minimum speed was set to 0.1 m/s. At initialisation, each pedestrian is assigned a maximum speed from the normal distribution with the mean 1.35 m/s and standard deviation 0.3.

\[
v = 1.4 - \frac{0.35}{\text{space}} \quad (m/s)
\]

(5.4)

An issue with the current implementation is that the time spent on the link cannot be changed mid-link, for example, if crowds begin to clear and the pedestrian speeds up. A solution to this is
to use smaller links.

**Deployment**

JACK Sim controls the simulation using an input file that contains a list of agents to instantiate. Agents who respond to a SETUP event will execute some initial plans before the BEGIN event is sent.

The input file contains the names of the input and output files, the warmup period (during which all pedestrians enter the environment using a uniform distribution) and a seed. The model will run until there are no more events to handle.

Unlike the CA model, where a for loop is used to set up the required number of pedestrians, all pedestrians need to be explicitly named. A program was developed that creates this file given a filename and the number of familiar and unfamiliar pedestrians.

**5.3.3 Operational assessment**

Some runs of the prototype were made in order to explore the behaviours and some sample graphical output is shown in Figure 5.21. The familiar pedestrians, as expected, all took the same paths through the environment. The unfamiliar pedestrians introduced some variability into the model.

The adaptive behaviour was raised when the agent dropped below a certain speed as expected, however due to the number and sameness of the familiar pedestrians, the real-time path was rarely different to the original path based on distance. This could be improved by creating individual representations of the environment graph for each pedestrian, with different opinions of travel time.
Figure 5.21: A example of the environmental output from the BDI prototype. Each path is represented as a link between two numbered nodes. Square nodes are entries/exits. Thicker lines and different colours mean that the level of service (density) is lower (higher) on those links.
Chapter 6

Model evaluation and discussion

In the previous chapter, two prototypes of pedestrian models were described. The first prototype employed agent-based cellular automata, which is a popular approach based on the literature for explorative models of behaviour. The second prototype used BDI agents and was implemented using JACK Intelligent Agents, an agent-oriented programming language.

Following the development of the prototypes, it is necessary to assess the processes and the products to explore the usefulness and development of agent-based models for pedestrian modelling.

In this chapter, the two prototypes will be evaluated using software engineering quality principles. The criteria will be defined before applying them to the prototypes. A discussion of two other simulation languages, with reference to the criteria, is presented. The two prototypes are also evaluated for their applicability to pedestrian modelling. As the experiments for the prototypes concerned different applications, the prototypes are not directly comparable and are evaluated separately.

6.1 Evaluation criteria

Software quality standards measure functionality, reliability, usability, efficiency, maintainability, and portability characteristics to determine the quality of a finished product [100]. However, for this project, only prototypes were developed and the quality criteria will take this into account.

Little research has been undertaken into evaluating tools that can be used for a specific activity, such as creating pedestrian models. A series of articles in the late 1990s in Software Engineering Notes looked at different methods for qualitative evaluations for deciding whether a tool was
appropriate. The case study methodology [53] will be adapted for this evaluation. This involves testing a method or tool on a real project and developing a lightweight assessment of the method or tool. However, only one evaluator will be used, therefore removing the variable of different evaluators.

Several sets of desirable features for models have been published. Axelrod [8] describes validity, usability, and extendability as goals of a simulation model. Law and Kelton [55] describe a wider range of features. Finally, Tobias and Hofmann [97] evaluate several libraries for agent-based social simulation and some of their criteria will be incorporated into our set of criteria.

6.1.1 Functionality

Functionality is “the degree to which the software satisfies stated needs” [76]. The following criteria will be considered:

- reasonable number of agents generated (Law and Kelton [55] prefer maximum model size);
- presence of an appropriate agent communication process;
- ease of generating agents;
- ease of generating networks;
- whether spatial location of agents is catered for;
- ease of input;
- provision for statistical output [55];
- support provided for modelling and simulation, such as timing and repeatable experiment functionalities.

6.1.2 Usability

Understandability, learnability, and operability are the subattributes of the usability criteria [100]. Understandability looks at the support provided and required in order to use the product, in particular the presence of appropriate documentation and other forms of support (e.g., mailing lists) for the intended user base [55]. This involves identifying the target audience.

Learnability looks at the time required to understand principles and to learn the product, using the provided resources and other materials.
6.1. EVALUATION CRITERIA

Operability looks at the programming/software engineering knowledge required and the software required to operate the product. This will involve an assumption about the appearance of the final product for the prototypes.

6.1.3 Efficiency

The execution speed of the model is important [55]. Large models are computationally intensive and from my experience, modellers do not have a supply of powerful computers purely for the purpose of model runs. The speed of the prototypes for different networks and population sizes will be explored.

6.1.4 Maintainability

Axelrod [8] mentions extensibility as a goal for simulation models, because this allows other researchers to adapt the model to explore new features and variations. Modularity and extensibility will be the main focus of this criteria.

6.1.5 Portability

For this evaluation, portability will cover the ability to use the product on different platforms with no change in results. This is important for repeatable simulations. L’Ecuyer [56] defines repeatability as “the ability to replicate exactly the same sequence of random numbers”. This assists in the verification step and also it ensures that different scenarios can be run with the same parameters.

6.1.6 Discussion of criteria

Reliability is excluded from this set of criteria as it relates to product performance, which is inappropriate for prototypes. However, some performance criteria is included in the efficiency criteria.

Some criteria are not applicable as only prototypes are being evaluated. Some parts of the prototypes have been profiled and their performance improved (for example, the collision resolving in the CA model), however there are probably other components that could be improved for performance. As final user interfaces have not been fully developed, expected interfaces have been discussed.
Only one evaluator is employed for the evaluation, which removes the variable of multiple evaluators. However, it means that only one point-of-view is present.

6.2 Functionality

Functionality usually refers to the end product, not prototypes. However, some assumptions about the functionality of the end product will be included here.

6.2.1 CA model

CA models typically suffer when the number of agents or the environment size is increased. The maximum number of agents is small (4000-5000), as a large amount of storage is required for the agents and the environment and the model frequently runs out of memory.

In this model, no communication process is required. All “communication” occurs between each pedestrian and the model controller. The pedestrians are told when to move. Any disputes over collisions are broken by the system.

The agents are generated using a for loop for the required count and instantiating a Pedestrian object.

For networks without blockages, no input file is required. The size of the environment is entered on the command line. If blockages are required, a GUI was created so that the user could click on the cells to be blocked. This was then saved to file and used as input.

The spatial location is explicit for agent-based CA models. All agents must have a location or be outside the model. The locations in this model were referred to by a cell number, due to the different environment types, however in most models agents would contain an xy coordinate.

The input and output were provided in custom text files. The output can be customised in the Simulation object, providing flexibility for the modeller. The movement of pedestrians can be visualised using a GUI.

Several output functions are included in the Update package, including entropy calculations and lane counts. The user can also add their own.

In CA models, timing is explicit and time-based, i.e., the model steps through each time step from zero to the determined finish point. No time is skipped. Seeds are also present in the model, allowing for repeatable simulations.
6.2. FUNCTIONALITY

6.2.2 BDI model

The model appeared to handle 12000 agents in a simple environment, however the model was running at a fraction of real-time. Approximately 7000 agents could be created for the model to run close to real-time.

The JACK platform provides communication functionality. However, in order to use it, an agent needs to know the name of the agent they want to communicate with. Unlike other toolkits such as JADE, no WhitePages or YellowPages agent is provided. However, the developer could implement one if required. No broadcast capability is provided either, requiring a dedicated agent to be developed. This would be useful to communicate with a group of pedestrians on a particular link.

In JACK, agents are easy to generate – the agent is constructed and then a user-defined start() method is called. In JACK Sim, the agents are created from an input file. The only parameters allowed are the name and type of the agent. The number of agents is also explicit. For this simulation, a program was required that created a file with the required number of agents to be read in as input.

JACK does not provide explicit functionality for keeping track of an agent’s spatial location. This is also noted as a property of agent-based models in the geosimulation framework in Chapter 3.

The input and output are developed by the user. For this prototype, the input was created in XML format. As Java provides XML support as part of its API, it is easy to set up. An additional feature of this support is the ability to validate a file against a DTD. From my experience, missing fields are often a problem with complex input files. The XML file itself can be edited with a text editor or an XML editor. The text-formatted output is used by a separate Java application to visualise the environment over time. If a standard DTD was introduced for transport output, then the model could output in the correct format for a dedicated graphical program to visualise.

A GUI was created that showed a static picture of the network, but did not allow editing of the network. The latter functionality would be a requirement for a final system, as well as The ability to read from a CAD drawing or GIS database.

Timing and clocks are explicit in JACK and JACK Sim. In JACK Sim, the timing is more transparent as it is managed by TimeSource and TimeDispatcher agents. Both event-based timing and real-time timing are available, however due to the temporal duration experienced in transport models, real-time clocks are inappropriate and were not used. JACK Sim creates repeatable
simulations and does not use seeds, so the developer must construct seeds explicitly for variable simulations.

6.3 Usability

6.3.1 CA model

For the CA model, the developer needs to know Java and understand object-oriented principles in order to customise the model to their needs. There are plenty of resources to assist in learning Java, including the freely-available Sun Java Tutorial [92].

Two user interfaces were provided to the model. The first enables the user to select blocked cells and save the environment model. The second shows a “playback” animation, which the user can step through or jump to particular points. Both interfaces allow the user to capture the current view as an image for including in reports.

6.3.2 BDI model

To use JACK, the developer needs to know Java and have a good understanding of agent concepts. Agent-Oriented Software provides documentation for JACK and JACK Sim and simple tutorials for JACK. A tutorial for the more complicated tasks in JACK Sim would be useful, in particular custom agent initialisation using JACK’s object modelling tool JACOB. A mailing list for JACK is also hosted and regularly used by AOS staff. Their description of their intended audience is accurate, as this application would be difficult to use for non-software engineers with little knowledge of agents. Some Javadocs for the API are provided.

For Prometheus, a basic understanding of agent concepts and and understanding of software design is required. Padgham and Winikoff [74] is a valuable resource and the Prometheus Design Tool is useful. It is regularly maintained and updated.

To use Prometheus, a Java Runtime Environment (JRE) is required which is common on most systems. For JACK, the Java Development Environment (JDE) and the JACK API are required. JACK is available free for either a trial or for academic use, however for other uses a commercial licence is required.

For the BDI prototype, two user interfaces were provided. The first displayed an overview of the environment. This would be enhanced so that the environment could be edited. The second displayed the output of the simulation. For ease of use, another user interface that assists in
For ease of development, a graphical user interface can be used to create JACK programs. However, this needs to be used from the start of the project, as it is not possible to import existing code. I found it easier to use a text editor or Eclipse to develop code as that was how I had learnt to use JACK and I found it difficult to create the code I wanted to using point-and-click techniques.

6.4 Efficiency

6.4.1 CA model

The CA prototype was profiled using the Eclipse profiler\(^1\). Three different environment sizes with three different densities were run with different seeds. The run time was measured and a snapshot of the profile data was saved.

The profile data showed the functions where the most time was spent during each run. For low densities (10%), the model spent most of the time in evaluating rules and retrieving cell and environment properties. For higher densities (30%), the model spent almost a third of its run time resolving collisions. For densities higher than 50%, the profiler suffered integer overflow and could not provide correct results.

The relationship between density and run time is polynomial. This can be attributed to the increased number of possible collisions (Figure 6.1).

Comparing the change in environment size with the same density, the relationship between environment size and run time is linear (Figure 6.2).

As mentioned earlier, the code for this prototype has not been fully optimised. Other projects use parallel computing approaches: spatial, where the environment is decomposed into smaller areas [66], and temporal, where the duration of the simulation is divided up [50]. The latter is problematic as the state of the world at the beginning of each timeslice is unknown.

6.4.2 BDI model

For this prototype, approximate timing was undertaken. Times were taken in milliseconds at five points: before the Java interpreter is invoked, when the setup phase begins, when the simulation

---

Figure 6.1: The execution times and linear fits for different environment sizes.

Figure 6.2: The execution times and linear fits for different densities.
Figure 6.3: The different execution times and the fits for each of the BDI model stages.

actually starts, when the simulation ends (before outputs are generated) and after the model is exited. Each scenario was run three times with three different seeds.

One random environment and four pedestrian volumes were used. The results (Figure 6.3) show that the model operates in linear time with respect to the number of agents initialised for the setup, simulation, and output stages of the model. This is to be expected as there are no complex interactions between agents. The startup phase was estimated to be closer to a hyperbolic function.

6.5 Maintainability

Two features of object-oriented programming are that it is modular and extensible [76]. In Java, classes can be organised into packages that contain all the objects and methods required for a particular task. The package can then be extended as necessary, as long as the public operation contract is not broken. Java interfaces are useful for defining the public operations required from an object.

The CA model provides the base classes to the user who must extend them to create a model.
The user can then define their own behaviours, properties, and collision rules.

JACK inherits these properties from Java. It also adds the concept of capabilities, which define a set of events, plans and beliefsets that are required to perform a task.

### 6.6 Portability

A key feature of Java is its portability [39]. The JACK framework is also available for several platforms.

The random number generator provided in the Java API uses a linear congruential pseudorandom number generator and will return the same series of numbers given the same seed. This is important for repeatable experiments.

The JACK Sim framework is also guaranteed to be repeatable for one agent only. For more than one agent, multiple threads are required and the order of events cannot be guaranteed. However, from the runs undertaken for the efficiency discussion, it appeared that the system was producing the same results for the same seeds.

### 6.7 Other simulation languages

Before exploring the usefulness of the models, the above criteria will be applied to two other agent-based simulation languages, Repast and Swarm, to see how JACK Sim compares with similar products. The analysis is based on the available documentation for the two languages.

#### 6.7.1 Repast

The aim of Repast is to develop “extremely flexible models of living social agents” [78]. It supports the development of agents and their behaviours. Models can be run in interactive or batch mode and it is easy to request multiple runs with different parameters.

Several libraries for the development of behaviours (e.g, genetic algorithms and neural networks) and domains (e.g, geographic information systems, game theory) are provided. The toolkit provides support for discrete event simulation. Statistical output can be collected by “loggers”.

The toolkit is available in three implementations. The Java version is portable across the platforms where a JRE is available. The .NET version provides multilingual functionality, as all the languages in the .NET framework can be used to develop models. A Python implementation is also provided, which uses a point-and-click interface to develop simple Repast models.
The toolkit supports many standards, including FIPA standards [70]. Tutorials, sample code, FAQ, and mailing lists are provided to users. The Javadocs of the Java version are also available. The toolkit can produce graphs, reports, and QuickTime movies.

### 6.7.2 Swarm

Swarm is an agent-based modelling platform that is used for designing, describing, and conducting experiments with agent-based models [93]. The original implementation was in Objective-C, an object-oriented language, and a Java wrapper for this implementation has been developed. The platform can be used on all major operating systems.

Swarm provides for nesting swarms of agents in a discrete-event environment. No environmental representations are available in the model [64].

Several forms of documentation are provided, in the form of mailing lists, FAQ, tutorials, user guides, demonstration models, contributed code, and Javadocs [93]. A tool for “probing” the simulation as it runs is provided. Graphs of the models can be retrieved. Support for Tcl/Tk user interfaces is also provided.

Swarm provides a library with many random number generators and random distributions. All have been tested and a discussion on the merits of different generators is included in the user guide [93].

### 6.7.3 Comparison to JACK Sim

Repast and Swarm are more mature than JACK Sim, so it is reasonable to expect that the former languages have more functionalities. Both projects are also open source and have large user bases, who are happy to share sample projects and libraries.

JACK Sim contains a similar amount of documentation to the two projects, except for a dedicated tutorial. The main omissions from JACK Sim are the domain-specific libraries and functionality for creating graphs of results. The intention would be for the users to develop this in Java, as it is expected that JACK users have extensive software engineering experience. JACK Sim only provides the simulation control and the language constructs for creating agents [2].
6.8 Usefulness

Ideally, to determine the usefulness of a model, it should be validated against the real world and other models. Several validation techniques are applicable, including expert observation of animations and statistical comparison of volumes and flows [87]. Unfortunately data were not available for validation of these prototypes.

For the CA prototype, there were no problems with replicating cellular automata theory when using a general purpose language. Behaviours developed by other researchers could be added easily. The use of CA is a neat abstraction of the way we walk and the incorporation of the behaviour is an advantage. However the CA framework cannot handle higher-level behaviours, such as activity planning.

The type of environment chosen for this model was an example of a small-scale enclosed space, as described in the framework in Chapter 4. Simple bi-directional corridors can be represented easily in a CA model.

JACK is a good representation of the BDI framework and although JACK itself has been used for simulations, the JACK Sim add-on provides more simulation capability. This model is the opposite of the CA model in several ways. Although the JACK package provides for beliefs, these are difficult to use for large and complex data sets, such as an individual representation of the environment, and the spatial location of the agents is not explicitly provided for. However, the model proved to be more useful for high-level planning than low-level movement. Several simple behaviours from Bovy and Stern [19] were able to be replicated.

The type of environment used for this model was an example of a mixed mode or open space environment. The graph structure provided a concise representation of this large area and path planning could be undertaken. A metric representation of the environment is required if finer details of movements are required.

In Section 4.2.1, the requirements of practitioners and clients for models was discussed in the context of the framework. The client’s needs are an understanding of the model scope and the ability to view model results in a variety of formats. The practitioner’s requirements are:

- ease of use;
- a clear understanding of parameters;
- the ability to modify parameters quickly and easily;
- the ability to read in environment data from various sources, including temporal constraints;
• flexibility with output;

• a reasonable running time.

The CA model meets these requirements with the exception of reasonable running time.

The BDI model is a lot closer to meeting these requirements. The ability to modify parameters easily is met, due to the separation of the scenario from the code. However a deep understanding of agent theory, programming, and the JACK package is required.
Chapter 7

Conclusions and further work

This thesis has investigated the early stages of the software development process for agent-based models of pedestrian behaviour, with an emphasis on their applicability and usefulness. The three processes that play a role in the development of pedestrian models – planning, modelling, and software development – provided a framework for the thesis.

Pedestrian behaviour is more unpredictable than other transport modes and consists of both conscious and unconscious movement. This means that classical transport modelling techniques, such as the four-step model, are sometimes inappropriate. Disaggregate models that treat each pedestrian as an individual are more commonly used.

The existing approaches to pedestrian modelling – microsimulation, mathematical, and agent-based – were reviewed and a framework developed to assist in the choice of approach for a given project. As pedestrian-oriented projects are becoming more popular, it will be necessary to undertake modelling as part of the planning process. There is little research into how people select the more appropriate tool for the job, but both formal and anecdotal evidence point to the tool that the developer is the most familiar with. This, however, may not be the best tool for the job. In order to provide extra assurance to the client that the consultant’s recommendations are relevant, a framework is required that works through a series of steps, uses common concepts, and is clear to all stakeholders. This also assists stakeholders in determining the scope more accurately earlier in the project, as they have the capabilities, advantages, and disadvantages of each approach.

Two prototypes of agent-based approaches – agent-based cellular automata and belief-desire-intention agents – were developed in order to explore the early stages of the software development process. Each prototype was evaluated using software engineering quality principles. The CA
prototype was easy to set up and understand and this is confirmed by the number of models using this technique. It is particularly suited to modelling detailed movement because of the explicit spatial representation. However, the lack of scalability causes problems for larger simulations and high-level behaviour, such as route choice, requires additional non-CA modules. Other research is exploring the use of parallel computing to increase efficiency. The use of a general-purpose language means that the model is constrained only by the language, however the developer must implement simulation-specific functionality.

The BDI prototype was more difficult to design and implement. The spatial location of the agents must be defined by the user. The diagrams and tools used in the Prometheus design methodology would not be unfamiliar to UML users, however some of the concepts require knowledge and understanding of agent concepts in order to create an effective design. JACK Intelligent Agents quite correctly states its audience to be software engineers with agent-oriented knowledge, however this precludes modellers and planners using the software to develop models effectively. Using a simulation language means that several simulation concepts are transparent to the developer, such as timing systems, and more time can be spent developing the agents behaviour. However, time needs to be spent understanding how the built-in concepts work, so that they are used properly. The BDI approach is useful for modelling human behaviour, but lack of knowledge about how pedestrians move at a low level means that it is limited to high-level movement and planning.

It is a mistake to believe that agents are the solution to many problems and people sometimes misunderstand what they are capable of. Agents are a design metaphor that is particularly useful for describing human behaviour. Ultimately, the success of the model depends on defining the stakeholders’ roles to avoid misunderstanding and choosing an appropriate approach for the task. Both of these involve training, communication, and experience.

### 7.1 Further work

As stated in the aim, this project was limited in scope by resources. If more resources were made available in the future, there are several opportunities for extensions and further work.
7.1. Framework

The framework requires validation with industry. This could be achieved by having readthroughs and using the experience of various consultants and planners. A large-scale study could also be used to test the success of using the framework on projects.

7.1.2 Agent architectures

The two approaches prototyped fell at either end of the reactive-deliberative scale. Hybrid architectures that create both reactive and deliberative behaviour also have potential for modelling spatial behaviour. An example is INTERRAP, where the agents consist of vertical layers representing behaviour, plans, and cooperation [59]. Reactive behaviour is generated by the behaviour-based layer. Events that cannot be responded to reactivity pass to the plan-based layer and possibly to the cooperation layer. Both reactive and deliberative behaviour are clearly present.

7.1.3 Model development

There is little research available on creating and evaluating agent-based models from a software engineering perspective. The main focus is on creating models that output the “right” result, rather than ensuring the model is robust and usable. As more projects require custom software solutions, the software development phase will become more important in the modelling process.

As discussed in Chapter 5, parts of the software development process are being adapted for agent-oriented software engineering. Lind [58] has developed some support for agent design patterns. In the methodologies reviewed in Luck et al. [59], none covered the verification or testing of agent-based software. Verification is difficult, especially for open and non-repeatable systems, however it is an important requirement for valid simulations. Poutakidis et al. [75] are currently developing a debugging tool for JACK.

Extensions to possible behaviours exist for the CA model. For example, pedestrians could be given the option to step backwards or not move at all in a timestep.

Due to the scope limitations on this thesis, the prototypes were unable to be validated as data are required to properly validate the model. If data were available, it would be particularly useful to compare with the modelled behaviour, especially the BDI model. The development of more low-level behaviour is certainly of interest and the implementation of more of the route choice theory from Bovy and Stern [19] is also possible.
Bibliography


Author/s:
RONALD, NICOLE AMY

Title:
Agent-based approaches to pedestrian modelling

Date:
2007-02

Citation:

Publication Status:
Unpublished

Persistent Link:
http://hdl.handle.net/11343/39212

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.