A FRAMEWORK FOR SOFTWARE MODELLING IN SOCIAL SCIENCE RESEARCH

by

Piper J. Jackson

B.Sc., Simon Fraser University, 2005

B.A. (Hons.), McGill University, 1996

a Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the

School of Computing Science

Faculty of Applied Sciences

© Piper J. Jackson 2013

SIMON FRASER UNIVERSITY

Summer 2013

All rights reserved.

However, in accordance with the Copyright Act of Canada, this work may be
reproduced without authorization under the conditions for “Fair Dealing.”
Therefore, limited reproduction of this work for the purposes of private study,
research, criticism, review and news reporting is likely to be in accordance
with the law, particularly if cited appropriately.
APPROVAL

Name: Piper J. Jackson
Degree: Doctor of Philosophy
Title of Thesis: A Framework for Software Modelling in Social Science Research

Examining Committee: Dr. Steven Pearce
Chair

Dr. Uwe Glässer
Senior Supervisor
Professor

Dr. Vahid Dabbaghian
Supervisor
Adjunct Professor, Mathematics
Associate Member, Computing Science

Dr. Lou Hafer
Internal Examiner
Associate Professor

Dr. Nathaniel Osgood
External Examiner
Associate Professor, University of Saskatchewan

Date Approved: April 29, 2013
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the “Institutional Repository” link of the SFU Library website (www.lib.sfu.ca) at http://summit/sfu.ca and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author’s written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2011
Abstract

Social science is critical to decision making at the policy level. Software modelling and simulation are innovative computational methods that provide alternative means of developing and testing theory relevant to policy decisions. Software modelling is capable of dealing with obstacles often encountered in traditional social science research, such as the difficulty of performing real-world experimentation. As a relatively new science, computational research in the social sciences faces significant challenges, both in terms of methodology and acceptance. However, there is great potential for computing to aid in the application of scientific thinking to the grave issues facing society. This is particularly true since technological advances and societal change continue to make our lives more complex. Policy decisions can have significant impacts in the lives of those affected; it is imperative we strive to develop novel and effective methods to inform these decisions.

This thesis focuses on the interaction of modelling, software development, and experimentation in computational social science research pursued by small teams of interdisciplinary scientists. I present an innovative software development framework designed for this kind of research. By integrating software throughout the research process for both modelling and experimentation, and utilizing a flexible and iterative development model, my framework addresses many pressing issues of computational social science: uncertainty due to lack of data or changing conditions; validation of models; usability; rapid adjustment to changes in direction; facilitating collaboration; and communication of results to peers and stakeholders. Case studies of projects developed using this software modelling framework are used to illustrate and discuss the approach. The case studies span several fields of the social sciences, including Criminology, Geography, Political Science, and Public Health.

**Keywords:** Abstract state machine; computer simulation; interdisciplinary research; mathematical modelling; social science; software development
To Marshall, and bright futures
“Clay is formed into vessels;
    from their emptiness
    the vessels are useful.”

— DAO DE JING
I would like to offer my sincerest thanks to my senior supervisor, Uwe Glässer, for his guidance and support over the years of my graduate studies at Simon Fraser University. I am truly indebted to your help and patience as I struggled to find my own area of expertise in computing science. It has not been easy to navigate the many promises and challenges of open research fields, and your positive demeanour and wealth of knowledge have been vital in passing through this stage of my career. I also have my other supervisor, Vahid Dabbaghian, to thank for guidance and support. Without your insistence that I remain active in Modelling of Complex Social Systems program, I doubt this thesis would have come to pass. Thank you both very much.

I owe a great deal of thanks to my colleagues on the projects I have participated in. It was an eye-opening experience working with you, both because of the fascinating knowledge contained in each area of expertise, but also from becoming aware of the differences and assumptions in our various fields. I am happy to say that collaborating with you has been a major source of inspiration for my research. Chris Bone, Patricia Brantingham, Egor Chalubeyeu, Roozbeh Farahbod, Philippe Giabbanelli, Christopher Giles, Jordan Ginther, Niki Huitson, Eric Joel, Ali Khalili Araghi, Bryan Kinney, Vijay Mago, Hilary Kim Morden, Simon Pratt, Andrew Reid, Valerie Spicer, Mona Vajihollahi, Hans Wehn, Kathryn Wuschke, and Hamed Yaghoubi Shahir: it was a pleasure to work with all of you.

Finally, I would very much like to thank my wife, son, friends, and extended family for their love and support through this process. I would especially like to thank my wife for her patience and hard-work during times I have been absent due to research. You have my heartfelt gratitude.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval</td>
<td>ii</td>
</tr>
<tr>
<td>Partial Copyright License</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Dedication</td>
<td>v</td>
</tr>
<tr>
<td>Quotation</td>
<td>vi</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>vii</td>
</tr>
<tr>
<td>Contents</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xiii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Contributions of the Thesis</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Structure</td>
<td>5</td>
</tr>
<tr>
<td><strong>I Foundations</strong></td>
<td>7</td>
</tr>
<tr>
<td><strong>2 Computational Social Science</strong></td>
<td>8</td>
</tr>
<tr>
<td>2.1 Modelling–General Issues</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Models of Social Systems</td>
<td>10</td>
</tr>
</tbody>
</table>
2.3 Verification and Validation of Simulations

3 Software Technology

3.1 Software Development

3.2 Formal Methods

3.2.1 Abstract State Machines

3.2.2 CoreASM

3.3 Reproducibility

4 Scientific Positioning

4.1 Falsifiability

4.2 Scientific Progress

4.3 Post-Normal Science

4.4 Uncertainty in Science

4.5 Generative Science

4.6 Participatory Action Research

II Framework for Software Modelling in Social Science Research

5 Framework Principles

5.1 Characterization of Target Problems

5.2 Exploratory Software Modelling Research

5.3 Intelligence

5.4 Communication

5.5 The Roles of Software

6 Framework Elements

6.1 Iterative Development Cycle

6.2 Software Features

6.2.1 Feedback

6.2.2 User Interface

6.2.3 Portability

6.3 Collaboration

6.4 Clarity
III Case studies

7 Agent-Based Modelling Projects

7.1 Introduction: Agent-Based Modelling

7.2 The Mastermind Project

7.2.1 Conceptual Modelling

7.2.2 Mathematical Modelling

7.2.3 Computational Modelling

7.2.4 Experimental Set-Up

7.2.5 Execution

7.2.6 Results

7.2.7 Iteration

7.3 SocialMind

8 Cellular Automata Projects

8.1 Introduction: Cellular Automata

8.2 Urban Migration

8.2.1 Conceptual Modelling

8.2.2 Mathematical Modelling

8.2.3 Experimentation

8.2.4 Iteration

8.3 Urban Migration–Vancouver

8.3.1 Conceptual Modelling

8.3.2 Mathematical Modelling

8.3.3 Computational Modelling

8.3.4 Experimentation

8.3.5 Iteration

8.4 Binge Drinking

8.4.1 Modelling issues

8.4.2 Conceptual Modelling

8.4.3 Mathematical Modelling

8.4.4 Computational Modelling

8.4.5 Experimentation

8.4.6 Iteration
9 Fuzzy Conceptual Map Projects

9.1 Introduction: Fuzzy Cognitive Maps .................................. 115
9.2 Insurge ................................................................. 116
  9.2.1 Conceptual Modelling ........................................... 117
  9.2.2 Mathematical Modelling ........................................... 120
  9.2.3 Computational Modelling ........................................... 124
  9.2.4 Experimentation .................................................. 125
  9.2.5 Iteration .......................................................... 126
9.3 Obesity FCM ............................................................ 128
  9.3.1 Conceptual and Mathematical Modelling ..................... 128
  9.3.2 Computational Modelling and Experimentation ............... 129
  9.3.3 Comments ........................................................ 131
9.4 Crowd Dynamics .......................................................... 131
  9.4.1 Conceptual Modelling ........................................... 132
  9.4.2 Mathematical Modelling ........................................... 133
  9.4.3 Experimentation .................................................. 133
  9.4.4 Iteration .......................................................... 134

10 Conclusions ............................................................... 141

10.1 Limitations ............................................................ 142
10.2 Future Work ........................................................... 143
10.3 Achievements .......................................................... 145
10.4 Final Words ............................................................ 146

Bibliography ................................................................. 148
List of Tables

8.1 Neighbourhood factor by zoning district type. . . . . . . . . . . . . . . . . . . 98
8.2 Sensitivity to social influence factor. . . . . . . . . . . . . . . . . . . . . . 101
List of Figures

2.1 Rossiter, Noble, and Bell’s model-centred epistemology of social system simulations. ................................................. 14
2.2 Sargent’s relationships between the real world and a simulation world with regard to verification and validation ......................... 15
4.1 The iterative flow of participatory action research. ......................... 32
5.1 Boyd’s OODA Loop ............................................................... 42
6.1 The project cycle, showing standard modelling progress, and typical paths of model change.............................................................. 48
6.2 Binge drinking cellular automata program: parameter set-up. .......... 52
6.3 Binge drinking cellular automata program: simulation playback. ...... 53
6.4 Binge drinking cellular automata program: population visualization. .... 54
6.5 CSDe: A control state diagram editor plugin for the Eclipse development environment, with automatic translation to CoreASM code. .......... 59
7.1 Java implementation of Mastermind. ............................................. 65
7.2 CoreASM implementation of Mastermind. ...................................... 66
7.3 Road network in SocialMind. ......................................................... 69
7.4 Social connections in SocialMind. .................................................. 70
8.1 System-wide process model of a single time step. ............................. 80
8.2 Decision making process of an individual household. ....................... 82
8.3 Moore neighbourhood superimposed on households ......................... 83
8.4 No attractors versus positive or negative attractors. ...................... 88
8.5 Low versus high occupancy. ...................................................... 90
8.6 Low versus high neighbourhood influence. 92
8.7 Vancouver zoning districts, 2009. 95
8.8 Dwellings occupied by usual residents, Vancouver, 2006. 96
8.9 Urban Migration simulation program using real geographic data. 99
8.10 The simulation run with social influence factor set at 0.1. 102
8.11 The simulation run with social influence factor set at 0.001. 103
8.12 Model of drinking transitions. 108
8.13 Binge drinking cellular automata. 111
8.14 Binging cellular automata experiments on population distributions. 112
8.15 Binging cellular automata experiments on influence strength. 113

9.1 Insurgency fuzzy cognitive map. 119
9.2 Insurgency CA cell interactions. 122
9.3 Fuzzy triangular membership functions. 123
9.4 The insurgency simulation program. 126
9.5 The initialization provides each individual with an instance of the FCM. 129
9.6 Fuzzy cognitive map of obesity. 130
9.7 Crowd simulator execution. 135
9.8 Crowd simulator results: FCM values over time. 138
9.9 Crowd simulator results: population levels over time. 139
9.10 Crowd simulator results: average behaviour over time. 140

xiv
Chapter 1

Introduction

With the emergence of cheap and powerful computation, social scientists have started to explore the potential of applying computational to their research topics [81]. Hummon and Fararo claim that the traditional two component model of science–theory and empirical research–needs to be expanded to include computation as its third component. Simulation can be thought of as the interaction of theory and computation components. High-level programming code, with the capacity to clearly represent both entities (data structures) and behaviour (algorithms), provides a conceptual link between the underlying mathematics of the simulation model and researchers’ understanding. Being able to test theories using mathematics promises insight formerly limited to the physical sciences, but it is important to remember that social structures are always at least partly interpretive in nature, since they are constructed and maintained by human activity. Gilbert claims that this is not problematic, and indeed may even be more faithful a representation than simulation of physical phenomena: a translation from a social construct to a computational construct is likely to be less problematic than translating something in the real world into a constructed computer representation [64].

Early attempts from the sixties and seventies have been followed by numerous and increasingly complex attempts to construct faithful copies of real-world social interactions using program code on a computer. Human beings are highly social animals interacting with and dependent on each other on many different levels (family, friends, business transactions, politics), it makes sense that we would use computers to explore these properties, in much the same way that we use language and reason to discuss them, and writing and other technologies to store and transmit such discussion. And yet, computational models of social
systems have yet to be accepted as a mainstream tool for making decisions based on our understanding of such systems.

The social sciences include Anthropology, Criminology, Economics, Education, Geography, Law, Political Science, Psychology and Sociology, among many other disciplines. Each of these attempts to collect observations and produce general insight into the relevant aspect of human society. The universal laws found in the natural sciences are exceedingly difficult to establish in applications of social inquiry, due in part both to the slipperiness of human behaviour in terms of evading definition and the uniqueness of each individual’s situation and personality. Still, the subjects of the social sciences are the forces and entities which fill our social life, including both the people we interact with and the traditions and institutions that mediate that interaction. We have no lack for motivation for understanding these things. Further, our understanding of the social is used at many levels of policy and decision making, both in government and industry. The social sciences are the most rigorous and expansive bodies of knowledge that we have for these matters, and their influence on decision makers is undeniable (mostly in the form of statistical analysis), simply because the only other options for justifying choices are either dogma or personal fiat. The advancement and correctness of the social sciences is of utmost concern to all members of society.

For the social sciences, applying computational techniques helps in overcoming some of the core limitations of studying social phenomena. Social scientists have always been limited by the inextricability of the subject of their research from its environment. Hence, it is difficult to study different factors influencing phenomena in isolation. Safety and ethical issues can be an obstacle to innovation, e.g., for criminologists, it is very difficult to get first-hand evidence of crimes while they are being perpetrated—an observer would most likely be legally required to try to prevent the crime rather than letting it take place. Developing response strategies to unpredictable and dangerous situations is difficult to accomplish in the field, since such situations are unpredictable and by their nature very difficult to control. Computational methods allow us to circumvent these problems by generating scenarios inside a virtual environment. In particular, modeling and simulation allow us to dynamically and interactively explore our ideas and existing knowledge.

Computational thinking about social phenomena, however, means thinking in terms of multiple layers of abstraction [159], which facilitates a systematic study of the phenomena by adjusting the level of detail given to the various factors under study. Computer models of social systems simulate dynamic aspects of individual behaviour to study characteristic
properties and dominant behavioural patterns of societies as a basis for reasoning about real-world phenomena. This way, one can perform experiments as a mental exercise or by means of computer simulation, analyzing possible outcomes where it is difficult, if not impossible, to observe such outcomes in real life. Computing facilitates the process of developing clear hypotheses that can be shared, and then critiqued and refined over time.

This thesis considers the use of software models to investigate social science phenomena. The construction of models in software based on well-established theory and available data allows us to explore the ramifications and consistency of our understanding in a way that is impossible in the real world. This is certainly not the only method for pursuing social science research, but its clear isolation from real (and thus open) environments make it an appealing choice for many research initiatives. Further, the framework presented here seeks to address the needs of social scientists in the use of software modelling as a research vehicle. I also argue that the use of software models is a valid solution to methodological concerns, such as those based on the theory of post-normal science. This work seeks to widen the use of software modelling in social science research due to the unique advantages it promises.

1.1 Contributions of the Thesis

This thesis presents an innovative approach to the building of software models of social systems. This approach is more refined than anything existing in the literature, and is particularly tuned to the needs of small teams of interdisciplinary researchers (approximately 2 to 12 people). The case studies this framework was developed from were limited to teams of this size, so the claims made here will be limited to such groups. Teams of this size benefit greatly from some structure to improve collaboration and communication, while at the same time remaining flexible enough to pursue exploratory directions of research. Larger teams will likely need more rigid processes in order to ensure coordination, while solo researchers may need to adhere to fewer of the guidelines described here, at least during the initial phases of their work when communication is not as vital. Still, teams of other sizes are also likely to benefit from various aspects of this framework. Some general suggestions for software development in computational modelling exist [67], but the framework here extends upon those and provides more structure, without overburdening the creative process of experimentation with too much documentation or formal requirements. In particular, the software development framework I describe in this thesis contributes to the existing practice
of computational modelling of social systems in the following ways:

1. **Facilitates effective and intelligent progress:** The core of this development framework is an iterative cycle focused on building software in multiple, incremental stages. Development should be dynamic, reacting to the discovery of problems, and changes in direction of research interest. This cycle shares similarities with the OODA-loop used in defence science to model intelligent, reactive behaviour. In particular, project progress should involve frequent reassessments of the current situation and goals, resulting in appropriate re-planning. This emphasis on flexibility, learning, and rapid response in the development side of software modelling is a critical feature in meeting the needs of the creative aspect of exploratory research.

2. **Supports collaboration among team members:** The effectiveness of interdisciplinary research is dependent on the effectiveness of team members to work together and share their expertise. This framework supports that interaction through the use of mathematics, formal methods, and visualization to ensure that the understanding of the models under consideration are clearly shared among all team members. The diagram tool Control State Diagram editor (CSDe) was designed primarily for this purpose. These practices also make it easy to share the model and results with peers and stakeholders, as well as enabling reproduction of results by other scientists.

3. **Addresses social science researcher concerns:** Software modelling is a good answer to many concerns of contemporary social science research. Software models can perform experiments that are not feasible in real-life, either for practical or ethical reasons. The concept of post-normal science has been raised by social scientists as a cause for concern, particularly from a methodological perspective. Post-normal science claims that there are fields of science for which uncertainty is significant and unavoidable, yet the need for advancement in those fields is critical. For such fields, post-normal science suggests that “normal” application of science is difficult, that is, using scientific knowledge for solving specific problems or making predictions. Working on ideas in software allows scientists to deepen their understanding, even when faced with uncertainty in terms of data or theory. New knowledge can be integrated as it becomes available, and confounding influences can be abstracted out of a model. Software modelling is a particularly attractive avenue for pursuing future research in the social sciences for these reasons.
1.2 Structure

The first main part of the thesis considers foundational knowledge for my research. Chapter 2 introduces previous research and ideas related to Computational Social Science. Chapter 4 discusses several topics which are important to the consideration of this kind of research as science. I introduce both well known descriptions of scientific activity (from Popper and Kuhn), as well as more recent theories (Post-Normal science). Chapter 3 looks at concepts in software technology that are relevant to my topic: software development, and formal methods. These are necessary to understand the approach toward building research software that I present in this thesis.

The central part of the thesis is concerned with the development framework I have used for the software modelling of social systems. Chapter 5 describes the principles that any such framework would need, and motivates the thinking behind the framework itself. I characterize the kind of problems that are of central interest to software modelling research, discuss the realities and expectations of interdisciplinary research. I also describe the characteristics the framework needs to embody: intelligence, communication and reproducibility. The elements making up the framework are described in Chapter 6. These include the iterative development cycle, important software features, support for collaboration, and methods for ensuring clarity of communication.

The third part of this thesis presents a number of case studies of projects developed using my framework. The three chapters in this part are divided by the primary modelling technique used for the projects described in that chapter. Chapter 7, agent-based modelling projects, looks at several versions of a single long-term project: Mastermind. Mastermind is a simulation of criminal offender activity in an urban landscape [20, 22]. Chapter 8 considers several cellular automata projects. These include two stages of an urban migration simulation (i.e., where people choose to live in a community) [37, 85], and a simulation of peer influence on binge drinking [84]. The final chapter in this part, Chapter 9, describes several (hybrid) fuzzy cognitive map modelling projects. The first is a simulation of insurgent activity in a population-centric war, like the occupation of Afghanistan [125]. The second is a model of how obesity interacts with social networks [60]. The final project described considers crowd dynamics at public events, with a focus on how disruptive behaviour can spread through the crowd. These projects are intended to show the effectiveness of my framework for successful guiding and supporting social system simulation research. The projects are
The thesis concludes with discussion of the limitations, future work, and accomplishments associated with this work. Computational modelling holds great promise for the social sciences, since it is a way of overcoming many of the limitations faced in everyday investigation into social activity and behaviour. It is always challenging, often expensive, and sometimes dangerous to try and perform experimental social research on human beings. One big reason for this is that it is difficult to establish and maintain a boundary on the social activity of a given group of people. We are not discrete, and our behaviour can be unpredictable. Being able to isolate and simplify a social system on a computer can allow for a kind of theoretical purity impossible in the real world. Social inquiry also involves some issues of dire impact in our lives: our physical safety, our long-term economic livelihood, our acceptance socially in the wider population of humans, our sanity. Playing with such factors in order to test out a hypothesis can have a permanent impact on people’s lives if attempted in real life, whether done as part of research or as a policy decision. Computational simulation allows users to test out ideas in a dynamic but logically consistent manner on a computer, prior to actually affecting any individual’s actual life and activities, and at a tiny expense. So modelling with software is clearly capable of contributing to social science practice and research in new and helpful ways. While still in its early stages, this approach is being applied to more and more research topics, and we are learning more all the time, both about the subjects of our investigation, and also about how best to conduct that investigation. With this thesis I present the justification, methodology, and evidence from my own work that shows how to carry out such research in an effective and productive manner, from the perspective of developing the research software.
Part I

Foundations
Chapter 2

Computational Social Science

In this chapter I will discuss concepts and literature related to the use of computing to model social science topics.

2.1 Modelling–General Issues

Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful. [19]

A model is a parsimonious version of something in the real world, intended to facilitate understanding by only including salient aspects of the subject matter. Models can help to reduce uncertainty about the future, test “what if” scenarios, and have the potential to reveal underlying structural characteristics and relationships. Models also are a formal explanation of understanding, and serve to communicate ideas and share in knowledge discovery [70]. Modelling as a practice is the process of constructing a representation of a concept or phenomena such that it can be manipulated mathematically and/or computationally. A classical example of this is an equation whose variables represent significant factors, such as forces in a mechanical physics formula. A more recent example is a multi-agent program which simulates economic activity within a community. Modelling is extremely useful for scientific purposes, since it enables several primary activities necessary for science. First, it allows us to test our understanding by creating an example situation to test the model on. We can then compare the result of the model to a real world equivalent. If they match, we gain confidence in our understanding, and if they do not, we have an indication of where we
need to probe further to improve our knowledge. Second, a model that seeks to capture the
causal structure of the target phenomena can help to explain the concept it represents by
providing a plausible demonstration of how such behaviour can be produced. Finally, a good
model can be used to predict. If we are confident in the results the model will produce, we
can use it to forecast what will happen given a set of starting conditions. Some models work
in the reverse fashion: given a set of final conditions, produce the conditions that caused them. Some models, such as equations, may work both ways. There are extremely useful
applications of this kind of prediction that are already in everyday use, such as calculating
trajectories, and predicting weather. The three functions of models outlined here are vital
to scientific activity, and thus modelling is important to science in general, but the kinds of
modelling vary greatly depending on the field of research.

A model is a simplified representation of real-world phenomena. They are one of the
fundamental tools of the sciences, since they allow us to isolate the characteristics of interest
so that they may used for testing, analysis or prediction. For example, the Newtonian
model of physics provides mathematical models for describing the motions of objects. These
models can be tested experimentally, comparing the results of the equations with the motion
of real objects. The equations can be analyzed for their mathematical properties, such
as the relationship between different conceptual entities, such as speed and acceleration.
And the equations can be used to predict the future motion of objects given a starting
state. Note that the equations do not necessarily consider every possible factor capable
of affecting the current state, nor are they necessarily completely correct (as is the case
for Newtonian mechanics), but they are a useful approximation for such phenomena under
certain assumptions.

In [67, 73], the authors describe a number of different modelling approaches that are
appropriate for the application domain under consideration (health sciences, social sciences),
with the aim of providing researchers interested in modelling complex systems a starting
point from which to decide which method to adopt or adapt for their purposes. Both of
these books crucially point out that it is important to consider multiple modelling methods,
since they all have benefits and drawbacks. A particular modelling approach may be more
appropriate for a given purpose than others. It can also be beneficial to construct multiple
models for a given research project, in order to enable comparison of those models. In [73],
the modelling process as a whole is described, from initial decisions such as whether to
use a qualitative or quantitative modelling technique. For quantitative models, there are
further qualities to consider, such as whether it should be deterministic or stochastic; static or dynamic; discrete or continuous. They also discuss data issues and the use of statistics. Multiple modelling techniques are outlined, along with reasons for selecting them. Although it is aimed at the modelling of healthcare issues, the book is comprehensive and general enough that its content is largely transferable to most other fields that involve social systems.

2.2 Models of Social Systems

Nigel Gilbert has been central in promoting the use of simulation models in social science research. For example, in [66] the motivation and general methodology for using such models for social simulation is presented. They point out that computational models are based on logic, like other types of mathematical and statistical modelling. However, they point out the appeal of agent-based modelling techniques for considering human social systems, particularly for systems with non-linear mathematical properties. They suggest the use of object-oriented programming languages as a tool to improve the software development process, both from the perspective of coding and clarity. A more practical introduction to the activities, concepts, and tools that are part of agent-based modelling is found in [65]. Verification and validation, a definition of agents, bounded rationality, available software packages, and several examples are discussed, among other topics. Another overview and motivation of the use of agent-based modelling for social science applications is provided in [107]. Their article ends with a number of useful recommendations for simulation research. These include an emphasis on the scientific nature such research needs to embody, in that exploration of the model space should also include structured experimentation, and tests for robustness and validity. They also highlight the importance of considering macro level structures and behaviours, even when building a model of autonomous agents.

An initial methodological standard for computational modelling in the social sciences is presented in [129]. This methodology aims at categorizing models based on design choices in order to allow for easier comparison of different models, and in order to allow for modelling knowledge to progress more effectively. Characteristics they think are useful for describing models include: whether or not all values of the scenario parameters are tested, or only a subset (full or partial exploration, where feasible); whether the concept of equilibrium is important to the analysis of the model, at either the macro- or micro-level; where parameter values have been estimated, or are known, if at all; among others. They also underline the
need for communication of validity tests, and that modelling research should be replicable, both by means of documentation, and standard code.

A survey of literature related to stakeholder-centric simulation design found a need for “iteration, collaboration and exploration of alternative scenarios” [127]. That article contains a number of suggestions for more effectively involving stakeholders in the development of models, although they admit it is an initial set of techniques. Importantly, human factors, such as maintaining stakeholder motivation, and technical factors, such as the importance of formal computing models, are emphasized in their approach. They point out that prototyping can be very useful for analyzing the current model, and as a tool for discussion. They note that it is not always possible to clearly delineate where domain expert and technical expert roles are separated; this is especially true when you have experts from more than one application domain working together. While in their own project they had difficulty due to the long time periods required for programming and debugging, they point out that Agile software development methods hold promise in this regard. A particularly complete methodology for prototyping and developing distributed agent systems is presented in [55]. The methodology described, ELDAMeth, includes an overall development lifecycle model (consisting of Modeling, Simulation, and Implementation), and automated code generation based on high-level models. While ELDAMeth is explicitly designed for building distributed agent systems, it draws upon agent based modelling ideas and practices. Simulation of the system model for prototyping and validation is considered to be a central step in the development process. The techniques and processes they present could be useful for producing agent based models, but the focus on an implemented software system as the ultimate goal does not match well with scientific research.

A model-centred epistemology for social system simulation research is presented by Rossiter, Noble, and Bell in [133] (see Fig. 2.1). The authors identify 3 challenges for social system modelling: System Complexity, Limited Accuracy, and Issues with Empirical Data. They emphasize the need for validation throughout the research process. Building upon [5,114], their core ideas are: considering science as a means of explaining the dynamics of phase spaces; that theories explain isolated, idealized systems; and model centredness as a scientific approach. An important point that they raise is that these projects can be seen as working on a set of models, and not a single one (even if only one model is ever being actively worked upon). The various versions constitute competing views of the target system. One model (presumably the last) may end up being superior the others, or it may
be the case that different models are stronger with regard to different goals, scales, or questions. The model centred approach is shown as an alternative to an axiomatic approach. Such an alternative is attractive for social science research due to the difficulty of producing and proving appropriate axioms. They highlight the difference between Descriptive Model usage, whose goal is to closely mirror the behaviour of a real life system, and Theoretical Model usage, which attempts to mirror the behaviour of a theory in an (somewhat) abstract system. They note that empirical data is never pure, but is always filtered by the choices, assumptions, and constraints faced by researchers. A research process model with feedback loops is generated progressively in the article. It focuses on the way in which different research activities interact, primarily with regard to validation. One other way they suggest to assure validity in the model is to incorporate stakeholder views. One final contribution of the paper is a proposed taxonomy of scientific positioning choices, including pairs such as Theoretical vs. Descriptive, Qualitative Validation vs. Quantitative Validation, and Simplicifying Refinement vs. Additive Refinement. In whole, the article provides a number of methods for considering the methodological choices made regarding a given social science simulation project, which allows better comparison with other projects of the same kind.

There are numerous multi-agent modelling toolkits available, which include programming libraries and/or languages that facilitate the production of simulation software. Common and important components include methods for visualizing models graphically, reducing the need for toolkit users to have specialist technical knowledge in that kind of programming. Two toolkits are of particular note. Repast has a wide user base and allows for the creation of models using ReLogo (a dialect designed for Repast), Java, and flowcharts [118]. MASON is another toolkit with which a variety of model types can be developed, and is notable in that both the simulation itself and its data visualization can be run concurrently [104].

2.3 Verification and Validation of Simulations

Generally, verification is defined as determining whether or not a model has been correctly implemented according to its design, whereas validation is defined as determining whether or not a model accurately mimics reality. In “Science in the Age of Computer Simulations”, Eric Winsberg explores the philosophical foundations of scientific investigation though simulation [160]. His own domain familiarity is with the simulation of physical systems, such as storms. He claims that the problems faced by computer simulations are similar to those
encountered by traditional experimentation. He also discusses the processes of verification
and validation as they are traditionally seen as fitting into two specific phases of the model-
ing process. He claims that in actual practice it is impossible to separate the two, and
that scientists use a variety of factors to determine whether or not they believe a specific
model is reliable. A typical reason for this is that an exact mathematical representation of
the system being modelled is computationally intractable, and some approximation must
be used instead. Another has to do with the discrete nature of computing: it is usually
impossible to exactly model a continuous process using discrete time steps, particularly
if interactions at small scales have non-linear effects. Thus scientists are required to use
techniques that may in themselves not be completely founded in science. Winsberg claims
that such assumptions do not necessarily endanger the scientific value of the model: he
emphasizes “reliability over truth”.

Robert Sargent has a different approach to solving the mismatch of expectation to prac-
tice when it comes to the role of verification and validation in simulation projects [135] (see
Fig. 2.2). He identifies various types of verification and validation and places these within
an iterative, interactive development life-cycle. At its simplest, he breaks down the process
into these activities: Conceptual Model Validation, Computerized Model Verification, and
Operational Validation. These can be further broken down, e.g., the operational validation
of software can be considered in terms of how well the execution output matches appropriate
real data (Results Validation). It can also be considered in terms of how well it supports
the theory used to build the model—and this validation can be performed either to validate
the model based on the theory, or to validate the theory based on the experimental results.
Another powerful insight in Sargent’s model of the modelling process is the division of the
real world from the simulation world, with system theories as the bridging element, bringing
together results and observations from both modes of research. (Although it is not made
clear in Fig. 2.2, the results of simulations can directly influence real-world experimentation,
such as by prioritizing what data to collect, or guiding experimental design.) Sargent’s
lifecycle model provides many opportunities for specific types of testing for validity or ve-
racity. It also considers whether the operational validity of the system can be established
objectively or subjectively, within the context of both observable and non-observable target
systems. Thus, his approach to verification and validation is useful for simulation projects
outside of applied fields, including social science and complex systems analysis.
Figure 2.1: Rossiter, Noble, and Bell’s model-centred epistemology of social system simulations. Adapted from [133].
Figure 2.2: Sargent’s relationships between the real world and a simulation world with regard to verification and validation. Adapted from [135].
Chapter 3

Software Technology

3.1 Software Development

Software development methodologies enable systematic production of software. The intention is to increase the reliability of the software development process, allowing for predictable results and an improved final product. This is similar in purpose to any other systematic process of creation, such as traditional forms of craftsmanship, and the modern factory model. However, due to its peculiar characteristics, the systemization of software production has not been without challenges or the need to try different approaches. Since its creation is dependent on technical expertise, software shares some characteristics with other technological items, like machines and buildings. Thus, engineering concepts are appropriate for software in some ways. However, since software is not a physical object, but rather a composition of logic and data, it faces unique difficulties and opportunities. For one thing, a design for software is never set in stone. If the designated requirements for a software project change, it is possible in principle to alter development to meet the new goals. There are also usually a multitude of different ways in which a given problem can be solved in software, but it is not usually the case that it is clear which is the best way to pursue. Partly this is due to the youth of this field, which still has many important issues open, both theoretically and methodologically. It is also due to some proven limitations on our abilities to develop software. The most famous of these is the Halting Problem, which limits what we can know about a program while we are in the process of creating it.

Agile methodologies have found widespread adoption in industry due to higher rates of success while using fewer resources. Here success is measured in terms of getting a
project done on time and on budget. The category of agile methodologies includes Extreme Programming (XP), the Rational Unified Process (RUP) and Scrum, among others. Agile methods are diverse in practice, but share a number of important characteristics. First of all, they emphasize the role of people in the development process. Thus, how people interact and relate to each other is considered as an important factor in the production of software. This includes clients as well as the people working on the software. They also emphasize flexibility in the face of change and obstacles, which is appropriate since this is a strength of software itself. A particularly important characteristic of agile methods is their subscription to an iterative mode of production. Instead of following an initial plan from start to finish (the classical “waterfall” style), the problem is broken down into minimal milestones which can each be evaluated when complete. This means that the project is continuously being tested and considered by all stakeholders. The result of this is that development is given many opportunities to adapt in the case that problems arise, or if the program does not match user needs, or if those needs change, or any other such issue.

Another reason agile methods are appropriate is because they are aimed at small teams. Small teams need less management overhead, which is also beneficial in an academic or mixed setting, where a formal hierarchy is unlikely to exist. The goal for an agile project is fulfilling requirements in a rapid and successful manner, which may not seem appropriate for research projects where the goal is the discovery of useful knowledge. However, the interactive and iterative nature of agile development ensures that the software simulation mirrors researcher understanding, is informative where it fails to work, and allows for rapid adjustment to match new ideas.

The agile software development paradigm is in general the most appropriate for working on computational models. The emphasis on flexibility to change is crucial here, since the level of understanding and current focus of interest is subject to ongoing change. Scientific fallibility means that we must be prepared for the discovery of errors in our model, and be able to try out promising alternatives when they appear. It is important to recognize the role of insight as both a driver and goal of working on this kind of project. When it comes to the model of the system, the attitude of “if it isn’t broken, don’t fix it” is not appropriate; instead it is a constant drive to improve something that is necessarily broken (to some extent). Another reason for an emphasis on agile methods is their focus on the role of human beings in the development process. Team members bring a variety of skills, knowledge, and needs to any project, and failing to take advantage of the good and/or deal
with the difficult can be catastrophic in terms of success of the project.

Experts on simulation modelling agree that the building of simulations is an iterative process, and that requirements change regularly during development [73,136]. Agile methods are suggested as one way to address these issues. The reasons cited for these issues include the unclear and poorly understood nature of the subjects of simulation, as well as insight into the subject given by intermediate versions of the software. However, in some cases, this insight into the problem is the product of real value to the customer, more so than the simulation program itself. Indeed, the modelling process is capable of profound influence on its participants: e.g., challenging existing theoretical models, and suggesting alternatives; encouraging new formalisms and metrics; building professional connections, and helping specialists to become aware of how their work relates to that of others. Simulation and modelling has the potential to draw upon advances in software engineering, while also requiring attention to its own unique aims and characteristics.

3.2 Formal Methods

Formal methods are a means by which a system is specified in a manner that is fundamentally mathematical. Due to the mathematical nature of logic, this is particularly applicable to software systems. A formal method provides a way of designing a system so that we can analyze or ensure mathematical and logical properties. Formal methods are useful during the specification of a system, since they require that a system be described in such a way that there is no ambiguity from a mathematical perspective. This does not mean that all details must be decided on from the beginning, as long as they can be contained within some existing structure within the specification. If necessary, such details can be added in afterward. Formal methods may also be used to prove system properties. An example of this is model checking, which involves checking that a system fulfills its specified requirement, such as avoiding dead-lock conditions or staying within safety boundaries.

3.2.1 Abstract State Machines

The Abstract State Machine (ASM) method is a formal method used for the design and validation of software systems [16]. Formal methods in general are a way of specifying things in such a way that their representation is fundamentally mathematical. ASM is an extension of the well-known Finite State Machine model with the additional use of first
order structures as abstract data structures. ASMs have two major concerns. The first is that design decisions that can be considered independent in nature should be separated. Since many design decisions are based on things that are not formalized or understood well, it is important to keep them abstract. If the understanding of an element changes, it is very useful if that element can be altered independently without causing unintended side-effects in other parts of the code (i.e., a separation of concerns). The second concern of ASMs is to separate design of the software system from its analysis. ASM code can be used to clearly map the conceptual model to its formalism, enabling validation in advance of software testing. ASMs have additional advantages: for one, as formal methods, automated processing can be performed upon them, allowing for systematic testing and verification of the code, such as through model-checking. There is also the ASM refinement method, which provides a mathematically sound means for the stepwise refinement of elements within an ASM specification.

The Sequential ASM Thesis states that any sequential algorithm can be simulated step-for-step by a sequential ASM [16]. This thesis demonstrates the usefulness of ASMs as a general model of computation. ASMs can also be used in a distributed or asynchronous manner. Asynchronous programs are capable of mimicking the asynchronous nature of real world interactions. This is not always desired in scientific computation, since hardware-level effects can introduce non-determinism into the execution of a simulation. If a simulation is expected to behave identically each time it is run with specific parameters, this is not a desirable characteristic. However, if non-determinism is not problematic for a given project, asynchronous ASMs provide a very straightforward and natural way of modelling social systems. Human agents can be represented at any level of abstraction appropriate to the aims of the experiment. This may result in ASM agents acting as groups of people in a social system, or it may result in a group of ASM agents acting as the various cognitive and physical activities of a single person within such a system. This flexibility, both in terms of level of abstraction, and in terms of synchronous or asynchronous execution, is a crucial strength of the ASM method.

3.2.2 CoreASM

CoreASM is an executable specification language based on the ASM method originating from the Software Technology Lab at Simon Fraser University. Since it is executable, it requires and provides more syntax when generating an ASM specification. However, it is still more
straightforward and comprehensible for a non-specialist in computational modelling than regular program code, such as Java or C++. Also, due to the relative conciseness of a specification, it is easy to make changes based on user requests and test those changes immediately.

Formal methods and agile methods may at first seem unlikely to work well together. Formal methods emphasize establishing correct design. Indeed, some accomplish this through extensive mathematical measures such as theorem proving or model checking. On the other hand, the agile methods aim to enable software development to deal with change during the development process. While many associate agile methods with fast-paced code production, the gain in speed promised by agile programming is due to limiting time lost due to changing requirements, which is all but inevitable for most software projects. As another way of improving productivity, agile methods discourage producing too much documentation for software, instead focusing on developing code that is elegant and comprehensible on its own. At the other extreme, formal methods are associated with slow software production, since they require that more effort be spent on the design stage of development. However, they too aim at improving the entire software development process by limiting time lost due to incorrect specifications. So both formal and agile methods are intended to encourage more effective software development, with less wasted effort. Thus the combination of these methodologies appears promising [12]. In [46], the authors show an example of a mixed approach, using a formal method (X-machines) that works well in conjunction with an agile development process.

### 3.3 Reproducibility

Reproducibility is a fundamental characteristic of science that is both challenged by, and aided by software technology. In principle, a scientific experiment should be documented in such a way that anyone should be able to reproduce that experiment in order to independently verify its results. Traditionally, this has meant describing the equipment used and the process followed in order to produce the results. With the massive shift towards experimentation performed only via computer program, different ways of ensuring reproducibility have become necessary. The growing length of computer code has meant it is impractical to include entire programs as part of the body of a scholarly article. Some journals allow
for the inclusion of computer code and data as part of the supplementary material accompanying an article, usually available for download from an online source. However, this is not necessarily a comprehensive solution since it may be difficult to run a program in a given language in some operating environments or hardware configurations. It is also not a solution for code meant for large-scale computing, since it can be difficult to gain access to sufficient resources. Another problematic case is when the data used in an article is confidential in some way, and may not be freely distributed. Thus, while widely lauded as a good principle, implementation of reproducible computational science on a wide scale has proven problematic, despite attempts in some fields to modernize the practice [28].

However, simply re-running the code on the same data may not be enough to demonstrate its correctness, particularly for complex phenomena. Those results may not be typical, or there could be hidden errors. Another, perhaps more important, function of distributing the source code of a computational experiment is allowing other experts to examine the correctness of the methodology in its entirety. The central algorithm of an experiment may be demonstrated effectively in publication, but its implementation may be problematic. In fact, it is fear of inspection in this manner that discourages many scientists from presenting their code [6]. Even if they are confident in how the code behaves, they may be worried that the code itself is too messy to be presented to others—they do not want the code to give them an unprofessional appearance. Another issue has to do with code reuse. If source code for an experiment is distributed, others may use that code in their own experiments. Given proper mention in order to recognize original contribution, this is a positive opportunity. Having a shared code base allows for projects to be more easily compared, and enables more efficient implementation, since well-established sections of code can be left alone, and only the parts which are of current interest subject to change. Indeed, sharing code will allow for certain elements to be honed by community interest and editing, eventually becoming canonical building blocks for other scientific work to be built upon. This is a digital analogue to how well recognized scientific theories and models are used as the foundation for further scientific advancements.
Chapter 4

Scientific Positioning

This chapter establishes the scientific perspective that frames this research. The exact meaning of the word science has changed over time, and is largely used with an implied but ultimately ambiguous meaning. In the twentieth century, two philosophers of science made particularly influential contributions to the discussion of what constitutes science. Karl Popper provided a definition of science which is not only clearly defined and well thought out, but is also workable, as evidenced by its continued use among working scientists [124]. Thomas Kuhn, in *The Structure of Scientific Revolutions*, outlined how the practice and expectations of science regularly change as knowledge develops [95]. Post-normal science is a recent response to Kuhn’s taxonomy, arguing that we are now experiencing a new mode of scientific practice [59]. The characterization of post-normal problems is a close fit to the problems of interest to this thesis. I also discuss several scientific topics of interest: uncertainty; Joshua Epstein’s generative science [48], developed to explain the value of agent-based modelling research; and participatory action research [158], a non-computational form of active research which nonetheless possesses some salient features.

4.1 Falsifiability

The definition of science proposed by Karl Popper is widely accepted, since it clearly demarcates science from the non-scientific and does this in a manner that is possible to test in practice. It has several key characteristics, which I will describe here briefly. For a more rigorous explanation, please refer to [124]. The first is that nothing can be proved conclusively, since an exception may be found or the current situation may change. However,
things may be proven false by the demonstration of a counter-example. Thus, scientific claims are those deemed to be falsifiable, and scientific knowledge consists of the claims that have not been disproven. From this we get the final characteristic: fallibility, which posits that all current knowledge may at some time be proven false—nothing is certain. However, those scientific claims that are most general (and thus most easy to disprove) and have withstood the greatest amount of scrutiny and testing can be considered to be the most reliable. They maintain this respected status until such a time that they are disproven and require alteration to adjust to newly discovered information. Popper states:

> It should be noticed that a positive decision can only temporarily support the theory, for subsequent negative decisions may always overthrow it. So long as theory withstands detailed and severe tests and is not superseded by another theory in the course of scientific progress, we may say that it has ‘proved its mettle’ or that it is ‘corroborated’ by past experience. [124]

Thus the professional interaction of scientists includes attempts to disprove each other’s claims, which in turn improves confidence in those claims not disproven. Popper declares this “friendly hostility” to be at the heart of the advance of scientific knowledge. Popper suggests that this process can be captured in these two rules:

(1) The game of science is, in principle, without end. He who decides one day that scientific statements do not call for any further test, and that they can be regarded as finally verified, retires from the game. (2) Once a hypothesis has been proposed and tested, and has proved its mettle, it may not be allowed to drop out without ‘good reason’. A ‘good reason’ may be, for instance: replacement of the hypothesis by another which is better testable; or the falsification of one of the consequences of the hypothesis. [124]

Popper rejects the use of inductive reasoning as non-scientific, for the reasons famously described by Hume: no number of observations can establish a universal rule, since the existence of an exception cannot be ruled out [80]. The typical example of this line of thought is the black swan: if one has only ever seen white swans, one might believe that only white swans exist, but this is not the case, since swans can also be black, albeit extremely rarely. While inductive reasoning cannot be used to conclusively establish a scientific statement, noticing general properties or patterns over a series of observations is a natural way to come
up with new hypotheses. If such a hypothesis is falsifiable, then this is a means of directing scientific exploration.

The challenge of the social sciences from the Popperian view is the difficulty of coming up with falsifiable claims. The daily lives of human beings cannot be isolated from the rest of the world without compromising the very system under study. In many cases, the consequences of experiments are unethical, particularly in the realm of safety and security, where people’s lives and livelihood are at risk. And yet it is due to the gravity of the subject matter that scientific discovery in these kind of areas is so valuable. Computational models provide several advantages which real world experimentation in the social sciences does not always have. First of all, a computational simulation is not an event in the real world, so there is no real danger in performing it in order to make observations. Second, a computational model is fundamentally a mathematical structure, thus facilitating the production of falsifiable hypotheses to test it with. Factors which in real life are difficult or controversial to quantify can be modelled numerically or fuzzily in a computational model, making it much easier to observe and test than the real thing.

A good example of this from my own research occurred in the Urban Migration modelling project [37]. In that, we wanted to model the degree to which people follow and maintain social norms. However, there are numerous demographic and other factors believed to be related, and it is not clear to what extent these factors affect social cohesiveness. Instead, we modelled social cohesiveness as a single value, rather than modelling its causes or motivations. This lowered the dimensionality of our model to only the variable we were interested in, which reduced complexity and made scientific testing and analysis of the model more feasible. While in this case we are studying the behaviour of the model and not the real world system, experimentation on the model can be performed in a manner that is consistent with Popper’s definition of science. Establishing the degree to which a computational model is relevant to the real-world system is still an open problem; yet, it is hard to argue that there is no relevance if the model is constructed using a process which is in itself based upon scientific understanding of the system. While computational models may not be as reliable as real-world experimentation when circumstances are ideal, they provide methods for applying scientific methodology where conditions are not amenable to performing traditional research.
4.2 Scientific Progress

Kuhn’s most noted contribution is with regard to the social nature of scientific progress [95]. He identifies three stages: prescience, normal science, and revolutionary science. Prescience is the state before there is sufficient understanding to have any scientific practice. Normal science marks the stage where scientific practice in a field is useful (in the sense that it can be used in a problem solving manner), and the knowledge composing that field can be extended without challenging its fundamental assumptions. When the central tenets of a science are shown to be problematic, and competing theories are proposed, a science goes through a revolutionary period until a new paradigm dominates, and normal scientific practice resumes. Kuhn’s characterization accurately models the transformation from Newtonian to Quantum physics, and the establishment of Plate Tectonics as a dominant theory in geology. This inclusion of the social nature of progress and practice in an understanding of science is vital, since science (as we know it) is practiced within the social venue of human society. While Kuhn’s criteria for explaining which theories are considered scientific differs from Popper (he has a list of general criteria, but outlines that it is fundamentally relative to the priorities of individual scientists), Popper’s falsifiability criteria fits well within Kuhn’s cycle, and the two theories complement each other in providing an overarching view of scientific progress.

4.3 Post-Normal Science

The term post-normal science has been used to identify a contemporary mode of science that cannot abide the assumptions or conditions of earlier scientific traditions [59]. The need for and appropriateness of post-normal science closely follows the pervasiveness of science and technology in a society. In a nutshell, classical ideas of science taking place in isolation and requiring complete validation cannot be held for fields, such as the social sciences, where the vast web of interconnected factors prevent any type of conclusive validation or negation. Further, the issues being considered are of such significant importance and urgency that abandoning scientific research is infeasible or irresponsible. In short, it applies to fields of research where both system uncertainty and decision consequences are high. Certainly this is true of those fields that fall within the social sciences, where policy decisions have a direct impact on all of our lives, and are often felt beyond borders. This has definitely been the
case for climate science, where decades of simulation research has provided overwhelming
guidance to reduce greenhouse gas emissions, even if individual simulation experiments
cannot be shown to be true in the classical sense. It is also the case in economics, where
adherence to classical thinking has failed to provide guidance in real markets, and has failed
to prevent financial disasters.

One way of considering post-normal science is that the normal science phase, as defined
by Kuhn, is no longer a possibility due to the acceleration of scientific advancement. The
constant adoption of new technology and policy creates an environment in which ignorance is
a constant. The overlapping of multiple dynamic factors prevents an ultimate identification
of the causes and effects of phenomena. It is still possible to gain knowledge and advance in
our understanding, but our confidence is constantly challenged by the introduction of new
issues and changing priorities. Establishing reliability is, in effect, a moving target.

Post-normal science introduces a further factor of interest into any understanding of
scientific progress: the environment. Funtowicz and Ravetz note that several characteristics
of contemporary human society prevent certain fields of science from ever stabilizing within
a mode of normal science. They note that we are now facing issues for which there is great
uncertainty yet are of imminent concern, thus our concept of acceptable scientific practice
must change. Climate change is the central example from which notions of post-normal sci-
ence emerged, yet others are plentiful: public health, criminology, and security all deal with
issues for which inaction may in itself be reckless. We are forced to apply scientific thinking,
despite the fact that accessible theory and/or data may not be sufficient to provide us with
overwhelming confidence in our conclusions. Note that the nature of technological progress
itself is one of the drivers of this uncertainty. As scientific knowledge increases, applications
of that knowledge are found, and these introduce new influences into the physical and social
environment. Since these influences are new, their complete range of effects are unknown.
Further, as recent influences overlap and compound, it becomes increasingly difficult to iso-
late their individual contributions. Thus, a post-normal condition for science is unlikely to
abate without the deceleration or weakening of scientific and technological advance. This
appears to be a state of affairs that scientists in many fields need to become used to.

Note that a post-normal view science does not condone non- or anti-scientific behaviour.
The seriousness of the subject problems means that we need to act in an increasingly effective
manner to deal with them: the price of failure is certainly no less than the price of inaction.
Instead, we must be highly observant and act in a careful and proactive manner to apply
scientific practice to difficult problems. This means that solutions need to be tested carefully and progressively so that they can be measured and new knowledge established. The large-scale dumping of iron into the Pacific ocean in 2012, based on a small amount of existing theory, was decidedly not a scientific action, despite the pressing nature of greenhouse gas accumulation in the environment. Iron seeding done in a careful and progressive manner in more isolated venues would be a much less reckless and definitely more scientific approach to testing the same posited solution to climate change. Post-normal science also appeals to a wider inclusion of membership in scientific activity: practitioners in any field are noted for already possessing knowledge that researchers with less practical experience may lack. By extending the practice of scientific investigation to a larger section of society, we take advantage of more resources in society available for scientific computation (here I am using the concept of society as a computational process from [48]). Certainly, the implications of scientific discovery with regard to these pressing problems is of interest to a wide variety of stakeholders, and modern communications and information technology enables collaboration on these problems to a degree never before possible.

4.4 Uncertainty in Science

There is a truth in the scientific endeavour that should not be left undiscussed: uncertainty lurks in many places. On paper, scientific knowledge can appear perfect, but in the field, this is not always the case. For many applications, we have axiomatic truths which we are very confident of. Where these apply, by all means they should be used. But science in its regular sense also includes many uncertainties. One constant is that the accuracy of any measurement is limited by the precision of the instrument: uncertainty here is bounded by a range within which the true measurement exists. Statistics is a powerful way of describing and analyzing events in aggregate, but does not claim any knowledge of the characteristics of a single event. In paleontology, geology, and astronomy, we are provided with much evidence from which we can piece together events in the past, but not a direct record of the past itself: reconstruction of timelines in these fields is dependent on the correctness of the many assumptions that our research is founded upon. Thus, the taxonomy of animal species is neither exact nor static in many cases: new data and new theory incites changes in an ongoing manner.

So why even bother with science, if so much of what we do is limited and confounded
by what we do not know? Thomas Kuhn’s well-accepted metric for the utility of scientific knowledge is based upon predictive capacity. But note that is not the same thing as a definition of what is scientific: Karl Popper’s falsifiability principle is a well-accepted example of the latter. Something can indeed be scientific without being particularly useful. Indeed, this will likely be the case for much scientific knowledge, either in its early stages while it is still developing, or even if it is well-developed but lacks a practical application. Space exploration is often criticized as a waste of resources on something that is not particularly useful considering humanity’s current crises. And while many of us will not agree with this criticism, it would be difficult to find anyone who would claim that it is not scientific. Science is an expansion of knowledge in attempt to find the truth. Since we may never know the truth, or even be able to know if we have found it, Popper’s falsifiability principle is a useful tool in propelling us forward: make claims that can be proven false, challenge them, and use those which survive the test. At least until they fail, or are superseded by superior claims.

In software engineering, one speaks of agility; in planning and scheduling, robustness. These are fundamentally the same property: a resilience against failure caused by uncertainty. With software engineering, the danger is in misunderstanding the needs of the client or the market. Miscommunication is a constant threat to the success of a project. In planning, unknown factors or unforeseen events can result in an unanticipated state of affairs: a good plan needs to be able to deal with this, and contain alternative options or methods of reproducing necessary conditions. For science, in the face of great uncertainty, we must also be able to deal with failure. In fact, it is failure that drives us forward, following Popper’s falsifiability principle, and thus we must not shirk from the likelihood of facing it. Confronting our uncertainty allows us to improve our knowledge, but it is only a useful process if we are able to react to failure in an efficient manner such that we are able to repeatedly pivot our claims so that some progress can be made. In other words, if our scientific research process takes place such that feedback takes place on an extended timeline, our knowledge is brittle and each failure is expensive. This is a state of affairs for which uncertainty is to be avoided, or even feared as a challenge to our understanding. On the other hand, if we are able to construct scientific claims, challenge them, and react to any results in a quick manner, failure will allow us to hone our knowledge continuously, even if the surrounding uncertainty is great. Of course, we must still ensure that both our claims and methods are scientific, and the rapidity of progress is greatly dependent on the quality of our results. In
other words, not all failures are equally helpful in identifying our mistakes.

### 4.5 Generative Science

In his seminal article on the subject, “Agent-based Computational Models and Generative Social Science”, Joshua Epstein describes the promising capabilities of agent-based computational models [48]. From this, he derives an approach he labels generativist, and which he sums up in the phrase: “If you didn’t grow it, you didn’t explain its emergence.” In other words, if you can produce a target state with a system of heterogeneous, interacting agents, you show that such a state is possible. This approach implies deduction, satisfying Popper’s criteria for science, since a falsifiable claim is being tested (can the target state be achieved with this model?). Note that the reverse claim is not necessarily true: if you explain the emergence in a system, then you can build it. Epstein points out that any computable program could, in theory, be represented as a set of equations describing a discrete dynamical system, where each variable is a location in memory. However, it may not be clear how to apply analytical approaches to such a set of equations, such as finding a solution or equilibria. Simulation can be a much more practical approach, particularly for problems of high computational complexity. Furthermore, while the system states produced by processing the results of a set of equations would be the same as a simulation, the equations themselves would not provide the same kind of explanation of the nature of the system. They would lack semantic power, and it is unlikely the relationship between elements in the system would be clear in any kind of sense a human would understand. Rich descriptive power is a notable advantage of computational modelling techniques [45].

Any software model must be entirely mathematical in nature, thus it is straightforward to make falsifiable claims: given a set of parameter values (or ranges of values) and starting conditions, an expected set of ending conditions should appear. Intermediate conditions may also be specified (e.g., a certain value may not drop below a stated threshold). Deciding on interesting claims useful for validating the model is of course a matter of expertise: deciding on appropriate starting conditions, parameters and expectations requires domain expert knowledge. Still, trivially falsifiable claims are also useful in checking the correctness of the system: for example, to ensure that some values never become negative, or that strict capacity limits are not being exceeded. In the Urban Migration Cellular Automata model, this turned out to be vital, since it was necessary to ensure that agents were not moving into
occupied residences. Visualization of the system behaviour displayed impossible population levels at one point in development, and an error in design was discovered and fixed.

The underlying question here is: “Is this kind of research scientific?” From Popper’s definition it is, since any computational model is by definition fundamentally mathematical (otherwise it would not run on a computer). All of the content and behaviour of a computer program can be defined logically and/or numerically. Thus it is possible to formulate statements and hypotheses about the model which can be tested in negatable ways. This doesn’t prevent anyone from making untestable claims about a model, or from making a model based on nonsense or gibberish. But it is not only definitely possible to consider a computational model in a scientific manner, such models are particularly amenable to scientific investigation. Our control over the environment and execution of the model means that our access to any metric of interest is limited only by our effort and ability to innovate. We can store or visualize the resulting data as we wish, and we have the full range of mathematical methods at our disposal to analyze it. Since simulation is by its nature empirical—it produces data—statistical methods are particularly of interest. Qualitative measures are also possible, if they have some numerical interpretation, such as by using fuzzy logic, or visual evaluation of data. So we can even use expert experience to validate the behaviour of a model, if this is an acceptable practice in the subject domain. We can even use the computational model as a part of a process of establishing a quantitative match to qualitative measures, by analyzing what numerical properties of the model are correlated with qualitative judgments. As a mathematical construct, a computational model is fully capable of being explored and examined in a scientific manner.

However, the caveat here is that this is with regard to investigation of the model itself. The link between the model and reality can be difficult to ground, since the model is constructed using theory and data, either of which may be incomplete or problematic. Thus a computational model is greatly dependent on the theory and data chosen to build it. Conversely, analysis of the theory and data can be performed using the model. For example, the logical consistency of a theory or group of theories can be tested by executing a corresponding model and comparing its behaviour with the behaviour associated with the theory. If it can be shown that the program models the theory well, this can help to find problems or inconsistencies within the theory. Epstein describes this process as theory-stressing. This is particularly useful for looking at theories expected to be universally applicable: in these cases, even synthetic data can be used to search for limitations in the
theory, and assess the reliability of various methods. It is possible to think of simulation as a meta-science: it allows us to analyze or extend other knowledge in a scientific manner.

4.6 Participatory Action Research

A mode of social science research worthy of comparison to the current work is Participatory Action Research (PAR). The key features of this kind of research are the involvement of a variety of stakeholders, and that research questions are directly applied to the behaviour of the subject organization [158]. Thus, experiments are devised, performed, and analyzed by the people working in the organization itself. A key characteristic here is an awareness of feedback loops in this process, and an iterative approach. New ideas are tested out, and subsequent plans are based on the results of their predecessors. PAR has been widely accepted in numerous domains, including industry, agriculture, and public service. It differs greatly from the research presented in this thesis in that it involves actual change on a real social system, rather than a simulation.

However, there are two enlightening similarities. The first is the emphasis on integrating multiple viewpoints. The expertise and knowledge of people from different backgrounds are utilized to provide greater insight into the domain, the data, and the research questions themselves. Involving different points of view is also valuable for preventing mistakes, misunderstandings, or even overt manipulation of the experimental process. The second similarity with the simulation research presented here is the use of an overtly iterative experimental process. As can be seen from Fig. 4.1, PAR cycles through planning, testing, and analysis phases. This allows the subject organization to pivot intelligently based on the results of different experiments. Fundamentally, since the results of these experiments are real changes in the state of that organization, there is an invested motivation for results to be acted upon immediately. Simulation research differs from this greatly since there is no immediate reward or penalty for experimental results—any experiment can be pursued without worry of damaging real-world effects. However, it is always valuable for researchers to be able to pivot well to results and analysis, since this improves productivity through both accelerating the process and also by directing research towards more interesting and relevant questions rather than wasteful ones. Although the activities of PAR and simulation research do not necessarily overlap, it is a relevant example of how collaborative, iterative research
Figure 4.1: The iterative flow of participatory action research. Adapted from [105].
in the social sciences can be successfully achieved. Furthermore, for projects where simulation provides guidance and insight for real-world experimentation (such as an intervention), PAR has precisely the kind of characteristics that match with the approach to simulation research presented here, thus providing a template for how the real-world components of such projects should be pursued.
Part II

Framework for Software Modelling in Social Science Research
Chapter 5

Framework Principles

Here I describe the principles upon which the framework is based. These are inherently part of the framework in the sense that they can be understood as, variously, the characteristics, heuristics, and goals of developing software for the purpose of doing social science simulation research. This framework is intended to support the effectiveness of producing software models of social systems. It provides structure that is missing in an ad hoc process, while at the same time enabling the creative flexibility and necessary for innovation and discovery. It also recognizes and facilitates the type of collaboration typical of academic and research environments. In this way, it is tailored for the kind of interdisciplinary research described in this thesis.

5.1 Characterization of Target Problems

The key feature that characterizes models that should be examined through exploratory simulation as opposed to closed-form analysis is that they contain elements which are difficult to operationalize, either on a micro or macro scale. In other words, it is not clear how the model could be converted into a mathematical form that can be analyzed directly, without any simulation, or experimentation. This difficulty can appear in several different forms.

i. Mathematical complexity: In this case, there is a clear mathematical representation of elements in the model, such as transformations or interactions, but there is no known way to simplify this beyond simply executing the simulation. Examples of this are cellular automata and chaotic pendulums. The micro steps of the model are straightforward
and easy to understand, but the link to the behaviour at the macro level is not clear. In other words, we are confident in how the model will develop over a single time-step, but cannot directly transform the model over an arbitrary length of time. In both of the examples given, this is due to the overlapping of many interactions over time such that local events are capable of causing unpredictable global results.

ii. Limited understanding: Lack of knowledge about the element under consideration is the challenge here. The main elements of a process may be understood, but there can be underlying or external features that have not received as much study or consideration. Even topics that are extensively studied and well understood in isolation or at a given scale can produce this problem, since a new model may require the inclusion of phenomena or concepts that have not yet been fully investigated in an academic sense.

iii. Lack of theory: It is possible for a great deal of specific knowledge or data regarding an issue to exist without a relevant theory to generalize that understanding. In this case, it is not clear how to operationalize this aspect of phenomena into a mathematical model. We may be able to creatively imagine one or many ways to do this, but we have no domain knowledge that will support any particular option.

iv. Difficult to represent analytically: Here, the elements under consideration are difficult to completely represent numerically and/or using known analytical mathematical operations. Things that are generally understood qualitatively would fall into this category. This also includes situations where it is unclear whether or not an interaction is local or global, or whether or not an entity consists of smaller elements. In these cases, we may possess sufficient experience to recognize and describe the phenomena very well in real life, but the transformation to mathematical model cannot be done with confidence. Thus the best option is to build a simulation model whose execution can be compared, both qualitatively and quantitatively, with our domain knowledge. Even if a transformation to an analytical model can be achieved, if the phenomena is complex enough that comprehensive data gathering cannot be done, then it will be impossible to validate the model confidently. Thus simulation is a practical option here too.

v. Agency: In reality, human decision making is not well understood, involves numerous interacting concerns, and has aspects that we conceptualize qualitatively (e.g., emotions), thus this challenge can be seen, in some ways, as an overlap of the previous
three. When humans are represented in a model, if their decision making process is represented using even a limited form of AI, it will be unclear how they will act over time. This is true for agents acting individually, although interaction of course increases the complexity of the behaviour. The difficulty of representing human agency increases dramatically as soon as you depart from the most simplifying of assumptions (e.g., complete rationality, homogeneity, omniscience, limited concerns).

5.2 Exploratory Software Modelling Research

Applying computational techniques and tools to develop scientific software or to support the work of researchers in other disciplines calls for special attention to the unique qualities of a research environment. It is easy to see superficial similarities that the interdisciplinary research environment holds with a production software development environment. For example, in both cases there can be ongoing changes to what is being built, otherwise known as requirements creep. One could say that this is a central advantage of working with software: it can be redesigned and rebuilt without repercussions in the physical world, so it makes sense to take advantage of this flexibility. So it is not surprising to see this characteristic in any endeavour that adopts software as a tool. Yet there are fundamental differences in who builds research software and what exactly is being constructed. Despite many varieties of development frameworks available today, they all assume a production model at their core: a product is developed for a client. There is a gap between current methods of software development and the needs of the research environment [90].

One striking difference in a research project is the relationship between the team members. Traditionally, a client (or set of clients) will make a request, and the software team will focus on implementing that request [119]. This relationship constitutes a cycle of communication, where each party has a distinct role in contributing to the final product. In a research environment, computer scientists collaborate with colleagues from other disciplines, due to the potential to generate distinct advances in research through the combination of the theoretical power of all the disciplines involved. All of the scientists working on a project hope to make a discovery relevant to their own work. This means that decisions made on what the project will cover are shared among team members.

Traditionally, a software development process is focused on producing a final product for an end-user. In the context of simulations of social systems, scientific research uses
programs as experiments, to test theories and generate ideas that lead to new ones. Testing
is useful at each stage of theory development, so a single project can generate a number of
programs; they should be thought of as a set of related experiments. In science, the core
methodology for discovery of new knowledge can be concisely described as the iteration of
the hypothesis-experiment-result-conclusion cycle \[124\]^1. In the case of the social sciences,
this includes exploratory techniques as well as classic inductive and deductive approaches.
The programs developed for scientific research are computational experiments, and in order
to accept the reality of iteration, it is necessary to envision the software developed for
a research project as a set of related experiments. In order to relate to one another, they
must share the same theoretical foundation. In other words, it is the conceptual schema that
holds the continuity between programs, and not their functionality. However, the specific
implementation of any program will be determined by the requirements of the experiment
it embodies. Furthermore, since the software being developed is for the use of the team
members themselves, the final configuration of that software is not a primary concern.
The configuration of a program is only a concern to the extent that team members find it
usable. These characteristics firmly shift the focus of the development cycle to the design
or prototyping phase instead of implementation \[130\].

Central to all of these unique characteristics of the research environment is the underlying
aim of scientific discovery. A completed program is not the ultimate goal: software is used
as a route to testing existing theories and creating new ones. Software is important because
of the results experiments can provide. It can act as a sandbox where scientists are able
to try out different ideas. It is also important because the transformation of theory into
something computable captures the concepts being considered in a mathematical form. This
mathematical model provides a blueprint for researchers to explain their ideas, or for peers
to analyze its validity. It allows different projects to be compared to one another, and it
acts as a foundation for further exploration of the subject matter. This is not to say that
a program is an inconsequential product of this effort: it embodies the hypotheses and
concepts of a scientific project, and enables both demonstration of those ideas as well as
further experimentation. However, its importance is secondary to the knowledge generated
by means of the research process. This process can be assisted by the adoption of current best

---

\(^1\)"A SCIENTIST, whether theorist or experimenter, puts forward statements, or systems of statements,
and tests them step by step. In the field of empirical sciences, more particularly, he constructs hypotheses,
or systems of theories, and tests them against experience by observation and experiment." \[124\]
coding practices [9]. Agile methods are well suited to research, particularly the importance placed on people and the idea of iterative development [56]. However, new techniques tailored specifically for the research environment can help to encourage software development that is more successful in generating new knowledge.

5.3 Intelligence

One hallmark of software development when compared to the construction of physical objects is the flexibility of the thing being created. While it is always stored somewhere on physical media, software is pure information, a specific arrangement of symbols, and this makes it easy to change. Thus, a piece of software can (and usually does) change over its period of development. This can be for numerous reasons, including changes in requirements, the clarification of those requirements, changes in available resources, or the platforms upon which the software will be run. Embracing the flexibility of software, and encouraging it as part of the development process, is usually referred to as agility within software engineering. This can be considered a dynamic approach to building software, reacting to changes as they appear. However, I will refer to this principle within the light of the software modelling research that is considered by this thesis as intelligence. The reason for this distinction is because the changes that are taking place in the software over its lifecycle are not due to outside effects beyond the control of the development team. Instead, they are an integral part of the research process that actively guide the development of the software so that the research goals are met. The term intelligence is not used here to imply some kind of intellectual superiority; instead it is used to emphasize a self-directed and goal-oriented nature. More than a matter of solely adapting to problems as they arise, this kind of software development is an active process, where the question is not “Are we still on track?” but instead “What will the next change be?”

Changes in the direction the research is taking can be due to the initiatives of various members of the team, and for various reasons, e.g., the identification and eradication of bugs and inappropriate model behaviour, or testing out new hypotheses based on the results of previous experiments. The ability of the software development process to pivot rapidly and effectively upon these changes in direction is, not surprisingly, central to modelling research via software. Rapid adjustments to the software encourage researchers to ask more questions, and to take more initiative in driving development throughout the research process,
even if they are themselves not comfortable with software development. This leads to a sense of ownership, and encourages researchers to ask more and more interesting questions. Certainly, developers should be clear if a given change is resource intensive and will take time to implement, but small changes should be encouraged. They are indeed at the heart of the experimental process: small steps in other directions, using previous results for comparison.

Considering exploratory software research as an intelligent process is important because it encapsulates a variety of activities being executed by (probably) several individuals in one inclusive arc. The investigation, discussion, modelling, engineering, and experimenting all centres around a single research focus, and is jointly focused on generating new insight into the subject. Thus, the activities should not occur at random, or in a rigid but inappropriate manner, but instead should successively and appropriately push the research forward, reacting to previous information, and searching for answers to newly raised questions. In this sense is the process as a whole intelligent. Actions are planned, their results are monitored, and successive activities planned accordingly.

I present here the OODA Loop as the archetypal model of intelligent reactive behaviour. I became familiar with the OODA loop through my own research in the field of automated planning [83], and although its original application is quite different from that of this thesis, I believe its generality allows it to serve as the inspiration for the reactive and intelligent behaviour of the software development process of this framework.

The OODA Loop

The OODA loop (Fig. 5.1), developed by the USAF Colonel John Boyd, is a model of how an intelligent, sensing agent interacts with a dynamic environment, well-established in both business and military contexts. It consists of four main steps: Observe the environment to discover what is happening; Orient the evidence observed, to establish its meaning in terms of one’s own situation and interests; Decide on what to do based on what was understood during orientation; and finally, Act on that decision. This is an iterative loop, with the results of the Act phase feeding into the next Observe phase. By continually following the OODA loop, an individual agent or group of agents can act in accordance with the situation they are in. Here, the OODA loop is employed as a definitive model of how both directed and reactive behaviour can be broken down to its constituent elements, and how those elements interact with each other. Importantly, the flexibility and effectiveness of an agent utilizing the OODA loop is dependent upon the time it takes to pass through each
OODA iteration. In other words, in a “tight” loop, the OODA agent is capable of adjusting appropriately to new conditions since the decision process is repeatedly quickly.

In fact, the OODA loop is a good model of dynamic, intelligent behaviour. It allows for both long term planning and short term planning, as well as making adjustments for new information, mistakes, new problems, and changes in available resources. I present it here as a useful way of considering scientific research, particularly with regard to complex social systems research. With every process undertaken as part of a research project, it is important to be able to react to any new developments, whether that process is reviewing background literature, formalizing a theoretical model into something mathematical, developing research software, testing that software, or running experimental scenarios. Information (sometimes in the form of data) is produced by all of these activities, and effective evaluation of the meaning of that information (i.e., Orientation) can help to guide further steps in the research process. In fact, one could claim that the OODA loop is a good characterization of scientific activity, with Observation corresponding to data gathering and processing, Orientation corresponding to evaluation and analysis, Decide corresponding with hypothesis and experiment generation, and Act with the execution of experiments and testing in general. The important divergence of this from the classical scientific model is its emphasis on iteration, and the ongoing nature of research.

This is particularly important for research that is being undertaken as part of a policy-making initiative: simulation results can help to guide long-term policy decisions, but such decisions can be strengthened through an iterative process to continually evaluate their appropriateness as the situation develops in real time. While this kind of dynamic or adaptive research-driven policy making is not yet common, I believe that it is the natural implication of both the science of complex systems and the process of intelligent decision making, as illustrated by the OODA loop (see Fig. 5.1).

I present the OODA loop here as a guide to addressing the characteristics of complex social systems research and post normal science in general. The model and software development practices I describe in this text deal with these characteristics, and fit well into a social science simulation research process. Further, in Section 6.2 I also describe specific software technologies and applications which support this process, and help to tighten the iterative research loop. The goal of this is twofold: first, to allow the research process to produce knowledge in a natural and productive manner, such that the results of one investigation guide the questions of the next; and second, to provide an alternative process for
research in field where a single infallible prediction is not a realistic goal of the scientific process, but instead a set of scenarios, which are updated and replaced as understanding and data develop over time.

5.4 Communication

It is common even in regular speech for people to misunderstand the meanings of others’ communication. However, for communication concerned with technical matters or expert knowledge, this is an even more likely occurrence. Not only are there terms specific in use only to one field, there are words whose everyday meaning differs greatly from their usage in certain technical fields. The communication problem is a recognized challenge for all software development projects [11, 56]. In interdisciplinary research, this is intensified because all team members are specialists. Researchers spend many years building up their domain expertise and cannot be expected to develop deep understanding of another field over a short period of time. By the very nature of research, the topics under investigation are cutting-edge. Finally, in order for the results of a project to be recognized, it is necessary to be able to communicate the workings of any computational elements to reviewers. For these reasons it is vital that there are methods for expressing the essence of an idea that
allow for critical inspection, validation and modification.

5.5 The Roles of Software

Based on the nature of social system simulation projects and the principles described in this chapter, the roles we expect computational modelling to play in a simulation modelling research initiative are outlined here.

**Reinforce Iterative Discovery** The executable nature of software means that it can be tested for various things such as accuracy and bugs whenever a change is introduced. Feedback from such testing can be used to iteratively improve either the program or the model itself. Of course, this is also possible with non-computational research, but software encourages this feedback loop through interactivity. To maintain this process, it is important for the mathematical model and computational models to remain flexible and capable of change.

**Visualization** Visual representation of data allows human users to examine and interpret using natural cognitive and pattern-matching capabilities, in addition to deductive reasoning. Ideally, a variety of visualizations should be available, allowing the user to alter their view of the data in order to consider different perspectives or subsets of the entire data set, which can be overwhelmingly rich in a social simulation.

**Formalization** Computational modelling forces researchers to precisely define the modelled elements of the subject phenomena so that it is possible to execute it on a computer. This process helps to frame existing domain knowledge in a mathematical manner. It also highlights areas which are poorly understood (which can become clear since they are difficult to formalize) and areas which are not important to the current research focus (which can become clear when defining them takes more effort than the value they add to the model).

**Testing Ideas** Computational models are ideal for testing hypotheses in a sandbox-like environment.
Raising Questions The modelling process can help raise questions that can be answered by more traditional research in the domain field. This can happen when information is missing to complete the design of the mathematical model, or if the computational model produces contradictory or conflicting results. In these cases, further literature review or discussion among experts may help to elucidate the missing information. It can also guide field research by suggesting questions to ask in future surveys. It is also possible that questions may be raised that are best addressed by more simulation research.

Demonstration A computational model is valuable if it can illustrate domain concepts to non-experts. The dynamic qualities of a simulation combined with visualization capabilities combine to provide a powerful means of explaining complex phenomena in a straightforward manner.

Communication of Ideas As a formal model of a concept, the computational model is a proposed theory that can be used to communicate new ideas in an interactive manner among peers. Likewise, the model should be transparent enough that peers can examine it critically to point out problems, ask for clarification, or use the ideas in other research.

Complex Calculation A computational model is often a combination of many simple rules and concepts, but in aggregate the behaviour of such a model can easily be beyond the comprehension or predictive capabilities of a human expert. In this way, a simulation program can act in the place of a human expert, correctly combining rules and concepts to produce aggregate (and sometimes emergent) results.

Reproducibility A computational model facilitates reproduction of experiments by other researchers, both in the sense of verifying the results through independent testing, and also in the sense that program elements can be easily shared with other researchers working on similar projects.
Chapter 6

Framework Elements

In this chapter I outline some specific solutions, both technological and methodological, that help to fulfill the requirements outlined in the previous chapter. The software development framework presented here possesses the following elements: an iterative cycle of development, software that supports and follows the research process, recognition of the essential role of collaboration, and an emphasis on clarity. These elements are designed to follow the principles described in the previous chapter, and to enable successful and productive social science research using computational models.

6.1 Iterative Development Cycle

Software development under this framework is recognized to take place in an iterative manner, with ongoing changes to the program code. The cycle outlined here is an expansion of the model development process first presented in [20], with an emphasis on how software development interacts with other research activity, such as testing and experimentation. The stages of the process are:

1. **Conceptual Modelling** A general description of the features and expectations of the model. It is here that the relevant domain theory and ideas are selected to lay a foundation for the model in reality. Model assumptions and limitations are also described here in terms of the domain theory.
2. **Mathematical Modelling** Equations, formulas, and mathematical statements are used to precisely define the structure of the model. In the beginning, only the most central aspects of the model will be explicitly formalized here, but others aspects may be added if it turns out that their mathematical formulation has an impact on the behaviour of the model, i.e., how they are modelled is academically of interest.

3. **Computational Modelling** This stage includes the expansion of the core mathematical mechanics established in the Mathematical Modelling step into a full-fledged program capable of producing results, as well as the debugging of the program code. This can include definitions in terms of pseudo-code, formal methods, and of course program code. The process of expanding general ideas to a full program will raise questions that may need to be answered by returning to the Conceptual or Mathematical Modelling step. Model calibration is also realized in this stage, based on feedback from model testing in the experimentation phase.

4. **Experimental Set-Up** This is where the initial state of a system and the duration of an experiment are specified. Combined with the user’s expectations, this composes the scientific claim or hypothesis of this experiment. Set-up options may need to be changed if they do not support a new question of interest.

5. **Execution** Here, the user can view the behaviour of the system as it passes through time during the simulation. The user is capable of shuttling back and forwards through time to compare relevant states, such as the beginning, midpoint, and end. Options for viewing all simulated variables and entities (and their likely aggregations) should be accessible here.

6. **Results Analysis** The data generated by the experiment is presented here for analysis. Typically, there is some plotting or graphical representation of selected variables over the time of an execution. Data output may also be accessed here for use in another tool (data mining, etc.). However, ideally an integrated visualization suite (pre-existing or custom) is included that gives users the freedom to modify their view
of the data as they consider it.

A general outline of the flow of the process is shown in Figure 6.1. The first three stages can be described as the Modelling Phase of the project, while the last three stages can be described as the Experimentation Phase. As can be seen by the many connecting arrows, these two phases have a great deal of interaction and feedback, and in practice are not necessarily discrete in terms of activity. Note that progressive movement through the process is in a single, ordered flow through the stages listed above (the downward arrows), whereas corrective or reactive movement can jump several steps backward to the appropriate previous stage (shown by the upward facing arrows). In principle, it is possible to jump backward to any previous node, but the figure only shows the most likely transitions. On the other hand, forward movement should be done in order. The process of advancing forward happens constantly, adjusting the conceptual, mathematical, and computational models to match the current assumptions, formulations, and questions of interest, interrupted by returns to previous stages when changes need to be made, based on problems encountered or interesting results. By controlling progressive development, we help to ensure that any changes are consistent with team members’ accepted understanding of the models that have been built, and to prevent errors due to misunderstandings from creeping in. In other words, we want to make sure that the conceptual, mathematical, and computational models always match each other. This helps to keep the creative process of experimentation on an executable program within a scientific process where any changes to that program are well understood, validated by domain experts, and documented (where necessary). The various options in terms of backward movement enable flexibility in terms of how the model can pivot appropriately based on feedback or the interpretation of results.

Rapid iteration is key since it allows new ideas generated during experimentation to be added to the program on a regular basis. This is a fundamental aspect of research software development since research by its very nature is exploratory. Development iteration should be kept to a very tight cycle, such as weekly updates, since responsiveness is as valuable here as it is in the interface itself. Participant ideas can be included and tested out soon after they are formulated, speeding up the research process. In fact, it is beneficial to the developer to distribute the latest version of the software after every change so that any unseen effects of those changes are discovered in isolation from other changes. Rapid iteration also means that bugs are more likely to be encountered soon after their introduction into the code (i.e., in the latest set of changes), making their source easier to identify. Bugs found in this
manner are easier to fix than those whose origin is unclear.

Note that this view of the modelling process is complementary to others discussed in this text. Robert Sargent’s diagram (see Section 2.3) emphasizes how different phases in simulation research can provide feedback for other stages, particularly for the purpose of verification and validation [135]. The view described by Rossiter, Noble, and Bell (see Section 2.2) shows the research process from a higher-level perspective: how simulation modelling fits into the general process of social science research. The cycle described here focuses on the interaction of software with modelling research as it progresses in a day-to-day fashion, including building, testing, and experimenting with the software [133]. Similarly, different models of the research process could be used to focus on other aspects of social system modelling research, such as the research and investigation that underpins the Conceptual
Modelling stage.

6.2 Software Features

For this kind of research, the actual software built is not as important as a final goal as the knowledge that is discovered or tested during experimentation. That said, this kind of modelling can benefit greatly by having software that shares a number of features that support the research work of users. For this reason, it is important to build software that helps team members achieve their research goals. So, ideally the software package used to implement the simulation system should be:

i. *Easy to change*, so that suggestions and requests for changes, as well as new ideas, can be implemented in a rapid manner. Obscure programming languages are problematic here, or complicated libraries the code is dependent upon (e.g., for graphics).

ii. *Easy to distribute*, so that each team member can easily remain up-to-date with regards to the latest build and test it out on any machine they have access to. Software that is difficult to compile or that requires specific operating systems or settings may be intimidating to some team members, and prevent adoption.

iii. *Easy to use*, so that all team members feel comfortable using it to test out their ideas. This can be accomplished in part by simplifying the user interface and layout of the program. It can also be accomplished by identifying and maintaining an effective user interface, so that users are not forced to relearn the program frequently. Changes to the user interface are acceptable depending on the level of their contribution to the program.

iv. *Easy to discuss*, so that the experiences and insight of users can be efficiently converted into improvements to the computational model and its software. This requires a user-friendly system for reporting errors and discussing ideas. This includes program-related methods for sharing experiments: results as data files, plots and charts as image files, and perhaps even scenario parameters as settings files. Time spent on establishing a shared vocabulary to describe important features and concepts is also valuable.

I will now discuss several specific ways of meeting these requirements.
6.2.1 Feedback

It is important to maintain a sense of ownership among all team members throughout the development process. This will engender cooperation and participation in the creation of a program. Domain experts should care about what the program does and how it does it. In other words, it should be easy for them to verify that it is behaving in a way that makes sense to them. The program itself should also be easy and even fun to use. This will encourage all team members to use the program, thus providing more feedback, which can be used to improve the program in the next iteration. In other words, the ease of use of a program increases the likelihood and amount of feedback one can hope to receive, and so it is central to the value of a program. A sense of fun encourages play, which is a prime method for intelligent animals to learn how to do something. Play also encourages experimentation and creative investigation, which are central parts of the research process.

Thus the standard criteria for judging quality of Human-Computer Interaction (HCI) should be noted. Responsiveness reduces frustration, and allows researchers to investigate the problem as they are thinking about it, with little or no delay. This will affect choices in modelling: granularity, for example, can reduce the speed of computation if it causes the size of data to increase by a great degree. Feedback for the user regarding program behaviour is likewise important. The richness of this feedback will improve user understanding of what is going on and what it means for their research. Thus it is important to have visual depictions of program behaviour in order to allow for verification of program behaviour. Bugs that are easy to miss in code often become easy to pick out when there is some sort of graphical rendering of what is going on. Visual depictions of experimental results are useful for providing insight into the meaning of a particular run, and for highlighting unusual or unexpected behaviour. For this purpose, graphs and plots can be automatically generated. The result data should also be stored in database format, such as a comma-separated value (.csv) file, which can be used by most database and spreadsheet applications. Such applications are capable of analyzing the data numerically or algorithmically, opening up many options for researchers in terms of how they wish to look at their experimental results.
6.2.2 User Interface

The counterpart to these feedback methods are ways for the user to set up experiments. These constitute the input methods for a program that does not allow for interaction during execution (all of the experiments we would consider at this point in time). These parameters need to be clearly related to how an experiment will be run. They should constitute the “scenario variables” for an experiment, defining the conditions for a given scenario. Some of these parameters will be obvious inclusions in the development of the program, but others will require time and feedback to discover. For example, it may be only through experimentation with one set of parameters that a user will realize that they would like to test variations in another dimension. It will also become clear through use or sensitivity analysis that some parameters have very little if any effect on execution results. These can be eliminated if they become cumbersome to the interface. However, it is important to realize that the parameters chosen and their permitted values are in principle not canonical. In other words, it is through the experimentation and exploration of their values that the experimental space is grown or pruned and interesting results are discovered.

If possible, each of the stages in the experimental process should be clearly included in the design of the simulation software, so that they are visible or accessible to the user to some degree. For example, tabs in the user interface can be used to correspond to the functionality associated with each stage. In practice, at different stages of program development it may not be possible to include such functionality since it is not complete, or the choices associated with that aspect are hard-coded into the software (and thus does not require user input). Still, this is an important guideline since it achieves two goals. First, assumptions and design decisions made by team members are clearly embodied in the software, and it is possible through the software to review the process by which the current software model came to be. This also means that sections of the model which are mocked out but not yet implemented can be examined and discussed. Ideally, the UI will enable changes to design decisions and modelling so that new experiments can be tried out simply through the software. It may be a good idea to limit some users from changing some elements, particularly the model code, but it should still be available for them to check over. In the case of something like the conceptual model, a box of text is likely sufficient: it would contain expectations, ideas, and references to relevant literature (perhaps in the form of hyperlinks). Having this as a reference is useful for some or all of the people using
the program, even if the project has moved beyond frequent updates of this aspect.

![Binge drinking cellular automata program: parameter set-up.](image)

Figure 6.2: Binge drinking cellular automata program: parameter set-up.

Figures 6.2, 6.3, and 6.4 show the Binge Drinking Cellular Automata program, a model of peer influence on the drinking behaviour of undergraduate students (for a description of the project, see Section 8.4). As can be seen in the screenshot, the program interface has the following tabs (which correspond to the stage in parentheses): Parameters (experimental design), Simulation (playback), Population (visualization), and Averages (visualization). In this case, there is also a Log tab which contains system output (including random number generator seed values). Many parts of the software can be reused between simulation projects, with the exception of the code related to the system model, and possibly the graphical output of the playback. I believe that this is due to the overall unity in the scientific method used to develop these models. It is important that the software reinforce and assist in following good scientific practices, and remain a unifying theme across different projects, ideally across different work groups, so that they may more easily understand each other’s work. Indeed, developing a clear and accessible user interface and then maintaining it will allow users to get to understand it better so that they are less intimidated by the software
and are more able to focus their attention on the computational model that it is simulating.

### 6.2.3 Portability

This is a key feature that is easy to overlook in the enthusiasm of computing scientists to adopt interesting, or perhaps even esoteric, technology. It is important to remember that team members will be using their own computers to run the software, and that they will not necessarily have any special technical knowledge relevant to installing software packages, setting up operating system environment variables, etc. Thus the portability of the software is very important. Programs written in languages supported by most operating systems, e.g., Java, are thus a good development choice. An even better option that is becoming easier is to release the program as a web application that is run in a browser. This has the added benefit of making your program easy to share with interested parties outside of your group on a website. A final option that we have not yet tried is development as
Figure 6.4: Binge drinking cellular automata program: population visualization.

a smart-phone application. Again, there is wide adoption of this kind of technology, and although the technical skills related to that kind of development are not yet standard, this is a promising option, as long as the user interface can be implemented usefully within the small size of a portable device.

6.3 Collaboration

While there certainly are gifted researchers in the field who are capable of achieving all of the stages in the research process outlined above all on their own, this framework is targeted at research that is pursued as a team project. This is largely due to the nature of cutting edge research: the knowledge and expertise of each team member is the result of a significant amount of experience and study, such that it is impractical in most cases to expect one team member to be able to perform numerous roles. Certainly, familiarity with each other's
CHAPTER 6. FRAMEWORK ELEMENTS

discipline is of great use as it helps with the greatest challenges in collaborative work: communication and decision-making. Still, even when faced with limited familiarity with other disciplines on the part of some or all members of an interdisciplinary project, the dedicated application of collaboration-centric practices can be an effective way of approaching this obstacle.

The first challenge is effective communication, which is discussed in Section 6.4. The other challenge is decision-making. With respect to the kind of research projects considered in this thesis, this refers to the process of discussion and debate that leads to decisions made in terms of the model design, assumptions, and program functionality. In some cases, mathematical algorithms for integrating the opinions of various experts exist, such as with fuzzy logic [109,147]. When this is possible, it is an attractive option, since it provides some rigour to aspects of the model design that may otherwise be difficult to justify, particularly when dealing with qualitative criteria. However, the formal nature of such a process may be discouraging for some groups, since it discourages direct interaction between expert participants and a more creative synthesis of their ideas. Thus, it is alternately possible for group members to share ideas and try to organically come up with communal decisions that are justifiable within the application field of the model subject. For this, it can be useful for one team member to stay out of this process and independently consider design decisions, particularly if those decisions have been well-posed (for example, in a survey or table). Comparison of conclusions can be useful in validating decisions, or highlighting areas that need further discussion. We have found that the input of non-experts can be useful in this process as well, as a kind of outside perspective on the subject phenomena. Reaching agreement on model design decisions can require an investment of time, but are achievable with some patience and humility.

It is important to note that project decisions do not necessarily require agreement in all cases. It is perfectly possible for alternate models to be considered experimentally by implementing both options in the software. This can be an effective avenue of exploration, and can also help different team members achieve their research goals. It is practical in two general cases. The first is when the alternatives being considered relate to one isolatable part of the model. In this case, both options can be designed in code, and the user of the software can pick the option they wish to test with each execution. The extra time it takes to build alternative model options can be much less than the effort it would take to get entrenched theoretical positions to accede or compromise, particularly when the
programmatic differences are fairly straightforward (as they often are). The second general case for implementing multiple models is when one model is a refinement or extension of another, simpler model. This means that the model is built and tested in two stages: the simpler one followed by the more complicated one. While some team members’ research goals may not be directly met by the simpler version, this process ensures that the eventual model is built upon a design that has been thoroughly considered in itself.

6.4 Clarity

It is important for all researchers involved in the project to be able to understand and identify how the computational model works, or in other words, what is going on “under the hood”. This does not mean that all members should know programming, however, their understanding of the model will allow them to recognize issues that need to be addressed and either make the alterations directly (through a UI perhaps) or be able to communicate directly to the development members. This is especially important with an experimental project, since vital characteristics may change with each experiment, and may diverge significantly from the design agreed upon in initial phases. Clear representation is necessary for this, and graphical feedback can provide a satisfying way of achieving this. It is important to remember that the goal of any such graphical output is to communicate the behaviour of the program, and not to give an illusion of validity or to otherwise hide inner mechanisms. It also will make it easier to confirm the validity of the computational model. A program with obscure innards is useless to a researcher: they have no confidence in its results (it is just bells and whistles to them), and it cannot be used academically, because the source of any results it gives cannot be explained, so it is not convincing to peer readers.

The parts of the program related to the simulation model (not technical aspects like the GUI, visualization, etc.) should always be visible to users. Having it visible at all times would encourage the writing of clear code, and allow users familiar with the programming language to verify that the computational description of the simulation entities matches their expectations. It also encourages peers to check over the relevant code while testing out the software, without forcing them to sift through the entirety of the code base. A straightforward specification language (e.g., Abstract State Machines [16]) is ideal for this purpose. Declarative programming languages are another option for producing code that emphasizes model content and relations over implementation details.
Thus it is important to ensure that communication regarding model content and structure be truly understood by all relevant parties. A number of approaches can help with this. As described in the software development chapter, formal methods, and in particular Abstract State Machines, can be valuable in laying out the model in means that is both human-readable and formal in a mathematical sense. This allows the design to be validated simultaneously in terms of expert knowledge, mathematical soundness, and through execution (in the case of CoreASM). Similar results can be achieved by writing code in such a way that it is closer to pseudo code. This is easier to do with a dynamically typed language, such as Python, where technical terms within the code are fewer and thus less likely to interfere with the examination of the code by a non-programmer. The trade-off with declarative languages is that it is harder to avoid bugs more easily identified with statically typed languages. Further, with expressive data structures and function naming, it is possible to write code in many common languages in such a way that it is not difficult to parse by someone with at least some familiarity with general program structure.

**Control State Diagram Editor—CSDe**

Control state diagrams (CSD) are a class of abstract state machine that can be depicted graphically. They are graph structures with three classes of node (states, rules, and guards) and edges which show the control flow through the nodes. Conditions direct the flow of execution; rules denote actions taken as part of state transitions. The expressive flexibility of this type of ASM is demonstrated by their capacity for representing many classical automata such as various extensions of finite state machines. Since, by definition, they can be depicted graphically, they are a sound foundation for the visual modelling of this particular class of ASMs. This simple visual formulation of a specification is appealing as a way to capture a formal idea in a manner that is accessible to people not familiar with the formatting of program code. While drawing CSDs is not particularly onerous, being able to create and edit them on a computer opened up some powerful possibilities. The most important is that, since each CSD can be represented as an ASM, automatic translation of a CSD into ASM code is possible. This code is not guaranteed to be executable as is, but it is a solid starting point for constructing such code, allowing the main structures and process to be identified before ensuring all syntactic requirements have been met.

The Control State Diagram editor (CSDe) is a software tool for creating and modifying Control State ASMs (see Section 6.4). It is distributed as a plugin for the Eclipse
software development suite. It was built for the Eclipse plug-in architecture and uses the Eclipse Graphical Modeling Framework as a foundation. The plugin allows the user to work with Control State Diagrams (CSDs) using a point-and-click schema. Both the simplicity of CSDs and the intuitiveness of the graphical interface work together to allow users to confidently contribute to the design, regardless of their technical background. To increase expressiveness, multiple conditions and rules for transitions between control states as well as the turbo ASM mechanism [16] of sequential behaviour are permitted within a rule block, in addition to the usual parallelism. The executable ASM language CoreASM also has a plug-in for working in Eclipse, making it a natural choice. The user is able to select the main structures (states, rules, conditionals) and relations (next, true, false) and arrange them in a drag-and-drop fashion. Once done, the diagram can be translated to CoreASM at the touch of a button. The translation algorithm analyzes the underlying graph structure of the diagram by looking at its corresponding XML file, then builds a CoreASM file with a corresponding structure using the content of the diagram elements. The final product pops up in the user’s development environment, providing an instantaneous translation into ASM program code. This automated translation from a diagram into code improves the ease of transition from high-level design towards subsequent stages of development.

Tools like CSDe encourage the direct involvement of non-computing experts in the design and development process of research software. Arbitrary design choices made by computing experts not intimately familiar with the social system under study are potentially dangerous and can lead to fatal design flaws due to misconceptions or oversights. However, it is usually difficult for non-computing team members to understand the development process and especially the formal representation of a system. Hence, it is necessary to make development as transparent as possible, for instance, by using visual representation means, such as **ASM control state diagrams (CSD)**\(^1\) as illustrated in Fig. 6.5.

Despite similarity to the more complicated UML activity diagrams, ASM CSDs do not require any special training to understand. Their simplicity allows the interdisciplinary reader to focus on the content of the description rather than the formalism. The accessibility and ease of use of CSDs make them an integral part of our design process. In our experience, the domain experts were able to understand a CSD, and even suggest changes to

---

\(^1\)Control state ASMs provide “a normal form for UML activity diagrams and allow the designer to define machines which below the main control structure of finite state machines provide synchronous parallelism and the possibility to manipulate data structures” [16].
it, regardless of their technical background. As such, CSDs act as both a means of clarifying communication between development partners and of enabling straight-forward validation.

The versatility of CSDe offers an efficient way to perform the early phases of design and validation in a manner that is non-technical yet still fundamentally formal. In most projects, domain experts and technical specialists must work together to come up with a valid design. However, the specialized knowledge that makes such a collaboration so useful also acts as an obstacle to communication, since they may not share a common vocabulary. CSDe circumvents these issues by providing a clear and precise method for working with ideas at a high-level. By describing concepts in a straightforward, visual manner, all project members should be able to participate fully, regardless of their background.
Part III

Case studies
Chapter 7

Agent-Based Modelling Projects

7.1 Introduction: Agent-Based Modelling

An appropriate technique for modelling social systems is agent based modelling (ABM). This is strongly related in structure to multi-agent systems (MAS) in computing research. However, whereas in MAS the focus is on the robustness and structure of the system and forms of interaction among agents, with ABMs the concern is the activity that is being represented by the computational agents [72]. The agents themselves represent individuals or groups as entities co-existing in an environment. There is no universally accepted definition of an agent, but generally, they must have goals, be capable of action, and make independent decisions. They may also have memory, relationships with other agents (sociality), and limited perception of their environment [65]. These qualities make ABM ideal for simulating the complex behaviour of living organisms, and in particular (from a social science perspective), people. Macy and Willer describe ABM as a transformation of purpose from prediction to thought experiment. They list a number of tips for ABM projects. These fall into two categories, the first of which are reminders to implement various validation processes within a simulation project. The second category concerns how simulations should fit into the practice of pursuing social science. ABMs should not only explore the theoretical possibilities of a scenario, but should also be capable of execution as an experiment, testing a specific hypothesis. And while ABM are built on the concept of the individual agent acting within a society, ABM simulationists should not lose sight of macrosocial factors, such as the interaction of different groups within a larger society [107].
7.2 The Mastermind Project

The Mastermind project began in 2004 at Simon Fraser University as a collaboration between the Software Technology Lab and the Institute for Canadian Urban Research Studies. Mastermind is a simulation environment for examining criminology computationally, employing formal modelling and simulation as tools to investigate offender behaviour in an urban environment. The project aims at developing computational models of criminal activity patterns, with a special focus on spatiotemporal characteristics of crime, potentially involving multiple offenders and multiple targets. The Mastermind project utilizes the ASM method and the CoreASM tool suite to address the specific requirements of developing computational models and analysis tools for the study of crime in a collaborative research environment.

7.2.1 Conceptual Modelling

One goal of Mastermind was to show how Activity Space and Awareness Space work. Activity space includes the places we move through in our regular activities, as we go to work and come home, visit friends and family, go shopping and so on. Awareness space encompasses activity space, and it is the area which we have general knowledge about, either through observation or through communication with the people we know [25]. For example, it is typical to know something about a neighbourhood through which we pass regularly, even if we do not travel down each road in that neighbourhood every day. These concepts are important for criminology because, according to Routine Activity Theory, criminals are normal with regard to the large part of their activities. They pursue crime opportunistically, when they notice a potential target within their awareness space. The process of simulating criminal target selection was another goal of the project. By constructing a simulated environment and having agents follow routine activities with the potential to select targets for crime, it was possible to show how these important ideas in environmental criminology could work together in a logical fashion.

Crime is understood to be comprised of four main elements: the law, the offender, the target and the location [24]. With Mastermind, we have constructed a multi-dimensional model of crime in order to study the interaction of these elements. Our focus is on the concepts of environmental criminology, which argues that in spite of their complexity, criminal events can be understood in the context of people’s movements in the course of everyday
routines [24]. Through movement within a given environment, possible offenders, characterized as agents, develop mental maps of the places they know (awareness space) and the places they regularly visit (activity space). At its core, Mastermind captures the essence of the Crime Pattern theory, i.e. crime occurs when a motivated individual encounters a suitable target. Figure 6.5 captures this behaviour in terms of a Control State ASM.

Crime is an event that occurs when an individual with some criminal readiness level encounters a suitable target in a situation sufficient to activate that readiness potential [25].

### 7.2.2 Mathematical Modelling

The main building block of Mastermind is a robust ASM ground model [15] developed extensively through several iterations required for checking the validity of the model with respect to the understanding of domain experts. For details on that model, please refer to [141]. The process of establishing the key properties, determining the right level of abstraction, and ensuring the validity of the model was greatly facilitated using the simple graphical notation provided by CSDe and by the ability to run experiments on abstract models in early stages of design using the CoreASM engine.

It is important to emphasize the role of the CoreASM tool environment in facing the challenges of two major phases of our project, namely formalization and validation. In an interdisciplinary research project, the communication problem is intensified, imposing serious challenges in ensuring a correct transformation from domain knowledge to computational artifacts. This difficulty is compounded due to differences between academic disciplines in terms of approach and underlying assumptions, not to mention the fact that real-life events, such as crime events, are not usually thought of in a discrete, mathematical manner. To this end, diagrams created by CSDe greatly facilitate an interactive design process where domain experts are able to directly check and correct a design.

### 7.2.3 Computational Modelling

The ground model has been further refined into more concrete models with specific details systematically added, an example of which is the simulation model of Mastermind implemented in Java. The Java version captures the navigation behaviour of offenders with a
high degree of detail. It also boasts a responsive user interface that allows for setting experiment parameters and changing the user perspective of the road network. The CoreASM executable ground model was refined to run more controlled experiments, which allows for a structured analysis of theories in a hypothetical world. These simple and comprehensible models provide domain experts with full control over the variables under study and their interdependence. Both versions also provide visualization features which are a priority for criminology publications.

Neither of the versions of Mastermind were particularly portable. That is, they were not easy to transfer between different computers. The first version uses OpenGL for its graphics functionality, which means that those graphic libraries need to be installed on any computer intending to run the program. This is not a difficult task for someone familiar with programming or operating system management, but can be intimidating and confusing for a general PC user. Likewise with the second version, coreASM needs to be installed on any computer that will run the program. Lack of portability proved to be a large obstacle for engaging domain experts (in this case criminologists) in the software itself. The graphical capabilities of OpenGL were in the end not worth the trouble when it came to sharing the software with other team members. Nowadays, many programming languages (such as Java) include graphical libraries that are quite powerful, so this issue is not as serious as it once was. Still the general lesson about portability is important: a program that can be shared by e-mail or web, and that can run without fuss on almost any computer will get much more use than one that has greater requirements. Being able to distribute each update in an uncomplicated way through a common means such as e-mail or the web supports the emphasis presented here in keeping the iterative research loop as tight as possible.

7.2.4 Experimental Set-Up

Figures 7.1 and 7.2 show snapshots of the first two implementations of Mastermind. The visualizations show agent movements between activity nodes, the formation of their activity spaces and the effects on crime hotspots. The CoreASM model is meant to study concepts at a higher level of abstraction, using a simple grid structure in the example. In contrast, the Java version runs on the real road network of downtown Vancouver, including Stanley Park, and captures a finer degree of detail and complexity. Experiments run on the Java version (comparing different navigation preferences) are described in [22].

On the other hand, using the CoreASM executable ground model as a basis, it has
been possible to derive specific refinements to generate more controlled experiments. The simplicity and control provided by such models allows for a structured analysis of theories in a hypothetical world, where the researcher has full control over the variables under study and their interdependence.

### 7.2.5 Execution

The CoreASM version raised an important issue: visualization. Feedback from domain experts highlighted the importance of an interface that was easy for them to read. Using a heat map style of colouring to indicate the amount of movement along road segments was one advancement from this discussion. Also, close integration with Microsoft Office allowed
Figure 7.2: The CoreASM implementation of Mastermind: this version is more abstract, focusing exclusively on specific elements.

easy conversion of data to coloured plots and tables in Microsoft Excel. From this point forward it became clear that this kind of functionality is important to include in all of our projects. First, the data should be displayed in such a format that it is clear and intuitive to read. Second, multiple displays of the data from the simulation should be available: it is simply hard to show everything clearly in a 2D or virtual 3D format.

7.2.6 Results

The CoreASM version of Mastermind was an effective demonstration of something known well in the study of property crime: people who spend more time moving about the environment, such as door-to-door salespeople and delivery people, have more potential to notice good criminal opportunities. The logic that supports this observation is fairly straightforward: assuming opportunities are present throughout the environment, more time spent travelling through the environment means more chances to notice those opportunities. Not only did the behaviour match the theory, it had a deducible mathematical reason: random
travellers were less efficient in getting from point A to point B, and thus spent a greater portion of their time travelling compared to other agents. Their greater hours spent travelling translated directly into more chances to notice criminal opportunities (or alternatively, better knowledge of an area of criminal opportunity).

7.2.7 Iteration

While designing the computational model of agent activity, it became clear at one point that we had no model of the criminal target selection mechanism. In other words, how criminals decide whether a certain target is a good choice for criminal activity. We addressed this deficiency in the conceptual model through discussion with domain experts. Knowledge on this particular topic was plentiful, but we were unable to generalize it into a rich model of target selection, so instead the model of target selection was kept very simple. This decision helped to keep our uncertainty regarding that topic clear, so that we were not building something we (as modellers) did not understand well or have great confidence with regard to a detailed implementation. The model of target selection we did develop very succinctly uses a stochastic function to encapsulate the behaviour we wanted to emulate but could not model more thoroughly. In this way, the model as a whole works, and we can substitute a more refined version of target selection when one is available.

Development of the full-fledged Mastermind simulation model in Java preceded the simpler, more abstract CoreASM model, and this was a valuable exercise to compare their utility as scientific models. The complexity of the Java version and the fact that it is considered as a black-box by domain experts introduces limitations on its academic usage. On the other hand, the CoreASM program code is easier for non-programmers to read. It is also well-suited for designing controlled experiments, by allowing the agile and precise construction of abstract models that contain only the current elements of interest. Taking advantage of the highly flexible plugin architecture offered by CoreASM, it was possible to rapidly develop the Mastermind Plugin to address the specific needs of criminologists, especially with respect to visualizing the results. In other words, the Mastermind Plugin encapsulates the mathematical structure of the ASM model in a comprehensible and familiar format for domain experts. This greatly facilitates communication with domain experts and analysis of the results for validation purposes.

One of the successes of Mastermind was the inclusion of real geographical data. This lent the simulation great realism, since the network the agents traverse is one dealt with by
real humans every day. The general point here is that if rich, accurate, and/or appropriate data is available, its inclusion in a simulation can be very beneficial. Conversely, theory regarding criminal target selection was difficult to generalize, and including it in the model was difficult. There was no lack of information, but it was too specific to individual crime types to be operationalized into a rule for how target selection works in general. However, it is an indispensable part of crime activity, and needed to be included. Thus it was modelled as a simple process, which worked well for the purposes of the project at that point. The level of detail also matched our confidence in the element—lack of knowledge was not hidden behind an arcane or complicated design. Modelling often raises these issues: how precisely should something be represented in a model, and what is our level of confidence of that element as a representation of reality. Further, such decisions should be included in any in-depth description of the model.

The results of work on the Mastermind project have been well-received by both researchers in academia and law enforcement officials. For more details on the project and the results, please refer to [21,22], and also the project website at www.stl.sfu.ca/projects/mastermind. The Mastermind project demonstrated the necessity of a robust and extensible, yet flexible, design that can be easily re-applied in new experiments. It also illustrated the importance of reducing the communication barriers and increasing the quality of communication between team members in a software development project that aims at generating results of research caliber.

7.3 SocialMind

SocialMind is a recent extension to the Mastermind project. The goal of this new stage is to better model the dynamic nature of activity space and awareness space. Agents in the system are all potential or active criminals, and they build up social links over time. For the purposes of this project, Awareness space has been defined to be the union of one’s own activity space with the activity space of one’s acquaintances. In other words, we know about the parts of a geographical environment that we personally visit, or that is visited to our friends, since they will relate information about the places they know through everyday communication. Our second important assumption is that activity space is built up through travel along the road network, and decays over time, in a manner similar to ant colony optimization models [40]. The general concept is that people know best the areas
they have visited most often and most recently. Each agent’s view of the road network is like a heat map built up through travel activity, and which cools over time.

The JUNG graph library has been used in SocialMind to store graph structures and display them. This includes both the road network in which the agents exist, as well as their network of social connections. These are displayed separately and update simultaneously. As can be seen in Fig. 7.4, the user interface includes options for manipulating the visual organization of the nodes according to existing algorithms, such as Kamada-Kawai, Fruchterman-Rheingold, or as a circle. JUNG makes working with graphs much easier, as in this case, where the arrangement of nodes in the display can be altered if, for example, the current arrangement is messy and difficult to interpret.

Figure 7.3: The road network in SocialMind: this map is built using the same data as in Fig. 7.1, but uses JUNG instead of OpenGL for visualization.
There are a number of refinements to the model which have been envisaged. First, social connections can be represented instead as a non-binary, fuzzy state, so that the quality of information shared by acquaintances is variable. Thus awareness would not simply be the union of awareness spaces, but instead construct a weighted awareness space generated considering the strength of relationships between an agent and their acquaintances. So, the awareness space of a close friend would be a much greater source of knowledge than that of someone not known very well. Transferring awareness space information through the network through common acquaintances has also been considered. Each of these refinements requires more discussion and design before implementation, but they are promising directions for our next stage of research.
SocialMind is still in its initial stages, but it is a good example of how a software modelling project can react to research interests. The changes over time in the Mastermind series of projects has been largely due to the different kinds of questions addressed:

- *Field-specific:* What is the effect of navigation type on criminal target selection?
- *Technical:* Which programming language and/or visualization library should be used? How should we deal with distribution and portability of the software?
- *Methodological:* How abstract or realistic should the model be for the current topic?

This flexibility is the hallmark of software as a medium for modelling. As can be seen here, it is important to prioritize methods and practices that facilitate that powerful attribute.
Chapter 8

Cellular Automata Projects

8.1 Introduction: Cellular Automata

Cellular automata (CA) model time (as steps) and space (as cells in a grid) in a simple but functional manner, making them both expressive and useful as a modelling approach. This grid is usually two-dimensional and toroidal in nature, but different configurations are possible. In a CA model, a population can be represented in a two dimensional square grid where each cell represents an individual in the population [31]. The state of each cell can vary depending on pre-designated rules. These rules are derived from an existing theoretical framework describing a particular phenomenon and are used to model what is happening in the real world. In the case of this model, both deterministic and stochastic rules are used. A CA model can effectively capture social interactions that happen over time [2, 75]. Since each cell has the capability of holding the information pertaining to that cell, changes can be recorded. In general, CA models measure time discretely, in other words, progress through time is represented as a series of time steps. The cells capture the information at each time step and their states can alter through successive time steps. A variety of transformation rules and neighbourhood definitions are well-established [82].

The cells in a CA can represent biological organisms, geographical areas, or people in a social network. There are even examples of CA found in the natural world, such as the shell patterns of cone snails. CA provide a simplified way of representing any kind of population, and are arguably the most general agent-based model. One criticism is that cells only have a linear number of connections to neighbours, which does not mimic the power law distribution found in scale-free networks. This is important since the scale-free
property can be found in many social phenomena, such as webpage connectivity. However, since all calculations are based on local information, the computational complexity of a CA is linear in terms of size. This means that grids of arbitrary size are possible with sufficient computing power. This characteristic combined with the simple structure means that CA are an attractive choice for modelling: they are both flexible and accessible for structural analysis. Fig. 8.13 shows a sample CA from my own research.

Since CA are a mathematical tool capable of modelling dynamic behaviour based on local relationships, they are appropriate for simulating phenomena with a geographical component. Notably, they have been used in numerous models of urban development [8,102]. In particular, two-dimensional CA can be displayed effectively on a computer screen or in print, which enables decision makers to visualize the interactions happening over a city or village and see the effect of selected policies. CA also support decision makers by providing an intuitive means to explore emergent behaviours (i.e., high level patterns emerging from low level interactions), a characteristic of complex social systems. Finally, since they scale efficiently with respect to the size of the simulation, CA are not limited to experiments containing few active elements, unlike some other modelling techniques. Regardless of the size of a CA grid, computation is of linear complexity in terms of the number of cells.

8.2 Urban Migration

Residential urban migration is an integral aspect of city and crime control planning. This type of movement does not occur uniformly over time and space. Rather, there are specific social dynamics occurring within cities that render certain areas more transient and others more stable [18, 24, 57, 89, 122, 134, 139, 154]. Furthermore, other social problems overlap the transient areas, whereas there are fewer issues in stable neighbourhoods [51, 57, 89, 122, 139]. For instance, there is more crime in areas where there is high residential mobility, but less in established neighbourhoods [57, 89]. While this is a simple illustration of the social problems related to higher residential mobility, the underlying mechanisms of urban migration continue to be of concern to a number of social agencies. The growing availability of data in this area provides researchers with the empirical knowledge to further investigate this urban phenomenon [3, 4, 7, 20, 86, 161, 162]. In order to effectively study the complex social processes involved in urban migration, a non-linear mathematical method needs to be applied [8, 75, 100, 101, 103, 153].
At the beginning of the 20th century, as American cities came to expand at a rapid speed, criminologists turned their attention to urban migration because they found it followed the aggregate crime patterns in a city [23, 122, 139]. In 1925, Park and Burgess developed the first concise criminological model to describe the relationship between urban migration and crime. The concentric zone model was derived from data collected in the city of Chicago and as such followed the urban development occurring in that city [122]. As an American city grows, the more affluent move to the periphery of the city while poorer residents stay close to the central business district where they have easier access to transit and work. Social disorganization is a centrally important concept emerging from this model. It is a reciprocal social process which accompanies the urban growth of American cities and with disorganization comes reorganization.

The increasing availability of data from the 1940s onward allowed McKay and Shaw to apply juvenile delinquency rates to the Concentric Zone Model in Chicago [139]. They found these rates followed the zones and delinquency was more concentrated in zones of transition where social disorganization was physically apparent. These results were also reflective of delinquency patterns in other cities. The authors conclude that crime is most prevalent in transitional zones where the majority of delinquents reside. Transitional neighbourhoods are defined as inner city areas where population movement is high, where there are more individuals who have lower education and median family income, and where housing is in a deteriorated state.

During the 1980s criminologists reviewed the link between crime and urban migration [18, 24, 134]. For instance, Brantingham and Brantingham (1984) study the crime patterns of several cities [24]. They see the Park and Burgess Concentric Zone Model as a compelling explanation for crime rates in urban areas from the 1930s to the 1950s. However, the zones are too large a unit to properly analyze and even McKay and Shaw abandoned this model in their later research. The areas of social disorganization are sometimes dispersed throughout the city and the patterns are not concentrically predictable. Rather, smaller urban areas are better suited units with which to investigate urban crime patterns. The Brantinghams found that three structural factors determined higher crime rates: residential mobility, low social economic status, and ethnic heterogeneity.

The reciprocal processes of social disorganization and reorganization are examined by Sampson and Groves (1989). In their study, they build on the determinants of social disorganization developed by McKay and Shaw, however, they also develop a social organization
construct which allows them to test the reciprocal nature of these two factors [134]. Linear regression analysis is used to look at this relationship. There are limitations to this approach as it cannot fully describe the dynamic interplay existing between these two constructs. Indeed, urban migration happens in a non-linear fashion in space and in time and is best studied using non-linear mathematical methods [4, 101, 103, 142, 144, 149, 152, 153].

More recently, the study of urban migration has gone further than the aggregate studies previously discussed to now include the micro factors that influence a household’s decision to move. More specifically, researchers have looked at the impact of victimization, both direct and indirect, on the decision to move [41]. This variable is also compared to other household specific factors such as unemployment, education and age. The results of these studies show the confluence of influences on this decision [10, 41, 154]. Indeed, a household’s decision to move is triggered by internal factors such as a change in income level, but also indirect factors such as neighbouring households being victimized by crime.

Additional research has shown that certain neighbourhood features such as school catchment areas or questionable establishments also influence the household’s decision to move [10]. These can be considered as positive or negative social attractors. Positive social attractors are associated with the institutions within the community that further strengthen the cohesion of a household to shared values and expectations. For instance, a school with a good reputation attracts households that value such positive community attributes. On the other hand, negative locations such as bars that are likely to attract crime may be viewed as negative social attractors [131]. The decision to move closer to either type of social attractor, be it positive or negative, would resonate with the social characteristics and attitudes of the household making such a decision.

The decision to move may be dependent upon multiple factors, such as trading up with the aim to save for retirement, a new birth in the family thus generating need for space, or other factors such as loss or gain of employment. However, in the decision to move, households will balance the pull of the community with the quality of the house [33]. Research findings indicate that a household looks for both better neighbourhoods and better housing when they move. Similarly when neighbourhoods begin to decline, a household will look to leave in order to secure the asset [89].

The Schelling one-dimensional cellular automata model of racial segregation due to residence selection was a ground breaking use of mathematical modelling of social phenomena [137]. A more recent and complex attempt at modelling racial segregation can be seen
in [27]. Over the past decade, mathematical modelling has been applied to the study of crime and urban development [4, 92, 100, 101, 103, 140, 142, 144, 149, 152, 162]. This type of method is highly applicable because it can model urban dynamics both spatially and temporally [8]. The mathematical approach used in this study is cellular automata (CA) modelling, which can effectively model the non-linear qualities of dynamic human interaction among the individuals in a community [8, 31]. A CA model can further assist in describing and understanding urban migration patterns [8]. This analytical method is suitable for this type of research because it can take into account both local interactions and also more distant influences. Here, a cell is used to signify the locations resided in by households and the transition rules for these cells are used replicate the social interactions between households. The goal of this study is to present a mathematically structured method to describe the dynamic social interaction of individuals in a community. A CA model of urban migration is developed to show the social relationship between households. This model effectively describes the impact of the social characteristics of a household on the decision to stay or move.

8.2.1 Conceptual Modelling

The objective of this CA model is to simulate the impact of social structure in the household on residential migration in an urban area. The underlying premise is that households are biased and want an appropriate place to live based on their social characteristics. We use the concept of social structure to represent these characteristics. Social structure refers to a combination of household variables such as the average age of residents, average income, number of parental figures, employment status, and criminal propensity of residents within the household. These variables affect the behaviour of the people residing in the household. A positive social structure value indicates adherence to social norms and lawful behaviour, while negative values indicate an emphasis on personal freedom and lack of community duty. Extreme negative values can also indicate criminal inclinations. Notably, the social structure of a household is influenced by others in the community. In this model, each cell can potentially be influenced by neighbours in its neighbourhood.

The social structure of each household is represented by a single value that changes over time based on the dynamics of the CA model. Deciding to model social structure as a single value was an important part of the iteration between the conceptual and mathematical modelling stages of this project. This approach avoids the difficulty of operationalizing the
individual factors and instead focuses upon their cumulative effect on residence choice. The decision to move emerges when there is a significant discrepancy between a household and its neighbours. Once the decision to move is formulated, the household selects a position in the model based on its own social structure and on neighbourhood social attractors. The model is executed using scenarios to demonstrate the decisions to move by households and the consequential effect of positive and negative social attractors in the neighbourhood.

This cellular automata model is based on four general assumptions. First, households frequently experience minor changes in their social structure due to relatively inconsequential events [7, 10, 29, 139]. However, a household will occasionally experience a significant change due to events such as a job promotion, employment loss, marriage or death in the family [18, 29, 134]. These events either increase or decrease the social structure of the household and can result in a move.

Second, the social structure of a household is influenced by the social structure of neighbouring households [18]. In essence, there is a social balance between households and when a sharp contrast emerges, the balance is offset and this can trigger a move [18, 122, 139]. This assumption is supported by research findings on how community structure can either positively or negatively influence behaviour. For example, neighbourhoods that contain residents with similar social characteristics promote shared supervision of community residents, thereby diminishing the opportunity for unregulated problematic activities [134]. Conversely, socially disordered neighbourhoods (such as those dominated by low-income, unemployed or single parent households) can impose a negative influence on their residents through a non-cohesive and less trusting community structure that offers minimal community supervision and fewer pro-social adult role models [24, 154].

Third, a household’s decision to move is influenced by the difference between its social structure and that of its neighbours. This may refer to situations in which social order declines and leads to poor community cohesiveness and increased criminal activity [29]. In addition, a similar discrepancy between the social structure of a household, and that of its neighbours, may be brought on by a new job or a growing family, either encouraging a move into an area with a higher social structure, or necessitating a move to an area with better access to schools. While the converse situation of gentrified neighbourhoods leading to displacement of current residents may exist, there appears to be limited evidence to support this process [57]. Regardless, our model generalizes this process and considers both possible types of social change.
The fourth assumption of the model states that a household’s decision to move is influenced by neighbouring households and the positive or negative social attractors \([10,20,26,92]\). A household will move to the most appropriate position considering both the neighbouring households and the proximity of social attractors.

There is no representation of immigration, emigration, births and deaths in the model at this stage. By assuming that these phenomena balance each other out, the model is simplified, thus allowing the key concepts to come forward. Any explicit inclusion of such phenomena would introduce additional complexity that is unwarranted at this stage of the project. In future iterations, these variables can be introduced as the influences of social structure become clearer. This model is theoretical in nature, and aims to act as a broad simulation of a complex concept, rather than an empirical model aimed at testing cause and effect. A complex subject is captured using a generalized variable which encompasses a broad range of factors included in the decision to move. This can include a household’s decision to move to a larger home, or to a neighbourhood with better access to schools, just as it captures the decision to move as a result of increased crime and social disorder occurring in the immediate neighbourhood.

### 8.2.2 Mathematical Modelling

The model is implemented on a grid of cells that replicates an urban area. Each cell represents a land parcel in which one or more households may reside, depending on the capacity of the cell. Not all cells are occupied by households in order to facilitate household migration. At each step of the model, the value representing the social structure of a household is subject to change based on a random variable that triggers a change in the household (assumption one from previous section). Next, the neighbourhood of each household is evaluated by calculating the average social structure of the surrounding cells. Households then make the decision to move when their social structure is different from the average social structure of the neighbourhood (assumption three). In order to ensure that households are not moving at every step of the model, a probabilistic function is used to determine when they will migrate. The chance of a move occurring is dependent on the dissatisfaction of the household with their neighbourhood, or in other words, the difference between their social structure and the average social structure of their neighbours.

A household with a different social structure from its neighbourhood engages in a search of all cells in the urban area in order to find a cell that best matches its social structure.
(assumption four). The household will remain in its current location if no vacant cells exist. If the household does locate a cell that meets these criteria, it will migrate to the new cell. All households that do not move and have been in their current neighbourhood for a minimal time period have their social structure recalculated based on the average social structure of the neighbourhood (assumption two). That is, the social structure value of a non-relocated household will move closer towards its neighbourhood value. The updating of social structure values concludes a single time step of the model.

At each time step in the model, two processes are performed to update the states of the cells and households. Updating the neighbourhoods consists of two parts:

1. Calculating the average value of each cell

2. Compiling a list of vacancies

In the case of a single-occupancy cell, the social structure of the cell itself, along with its occupied neighbours, is used to generate the average value. For a higher capacity cell, the social structure value of the occupants in that cell is used to generate the neighbourhood average. Cells that are not completely full are compiled into a list of vacancies, which is organized into groups based on social structure. Figure 8.1 is comprised of this process and its two parts, along with the process to update the households.

In this process phase, the changes in each household for one time interval are processed. If the household decides to move and an available location exists, they will relocate. Of the available locations, it will select from the locations with social structure closest to its own. If no available location exists, the household will stay at their current location, and if they have been there for more time than the threshold value, they will be socially influenced by their neighbours. All households will then have their social structure adjusted by a random value, in order to reflect changes experienced due to personal events in the family. This random value follows a normal distribution, so it will be a very small change in the majority of cases, with the occasional large change in a positive or negative direction. This models the changes that occur in daily activities in the sense that both minute everyday developments and life-changing disruptions of both the positive and negative variety are represented with appropriate intensity and frequency. The final step of this phase is to truncate the social structure value in the case that it has gone beyond the maximum or minimum value.

At each step, all households undergo varying amounts of change to their social structure. This change in social structure is influenced strictly by internal factors. In other words, an
event such as the death of a family member or a promotion at work is responsible for a large alteration of the social structure. External, or neighbourhood factors, begin to influence a household when it has remained in its current location for more than the minimum amount of time needed to settle in and start interacting with neighbours.

Figure 8.2 depicts the decision-making process undertaken by each household for each time step. This process is driven by the social structure of said household relative to its neighbourhood. Based on this relative difference, the household probabilistically determines whether or not it wants to move. A household which possesses both the willingness and ability to move identifies an appropriate vacant area from the queue and proceeds to move there. Once moved to its new location, the household is assigned a time stayed value of zero. Once the time stayed has passed the threshold, a household continues to consider the option of moving at each time step; however, it will not move until it is sufficiently dissatisfied with the current location and there is an alternative location available. If it stays at its current
location, it is only subject to the influence of its neighbours as well as changes in its personal life. Note that it is possible (although unlikely) for a household to move to a location that matches its social structure more poorly than its current location. However, it may have an easier time influencing its neighbours at the new location, and bringing the social structure closer to its own, so this possibility is permitted.

This CA model attempts to replicate the physical interaction between households within a neighbourhood. The cityscape, represented in this model with a grid, can easily be conceptualized using the most popular symmetric CA neighbourhoods such as a von Neumann or Moore neighbourhood (Fig. 8.3).

The location of cells in the grid is denoted by \((i,j)\) and \(N_{ij}\) shows a neighbourhood of \((i,j)\). The parameters used in this model are classified into two main classes:

1. household parameters: denoted by \(H\)
2. location parameters: specific to each cell in the grid

This model is scenario-based and the parameters with the rules that govern households and the locations in the grid are meant to reflect the social dynamics underlying urban migration.

**Household Parameters**

Household parameters represent the social characteristics of the households at each time step.

- \(T\): Time to settle into neighbourhood
- \(T_{ij}(t)\): The length of stay in \((i,j)\) at time \(t\). This is a counter and it resets to 0 when the household moves.
- \(s(t)\): The social structure of the household \(H\) at time \(t\) and varies from \(S^-\) to \(S^+\). It is randomly set at the beginning of the simulation and varies after that. This value is stored internally as a decimal value, thus is capable of minute changes. When examined externally (by other cells or when generating \(V_{ij}(t)\), below), it is rounded to the closest integer value. This discrepancy in numerical representation is intended to capture the difference between true knowledge of one’s own situation, and rough or imperfect knowledge of other households and locations.
Figure 8.2: Decision making process of an individual household.
There is a lapse time \((T)\) built in to the households to allow them to settle into a neighbourhood. During this time, they are not influenced by their neighbours. After \(T\) has lapsed, the social influence factor is triggered. The social structure \(s(t)\) captures the multiple factors in a household which can influence a household to move or stay. As described previously, it depicts the social characteristics of a household and varies over time according to rules that model various events and interactions.

\(H_{ij}(t)\) shows the state of the household \((H)\) at a given time \((t)\). This is represented by:

\[
H_{ij}(t) = [s(t), T_{ij}(t), T]
\] (8.1)
Location Parameters

Location parameters are characteristic parameters of each cell \((i,j)\) at each time step. These parameters depend on the surrounding cells.

- \(C_{ij}\): Represent the maximum capacity of residents at \((i,j)\). In this model the capacity for each cell is one household per cell (low density) or more than one household per cell (high density).

- \(C_{ij}(t)\): This represents the number of households resident in cell \((i,j)\) at time \(t\). The number of current resident households must always be less than or equal to the maximum cell capacity. \(C_{ij}(t)\) can vary from zero (not occupied) to the maximum capacity (fully occupied) therefore \(C_{ij}(t) \leq C_{ij}\).

- \(V_{ij}(t)\): Value of the cell \((i,j)\) at time \(t\) based on the neighbourhood social structure. \(V_{ij}(t)\) is the average of the households in the neighbourhood.

\(S_{ij}(t)\) is the average social structure of the households that live in \((i,j)\) when \(C_{ij} > 1\). Clearly, \(S_{ij}(t) = s(t)\) in the case when \(C_{ij} = 1\). Let

\[
V_{ij}(t) = \sum S_{i'j'}(t)
\]

where \((i',j')\) varies in the neighbourhood \(N_{ij}\). Then

\[
\begin{cases}
V_{ij}(t) = (1/|N_{ij}|) \sum S_{ij}(t) & \text{if } C_{ij} = 1 \\
V_{ij}(t) = S_{ij}(t) & \text{if } C_{ij} > 1
\end{cases}
\]

where \((i,j)\) varies in \(N_{ij}\).

In other words, if a household lives in a cell \((i,j)\) with capacity 1 then the value of social structure of its neighbourhood is the average of the social structure of the cells in that neighbourhood. This value is the average of social structure of households in the cell \((i,j)\), in the case that it lives in a higher capacity cell.

Since there is no specific definition for the parameter \(s(t)\) and its value depends on a given what-if scenario, \(s(t)\) is labeled a scenario parameter. Following that, \(V_{ij}(t)\) is a scenario parameter. The parameters with specific values are deterministic parameters. In the future, these values can be obtained by collecting data from previous studies. In this model, the parameters \(T, T_{ij}(t), C_{ij}\) and \(C_{ij}(t)\) are deterministic.
Updating Cells

In this model, cells will be updated according to the following three rules. The first rule updates the social structure of a household and the last two rules update cells based on when and where a household moves. At each time step, all cells in the grid will be updated simultaneously.

**Update for** \( s(t) \) The parameter \( s(t) \) can change over time and there are rules that govern this change. This is a scenario-based variable that changes gradually. There are two updates for \( s(t) \): one is internal to the household and one is external. The internal factors cannot be changed by the household.

**Internal influences on** \( s(t) \)

The internal changes in \( s(t) \) are meant to illustrate how certain factors or events in a household can change the social structure of the household. Parameter \( s(t) \) is dependent on these changes and they cannot be avoided. This includes both minor and major circumstances, such as a death in the family, a new birth, the loss of a job or a raise at work. These internal changes affect the social structure of the household, and can alter it enough to put the household in a position to move. At each step, the model is updated according to this internal \( s(t) \) rule:

\[
    s(t) = \max\{\min\{s(t-1) + \epsilon, S^+\}, S^-\} \tag{8.4}
\]

The parameter \( \epsilon \) is a randomly determined value with a normal distribution centred on zero. In this rule, \( \epsilon \) represents the change in social structure that a household experiences at each time step. This change varies at every time step for each individual household. The standard distribution of this value is a scenario parameter.

**External influences on** \( s(t) \)

After \( T \), the time required to settle into a neighbourhood, the household’s social structure can be influenced by other households in the neighbourhood. If the social structure of the household is less than the average of the neighbourhood, then the household will receive a positive influence. However, if the social structure of the household is more than the average of the neighbourhood, then the household will
receive a negative influence. These changes are described in the following formulae. If \( T_{ij}(t) > T \) then

\[
\begin{cases}
  s(t) = s(t-1) + c & \text{if } s(t-1) \leq V_{ij}(t) \\
  s(t) = s(t-1) - c & \text{if } s(t-1) > V_{ij}(t).
\end{cases}
\]  

(8.5)

The value of \( c \) indicates the amount of social influence a neighbourhood has on a household. It is a fixed scenario parameter. It can be considered as the incremental effect on the social structure of a household because of the received social interactions from a neighbourhood.

**Moving from \((i,j)\)** When the social structure of a household is close to the average, the probability of moving is significantly lower. The greater the difference between the household and average, the greater the likelihood of moving. If \( T_{ij}(t) > T \) then it moves with the probability

\[
P(t) = \frac{|V_{ij}(t) - s(t)|}{(S^+ - S^-)}
\]

(8.6)

where \( S^+ \) and \( S^- \) are the upper and lower bounds for social structures \( s(t) \), respectively.

**Moving to \((i,j)\)** When a household moves it will select the most appropriate location, which is the closest one to its own social structure with available capacity. This move is dependent upon the capacity of the cells. Therefore \( s(t) \approx V_{ij}(t) \) and \( C_{ij} > C_{ij}(t) \). As soon as \( H \) moves to \((i,j)\) the time counter resets and \( T_{ij}(t) = 0 \).

**Attractor Influence on the Decision to Move** When a household is deciding to move it will consider locations which have a social structure closest to theirs. Once these locations are identified, the positive or negative attractors in the model are the final step in the decision making process. These attractors can be considered as tiebreakers: if there are two identical locations that are available, the location closest to the appropriate attractor will be selected. There are currently only positive and negative social attractors. Households that have a positive social structure (\( > 0 \)) are only influenced by positive attractors, whereas households with a negative social structure (\( < 0 \)) are influenced by negative attractors. In the case where a household has a zero value it is influenced by both attractors.
8.2.3 Experimentation

For simulation of this CA model, a two dimensional 50 × 100 cell array type was used. Each element of the cell array is a vector storing household parameters, location parameters, internal influences and external influences. These vector elements were updated with time following the transition rules. The model was run for 5000 iterations. The time threshold is set at ten iterations by default, indicating the amount of time a household will spend in a cell before being influenced by immediate neighbours. The move threshold refers to the neighbourhood social structure for a cell, and simulates the required difference between a household and its neighbourhood in order for it to move. Three contrasting household distributions were simulated.

No Attractors versus Positive or Negative Attractors

An initial run of the model, using the default inputs, reveals the spatial patterning displayed in the No Attractors images below. After 5000 iterations, there are distinct clusters of households displaying high social structure values displayed in green cells, and those displaying low social structures are displayed in red cells. There appears to be a transition zone between the areas of high and low social structures, where values average out to near zero displayed by the light green, to white, to light red cells. There are several individual cells with low social structure amidst a generally high socially structured area and, likewise, several display high social structures in lower social structure areas. These different cells within the clusters are multi-household cells, and as such, are more influenced by neighbours within the same complex than single households around them.

A very different spatial pattern is revealed when features are added to the model to represent a positive and a negative social attractor. Almost immediately, households displaying positive social structures start to cluster around the positive social attractors which are located in the upper left-hand corner. Likewise, households with negative social structures start to cluster around the negative social attractor. These patterns are strengthened throughout further iterations, with a near-equal division between households with positive and negative social structures (see Figure 8.4).
Figure 8.4: No attractors versus positive or negative attractors.
Low Occupancy Versus High Occupancy

The default input for single occupancy households in this CA model is 90 percent, therefore the other 10 percent is multiple occupancy, representing multi-family dwellings such as apartments or condominiums. The default maximum occupancy for multiple household cells is 10 households. The occupancy rate is set at 90 percent, indicating that ten percent of available spaces are left unoccupied.

The impact of the amount of available space becomes clear when the results from a low-occupancy run are compared with a high-occupancy run. Households tend to group with similar households if given increased relocation options and defined clusters will form where the social structures of the households are similar. The unoccupied cells, defined by the light grey cells, increase in the low occupancy rate scenario. Further, in reducing occupancy levels, the transition zones which separate areas of high and low social structure which are denoted by the light red, white, and light green cells, increase in size. In this scenario the occupancy is set at 0.7.

Households have far fewer available cells to move into when this model is run with a higher occupancy rate of 0.999. As such, the households are increasingly impacted by the social structure of their neighbours. As a result, there are strict divisions between areas of high and low social structures. While smaller clusters of high or low social structure appear in the initial iterations of the model, these small islands quickly disappear and are replaced by larger zones of either low or high social structure (see Figure 8.5).

Low neighbourhood Influence Versus High Influence

The influence factor $c$, or the amount of impact that neighbouring households have on one another, is set at a default level of 0.01. The personal factor $\epsilon$, or the amount of impact that a random personal event may impact a household’s overall social structure, is set at 0.1.

Interesting patterns emerge when the social influence households have on each other is explored. The social influence factor has a larger impact on the model when there are fewer opportunities for movement. Early in the simulation, there is significant clustering of similar social value households, but these areas are quickly washed out as households with very high or very low social structures move closer to a zero value. After multiple simulations, it appears that a low positive social structure value dominates, with much of
<table>
<thead>
<tr>
<th>Time Step</th>
<th>Low Occupancy</th>
<th>High Occupancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
</tr>
<tr>
<td>800</td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td>50</td>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 8.5: Low versus high occupancy.
the display appearing as very light green (Influence = 0.1).

An interesting contrast in this simulation can be seen when exploring the impact of low social influence. Households tend to move to areas that are close to their personal social structure when they are not influenced by their neighbours. This creates another dramatic division between very high and very low social structured neighbourhoods. The results show a near even split, with only a handful of low social structured cells found in the high social structure zone, and vice versa. In this scenario, the influence factor is set at 0.001 (see Figure 8.6).

8.2.4 Iteration

The visualization of the program proved its value twice during the implementation process. The first was by highlighting a bug whereby CA cells were being populated by more agents than their intended limit. This bug was not clear when looking at the aggregated final results of an execution, and it was difficult to find in the code (it had to do with the order in which updates were processed across several class files). The second time visualization proved helpful was by displaying a characteristic of the computational model that was undesirable, without necessarily being a bug. Residence updates and changes were being processed in the grid in row by row order, moving from the top to the bottom. This lead to a clustering of similar values in the top left corner of the grid, and another cluster of opposite values in the bottom right. This ordered behaviour was not representative of independent household activity, so instead the updates were set to be processed in random order, thus fixing this issue to our satisfaction.

One important part of iterative development is how later stages can help shine some light on issues in earlier stages. For example, computational modelling can raise questions that need to be clarified in the mathematical modelling stage. For example, in this project, the threshold for determining when a household will start looking for new residences was established after trying several implementations of the concept. In another vein, the $\epsilon$ value which models life events was a situation where the most straightforward implementation (a pseudo-random number with a Gaussian distribution) was conceptually an eloquent fit for the effect we wanted in the model, and helped to solidify and build confidence in our model design.

Once this model is moved from the abstract into the concrete and applied field of urban migration, key variables can be introduced into the model. Census data is a logical starting
CHAPTER 8. CELLULAR AUTOMATA PROJECTS

Figure 8.6: Low versus high neighbourhood influence.
point for expanding and elaborating on the variables pertaining to the social structure of a household. The literature emphasizes the importance of socio-economic variables such as income, education level, and housing tenure [151]. Furthermore, the physical characteristics such as housing type or condition, and proximity to educational facilities, influence residential mobility. The measures for these variables can be found in the Canadian census, acquired every 5 years, with the most recent census occurring in May 2006. This data can be scaled to fit this CA model, and at the same time, other data sources can be introduced. Research indicates that neighbourhood-level crime and victimization are both relevant factors influencing residential migration (see [41] and [151]). Individual household-level crime data is available for inclusion in further iterations of this model. Finally, mobility data can be acquired from the Canadian Census, as a method of verifying further iterations of this model.

There are limitations apparent in this model of migration patterns in a residential urban area. First, the model design, created to represent an urban neighbourhood, is generalized into a series of cells, each capable of containing one or more households with a randomly assigned social structure. This generalization is not representative of a real urban landscape. This model does not take into account all of the features in the built environment that would attract or detract residents, because it only includes residential land use. Section 8.3 describes an extension of the model presented here to include more real geographical features.

8.3 Urban Migration–Vancouver

The project described in Section 8.2 focused on the general phenomena of residential movement and neighbourhood formation. As such, the environment included in the model was abstract in nature. Here, the model has been extended to include geographical features based on real data. Land zoning data from the City of Vancouver’s Open Data Catalogue\(^1\) allowed us to consider how zoning types could affect choices of residence. This section begins by outlining the background of the subject matter and modelling process, which is followed by a summary of the model design. The methods used to extend the model by integrating real geographical data to generate the simulation environment are described.

\(^1\)Website: \(\text{http://vancouver.ca/your-government/open-data-catalogue.aspx}\)
and results from test runs on the simulation model are presented. Finally, challenges encountered during this process are discussed, along with the techniques used to deal with them.

8.3.1 Conceptual Modelling

The goal in this stage of the project is to implement a geographic environment that matches the general residential features of a real environment. Vancouver is selected due to its diverse neighbourhoods and the availability of data from the City of Vancouver’s Open Data Catalogue. A land use map was generated using the City of Vancouver’s district zoning data (Fig. 8.7). Since the 75 different zoning classifications are too cumbersome to work with, these are aggregated into 9 general classes of zoning types. While the zoning data gives a general idea of where people in the city live, it is not sufficient for generating an environment for the simulation. Some parks are classified as residential area (including the spacious Stanley Park located downtown), despite having no residences. Further, the Comprehensive Development classification is used as a catch-all to mark areas that need special regulation. Both the mixed commercial and condominium high-rises of Yaletown and the luxurious homes of Shaughnessy fit into this classification, despite having radically different residential densities. Due to these problems, a map of dwelling density is clearly necessary.

The residential density map (Fig. 8.8) is prepared by using the Dwellings Occupied by Usual Residents data at the dissemination block level from the 2006 Census of Canada. Dissemination blocks vary in size, so some larger blocks have a high number of dwellings while being quite sparse. To adjust for this, the area of each block is calculated using the ArcGIS Spatial Analyst\(^2\) extension. The total dwelling count for each block is then divided by its size to generate a final density value of dwellings per hectare. The resulting map shows Vancouver’s concentration of residential density in the downtown core and relative sparseness throughout much of the west side. Each of these maps is rasterized and then saved as a text file. The program can then import the values from a matching set of zoning and density text files to construct an urban environment with cell values that match the raster grid. Files with cell sizes of 25, 50 and 100 square meters are prepared in order to test the model at different levels of spatial resolution.

\(^2\)Product information: http://www.esri.com/software/arcgis/extensions/spatialanalyst
Figure 8.7: Vancouver zoning districts, 2009. Source: City of Vancouver, Open Data Catalogue.
Figure 8.8: Dwellings occupied by usual residents, Vancouver, 2006. Source: Census of Canada.
8.3.2 Mathematical Modelling

The addition of an environment modeled after the real world required changes in other parts of the simulation model. Many of these changes are purely graphical, such as adding functionality to display the zoning district types on the map. Another change is to show dwelling density values using a logarithmic scale, in a similar way to how density levels are classified in Figure 8.8. Other changes are of a structural nature. In particular, the effect of the land use types on household behaviour needs to be determined. Since the zoning districts show the land use of a cell, they can be used to determine how much the effect the actions and behaviour of the residents have on the social structure of the neighbourhood. An example of this is in a primarily commercial area where there are many residences above places of business, but the character of the neighbourhood is determined more by the businesses than by the residents. The mirror situation is a neighbourhood composed primarily of resident homeowners: the behaviour of the residents is the main source of neighbourhood social structure. In the original model, low density cells used a radius one Moore neighbourhood to calculate social structure, while high density cells did not consider any neighbouring cells for this calculation. These ideas were composed to come up with a neighbourhood factor variable, determined by the zoning district type. The neighbourhood factor determines the weighting of neighbouring cells in determining the cell social structure. It is noted as $F$, such that $F_{ij}$ is the neighbourhood factor for the cell at grid location $(i, j)$. A radius one Moore neighbourhood is still used, so the number of neighbourhood cells on a square grid is 8. Given that $S_{ij}(t)$ is the average social structure of all residents at cell $(i, j)$ at time step $t$, and with 8 neighbouring cells $(i', j')$, the social structure of cell $(i, j)$ at time step $t$ is denoted as $V_{ij}(t)$ such that:

$$V_{ij}(t) = \frac{S_{ij}(t) + F_{ij} \times \sum S_{i'j'}(t)}{1 + F_{ij} \times 8}$$

(8.7)

Note that for the case $F_{ij} = 0$, $V_{ij}(t) = S_{ij}(t)$. This is identical to the previous version, where high density cells ignore neighbouring cells when determining cell social structure. Likewise, for $F_{ij} = 1$, $V_{ij}(t)$ is an average of the $S_{ij}(t)$ for all of the cells in the $3 \times 3$ neighbourhood. Thus, the options of the original version are still present, but intermediate levels of neighbourhood interaction are now possible. One difference from the previous version is that the neighbourhood factor is determined only by zoning type, and not the
actual density of a cell. While one can expect a high correspondence between density and zoning district type, this model allows for some variance in behaviour. For example, it is now possible to model high levels of interaction in a densely populated neighbourhood that is primarily residential, as might be seen in the eastern half of Vancouver.

### 8.3.3 Computational Modelling

One issue that turned out to be very important when bringing real data into a simulation model is scale. The effect of changing the temporal scale has already been discussed, but the choice of spatial scale was also critical. Although it was relatively easy to prepare the data at a variety of resolutions, actually getting each of those data sets to run was more of a challenge. In the case of the highest resolution, 25 m$^2$ cells, the program was unable to run at all due to insufficient memory. With the cells at 50 m$^2$ in size, the program was able to run successfully, but the times are an order of magnitude slower than at the lowest resolution (100 m$^2$). Initially, households searching for a new location would compare each potential location with all of the appropriate attractors in the entire environment. This resulted in the run time for scenarios with attractors taking roughly twice as much time as they currently do. However, since attractors do not move in the model, a preprocessing stage was added in which the relative locations of attractors to each cell are calculated. This has dramatically reduced the run time for scenarios with attractors in them, in exchange for a longer initialization time. This trade-off is particularly worth it for long runs of a thousand time steps or more. Eliminating redundant work and improving efficiency can often be ignored in research software development when working on abstract models due to the processing power of modern CPUs. Using large data sets from real sources changes this: it is possible for the CPU, memory or both to be insufficient to run the program at a satisfying pace.
CHAPTER 8. CELLULAR AUTOMATA PROJECTS

Figure 8.9: Urban Migration simulation program using real geographic data. This view displays the current vacancy rate of residences.
8.3.4 Experimentation

The original model was typically run on a $50 \times 100$ grid of cells that contained a total of 9500 households on average. The capacity of the City of Vancouver from the data generated is 253,680 dwellings. Running the simulation model with this massive increase in households resulted in the program grinding to a near standstill. Upon analysis, the part of household behaviour that demanded the most processor resources was related to residence change: the decision to move and the process of finding a new home location. The solution to this problem is to change the temporal scale of the model. Previously, time steps corresponded to one month, that being the shortest duration conceivable between typical residence changes. With a time step of that length, all households needed an opportunity to evaluate their satisfaction with their current location and decide whether or not to move. By changing the duration of a time step to a single day, it became acceptable to only give a small proportion of the total household population the opportunity to consider movement. Social structure changes due to social interaction and personal factors could still occur on a daily basis. This change in scale enabled the program to run at a reasonable speed. However, some adjustments to the model were required, such as adjusting parameters to match the new temporal scale where necessary. The implications of making this change are discussed in the next section.

With the change in temporal scale, fewer households are given the opportunity to move during each time step. Since the cells are subject to influence from their neighbours in the steps that they do not change location, it seemed likely that the new model would be sensitive to changes in the social influence factor variable. This variable determines the magnitude of change in social structure of a household being influenced by its neighbours. In order to test this, the simulation was run for 50 steps using a range of values for the social influence factor, and the standard deviation of the social structure values of the households was recorded at both the beginning and end of a run. Table II lists the average standard deviations for five runs of each value chosen for the social influence factor. No standard deviation score for any run varied more than 1% from the average.

Not surprisingly, the initial standard deviations are almost identical, since they are dependent only upon initialization of the households, prior to any opportunity for social influence. The final standard deviations show the impact of the social influence factor: large values result in small amounts of variance in social structure. In other words, when the
CHAPTER 8. CELLULAR AUTOMATA PROJECTS

<table>
<thead>
<tr>
<th>Social Influence Factor</th>
<th>Initial Social Structure Standard Deviation</th>
<th>Final Social Structure Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2.020462</td>
<td>0.4595</td>
</tr>
<tr>
<td>0.01</td>
<td>2.021267</td>
<td>1.857773</td>
</tr>
<tr>
<td>0.001</td>
<td>2.020333</td>
<td>2.111683</td>
</tr>
<tr>
<td>0.0001</td>
<td>2.021514</td>
<td>2.139422</td>
</tr>
</tbody>
</table>

Table 8.2: Sensitivity to social influence factor.

The magnitude of social influence is high, the social structure values of the households move closer to each other. This takes place through local interactions, but occurs throughout the system. This results in an averaging effect that causes social structure values to approach zero over time. Figures 8.10 and 8.11 illustrate the effect of the social influence factor. The non-residential areas are colored black; this includes parks and industrial areas. The remaining areas are residential, colored in a greyscale continuum to denote social structure, where white is positive and black is negative. Grey coloring shows areas where social structure equals or is close to zero, and these clearly dominate in Figure 8.10, a high social influence factor scenario. In Figure 8.11, there is more variation in social structure values, as shown by speckling cell coloring in the residential areas.

8.3.5 Iteration

The project experienced good participation from domain experts since running experiments on new scenario parameter values was easy to use and gave rapid feedback. The visual nature of CA helped greatly with this. The process of extending an abstract model with real data was also successful, although there were some technical issues because of different methods of measurement: the national census areas used to determine residential density did not match up perfectly with the zoning districts used by the City of Vancouver to establish land use. We witnessed some of the averaging typically found in decimal or continuous value CAs due to cells equally influencing each other. In this kind of model, the total of all the values of the cells can be thought of as the energy in the system, which is eventually distributed evenly over time. Adding cells with features that added resistance, in particular with multi-household dwellings, allowed clusters to form and neighbourhoods with distinct characteristics to appear. In effect, the cells less influenced by their neighbours, the ones that might be labelled as “stubborn”, had a leader effect on the system, and were able to guide the values of their neighbours disproportionately. In other words, in order to prevent a
Figure 8.10: The simulation run with social influence factor set at 0.1.
Figure 8.11: The simulation run with social influence factor set at 0.001.
domination of average values across the system, it is important that cells do not have equal levels of influence upon each other. This can either be represented as an influential entity producing more influence upon those it interacts with, and/or by making such an entity more resistant to influence from others. In either situation, the influential entity changes others more than it is changed by them. Distinct differences in the values of neighbours were found in our experiments with this model, generating something which resembled an “edge effect”.

8.4 Binge Drinking

The heavy consumption and abuse of alcohol among post-secondary students has gained considerable attention in recent decades, influencing a significant body of academic research [47,79,111,157]. Heavy episodic alcohol consumption, known as binge drinking, continues to be a popular social activity among post-secondary students, with a larger proportion of this population engaging in binge drinking than non-students of the same age [79,148]. Binge drinking is defined as the consumption of five or more drinks in a single session\(^3\) (see [63,123]) and has been associated with a number of negative effects, including many with health, behavioural and social consequences [36,78]. Alcohol-related health and wellness concerns are particularly well-documented in recent research. Long-term alcohol abuse is commonly associated with direct toxic effects such as liver and kidney damage [39]. Various health risks impact the post-secondary population in particular, including illness, injury, risky sexual behaviour, alcohol dependence, and death [42,79]. Wechsler and Nelson [156] report that an estimated 1700 college-aged students die from alcohol-related injuries every year, a large proportion of which are associated with motor vehicle accidents. Heavy alcohol consumption has also been linked to poor academic attendance and performance, as well as criminal and deviant behaviour, including physical and sexual assaults, vandalism, weapon use, drug use and arrest [47,78,79,112,155]. Second-hand impacts of heavy alcohol consumption have also been documented among non-bingers and abstainers within the post-secondary environment, including personal and property victimization, and interrupted study and sleep patterns [42,111].

\(^3\)There is significant variation and debate in the definition of binge drinking among academic literature. Some researchers include gender-specific definitions, such as five drinks per sitting for males, and four for females [77,79,156]. Others add time constraints to further define a drinking session [78].
Although its consequences are well documented, binge drinking is a complex behaviour associated with and influenced by a variety of environmental, biological and social factors. Within the post-secondary setting, age, gender, family history and ease of accessibility to alcohol, among other factors, have been found to be related to the prevalence of binge drinking [148,157]. In addition, recent research has further stressed the importance of social influences on post-secondary student binge drinking. Such behaviour is more prevalent among students involved in athletics and social organizations including fraternities and sororities [42,156,157]. The (actual or perceived) drinking patterns of peers and the approval of friends may also influence one’s alcohol consumption [97,111,156]. These findings support the theoretical contributions of social learning theory, which proposes that human behaviour, including binge drinking, is learned from interactions through peer groups and exposure to alternate values and norms [42]. With this in mind, investigating the effects of peer influences on binge drinking behaviour may provide a better understanding of alcohol consumption in post-secondary students.

8.4.1 Modelling issues

While conventional statistical techniques are able to effectively demonstrate the importance of peer influence on binge drinking, they are limited in their ability to answer more complex questions about such behaviour. For example, one may be interested in understanding how different types of social interactions affect the development of binge drinking among groups of college and university students. Similarly, one may be interested in understanding the effect of environmental factors on binge drinking behaviour. Through the use of non-linear mathematical modelling techniques, these areas of interest may be explored.

In addition, such techniques may be employed in simulation models to test a variety of scenarios where “What if?” questions may be posed. For example, one may want to know if binge drinking behaviour develops differently among populations that have different proportions of drinker types. Alternatively, one may be interested in knowing if populations of binge drinkers change over time when various social and environmental influences change. Through the use of non-linear mathematical modelling, it is possible to capture these complex dynamics and answer research questions that could influence public policy.

With respect to peer influences on drinking behaviour, several non-linear mathematical models have been proposed. For example, Gorman, et al. developed an agent-based model to examine social dynamics and environmental influences on agents’ drinking behaviours [71].
Through a variety of simulations they were able to demonstrate that contacts between agents were important factors in the social dynamics that influenced drinking. Similarly, Ormerod and Wiltshire developed an agent-based model to analyze the growth of binge drinking in the United Kingdom [121]. Through development of a theoretical model and calibration with survey-based data, the authors were able to show that the imitative behaviour spreading across social networks is a reasonable hypothesis to account for the patterns of binge drinking that had been observed in recent years.

Agent-based models, however, have some limitations. Applications of agent-based models in the social sciences often involve human agents with complex behaviour and psychology that are difficult to quantify and calibrate. As a result, caution must be taken when interpreting the quantitative outcome of such models when the accuracy of the inputs is questionable [14]. In addition, agent-based models require the description of individual units which can be computationally intensive and time consuming [14], limiting the number of agents included in a simulation, as well as their detail and level of interaction.

### 8.4.2 Conceptual Modelling

This CA model integrates social influences and transition rules. The cells in the grid interact as individuals would in a social community. The cells change over time as they receive and give social influence to their neighbours. After each iteration, the grid is updated to reflect the modifications. Since this is a scenario-based model, the variables can be set according to input data and adjusted to reflect possible changes in the community. Although the cells are stationary, the state of the cell can vary. This reflects the change in social state individuals may experience during their life course. These changes occur as a result of social influences and experiences. The von Neumann neighbourhood type was selected, and the average of the surrounding cells is used to describe these social interactions. Further, at any given time only a random subset (i.e., one to four) of the neighbours exert social influence on a cell.

The development of this model began with surveying existing literature in order to generate a conceptual model of the phenomena under study. For binge drinking, while its characteristics and effects have been well studied, the role of peer pressure is less well understood. Clearly it is important: within the post-secondary setting, direct peer influences may include pressure to consume alcohol by offering a drink, buying a round, or encouraging drinking games [17]. Social influences may also be indirect or passive in nature, associated with perceived norms of heavy drinking among peer groups, and general accessibility of
alcohol within the post-secondary education setting [111, 157]. Ambiguity can be discour-
aging for a modelling project, but it is precisely the difficulty of performing real-world
experimentation and study on this topic that makes this kind of attempt useful.

8.4.3 Mathematical Modelling

In this project, CA modelling was selected as a means to focus solely on the elements of
care: individual state (binge drinking) and local interactions (peer pressure). CA models
are well suited for exploring the dynamics that occur within a population, and are useful for
visualizing the clustering behaviour of communities. With this more abstract approach, it
is possible to simulate large populations with reasonable computational requirements. This
model represents a social community of individuals with a high-risk of binging behaviour
that extends beyond the physical boundaries of a specific geographical area. Specifically, a
community of post-secondary students and their direct social acquaintances is considered.
This community consists of three types of individuals.

- Non-Binger (NB)
- Occasionally Binger (OB)
- Frequently Binger (FB)

An individual can only play a single role at a time. Over time individuals can transition
from one state to the next based on predetermined rules. For example, an OB can become
a FB due to social interaction, and later become a NB following a health problem. The
purpose of this study is to analyze the evolution of a fixed population in a community of
such individuals.

Let \( k \in \{1, 2, 3\} \) denote the state of each individual, where 1 is for NB, 2 is for OB and 3
is for FB. Let the location of each individual \( s \) in the grid be denoted by \((i, j)\). Let \( N_s \) show
the neighbourhood of \( s \) (i.e., the set of individuals neighbouring \( s \)). Assume \( 1 \leq N_s \leq 4 \).
For each individual \( s \) in the grid \( C_s(t) \) is defined as the social counter of the individual \( s \)
at time \( t \). Suppose \( s \) is of type \( k' \) for \( k' \in \{1, 2, 3\} \) and \( v_{kk'} \) denotes the values of the social
influence of an individual of type \( k \) on \( s \) in the neighbourhood \( N_s \). Then define

\[
C_s(t) = C_s(t - 1) + \epsilon + \sum_{k \in N_s} v_{kk'}
\]  
(8.8)
Figure 8.12: Model of drinking transitions.

The parameter $\epsilon$ is a randomly determined value with a normal distribution centred on zero. Using Figure 8.12, values of $v_{kk'}$ are $\alpha > 0$ or $\beta < 0$ based on the type of surrounding neighbours. This provides one value for all positive interactions, and another different value for all negative interactions. This helps to keep the model simple while still permitting asymmetrical levels of positive and negative influence.

**Rules:**

It is assumed that at the initial state $C_s(t) = 0$ for each cell $s$ in the grid.

**Case I:** $s$ is a NB ($s = 1$)
- if $C_s(t) < -1$ for $T$ time steps then $s$ becomes an OB ($s = 2$)
- if $C_s(t) < -10$ then $s$ becomes a FB ($s = 3$) in the next time step

**Case II:** $s$ is a OB ($s = 2$)
- if $C_s(t) < -1$ for $T$ time steps then $s$ becomes a FB ($s = 3$)
- if $C_s(t) < -10$ then $s$ becomes a FB ($s = 3$) in the next time step
- if $C_s(t) > 1$ for $T$ time steps then $s$ becomes a NB ($s = 1$)
- if $C_s(t) > 10$ then $s$ becomes a NB ($s = 1$) in the next time step

**Case III:** $s$ is a FB ($s = 3$)
• if $C_s(t) > 1$ for $T$ time steps then $s$ becomes an OB ($s = 2$)

• if $C_s(t) > 10$ then $s$ becomes an OB ($s = 1$) in the next time step

Here $T$ is the number of time steps needed to effect change in individuals, and 1 and $-1$ are considered as threshold values for gradually changing the states of individuals. The thresholds 10 and $-10$ are considered for major circumstances that force individuals to change their states to NB or FB, respectively.

8.4.4 Computational Modelling

The binge drinking cellular automata application was developed in Java, and as such can run on any common operating system. Many parameters can be altered, including the dimensions of the grid and length of the simulation. Currently, grids of roughly 10,000 cells or less are supported. Execution time for an experiment can vary between a fraction of a second up to one minute for large grids and/or long simulations (2000 or more steps). Short execution times for experiments are a priority since it allows for more responsive and interactive exploration of the configurations of the simulation. The program features tabbed output allowing visualization of the cellular automata itself, as shown in Figure 8.13, as well as plots of interesting metrics, including population distributions and average cell value. Plot functionality is supported by the versatile JFreeChart library.

8.4.5 Experimentation

Due to the exploratory nature of the development of this model, many possible configurations of parameter values and options were available for running experiments. The experiments described here used a 50 cell by 50 cell grid, and were run for 600 steps. Various options were tested on our threshold model using the base parameter values. Plots of the distributions of the populations of cell classes can be seen in Figure 8.14. The options tested were

• Whether or not cells in the extreme binging categories (NB and FB) are permitted to change

• Whether initial cell values are distributed evenly across categories, or are distributed based on relationships found in a survey.
CHAPTER 8. CELLULAR AUTOMATA PROJECTS

Testing whether or not cells in the most extreme categories could change was investigated since it seemed conceivable that people entrenched in a given behaviour would not be susceptible to peer influence. After all four possible combinations of the options were run, some patterns emerged. If the cells in extreme categories do not change, then the CA as a whole quickly becomes dominated by the extreme categories. Any OB cells are eventually influenced by neighbours of one extreme or the other, until no mid-value cells remain. If cells of the extreme values can change, a short initial period is characterized by a flourishing of mid-values, but these are soon after absorbed into large, distinct clusters of extreme value. The use of the survey relationships for setting up experiments had a clear effect, since NB make up more than half of the total population. In this case, NB eventually dominated all the cells if extreme values were capable of changing; if extremes could not change, the OB cells still overwhelmingly changed state to NB. Notably, in all of these experiments the extreme classes end up dominating the cell grid.

Experiments varying the strength of influence were also performed. The strength of positive influences (i.e., against binge drinking) is determined by $\alpha$; the strength of negative influences by $\beta$. They are usually set at 0.02, which allows for gradual but noticeable change over the lifetime of an experiment. If these values are equal, changing them simply alters the rate of change in the model. However, the model behaves differently if $\alpha$ and $\beta$ are not equal. With $\alpha$ set to less than $\beta$, if extremes cannot be influenced, the stronger force initially converts more OB cells, but once there are primarily only extremes left, the effect is minimal, since the remaining cells are entrenched in their behaviour. If extremes can be influenced, the stronger force wins out eventually, dominating all cells. However, converting FB to NB, or vice-versa, is still a slow process. Figure 8.15 shows two runs of the model with $\beta$ (negative influence) higher than $\alpha$ (positive influence).

In the base model, OB have no influence on neighbours of the same type (OB). One alternate to this is for OB to have an effect equal to $\alpha - \beta$. Thus, they have a positive effect on each other if $\alpha$ is greater than $\beta$, and a negative effect if the reverse is true. However, this alternate rule simply accelerates any overall change in cell value. OB cells drift towards extreme values more quickly, and the overall state of the system approaches a steady state rapidly. This is true whether or not extreme values can be influenced. Since $\beta$ determines the strength of negative influence, in experiments where $\beta > \alpha$, FB cells dominated. The reverse was true for alpha, leading to domination by NB cells when $\alpha > \beta$. 
Figure 8.13: Binge drinking cellular automata: (a) example initial conditions, (b) after 100 steps, (c) after 200 steps, (d) after 300 steps.
Figure 8.14: Experiments on population distributions (population of each type over time): (a) even distribution, extremes do not change, (b) even distribution, extremes can change, (c) survey distribution, extremes do not change, and (d) survey distribution, extremes can change.
Figure 8.15: Experiments on influence strength, $\alpha = 0.02, \beta = 0.03$ (population of each type over time): (a) even distribution, extremes do not change, and (b) even distribution, extremes can change.

8.4.6 Iteration

Our initial mathematical model of binging behaviour used influence that was proportional to the difference in behaviour between agents. The justification to this was that agents would become like the people they interact with, and that a strong contrast would produce a larger change. We experimented with incremental change in the decimal values, and then with probabilistic changes in integer values, but both of these had the same result: the values of the entire CA population would merge towards the average value. The system could be seen as having a total amount of energy, which became equally distributed through this mathematical model. A plot of the average system value over time confirmed this interpretation (i.e., it did not change). This result led to changes in the mathematical model (primarily the parameters $\alpha$ and $\beta$) which were capable of producing clusters of similar agents, along with relatively stable contrasts between neighbours, much more like a real human population.

The data we used in our experimentation classified bingers in four categories (non-binger, low binger, medium binger, and frequent binger). We were unsure whether to use those four categories, or to combine non-bingers and low-bingers together. After experimenting with both, the results were very similar, and we found that it was easier to explain the model using fewer designations, so the results presented here were based on the experiments using
three categories. In other words, the added descriptive complexity of the four-category model did not add to its value in exploring or explaining the phenomena of interest.

The models included in this project present initial exploratory experimentation into the social factors associated with binge drinking behaviour. The cellular automata approach has proven to be a useful method for investigating peer influences, as it allows for both local and global population effects to be considered while taking into account the dynamics of various types of social and environmental influences. In the current work a simple approach was adopted that only considered local (neighbour) effects. We experimented with variation in the following parameters: flexibility of binge drinking classifications, the distribution of initial behaviour classifications, and the strength of positive and negative types of influences. Results of the experiments revealed intriguing patterns of social behaviour, including considerable variation in the speed at which individuals change their binge drinking habits. Such findings are encouraging at this early stage as they could lead to more significant discoveries in future research. For example, with further refinement to the model including the specification of positive and negative influences, the results of experimentations could lead to important policy implications for effective intervention strategies. It can also be used to support research employing more traditional methods, such as by suggesting what kind of questions should be asked in future surveys. This project shows how simulation modelling can be used even during exploratory phases of research, or to conduct theory-stressing [48].
Chapter 9

Fuzzy Conceptual Map Projects

9.1 Introduction: Fuzzy Cognitive Maps

Fuzzy cognitive maps (FCMs) have been used to address the complex social situations in which a variety of types of interactions are encountered, ranging from diabetes among Canadian aboriginals [68] to homelessness [108] and international politics [117,150]. FCMs have a network structure: they are composed of nodes representing the factors (e.g., obesity, discrimination) connected by directed edges standing for causal relationships (e.g., impact of obesity on discrimination). An in-depth technical description can be found in [69]. In an FCM, the strength of an edge is assigned by collecting expert opinions about that edge and combining them using Fuzzy Logic Theory, which is a technique of choice since it “resembles human reasoning under approximate information and inaccurate data to generate decisions under uncertain environments. It is designed to mathematically represent uncertainty and vagueness, and to provide formalized tools for dealing with imprecision in real-world problems” [99]. Once all edges have been assigned a strength based on that process, a “case” (e.g., the description of a patient) can be built by assigning a value to each factor of the FCM. Then, executing the FCM consists of repeatedly updating the values of the factors until a subset of them stabilizes. It is important to note that each step in this iterative process does not correspond to a well-defined time interval: it is typical for the values of the FCM to change rapidly at in the beginning, and then take longer to converge upon a final state.

FCMs combine Fuzzy logic rules to formulate a Fuzzy Inference System that yields a numeric value which summarizes the overall opinions of domain experts. Repeating this
process for each of the relationships involved in rebelliousness gives birth to a particular network called a fuzzy cognitive map (FCM) [94]. This technique has successfully been applied to critical situations in which making even small mistakes may have tremendous effects [146]: for example, it was used to evaluate the vulnerability of facilities to terrorism [1]. Its use is also well documented in the context of political crisis, for instance in the case of the Republic of Macedonia [150] or Cyprus [117]. In this network, the nodes represent domain concepts (e.g., ethnic diversity of society, level of political representation) connected by edges standing for causal representations. Concepts are fuzzy, as they hold to a given extent measured from 0 to 1. Similarly, edges take weights in the interval [0, 1] using the aforementioned process. Edges also have a sign: they are either positive or negative to indicate that the target concept respectively increases or decreases with the source concept. An iterative process is applied on the overall network to evolve it until it stabilizes under problem-specific criteria.

FCMs are able to represent the context in a way that is meaningful both mathematically and in terms of domain expert understanding, which makes the technique valuable for interdisciplinary endeavours. Furthermore, FCMs are able to include many relevant aspects, that is, aspects considered to be important and capable of being modelled under current conditions. This allows us to avoid the premature elimination of factors that may later turn out to be important. This modelling point is discussed in [45] and the underlying logic against simplicity as a primary modelling heuristic can be found in [44].

In this chapter, I describe three case studies of FCM projects. In fact, all of these projects utilize hybrid modelling techniques, integrating FCMs with another modelling technique in order to construct a model close to the target phenomena. The first (Insurge) and third (Crowd Behaviour) combine FCMs with CAs (which are described in Ch. 8), albeit in very different ways. The second (Obesity) combines FCMs with a social network graph. In all three cases, the method for combining the modelling techniques was original, and as such, required considerable contemplation, research, and discussion to design. In turn, this meant that developing the accompanying software was not straightforward.

9.2 Insurge

As terrorism and civil war continue to be issues of great political importance, researchers and policy makers are looking for more and better ways to understand these phenomena.
They occur most often in the context of insurgency, that is, the widespread and often violent rebellion by a large movement within society against its government. Insurgency presents challenges to local governments trying to improve their ability to represent their constituents and to deliver political goods, and also to governments involved in foreign diplomatic or military activities that require supporting a government against domestic insurrection. To face these challenges, decision makers must determine the effects of decisions upon the loyalties of their population. Therefore, such governments need models that will help them to understand how myriad economic, cultural, institutional, technological, and historical variables affect the likelihood of a person or a community participating in an insurgency.

The literature on civil wars offers clues as to the relevant factors and their relationships via statistical analyses of datasets, or nuanced case studies comprising qualitative observations of anthropologists, journalists, historians, military experts, and participants themselves. However, these approaches have issues that make the creation of a credible model particularly difficult to achieve. Qualitative observations can be vague, and wide discrepancies can occur between two accounts of the same event. Furthermore, these observations can focus on particular aspects of war and thus may only involve a small number of factors. On the other hand, statistical analyses are more assertive and comprehensive but are still limited due to the methodological constraints of data collection during war, as well as the difficulties of inferring complex relationships.

The goal of this project was to capture domain expert knowledge in a mathematical manner in order to provide insight to the fundamental nature of the phenomena under study. This is a vital step in the study of complex systems, since developing conclusions and results are not only challenging, but also usually limited to very specific situations. To this end, a modelling technique capable of capturing the complex array of interacting factors that together determine loyalty or rebelliousness was conceived. Our approach constitutes an innovative coupling of FCM and CA, thereby having the advantage of handling vague or conflicting data from the former while gaining the ability to model geographical dynamics from the latter. In this section, I will present the model of insurgent behaviour that was developed using this new technique.

9.2.1 Conceptual Modelling

At the core of the conceptual design of the model is a causal map, which consists of selecting representative factors and identifying their relationships as either positive (i.e., the presence
of a factor increases another) or negative.

The rebelliousness of a community indicates the extent to which it will participate in insurgent activities. It is thus the most closely monitored factor for this scenario. It is determined by two factors, representing motive and opportunity. In this case, the motive comprises the socio-economic advantage to insurgency, which collectively incorporates the social, political, and economic reasons for which an individual or community would want to rebel. Opportunities consist of the mechanisms and material conditions that make rebellious acts possible on an incidental basis. The Insurge model also accounts for the self-reinforcing effect of rebellion, in which existing rebelliousness facilitates further rebelliousness through mechanisms such as insurgent recruitment networks and the solidification of ascribed political loyalties [88]. This is primarily achieved through the interactions of variables in each FCM, but is reinforced locally, and transmitted across the map through the CA rules.

The socio-economic advantage to insurgency depends on several factors. The ability of insurgents to control the population and to use discriminating violence determines the extent to which they can offer enticements and coercion to the local community, as does the power of the government to also employ discriminate violence. This is central for scholars such as Kalyvas, for whom “control–regardless of the ‘true’ preferences of the population–precludes options other than collaboration by creating credible benefits for collaborators and, more importantly, sanctions for defectors” [88]. Community economic factors also play a powerful role. The rate or level of economic development determines the ‘opportunity costs’ of participation, where economic recession makes participation in rebellion less risky or costly in comparison to times of economic boom [35]. Natural resources vulnerable to looting or military capture present a tantalizing incentive to join armed groups [43]. For example, in the developing world, ‘conflict diamonds’ or the drug trade influenced the development and proliferation of militias [91] described as a class of ‘feral’ insurgent [106], since their activism is a means of survival instead of an institutional mechanism to secure popular support. High unemployment among young men produces a likely pool for insurgent recruitment, as they are looking for excitement and innovative opportunities to advance economically and socially [50]. On a more micro level, a household so poor that its members have little left to lose and no means to sustain themselves during times of hardship is particularly likely to join whichever group happens to offer the best immediate chance of survival [87]. Indeed, while Pakistani rebels often had good educations–unlike their Afghani counterparts–both Pakistani and Afghani rebels alike came from impoverished and large households [49].
Figure 9.1: Insurgency fuzzy cognitive map: each edge is either positive (full line) or negative (dashed line).
The opportunity to rebel is strongly linked to the strength of the insurgent organization, which affects whether recruitment mechanisms, arms [30], information [88], and logistical capacity make it possible to rebel in an organized or meaningful way. The strength of an insurgent organization increases in the presence of weak government institutions, as security forces are limited in their ability to identify and neutralize insurgent agents [53, 88, 93, 106]. Demographic precursors for insurgency establish conditions of instability that can be exploited by insurgent groups to provide political justifications and normative space for rebellion. An existing territorial dispute can evolve into an enduring ethnic or sectarian rivalry, providing the fault-lines for civil conflict and increased fractionalization [58]. The presence of foreign military forces staging an intervention can inhibit the power of any party to achieve significant political progress through force of arms, paradoxically often making conflict longer-lasting and more intractable [128]. A recent previous civil war can ensure that the population has both lurking hostilities and access to weapons. The Balkan wars are one example of the facilitating effect that weapons-saturation has upon making participation in civil wars feasible, as de Graaf notes in discussing the omnipresence of firearms throughout the former Yugoslavia [38]. A supportive foreign diaspora can make funding insurgent activities easier, while the social exclusion of certain groups makes conflict increasingly easier to justify and prosecute as the excluded group grows in size.

9.2.2 Mathematical Modelling

The conceptual model was used to develop an FCM by ascertaining the strength of the relationships in the the causal map by gathering evidence and combining it using Fuzzy Logic Theory. Then, the coupling with a cellular automaton is used to handle the geographical dynamics. This coupling of FCMs with CA allows for highly accurate models of the influence of geography upon rebelliousness. Informally, all cells of the cellular automaton contain an FCM having the same structure: this states that the same concepts are at work in each neighbourhood, and that they have the same consequences. However, the concepts of an FCM have cell-specific values. For example, the value of household poverty depends on the cell, but it is (structurally) assumed to provide a socio-economic advantage to insurgency regardless of the cell. This is illustrated in Figure 9.2: the same two factors are found in all cells and are related in the same way, but their values differ. In the illustrated situation, the values represent a terrain that is gradually less hospitable following a south-west to north-west gradient (e.g., mountains are found in the north-west whereas there are plains.
in the south-west).

The key connection between the dynamics of cellular automata and fuzzy cognitive maps resides in changing the values of some concepts in an FCM based on the values taken by (possibly distinct) concepts in neighbouring cells. In doing so, we say that FCMs are influencing one another. Figure 9.2 shows that the ability of insurgents to control movements in one cell is influenced by their ability to do so in the surrounding cells. This provides a simple example of a factor being ‘matched’ to the same one. Complex matchings can also be designed, in which one factor can be influenced by several. In a more complex scenario, the ability of insurgents to control population movements also depends on the geography of the neighbourhood, since controlling a well-placed area provides an additional advantage to influence surrounding communities. This was demonstrated during the Second Intifada, when the Israeli settlement of Gilo came under mortar and sniper fire from the surrounding Palestinian town of Beit Jala [74].

The overall process consists of applying the influence between the FCMs (similarly to the classical updating step of a cellular automaton), evolving each FCM until it stabilizes (i.e., until the values converge within a given stability threshold), and repeating the process for the number of time steps specified in the simulation. The effect of this procedure is to establish realistic predictions in each cell based on local dynamics. All generic aspects of this procedure are mathematically specified in the abstract framework that we recently introduced [61]. Problem specific aspects (e.g., how FCMs influence each other) are developed in Section 9.2.3.

The model developed for this project uses a coupling of the CA and FCM methods. Geographical space is represented as cells in a CA; each cell is a space with features based on its geographical and demographical characteristics. These features are captured and modulated by an FCM. In effect, each cell is an FCM, influencing its neighbours at each iteration, then the FCM method is used to stabilize to a consistent state if any disruption is caused by that influence. Thus there is local influence rippling across the map (i.e., the CA), but the features for each area of the map are mediated through and modulated by expert understanding on how such features interact in an area vulnerable to insurgency (i.e., the FCM). The integration of the FCM with the CA required significant verification because of the complexity of the mathematics involved. The general principles of FCMs are straightforward, but it can be difficult to identify incorrect results due to the number of variables, connections and iterations it takes to execute the FCM algorithm. In the end,
Figure 9.2: All cells contain an FCM with the same concepts, linked in the same way. The values of these concepts depend on the cell, and here we see that insurgent control is affected by the same factor in neighbouring cells, but terrain is unaffected by neighbours.

This required test FCMs calculated step by step to compare against the program behaviour.

The FCM captures the precise strength of the 44 discrete relationships between factors by giving each edge a weight (Figure 9.1). To determine the weight of a given edge, evidence was gathered from nuanced readings of peer-reviewed articles. Following a standard procedure (e.g., see word bank in [108]), the parts of selected articles that spoke to the relationship under study were examined by a field expert and categorized using a fuzzy linguistic term from the set \{Very Weak (VW), Weak (W), Medium (M), Strong (S), Very Strong (VS)\}.

For example, Fearon and Laitin reported that “the effect of primary commodity exports is considerable: [...] a country with no natural resource export only has a probability of war-start of 1%” compared to 22% when exporting [50], but Ross considered that “the claim that primary commodity exports are linked to civil war appears fragile and should be treated with caution” [132]. These views were used to estimate the relationship from Resource Vulnerability to Military Capture to Geographic Enabler for Insurgence; Fearon and Laitin were categorized as strong as they saw the effect of commodity exports as “considerable” [50], while Ross was categorized under weak since he sees the link as “fragile” [132]. The different
CHAPTER 9. FUZZY CONCEPTUAL MAP PROJECTS

categories assigned to the evidence constitute a knowledge base which is combined using Fuzzy Logic Theory [164]. The membership functions used in this process are described by the standard equations [163] illustrated in Figure 9.3.

\[
\mu_{VW}(x) = \max(\min(0, \frac{0.25 - x}{0.25}), 0) \tag{9.1}
\]

\[
\mu_{W}(x) = \max(\min(\frac{x}{0.25}, \frac{0.5 - x}{0.5 - 0.25}), 0) \tag{9.2}
\]

\[
\mu_{M}(x) = \max(\min(\frac{x - 0.25}{0.5 - 0.25}, \frac{0.75 - x}{0.75 - 0.5}), 0) \tag{9.3}
\]

\[
\mu_{S}(x) = \max(\min(\frac{x - 0.5}{0.75 - 0.5}, \frac{1 - x}{1 - 0.75}), 0) \tag{9.4}
\]

\[
\mu_{VS}(x) = \max(\min(\frac{x - 0.75}{1 - 0.75}, 1), 0) \tag{9.5}
\]

The process was carried out for each edge, using the Mamdani algorithm, the sum method of aggregation, and the centroid method for defuzzification. A threshold function (also known as a transfer function) was used, in this case, the traditional hyperbolic tangent \( f(x) = \tanh \frac{e^x - e^{-x}}{e^x + e^{-x}} \). These technical choices are common practice, and the interested reader may refer to [143] for the technical aspects. Under these choices, a knowledge base such as illustrated in Section 9.1 (2 Strong and 3 Very Strong) would translate to 0.797.

Figure 9.3: Perceptions for concepts of strength correspond to membership functions, which can overlap. In this example, triangular membership functions are used.


### 9.2.3 Computational Modelling

Each cell of the cellular automaton contains the FCM depicted in Fig. 9.1. In each cell, all concepts are provided an initial value. During the simulation, values in a cell can be influenced by values from surrounding cells. This has not been mathematically specified in depth [61], thus additional formal notation is needed. Consider a concept $a_i$ in a cell $i$ influenced by a (possibly distinct) concept $b_j$ in another cell $j$. The values of these concepts change with time, thus the values of these concepts at time $t$ are referred to as $a_{i,t}$ and $b_{j,t}$.

Functions specify how the influences are conveyed during the simulation, possibly due to particular events. Thus, a function is created at time $\tau$, and called at each time step $t > \tau$ of the simulation. The simplest kind of function is an ongoing process such as the facilitation of recruitment in an insurgent network if there is an existing one in the neighbourhood. In this scenario, the influenced network can grow depending on the strength of its social ties $\rho$ with insurgents: $f_\tau(a_i, b_j, t) = a_{i,t} + \rho \times a_{i,t} \times b_{j,t}$. Note that $\tau$ is not used to specify this process, as it is ongoing. In contrast, a function can depict an event such as gaining control of higher ground terrain in $b_j$. This immediately provides an advantage on control in $a_i$. Thus, if the event happened at time $\tau$ in $b_j$, the control exerted on $a_i$ at time $\tau + 1$ is at least as much as in $b_j$:

$$
\begin{align*}
    f_\tau(a_i, b_j, t) &= \begin{cases} 
    \min(b_{j,t}, \max(a_{i,t}, b_{j,t})), & \text{if } t = \tau + 1 \\
    a_{i,t}, & \text{if } t \neq \tau + 1
    \end{cases} 
\end{align*}
$$

(9.6)

Finally, processes may be decaying. For example, the impact of demographic precursors is stronger near the beginning of a civil war’s outbreak, but they play a weaker role as time goes on compared to mechanisms of control and economics. This can lead to functions decaying at a certain rate $\lambda$, such as $f_\tau(a_i, b_j, t) = a_{i,t} + b_{j,t} \times e^{-\lambda \times (t-\tau)}$. All three functions exemplified here could be used in the broad scenario under consideration. However, their values will need to be carefully calibrated based on the exact situation. Furthermore, these functions were presented in order of complexity, from default transformations (i.e., ongoing processes) to decaying processes. A trade-off should thus be exerted between the increase in realism gained by choosing more complex functions and the loss in our ability to analytically examine the system.
9.2.4 Experimentation

Several steps are required to further put the proof-of-concept developed here to practice. The development of simulation software that has potential for use by decision-makers has begun. Our software currently makes it feasible to test different theories in both an efficient and intuitive way. The efficiency is provided by a short running time: simulating a city of 40 by 40 cells over 200 time steps takes approximately 58 seconds to compute on a personal computer (CPU running at 1.67 GHz, 1 GB RAM). Therefore, users can easily explore and refine the parameter space (i.e., identify which experimental set-ups are of interest) as well as explore the dynamic behaviour of the system modelled (e.g., bifurcation points). Several elements contribute to the intuitiveness of the software. For example, all steps of a simulation can be saved and put together, allowing users to navigate through the simulation in a movie-style; the colours displayed during the animation provide information on the state of a selected variable for each neighbourhood.

One obstacle that appeared after initial implementation was the visualization of CA cells with many internal values. The program allowed for a mono-colour representation of the CA map using only a single variable at a time. This view was not very useful for the domain expert, since interactions between different variables were basically impossible to see. The hybrid nature of the model also made it difficult for the domain expert to understand how to use the program to experiment on the underlying model, since the specific manner in which the two modelling techniques interact was hidden inside the program. A satisfactory solution to this problem was not produced for this particular project, although one approach that worked for a two-valued CA can be seen in Section 9.4.

The complicated structure of most FCMs was an obstacle to the testing of the FCM component of the program code. In the end, it was necessary to devise an example small enough to calculate by hand, and then compare the results of that with running that example in the software. An important bug was found, based on the assumption that variable values in the model should be in the range $[0,1]$, whereas as the FCM algorithm expected them to be in the range $[-1,1]$. This required redefinitions in the mathematical modelling stage, but was easy to fix.
9.2.5 Iteration

The combination of FCMs and CA introduced in this section offers the possibility for scholars and decision-makers in civil wars and insurgency to overcome several methodological obstacles that have otherwise limited the possibility of predicting insurgency based on a wide array of factors. First and foremost, it allows for complex models to be built through the synthesis of qualitative opinions, which comprise the bulk of scholarship on individual and small-group level analyses of political loyalty and violence in armed conflict settings. This benefit of FCMs is not limited to the particular model presented in this paper, but also extends to the synthesis of other qualitative literature, such as multiple psychological models on radicalization and violence, which note similar and interrelated factors but characterize
the relationships between them differently [110, 116]. Another obstacle addressed by our approach is the incorporation of geography into a model. The overwhelming consensus of those studying insurgency is that geography plays a key role. Cellular automata provides a technique for linking multiple agents with both one another and with certain features of their environment. This allows for the simulation of shifts in territorial position and unit interaction, both factors of great significance in military theory but typically omitted from most current sociological models of conflict [88].

Both the general domain definition and technical structure of our model is complete. Therefore, two interrelated tasks remain. The first one, and the most crucial, is to enrich the knowledge base on which our FCM has been developed. Our process of gathering evidence speaking to a wide array of relationships was satisfactory regarding the conditions that precede the outbreak of hostilities, as they are well documented, but was hindered by the limited data on the changes occurring during hostilities. Until more data becomes available via an improvement in security and support in conflict zones, convening an expert panel could be viable. It has been demonstrated that experts can be in high agreement when prompted to evaluate very complex relationships, and that a model such as ours could reliably be based on their judgments [62]. The second task is to develop simulation scenarios which will allow us to test both the mathematical model and the accepted domain wisdom.

Several important issues were highlighted by the project with regard to the integration of FCMs and CA. One thing that became clear to us was the functions used to determine how neighbouring values affect the values within an FCM could be arbitrarily complex. For example, a value such as Motivation (to perform insurgent activities) could be affected by a number of values in surrounding areas, and even by an interesting combination (e.g., a polynomial function) of those values. While some expert advice could lead naturally to certain mathematical formulations of interaction, in general the process of developing them could be quite haphazard. One option would be to limit the type of interactions to a certain set of operations (such as fuzzy AND and OR). That method would produce a model consistent in terms of design, but it would be important to take care not to allow the methodology to inform the model to an excessive extent. In other words, if the method is not able to produce formulations that are a reasonably good match to the descriptions provided by expert knowledge, then it is the method and not the model that should change. This is a typical example of the issue of balancing the elegance of a model with its accuracy in representing real world phenomena. The specific instance of this problem for this project related to the
identification of domain appropriate CA rules from a vast number of conceivable rules.

9.3 Obesity FCM

In Canada, one-third of adults are overweight and one-fourth are obese [126], which is similar to the situation in the United States [120]. This high prevalence together with its consequences on population health and the associated treatment costs has made obesity one of today’s most pressing health issues. An approach that is increasingly being advocated is to recognize that obesity is a complex problem [52, 96], impacted by factors such as the psycho-social context (e.g., gender, depression) and the behaviour of peers. These two aspects have been treated extensively but independently. The aim of the model produced for this project is to foster our understanding of the interplay between these key contributors to obesity [60]. The model integrates a social network of peers (e.g., friends, family, co-workers, etc.) along with a fuzzy cognitive map of the factors affecting obesity.

9.3.1 Conceptual and Mathematical Modelling

One’s obesity is shaped by an environment made of many interacting psycho-social factors. The complexity of obesity is not a mere consequence of the number of these factors or the amount of their interactions. A key challenge is to be found in the nature of these interactions: they can be vague, difficult to interpret, or uncertain. Typically, one FCM is used and presented with different cases for which it makes predictions. This framework differs by using one FCM to represent the environment of each individual. The structure of the FCM is the same for all individuals since that represents causalities, but the values of the factors are specific to the individual. For example, Figure 9.5 shows three individuals, each with an FCM. It is known that obesity generates a weight stigma, and this link is found in all individuals. However, it will only have an effect for Lilian and Garnet, who are the ones experiencing obesity.

The key idea of this framework is that individuals influence and are influenced by others on some factors of their FCMs, which are then updated to account for how the environment mediates these social influences. For example, in Figure 9.5, Martha might influence her friend Lillian by promoting exercise, thus the value of Lillian’s exercise might initially go up. Then, Lillian’s FCM is updated, and her socio-economic status will start to mediate
Martha’s influence, since not being able to afford exercise might not allow Lillian to undertake the changes advocated by her friend. Overall, this framework can model both a realistic population structure, generated using scale-free or small-world networks, and an accurate environment thanks to a finely-tuned expert system.

Figure 9.5: The initialization provides each individual with an instance of the FCM. Some concepts of the FCM can be influenced by peers (in black) while others (in white) cannot.

9.3.2 Computational Modelling and Experimentation

To develop and test this model, a software suite designed for building FCM simulations called CoCo was created. CoCo is built upon four pillars: a concept map editor, a coupling editor (for designing how nodes can affect each other), and testing environments for both social and geographic simulations. The concept map editor can be used for experimenting on an individual FCM, which can be useful for providing insight or identifying problems before stepping up to a network of FCMs. This model was run using the social simulator, which provides a wide variety of sample social network models, including simple networks such as cycle or complete graphs, or complex networks like small-world and scale-free. The software provides analysis of the graph structure in addition to plots of the system variables over the execution of a simulation run. Fig. 9.6 shows a screenshot of the software.

The most significant innovation in this project from a modelling perspective is the interaction editor. This was the product of a challenge: in both this and the Insurge project, the question of how FCMs should influence each other was not easy to answer. The CA
mechanic of neighbouring entities influencing each other using specified rules seemed a good fit, but determining what exactly constitutes a good influence rule was open to many possibilities. Since each FCM contains a number of different values, and these values could possibly affect different values in neighbours, either by themselves or in combination with other values, the range of possible configurations for this kind of hybrid model is intimidating. This is further intensified by the fact that those rules do not need to be similar across values, nor do they need to be linear functions. To deal with this challenge, an interaction editor was added to the software such that a user would be able to create or adjust rules and test them on the fly. Fundamentally, rules are being used in this way as scenario parameters, rather than something static within the model design. This provides a very different view on what constitutes a computational model, and gives the user a lot of flexibility. However, the rules must be written in the Java programming language, limiting this functionality to users with programming expertise. Enabling this kind of functionality more generally, and making it more accessible is a goal for future projects.
9.3.3 Comments
This project (including both the model and software) was primarily the work of a single researcher, Philippe Giabbanelli, and continues from his previous individual research into this topic. As such, the iterative cycle of software development was not as strictly followed as in other projects described in this thesis. Still, collaboration was vital, both for the inclusion of the interaction editor (as described above), and in establishing an overall structure for the software so that it supported all phases of modelling and experimentation. Thus the framework and approach presented in this thesis was a supportive influence on both the modelling process and the software tool design. The resulting software package, CoCo, was thus a more general tool, capable of supporting multiple researchers through the research process, from FCM modelling and testing, up to simulation of interaction using a social network.

9.4 Crowd Dynamics
The Crowd Behaviour modelling project was the product of the desire to apply computational modelling methods to the growing concern for appropriate civil response to problems related to crowd events. With several serious crowd problems in its history, most notably the hockey riots in 1994 and 2011, this is an issue of particular interest in Vancouver. Incidents around the world, such as the Occupy movement in Manhattan, and the Spring Uprisings in North Africa, also motivated interest in this issue. The questions being considered here surround the transformation of a peaceful gathering into a disruption: why does this happen, at what point, and due to which conditions? Also of interest were questions related to appropriate and effective police response, but investigation into that will come at a later point.

Our crowd event expert from the Vancouver Police provided a useful model of the behaviour of crowd participants. There are three general categories: A, disrupters who are actively looking to cause trouble; B, observers with the capacity to become participants in disruption if sufficiently motivated; and C, people with no interest in causing trouble, and who may even act as guardians against it. There is a great wealth of existing research into crowd behaviour. Concepts of note included: the idea of group mind and the psychology of crowds as a unit [98]; crowd crystals, people around whom crowd behaviour forms [115]; the phenomenon of hooliganism, well known in relation to European soccer events [145];
the 6 factors of collective behaviour [54]; deindividuation theory, which considers how and when people stop acting as an individual and become psychologically absorbed into the crowd [165]; and the emotions of body movement, which establishes the extreme rapidity with which people are able to read and react to the general emotional state of a crowd [113]. With this background research, it was possible to begin building our model.

### 9.4.1 Conceptual Modelling

Our model is built from two interacting components: a fuzzy cognitive map, representing macro factors, and a cellular automaton, representing micro factors, particularly the individual interactions of members of the crowd. The FCM was built using surveys of group member knowledge. In order to keep the complexity of the FCM from exploding, a vast array of individual factors were grouped into six general types of factors:

- **Effective social control mechanisms**: police, city, transit
- **Structured environmental factors**: road design, event location
- **Unfavourable situational factors**: suitable target, podiums in the environment
- **Unstructured technological connectivity**: text messaging, Twitter, Facebook
- **Volatile demographics**: younger people, intoxication, gender distribution
- **High risk event**: divisive event, non-family oriented

The definitions of each of these factors was refined iteratively in order to ensure that their meaning was truly shared between group members and easy to communicate to colleagues, regardless of background. Surveys of group members were also used to establish the connections between factors and the weights of those factors. These surveys were followed by discussion among most group members and then compared against the results of a member not involved in the discussion, to avoid group-think. One further factor is included to represent the likelihood of the event becoming disruptive. This factor can be considered to be the output of the other factors.
9.4.2 Mathematical Modelling

The cellular automaton component of the model represents how individual behaviour can change due to interaction with other crowd members. Each cell represents a crowd participant, and their neighbours in the grid are the other participants they interact with. This would usually be the people in the crowd closest to them physically, but it is generalized to include other interactions that may not be spatial in nature, e.g., over cell-phone. Each person has a stable character based on the three categories outlined above (A, B, or C). This does not change over the time frame of the simulation, since a crowd event will likely not last longer than several hours. Individuals also have an activity level, which ranges from $-1$ to $1$, inclusive. Negative values represent disruptive behaviour (making noise, defacing property, crime), and positive values represent active guardianship (berating misbehaviour, taking pictures, alerting police). Values near zero would be considered observer-type activity, whereas values close to $-1$ or $1$ are more active and extreme. These values change over time using transitions rules, which in this case are fuzzy in nature. The Takagi-Sugeno-Kang fuzzy methodology was chosen, where each transition rule is defined as a mathematical function [147]. This is computationally efficient, and also allows each rule to be represented concisely in a mathematical manner (e.g., linear, exponential), which was well-accepted by group members. Nine rules in total were required, one for each combination of character type (A, B, C) and behaviour (Disrupting, Observing, Guarding). Each of these rules are applied to cells at each iteration, at a level corresponding to the values of each cell and each neighbour. For example, an A person (value = $-1$) interacting with a moderately Guarding neighbour (value = $0.5$), would have two rules applied at half-strength (A vs. Observing, and A vs. Guarding), while the results remaining rules would be set to zero. This project is still ongoing, and the definition of individual model characteristics is still changing, so a more explicit mathematical definition of the model is not possible at this point.

9.4.3 Experimentation

The program used to compute the simulation was based on earlier projects, such as the Insurgency project. However, in this case, group members made it clear that it was important to be able to see both an individual character and current behaviour level simultaneously. Thus, a visualization option is included that draws the colour corresponding to character
type in the top left corner of the cell. This contrast between the small square and its sur-
rounding provides an easy way to compare the two values. Colours range from red for $-1$, yellow for 0, and blue for 1, with secondary shades (oranges and greens) for intermediate colours. An observer who has been swept up in the excitement can be identified as a red square containing a smaller yellow square; a person behaving closer to their main intentions does not have this contrast, as the smaller square and its surrounding closely match each other in terms of colour.

9.4.4 Iteration

This project was a challenging attempt to merge cellular automata modelling and fuzzy cognitive map in a manner that suited the nature of the phenomena under investigation. Two main classes of entities in the system were identified: the individual people in the crowd, and the global variables affecting the environment (including police action). Cellular automata were recognized as a good match for the modelling the changing state of the people in the crowd, which the available literature suggested is highly influenced by local interactions (crowd mind). The abstract yet well defined structure of FCMs was determined by our group to be a good match for the global effects in the system. In the end, the choice to have the two models affect each other over time is a novel way of capturing this complex real world interaction in a clear mathematical manner. While analysis of a hybrid model like this is much more difficult that with a single, more well understood modelling technique, a hybrid model has great potential for investigating a complex phenomenon like crowd behaviour dynamics.

This project went through several important changes during development:

- We began with an FCM which contained every factor we could find related to riot formation, but it became clear that this was too unwieldy a structure. It would require too much effort respective to our resources in order to establish the strength of links between the many factors, and even if completed, the resulting FCM would be too complicated for human experts to understand. Related factors were combined into general topics until six main influencing factors on riot formation were identified, and construction of the FCM proceeded from there.

- Testing and code analysis eventually revealed an elusive bug that caused almost all experiments to result in a full-scale riot. The source of the bug itself was a very
Figure 9.7: Crowd simulator execution: (a) initial state, and (b) final state. Note that while character (internal cell) and behaviour (external cell) match initially, some cells show noticeable contrast between the two by the end of the simulation, due to influence by neighbours and the environment FCM.
small thing, but it was important to search for the cause of the unexpected model behaviour, in order to see if it was caused due to a problem with the model definition, or its implementation. Fixing the technical problem was of course much less effort than having to re-evaluate the model.

• Initially, we considered representing crowd members as vectors of various inclinations (peaceful, disruptive, etc.). However, we decided upon using CA to represent the human agents for our initial version due to the simplicity and straightforwardness of that modelling technique. Furthermore, movement through space could be added in a later version by allowing agents to move through the grid, populating some cells while vacating others. Realizing that movement through the physical environment was not a priority for our initial model was an important step in development.

• Early discussion highlighted the need to represent guardian behaviour in model, and not only disruptive or criminal activity. This was based on the definition of the minimal elements of crime in routine activity theory as the intersection of three factors: a target, an offender, and the lack of a guardian [34]. The number and effectiveness of guardians provided a good explanation for the success or failure of a small number of troublemakers in inciting a crowd to riot. For example, it is likely that someone jumping up on a podium and shouting obscenities will have a very different effect at a sporting event than at a holiday parade. There are of course other factors that need to be considered, but including the capacity for agents to discourage others from acting in a disruptive manner was vital to the model presented here.

• As part of the mathematical modelling process, we realized that the CA component and the FCM component should be updated at different speeds within the simulation. The CA component represents the immediate interactions of crowd members with each other, and changes in value there show differences in activity as decisions are made and crowd members react to one another. Thus, a rapid pace of updating seemed appropriate (e.g., each time step corresponds to 1 minute of simulated time). On the other hand, the FCM represents the state of the environment based on the aggregated interactions of various global processes. This suggested a slower pace of updating (e.g., each FCM evaluation process is calculated for each 10 minutes of simulated time).

• Finally, initial testing established that constructing interesting scenarios is a challenge.
This is the case both when considering which combinations of the many variables are interesting and relevant (i.e., represent real possible scenarios of interest), and also in terms of coming up with an interface that allows users to easily set those values in a comprehensible manner. Requiring users to set up experiments using a complicated interface would reduce the number of different scenarios tested, and would be an obstacle to iteratively changing parameters in order to test new scenarios based on the results of the previous ones. This is an open problem for this project, but it is our highest priority currently.

This project was particularly successful from a collaboration perspective. While there were clearly assigned roles for the team members, based on academic background, ensuring that all project elements were well understood by all team members was an agreed upon priority. Team member experience was relied upon for the transformation of domain knowledge into a mathematical model, particularly with regard to generating the global FCM. A method of checking group consensus against the conclusions of an absent member was employed, to ensure that the modelling decisions were sound. Tension with regard to program functionality expectations was welcomed, since resolving it (by discussion of differing perspectives) led to some important inclusions in the program. For example, the option to show (static) agent character alongside (dynamic) agent behaviour by using a smaller coloured square within the main square was an interesting solution for visualizing a 2D CA with two variables. The difficulty in the Insurge project of visualizing the numerous cell variables could perhaps be solved by a technique such as this. Our collaboration process was supported by formal documentation of group brainstorming, such as surveys with tables showing group responses. This was an additional help in converting qualitative domain knowledge into a mathematical form, and clearly integrated the contributions of all participating team members.
Figure 9.8: Crowd simulator results: values of the FCM variables over time.
Figure 9.9: Crowd simulator results: population levels for different classes of behaviour over time.
Figure 9.10: Crowd simulator results: average behaviour level of the entire population over time.
Chapter 10

Conclusions

Computing research in the social sciences is maturing, and a growing emphasis on interdisciplinary team projects can be expected. As researchers, we would like to expand the subject matter we apply this method to, and we would like to improve the quality of the feedback we receive from our work. As such, there is a growing need for more structured ways of going about producing computational social science research. The ability to combine the expertise from disparate fields, and to investigate new ideas in a collaborative manner, appeal to scientists from both practical and social perspectives. This need for technical as well as domain experts is understandable considering the continuing advance of computing technology and modelling methodologies. It is important that we do not calcify our investigative activity so much that our scientific creativity is stifled, and we are no longer able to efficiently consider promising new ideas. But we also need to ensure that our work is valuable as science, and can contribute to the greater project of expanding human knowledge. Thus, frameworks for efficient and productive research project management, including software development as a core endeavour, are clearly needed.

In this thesis, I have described a need for supporting this new kind of research; discussed relevant foundational concepts; presented a software development framework that addresses the needs of that research; and used several examples of projects coming out of this research practice as illustrations of its effectiveness. The framework presented here for the construction and development of computational models of social systems is in part the result of collaborative work on those computational modelling projects in the social sciences. It successfully brings the varied elements required for producing a computational model together
in a way that emphasizes the integrity of the model and facilitates feedback-driven improvement. The number of projects successfully brought from concept to results and publication through the application of this approach demonstrates its effectiveness and appropriateness for this kind of research. To wrap up this thesis and the explanation of my work on this topic, I will discuss the limitations, future directions, and accomplishments of this research.

10.1 Limitations

This framework is not intended for use on large projects, where heavier-weight methods are more appropriate. This can also be true for simulation projects that are not exploratory in nature, but are instead focused on implementing well-known theory with extensive data. For projects of this kind, rapid iterations in software development may not be as feasible nor advisable, since maintaining a consistent and working code base can be more of a challenge. However, it is important to point out that there is a growing case to be made for integrating agile methods with traditional plan-based methods, even in large projects, in order to provide greater capacity for adaptation to changing requirements [13]. This suggests that some aspects of the framework presented in this thesis will still be useful even in larger projects. On the other hand, the framework may be less useful for individual scientists who are capable of handling both the technical and domain aspects of the research. For them, collaboration is not a source of obstacles, and they can instantly change the direction of their software development as they wish. However, even for individual researchers, I believe that the framework provides a good suggestion for how to arrange their activities. This is because ensuring the scientific integrity of the software model is a crucial concern here. The framework also emphasizes and provides methods for communicating the structure of the model, which is useful for sharing the work with peers and stakeholders. Further, the general principles governing the framework should apply to their research process.

While I have situated this research as scientific in the sense that development of the model and experimentation upon it are performed in a scientific manner, I do not present any method for validating that the software model accurately mirrors the behaviour of a real-life system. This means that the programs developed as part of exploratory social science research might not be useful to accurately predict future events, at least while the model is still being represented at an abstract level. This can make selling this kind of research to stakeholders more difficult, because the models developed are unlikely to be a
‘silver bullet’, capable of providing the exact solution to a specific problem. In those cases, producing a solution through analytic methods is superior, if possible. Still, even in these cases, the kind of exploratory research presented here can be useful in providing direction and understanding regarding a specific issue. If sufficient data and well-established theory exist, it could be useful to use the software development framework presented here to guide design choices, and even for modelling, if the interaction of the component theories is still not yet well understood. However, for building a high-fidelity model of greater scale than those I have discussed, a software development process involving more planning and heavy weight methods may be more appropriate.

10.2 Future Work

With regard to specific domain research, I am particularly interested in applying this kind of software modelling to the exploration of economic theory. With the continued current decay of national and global financial systems, economic theory once thought canonical has been challenged. One simple reason traditional economic models have lost some credibility is that many are based on a premise of a system striving towards equilibrium. Yet, there is little evidence of this in the real world. Financial systems repeatedly head towards extreme states, and require external intervention to manage. Thus, research into economic behaviour that accepts that systems are dynamic (perhaps even chaotic) appears likely fertile ground for discovery, and software modelling as we have pursued it here would be a good approach for considering multiple models and theories in an intelligent manner. More generally, economics has traditionally been one of the social sciences with the most formally specified models of human behaviour, even beyond financial activity, and I believe that makes it of particular interest to software modellers of social science. The famous Sugarland project considers a great number of theories about agent interaction, and that vector of research should be pursued while integrating even more economic thinking (such as by including technological innovation within the model) [48].

From a modelling perspective, it is still a challenge to judge or categorize the stability of the various social phenomena considered here. Some phenomena, such as with the crowd behaviour project, change over time due to technology. Up until the 1990s, all communication within a crowd would be local—people talking to each other, or using visual signals. Over the last ten years, we have seen widespread adoption of mobile computing devices.
This is even true outside of the wealthy countries once thought to be at the forefront of technological change, as can be seen by the use of mobile phones in political activity in North Africa and the Middle East. Thus the crowd behaviour models may need to change over time as new technologies change how people interact during a crowd event. However, not all social interactions are necessarily technologically mediated. With the binge drinking project, it is difficult to imagine how this kind of behaviour would change within the foreseeable future. So it would be useful from a theoretical and model building perspective to be able to categorize the stability of different subject phenomena. This will help us to judge how enduring our understanding of such phenomena may be. It may be that for some phenomena, although software modelling provides a helpful avenue of exploring the theoretical space, ultimately accurate traditional (e.g., analytical) models will be found and provide more effective ways of embodying the knowledge produced by the research process, whereas for other phenomena, software models may provide the most concise and accurate models (at least until more effective methods are developed). That said, as pointed out in [76], there is a cultural aspect to perhaps all social phenomena that suggests that complex models may ultimately be the best and most natural way of considering them.

I have also been interested in establishing a general abstract model for computational simulations. The lambda calculus appears to be the best option for formulating this model. The lambda calculus is a general model of computing [32], like Turing machines, but the use of functions aligns it well with both the mathematical and computational modelling stages of our research cycle. The question that is raised by attempting to capture all computational social system simulation in lambda calculus is whether or not all of the functions we are concerned with are computable. There may indeed be processes in real life which are analogous to non-computable functions, such as each person’s emotional reaction to stimuli. However, I suspect that even such phenomena can be modelled sufficiently well using computable functions (or even simple stochasticity). Developing a solid characterization of social system models in the lambda calculus would enable strong computational claims to be made about them, such as limitations or computability. I was unable to spend much time on this topic during my doctoral research, however I think it has some intriguing possibilities. However, existing work in this direction is promising, such as the FRABjouS language and framework, which uses the functional reactive programming paradigm to overcome the lack of representation of time found in purely functional programming [138].
10.3 Achievements

Here I will describe the achievements and contributions made by this thesis and its associated research.

1. A development cycle for software modelling in the social sciences: This thesis describes a framework for the software development aspect of interdisciplinary computational modelling in the social sciences. I have outlined the characteristics and elements that make up this framework. At its heart is an iterative development cycle that is designed to enable rapid and intelligent responses to research team needs as they arise, either in the form of mistakes that need to be fixed, misunderstandings that need to be cleared up, or new research questions that need to be tested. The cycle is presented in Chapter 6. One of John Boyd’s notable insights with his OODA loop is the fact that the effectiveness of interaction and reaction provided by the loop is tied to the speed it takes for each iteration of the loop to process. Quick processing means that more decisions can be made, and there are more opportunities to adjust to feedback. Exploratory research is no different: a rapidly updating research cycle means that more experiments can be made, and results can be used to provide more guidance to the progress of the project. The influence of the research cycle can be seen in the wide variety of research projects used as case studies here, and by the ways in which those projects were able to adjust over time to changes in research direction. As I explain in Chapter 5, this type of software development is intended to be intelligent, in the sense that it adapts to the project in a flexible and effective manner.

2. Collaboration through formal methods and visualization: Throughout the collaborative work I have participated in, I have integrated methods and tools to improve the ability of team members to work together and share information without misunderstanding. This has been achieved partly through the use of formal methods, such as the CoreASM Abstract State Machine language. This language allows for the creation of executable code at an abstract level, which can be understood without a high level of technical proficiency in programming. I also constructed the Control State Diagram editor (CSDe) to allow CoreASM users to design and share ASMs in a visual manner. These diagrams can be automatically converted into program code, and accelerate the process of developing full programs. I have also embraced the use
of visualization in my simulation software. This includes both full graphical display of model behaviour during the execution of an experiment (either in real-time or as a playback), as well as visualization of data once the results of the experiment have been generated after execution. These multiple views assist in not only analysis of model behaviour, but are valuable tools for the debugging and validation of software models. While a software model may ultimately be a mathematical entity, and thus not explicitly require graphical visualization, good visuals are a boon to a project. They not only provide multiple perspectives from which to consider the model, but also engage users who would find a purely numerical representation of the model obtuse and onerous to use.

3. A methodology pertinent to social science needs and concerns: As can be seen in the case studies included in this thesis, software modelling allows researchers to consider many subjects that would be very difficult to study in real situations. They allow phenomena to be represented in an abstract, manageable manner, while at the same time including any relevant data that is available. These models can then be used to test our understanding of the phenomena, and build new theories from any discoveries made. This mode of research is well suited to fields that are seen as falling under the category of post-normal science, since research can be undertaken safely, integrating the knowledge of different stakeholders, without directly putting any real-life system at risk. Further, uncertainty can be mathematically incorporated into the model, or such aspects can be left out if considered irrelevant, thus reducing stochastic noise in the results. Finally, ease of use for all team members is emphasized, as is the need to encourage a sense of ownership regarding the software, so that technology is a vehicle, and not an obstacle, to research.

10.4 Final Words

While there are still methodological, institutional, and philosophical challenges that face computational modelling and simulation in the social sciences, the promise of this means of expanding knowledge is too great to forgo. Software allows us to model our thoughts in a mathematical manner, and to be able to iteratively experiment and adjust as necessary. In a way, we are using the computation to extend our minds so that we can consider theories using powerful deductive and empirical methods normally impossible. Certainly, this kind
of experimentation is not sufficient in isolation to solve all of our pressing issues. However, software modelling is a new tool that allows us to develop our thoughts and expand our knowledge in previously inaccessible ways, leveraging both traditional and new methods for innovative purposes.

Finally and perhaps most importantly, general acceptance of a development framework by stakeholders in research and policy making institutions could be one of the factors that leads to computational modelling being more widely used as a tool for inquiry. It provides structure for the careful production and refinement of knowledge, and mechanisms for independently establishing the value of ideas and reusing them. There is still a great deal of work to be done before simulation is as widely used in research and decision making as statistics (for example), but considering the potential for a method that exploits the algorithmic and mathematical nature of computing for the purpose of advancing science, it is a worthwhile goal to aim for.
Bibliography


[38] Bob de Graaf. The wars in former Yugoslavia in the 1990s: bringing the state back in'. In Isabelle Duyvesteyn and Jan Angstrom, editors, Rethinking the Nature of War. Frank Cass, 2005.


[118] Michael J North, Nicholson T Collier, and Jerry R Vos. Experiences creating three implemen-
tations of the repast agent modeling toolkit. ACM Transactions on Modeling and Computer
[120] Cynthia L Ogden and Margaret D Carroll. Prevalence of overweight, obesity, and extreme
for Health Statistics, June 2010.
[121] Paul Ormerod and Greg Wiltshire. binge drinking in the uk: a social network phenomenon.
[123] Alex R Piquero, Chris L Gibson, and Stephen G Tibbetts. Does self-control account for the
relationship between binge drinking and alcohol-related behaviours? Criminal Behaviour and
[125] Simon F Pratt, Philippe J Giabbanelli, Piper Jackson, and Vijay K Mago. Rebel with many
causes: A computational model of insurgency. In Intelligence and Security Informatics (ISI),
[126] Public Health Agency of Canada and the Canadian Institute for Health Information. Obesity
in Canada. 2011.
[127] Ana Maria Ramanath and Nigel Gilbert. The design of participatory agent-based social sim-
[128] Patrick M Regan. Third-party interventions and the duration of intrastate conflicts’ journal
[129] Matteo Richiardi, Roberto Leombruni, Nicole Saam, and Michele Sonnessa. A common pro-
tocol for agent-based social simulation. Journal of artificial societies and social simulation, 9,
2006.
[130] Christopher D Rickett, Sung-Eun Choi, Craig E Rasmussen, and Matthew J Sottile. Rapid
prototyping frameworks for developing scientific applications: A case study. The Journal of
[131] Dennis W Roncek and Pamela A Maier. Bars, blocks, and crimes revisited: Linking the theory
[133] Stuart Rossiter, Jason Noble, and Keith RW Bell. Social simulations: Improving interdisci-
plinary understanding of scientific positioning and validity. Journal of Artificial Societies and
[134] Robert J Sampson and W Byron Groves. Community structure and crime: Testing social-


