

A Software Engineering Approach for Agent-Based Modelling and Simulation of Public Goods Games

Tuong Manh Vu

Intelligent Modelling & Analysis Research Group
School of Computer Science
University of Nottingham

This dissertation is submitted for the degree of
Doctor of Philosophy

May 2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Dr Peer-Olaf Siebers, for his support of my PhD study, for his patience, and his vast knowledge. I'm also extremely grateful to my second supervisor, Dr Christian Wagner, for his insights and invaluable suggestions. Thank you both for being great mentors, for giving me the opportunity to walk this path, and allowing me to grow as a researcher.

Besides, I would like to thank my colleagues in the IMA (Intelligence Modelling and Analysis) research group for stimulating discussions, for the encouragement, and for all your support in the last four years. Additionally, I really appreciate researchers from other departments who have helped me with three case studies in my research: Alejandro Lee Penagos for discussion on the first case study, Anya Skatova as the domain expert in the second case study, and Francesco Fallucchi and Lucas Molleman for discussing ideas for the third case study. Especially very special thanks to Theodore Turocy for helping with data acquisition and helpful suggestions.

More importantly, I immensely grateful to my family: my parents, my brother, and my aunt for wholeheartedly supporting me to walk my own path, for your caring and unconditional love. Furthermore, I dedicate this thesis to the memory of my grandfather and grandmother, who passed away during my PhD studies. They always supported me and I know they would be proud to see me achieve a doctorate.

Last but not least, a big thank you to all my friends, especially my house mates and Vietnamese friends, for making Nottingham and UK my second home.

Publications

This section presents a list of publications formed as part of the research work for this thesis.

1. Vu, T. M., Siebers, P. O., Skatova, A., & Turocy, T. (2015). Modelling the Effect of Individual Differences in Punishment Sensitivity on Behaviour in a Public Goods Game. *The 14th International Conference on Modeling and Applied Simulation (MAS 2015)*.
2. Vu, T. M., Siebers, P. O., & Wagner, C. (2014). The Potential of Object-Oriented Analysis and Design for Agent-based Computational Economics. *The 2014 European Simulation and Modelling Conference (ESM'2014)*.
3. Dawson, D., Siebers, P. O., & Vu, T. M. (2014). Opening pandora's box: Some insight into the inner workings of an Agent-Based Simulation environment. *2014 Federated Conference on Computer Science and Information Systems (FedCSIS)*.
4. Vu, T. M., Siebers, P. O., & Wagner, C. (2013). Comparison of crisp systems and fuzzy systems in agent-based simulation: A case study of soccer penalties. *2013 13th UK Workshop on Computational Intelligence (UKCI)*.

Abstract

In Agent-based Modelling and Simulation (ABMS), a system is modelled as a collection of agents, which are autonomous decision-making units with diverse characteristics. The interaction of the individual behaviours of the agents results in the global behaviour of the system. Because ABMS offers a methodology to create an artificial society in which actors with their behaviour can be designed and results of their interaction can be observed, it has gained attention in social sciences such as Economics, Ecology, Social Psychology, and Sociology.

In Economics, ABMS has been used to model many strategic situations. One of the popular strategic situations is the Public Goods Game (PGG). In the PGG, participants secretly choose how many of their private money units to put into a public pot. Social scientists can conduct laboratory experiments of PGGs to study human behaviours in strategic situations. Research findings from these laboratory studies have inspired studies using computational agents and vice versa. However, there is a lack of guidelines regarding the detailed development process and the modelling of agent behaviour for agent-based models of PGGs. We believe that this has contributed to ABMS of PGG not having been used to its full potential.

This thesis aims to leverage the potential of ABMS of PGG, focusing on the development methodology of ABMS and the modelling of agent behaviour. We construct a development framework with incorporated software engineering techniques, then tailored it to ABMS of PGG. The framework uses the Unified Modelling Language (UML) as a standard specification language, and includes a simulation development lifecycle, a step-by-step development guideline, and a short guide for modelling agent behaviour with statecharts. It utilizes software engineering methods to provide a structured approach to identify agent interactions, and design simulation architecture and agent behaviour. The framework is named ABOOMS (Agent-Based Object-Oriented Modelling and Simulation).

After applying the ABOOMS framework to three case studies, the framework demonstrates flexibility in development with two different modelling principles (Keep-It-Simple-Stupid vs. Keep-It-Descriptive-Stupid), capability in supporting complex psychological mechanisms, and ability to model dynamic behaviours in both discrete and continuous time. Additionally, the thesis developed an agent-based model of a PGG in a continuous-time setting. To the best of our knowledge such agent-based models do not exist. During the development, a new social

preference, Generous Conditional Cooperators, was introduced to better explain the behavioural dynamics in continuous-time PGG. Experimentation with the agent-based model generated dynamics that are not presented in discrete-time setting. Thus, it is important to study both discrete and continuous time PGG, with laboratory experiment and ABMS. Our new framework allows to do the latter in a structured way.

With the ABOOMS framework, economists can develop PGG simulation models in a structured way and communicate them with a formal model specification. The thesis also showed that there is a need for further investigation on behaviours in continuous-time PGG. For future works, the framework can be tested with variations of PGG or other related strategic interactions.

Table of contents

List of figures	xii
List of tables	xiv
List of pseudocodes	xiv
List of abbreviations	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Contribution	4
1.3 Thesis Outline	4
2 Background	5
2.1 A Review of Game Theory, Experimental and Behavioural Economics	6
2.1.1 Analytical Game Theory	6
2.1.2 Behavioural Game Theory	10
2.1.3 Experimental Economics	12
2.1.4 Behavioural Economics	14
2.2 Agent-Based Modelling and Simulation	16
2.2.1 Modelling and Simulation	16
2.2.2 Agent Concept	19
2.2.3 Multi Agent System	21
2.2.4 Agent-Based Modelling and Simulation	22
2.2.5 Agent-based Computational Economics	24
2.2.6 Compare and Contrast between Multi Agent System and Agent-based Computational Economics	25
2.2.7 Basic Software Engineering Concepts	27
2.3 The Public Goods Game	29

2.3.1	Public Goods and the Free Rider Problem	29
2.3.2	The Analytical Game Theory Approach	30
2.3.3	The Laboratory Experiment Approach	31
2.3.4	The Agent-Based Modelling and Simulation Approach	37
2.4	Research Gap	37
2.4.1	Development Methodology Aspect	38
2.4.2	Agent Behaviour Modelling Aspect	39
2.5	Summary	40
3	Research Methodology	41
3.1	Aim and Objectives	41
3.2	Methodology	42
3.3	Research Activities	43
3.4	Choice of Simulation Software	43
3.5	Design of the Initial Framework	48
3.5.1	The ABOOMS Framework Overview: the Macro and Micro Process	48
3.5.2	Design of the Macro Process	48
3.5.3	Design of the Micro Process	54
3.5.4	The ABOOMS Framework v1	63
3.6	Summary	66
4	Case Study 1: Keep-It-Simple-Stupid vs. Keep-It-Descriptive-Stupid	67
4.1	Case Study of Washing Dishes in a Shared House	67
4.2	Conceptual Modelling - Three Common tasks	68
4.2.1	Understand the problem situation	68
4.2.2	Collect and analyse data and related theories	68
4.2.3	Determine objectives of the simulation study	69
4.3	Modelling using KISS Approach	69
4.3.1	Conceptual Modelling - Design the conceptual model	69
4.3.2	Analysis	71
4.3.3	Design	74
4.3.4	Implementation	79
4.4	Modelling using KIDS Approach	82
4.4.1	Conceptual Modelling - Design the conceptual model	82
4.4.2	Analysis	84
4.4.3	Design	86
4.4.4	Implementation	88

4.5	Experiments and Results	91
4.6	Discussion	93
4.6.1	The simulation studies	93
4.6.2	OOAD and UML	94
4.6.3	Can everything be agents?	96
4.7	Framework Revision and the ABOOMS Framework v2	97
4.7.1	Revision	97
4.7.2	The ABOOMS Framework v2	99
4.8	Summary	101
5	Case Study 2: A Psychological Model of Punishment Sensitivity	103
5.1	Case Study of Punishment Sensitivity in the Public Goods Game	103
5.2	Conceptual Modelling	104
5.2.1	Understand the problem situation	104
5.2.2	Collect and analyse data and related theories	105
5.2.3	Determine objectives of the simulation study	105
5.2.4	Identify the settings of the PGG	106
5.2.5	Design the conceptual model	106
5.3	Analysis	107
5.3.1	Identify agents, artifacts, and external actors	107
5.3.2	Identify use cases	108
5.3.3	Document use cases	108
5.3.4	Draw use case diagram	109
5.4	Design	110
5.4.1	Structural design	110
5.4.2	Behavioural design: gameplay	112
5.4.3	Behavioural design: individual differences in punishment sensitivity	114
5.5	Implementation	117
5.5.1	Structural implementation	117
5.5.2	Behavioural implementation	118
5.5.3	Alternative implementation of punishment sensitivity	123
5.6	Experiments and Results	126
5.6.1	Validation experiment	126
5.6.2	Experiments with different punishment conditions	130
5.6.3	Experiments with different ratios of strategies	132
5.6.4	Experiments with different ratios of anxiety	133
5.7	Discussion	134

5.7.1	Simulation and experiments	134
5.7.2	Statechart for Behavioural Modelling	135
5.8	Framework Revision and the ABOOMS Framework v3	136
5.8.1	Revision	136
5.8.2	The ABOOMS Framework v3	137
5.9	Summary	140
6	Case Study 3: Continuous-time Setting	141
6.1	Case Study of Continuous-time Public Goods Game	141
6.2	Conceptual Modelling	143
6.2.1	Understand the problem situation	143
6.2.2	Collect and analyse data and related theories	144
6.2.3	Determine objectives of the simulation study	145
6.2.4	Identify the settings of the PGG	145
6.2.5	Design the conceptual model	147
6.3	Analysis	148
6.3.1	Identify actors	148
6.3.2	Identify use cases	148
6.3.3	Document use cases	149
6.3.4	Draw use case diagram	150
6.4	Design	150
6.4.1	Structural design	150
6.4.2	Behavioural design	151
6.5	Implementation	153
6.6	Experiments and Results	156
6.6.1	The presence of GCCs	156
6.6.2	Evaluation rate of CCs in a simulation without GCC	157
6.6.3	Evaluation rate and generosity rate of GCCs	159
6.7	Discussion	160
6.8	Summary	161
7	Conclusion and Future Work	162
7.1	Summary of Research	162
7.2	Uses of the ABOOMS Framework	163
7.2.1	Purposes and General Uses	163
7.2.2	The Influence of Software Engineering Methods	164
7.2.3	Use with Procedural Programming	166

7.3	Achievement of Aim and Objectives	166
7.4	Contribution to Knowledge	167
7.5	Limitations and Future Works	168
References		170
Appendix A The NetLogo Implementation of Case Study 1 - KISS approach		181
Appendix B Contributions in Non-Communication Continuous-time PGG		185
Appendix C Documentation of the ABOOMS Framework		188
C.1	The macro process	188
C.2	The micro process	188
C.3	Behavioural modelling with statechart	191

List of figures

2.1	Outline of the Background Chapter	5
2.2	The interactions between MAS, ABMS, ACE	16
2.3	Simulation processes used in Computer Science vs. Experimental Economics	27
2.4	Toward a more general behavioural theory of collective-action dilemmas (Po- teete et al., 2010)	35
3.1	Research activities	44
3.2	A screenshot of a NetLogo model	46
3.3	AnyLogic IDE	47
3.4	The lifecycle of a simulation study (Robinson, 2004)	49
3.5	Overview lifecycle of the ABOOMS framework v0	54
3.6	An object-oriented development process	55
3.7	Use case diagram of an ATM system	57
3.8	Customer class diagram (which is part of an ATM system)	57
3.9	Class diagram of an ATM system	58
3.10	A state machine diagram of an ATM system	59
3.11	A sequence diagram of an ATM system	60
3.12	An activity diagram of an ATM system	62
3.13	Overview lifecycle of the ABOOMS framework v1	64
4.1	Use case diagram - KISS approach	73
4.2	Use case diagram - KISS approach with Experimenter	73
4.3	Class diagram - KISS approach	75
4.4	Activity diagram - KISS approach	76
4.5	Sequence diagram - KISS approach	77
4.6	AnyLogic IDE with the project of the KISS approach	80
4.7	AnyLogic implementation of the KISS approach	83
4.8	Class diagram - KIDS approach	87

4.9	Statechart - KIDS approach	87
4.10	AnyLogic implementation of KIDS approach	89
4.11	Example of a punishing situation	91
4.12	Mean cooperativeness over time	92
4.13	Overview lifecycle of the ABOOMS framework v2	100
5.1	Use case diagram	110
5.2	Class diagram	111
5.3	Gameplay statecharts of a Main artifact and a Person agent	113
5.4	Sequence diagram	115
5.5	PunishmentSensitivity statechart of Person agent	116
5.6	Strategy change of Conditional Cooperators	117
5.7	Strategy change of Full Cooperator and Full Defector	117
5.8	An AnyLogic screenshot of the simulation of case study 2	119
5.9	Implementation of statecharts	120
5.10	An example of statechart with probability transitions	122
5.11	Strategy statechart	123
5.12	Average investment over four blocks	128
5.13	Average investment over 40 rounds	129
5.14	Agent states of punishment sensitivity statechart	129
5.15	Strategy of agents over time	130
5.16	Punishment was periodically implemented on rounds in an expected punishment condition	131
5.17	Implemented punishment blocks occurred periodically with randomly <i>two out of ten</i> punishment games per block	131
5.18	Implemented punishment blocks occurred periodically with randomly <i>five out of ten</i> punishment games per block	132
5.19	Total investment in different ratios of strategy	133
5.20	Investment over time in different ratios of anxiety	134
5.21	Overview lifecycle of the ABOOMS framework v3	138
6.1	(a) Median and (b) mean contributions by period and treatment (Oprea et al., 2014)	142
6.2	Screenshot of a continuous-time laboratory experiment (Oprea et al., 2014)	143
6.3	Median and mean of contributions over 10 sub-periods of 1 minute (Oprea et al., 2014)	145
6.4	Use case diagram	150

6.5	Class diagram	151
6.6	Statecharts of the four types of agents	153
6.7	AnyLogic implementation screenshots	155
6.8	Average contributions of Experiment 1, 2, and 3	157
6.9	Contribution in different groups	158
6.10	Experiment 5 - Average contribution of different generosityRate	160
B.1	Group 1	185
B.2	Group 2	186
B.3	Group 3	186
B.4	Group 4	187
B.5	Group 5	187
C.1	Overview lifecycle of the ABOOMS framework	189

List of tables

2.1	Payoff tables of Prisoner's Dilemma	8
2.2	Games categorized by rivalry and excludability	30
2.3	A payoff table of PGG ($C > V > C/2$)	30
2.4	A payoff table of PGG with punishment	31
2.5	Variables and their effects on contribution, adapted from Ledyard (1995) . . .	33
4.1	Model Scope of Case Study 1 - KISS approach	71
4.2	Model Level of Detail of Case Study 1 - KISS approach	71
4.3	Model Scope of Case Study 2 - KIDS approach	85
4.4	Model Level of Detail of Case Study 2 - KIDS approach	85
4.5	Transitions in the statechart of Person agent	90
5.1	Model Scope	107
5.2	Model Level of Detail	107
5.3	Person agent initialization	112
5.4	Transitions in statecharts of Main and Person	121
5.5	Transitions in Person's strategy statechart	124
5.6	Experiments with different ratios of anxiety	133
6.1	Individuals in no-communication continuous-time experiments	146
6.2	Number of each type in no-communication continuous-time experiments . . .	146
6.3	Model Scope	148
6.4	Model Level of Detail	148
6.5	Parameters of Experiment 1, 2 ,and 3	156
6.6	Parameters of Experiment 4	159
6.7	Average contribution with different rates of GCCs	159
6.8	Parameters of Experiment 5	160

List of pseudocodes

4.1	Initialization of Person agent	79
4.2	Contribution pseudocode	81
4.3	Punishment heuristic	81
4.4	Calculate outcome with expected punishment	82
5.1	Investment decision	125
5.2	Strategy change	127
6.1	Initialization of GCCs and CCs	154

List of abbreviations

ABMS	Agent-Based Modelling and Simulation
ABOOMS	Agent-Based Object-Oriented Modelling and Simulation
ACE	Agent-Based Computational Economics
AOSE	Agent-Oriented Software Engineering
BIS	Behavioural Inhibition Scale
CC	Conditional Cooperator
CPR	Common Pool Resource
DSGE	Dynamic Stochastic General Equilibrium
FC	Full Cooperation
FD	Full Defection
GCC	Generous Conditional Cooperator
IDE	Integrated Development Environment
IODA	Interaction-Oriented Design of Agent simulations
KIDS	Keep It Descriptive Stupid
KISS	Keep It Simple Stupid
MAS	Multi Agent System
MPCR	Marginal Per Capita Return
MU	Money Unit
NCC	Normal Conditional Cooperation
OOAD	Object-Oriented Analysis and Design
OOP	Object-Oriented Programming
PGG	Public Goods Game
SCC	Strong Conditional Cooperation
SE	Software Engineering
UML	Unified Modelling Language
VBM	Variable-Based Modelling
WCC	Weak Conditional Cooperation

Chapter 1

Introduction

1.1 Motivation

An economy is a complex adaptive system. In order to understand the economy and make predictions, economists build models using different methods. However, the 2008 economic crisis has revealed the need for a new modelling method. During that period, the best models belonged to two types, both with fatal flaws ([Farmer and Foley, 2009](#)). The first type is econometrics, which is the application of mathematics and statistics methods to the empirical measurement of relationships theorized by an underlying theory ([Greene, 2012](#)). As long as things stay similar, econometric models make successful forecasts but fail in the event of great change such as in economic crisis. The second type is dynamic stochastic general equilibrium (DSGE), which assumes a perfect world and by its nature rules out uncommon events such as an economic crisis ([Farmer and Foley, 2009](#)). DSGE models include representative agents “who produces a good that can either be consumed or invested to increase future output and consumption” ([Wickens, 2008](#)) and use optimisation to make decision. [Farmer and Foley \(2009\)](#) suggested that Agent-Based Modelling and Simulation (ABMS) is “a better way”, but not well developed in Economics “because of historical choices made to address the complexity of the economy and the importance of human reasoning and adaptability.”

So, what is ABMS? In ABMS, a system is modelled as a collection of agents, which are autonomous decision-making units with diverse characteristics ([Macal and North, 2010](#)). The individual agents interact with each other and their environment at the micro level to produce complex collective behaviour patterns at the macro level, often referred to as system level emergence. In Social Science, ABMS allows for the representation, experimentation and analysis of complex social phenomena to gain the understanding of the dynamics behind them ([Elsenbroich and Gilbert, 2014](#)).

Also ABMS is a suitable approach for Economics, because a system in ABMS is a dynamic complex system comprised of many agents, which is similar to a system of an economy. Economies are complex dynamic systems, in which large numbers of decision makers interacting at micro level create global regularities (Teshatsion, 2006). Economists have modelled economies for hundreds of years. During the last quarter of the twentieth century, the dominant paradigm in Economics is the rational expectation approach, which assumes people have perfect information, adapt instantly and rationally, and maximize their personal advantage (Farmer and Foley, 2009). However, decision-making of real people is also based on peer pressure, fear, or overconfidence (Farmer and Foley, 2009). In addition, there is a trade-off between capturing social systems with a high level of fidelity and analytical tractability (Macal, 2016). For simplification, common assumptions of single representative agents with homogeneous behaviours are made. These assumptions create models with significant differences compared to the real-world system that they are intended to represent. In order to improve the realism of economic models, several economists started to use other methodologies such as laboratory experiment and ABMS to study the additional aspects of human behaviours (peer pressure, emotions, etc.) in economic settings (Teshatsion, 2006). Research findings from laboratory experiments of human subject behaviour have inspired studies using computational agents in ABMS and vice versa (Duffy, 2006). To explain the interaction between laboratory experiment and ABMS, we focus on and discuss in detail Public Goods Game (PGG), a popular game used in laboratory experiments and agent-based models.

The PGGs have been investigated thoroughly in Economics. It is a standard experimental economics approach to study human cooperation. In Experimental Economics, a laboratory public goods experiment consists of many participants, who are matched into groups (usually of four people). They are given an endowment of Money Unit (MU) which they can keep for themselves or invest into a public account. The invested money is multiplied and distributed equally to all group members. In this game each player faces a dilemma: to invest into public good and potentially get a higher return through cooperation in the future; or to free ride on other people's contributions. Classical Economics predicts that no one will contribute assuming the rational agents want to maximize their payoff. However, in real life, people do collaborate. Therefore, in Experimental Economics, many laboratory experiments with different settings have been conducted to investigate the reasons people cooperate.

However, laboratory experiments of the PGG have some limitations on small numbers of participants, games, or different experimental settings. ABMS offers ways of addressing these limitations because a computational model is more flexible than the laboratory experiment. Nonetheless, the modeller needs to carefully design, calibrate, and validate the agent-based models before experimentation. Currently in Economics, the agent-based models have the

agents play the PGG in the manner similar to the laboratory. The experiment in the model consists of a series of games. In each round, all agents make decisions how much they want to contribute, then receive payoff, and if applicable receive information on the average investment of the group. The agents' behaviours are designed, and then the model is calibrated based on the data collected from the laboratory. After that, the model can be experimented with more agents, different settings of games, etc. While these agent-based models can be used to validate the results from the laboratory experiments, they have also enabled a deeper understanding of participants' behaviours in the lab (Dinther, 2008). However, in many of these models, the design of agent decision-making is based on utility functions of microeconomic models. A utility function is derived from theory and the modeller's perception of the system, then agents make their contribution decision by choosing the best (maximum or minimum) value from the function. The problems with this approach are that the human decision-making process is much more complex, and in real life, people have to make decisions in continuous time and constantly changing environment. In the end, it can be said that even though ABMS shows great potential, the design of the agent-based models in Economics and other social sciences needs to be improved to allow a better capture of the realism of human behaviour in these experiments.

A potential option to improve the design of agents-based models in Economics is applying Software Engineering (SE) techniques from Computer Science. Agent technology originated from Computer Science and has developed a well-established field named Multi Agent System (MAS). There is an "unwarranted gap" between MAS used in Computer Science and ABMS used in Social Science (Conte et al., 1998). For example, agent-based models are used for different purposes in Computer Science and Economics. While computer scientists design agents for controlling complex systems or for instilling objects with autonomy and intelligence, economists use agents for gaining a better understanding of the dynamics within economic systems. For the modelling task both communities use their specialised modelling approaches which are very different from each other. We believe that ABMS of the PGG has not been used to its full potential and that the tools used for designing software agents in MAS could help unleash this potential.

Therefore, in this thesis, we want to leverage the potential of ABMS of the PGG, focusing on development methodology and agent behaviour modelling. We chose to focus on the PGG because it is a popular game and there have been many studies on many variations of the game in Economics. In order that the design process and techniques can later be used by social scientists, we developed a framework, in which SE techniques are applied for identifying, designing and implementing agents in the PGG. This framework focuses on development lifecycle and the modelling of agent behaviour, and provides a guideline for agent specification

at many levels of abstraction using visual modelling formalisms. This research can serve as a bridge between two disciplines: the framework provides a structured approach for economists to develop and document a simulation; while computer scientists can have access to theories and models of human behaviour in social science using the framework as a communication medium.

1.2 Thesis Contribution

As the main contribution to knowledge, the thesis makes to the body of knowledge is a methodology of developing agent-based models of PGGs using well established tools from SE. The methodology is manifested in form of a framework and its effectiveness is tested with three case studies. These case studies demonstrate flexibility in development with two different modelling principles (Keep-It-Simple-Stupid vs. Keep-It-Descriptive-Stupid), capability in supporting complex psychological mechanisms, and ability to model dynamic behaviours in both discrete and continuous time. Also, for the first time, this thesis presents an agent-based model of the PGG in a continuous-time setting that is rarely considered in Economics, and our simulation incorporates different modes of strategic reasoning being triggered by continuous time compared to discrete time.

1.3 Thesis Outline

This thesis is structured in the following manner. Chapter 2 presents relevant background information and is divided into four sections: the first introduces Game Theory, Experimental and Behavioural Economics; the second reviews the landscape and development of ABMS; thirdly, PGG and relevant research approaches are reviewed; lastly, the research gap found in the literature along with motivation for the research is outlined. Chapter 3 presents the research methodology of case studies, describes research activities, and explains the design of the initial framework. Chapters 4, 5, and 6 report on the case studies 1, 2, and 3 respectively. Chapter 7 concludes the thesis, revisits the aim and objectives, presents contribution to knowledge, and discusses limitations and future directions for this research.

Chapter 2

Background

This chapter presents an introduction to the fields of Game Theory, Experimental and Behavioural Economics in Section 2.1, and Agent-Based Modelling and Simulation in Section 2.2. Then Section 2.3 discusses Public Goods Game and overview of three research approaches, namely Game Theory, Laboratory Experiment, and Agent-Based Modelling and Simulation. After discussing the literate streams, the research gap is derived in Section 2.4. Figure 2.1 summarizes the outline of this chapter.

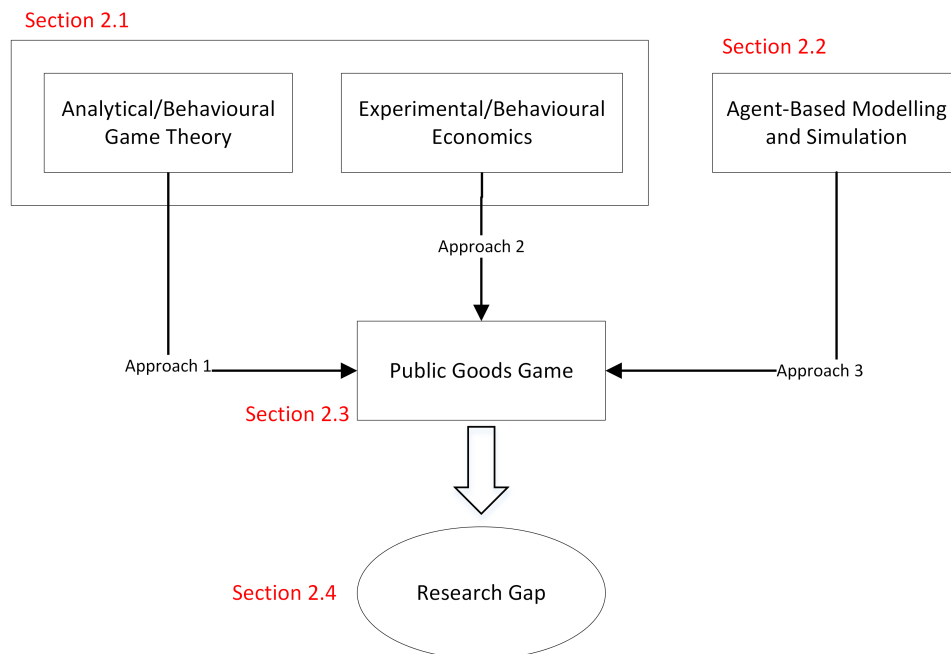


Fig. 2.1 Outline of the Background Chapter

2.1 A Review of Analytical Game Theory, Behavioural Game Theory, Experimental Economics, and Behavioural Economics

This section introduces the basic concepts and historic context of Game Theory and Economics, focuses on the interactions of four closely-related sub-fields: Analytical Game Theory, Behavioural Game Theory, Experimental Economics, and Behavioural Economics. From the development of Analytical Game Theory, Behavioural Game Theory was born; while Experimental Economics and Behavioural Economics emerged from the development of Game Theory.

2.1.1 Analytical Game Theory

Games in Game Theory

Game theory, a field of Mathematics, originated in the works of [Borel \(1921\)](#) and [Von Neumann \(1928\)](#). In 1921, Emilie Borel published several papers on theory of games using poker as an example. However, he did not develop this idea very far. In 1928, seven years after Borel, John Von Neumann published his first paper on game theory, and later continued to develop and popularize game theory. In 1944, the book “Theory of Games and Economic Behavior” ([Von Neumann and Morgenstern, 1944](#)) by Jon Von Neumann (a mathematician) and Oskar Morgenstern (an economist) has revolutionized Economics ¹. Game theory at this time is called *Classical Game Theory* or *Analytical Game Theory*.

Game theory is about human behaviour in *strategic situations*. A strategic situation can be called a *game*, which is played by two or many people (*players*) by choosing between different courses of action (*strategies*). You play these kinds of games throughout your entire life: with your parents, brothers, sisters, friends, enemies, and even strangers. Some examples are a buyer plays a negotiating game with a seller of a product, or bidders play a game against each other at an auction for an art masterpiece. On a larger scale, similar games are played by politicians, businesspeople, leaders, etc. For instance, in the 1960s, Kennedy and Kruschev played a game in the Cuban missile crisis that might have lead to a nuclear war. Game theory studies these *games* and helps you to better recognize these games in real life and understand the surrounding world.

Game theory can be used to reason about decision making. When interacting with others in a game, people, who are assumed to behave rationally, know their objectives, think carefully,

¹The influence of this book on economic theory is described more in Section [2.1.3](#)

and decide what strategies to go for in order to achieve the desired outcomes. For example, in table tennis, when serving, a player needs to decide which side of the table to serve, how much spin to put on the ball, and how fast the ball will go. If a player know the physical skills and weaknesses of their opponents, s/he can choose strategies that exploit these weaknesses. However, the other players also may do similar strategic thinking about you at the same time. Game theorists analyse this interactive decision-making process, but not guarantee a solution to every problem. It can be that some chance elements happen, or the player does not act in a rational manner. However, game theory study still is useful, because we all at least try to think and perform sensibly in a strategic situation. It provides general principles for thinking in a certain strategic situation. Good decision makers mixed game theory with their own experience or instinct to deal with a specific situation. In addition to people decision-making, researchers have some quite noticeable success in analysing and explaining insect and plant behaviours using game theory because their genes are programmed with rational behaviours; the irrational ones could have been eliminated by nature ([Binmore, 2012](#)).

Prisoner's Dilemma

Game theorists study toy games, relatively simple versions of real life situations, in order to remove the irrelevant clutter of real world problems and focus on the strategic issues. There are four essential elements in a game ([Rasmusen and Blackwell, 1994](#)): the players, the information and actions available to each player at each decision point, and the payoff for each outcome. Many games have been formed and studied. One of the most famous classic games is the *Prisoner's Dilemma*, which has been studied in many variations and thousands of experiments. To better understand how a game theorist analyses a game, Prisoner's Dilemma is thoroughly discussed in the following.

In a Prisoner's Dilemma game, there are two prisoners, who are not able to communicate with each other. The police does not have enough evidence to convict them for a serious offence, and can sentence both of them to a year in prison on a lesser charge. So the police simultaneously offer prisoner A and prisoner B the following:

- If A and B both confess, they serve 3 years in prison.
- If A confesses and B denies, A will be set free and B will serve 5 years in prison (and vice versa).
- If A and B both deny, both of them serve 1 year in prison.

Both prisoner A and B are players in a two-person simultaneous-move game in which they have to choose between confessing and denying the crime. The strategies Confess and Deny

can be called Defect and Cooperate to describe the interaction between two players. Cooperate means to take the actions that help the other player; while Deny means to betray the other. If they both trust each other and deny, the jail sentence is only a year. However, if A does not trust B and confesses while B denies, then B gets a 5-year jail sentence but A will go free. If both confess, they both get 3-year terms.

Game theorists assign *payoffs* to every possible outcome of the game to assess what might happen. In this case, the payoff is the number of years of the sentence. The outcomes and choices of the Prisoner's Dilemma described above can be represented in a payoff table (see Table 2.1a). The Prisoner's Dilemma game can be represented algebraically with abstract payoffs in Table 2.1b (T, R, P, S stand for Temptation, Reward, Punishment, Sucker's payoffs). The prisoner can choose their strategy based on the strategy of the other prisoner. But he does not know the other's strategy, or at least trust that the other prisoner will cooperate with him. However, he can guess what the other will do and try to *maximize his payoff*, assuming that he is a selfish rational prisoner.

Consider the prisoner's dilemma from perspective of prisoner A. He has to make this choice based on what he believes B will choose. If B denies, A's best response is to confess since he will go free compared with 1 year jail. But if B confesses, A's *best response* is to confess, he will get 2 years instead of 3. Therefore, Confess is the best strategy for A regardless of his belief about B's choice. Confess is called a *dominant strategy* and Deny is a *dominated strategy*. Since prisoner B is playing the same game, Confess is also the dominant strategy for him. Therefore, the final result is that both will confess and get 3-year jail sentences. Even though they can both deny and get just 1 year each, they do not do it because the choice (Deny, Deny) is unstable, which means that the other prisoner can get a better payoff by changing from Deny to Confess. The state that both prisoners confess is called *Nash Equilibrium*, which

		B	
		denies	confesses
A	denies	-1, -1	-5, 0
	confesses	0, -5	-3, -3

(a) A payoff table

		B	
		cooperate	defect
A	cooperate	R, R	S, T
	defect	T, S	P, P

(b) A abstract payoff table ($T > R > P > S$)

Table 2.1 Payoff tables of Prisoner's Dilemma

is named after a mathematician John Forbes Nash. More formally, Nash Equilibrium is a list of strategies, one for each player, such that no player can get a better payoff by switching to another strategy.

It can be said that in a Nash Equilibrium, each player chooses the best response to the other's choice. But if the two choices are made simultaneously, how can one know the choice of the other to respond? Still people play simultaneous-move games and make choices in real life. There are several methods a person can use to try to figure out what the other's choice will be: blind guess, experience from the past of playing the same game or knowing the other player, or logical deduction. Game theorists use a concept *belief* to describe what one thinks that the other will be choosing in simultaneous-move games. So in Nash Equilibrium, each player must have correct beliefs about strategies of the others and use the strategy which is best for him, given beliefs about strategies of others.

Types of games

Games in Game Theory can be categorized based on their characteristics ([Wilkinson and Klaes, 2012](#)):

1. Cooperative vs. Non-Cooperative games

In cooperative games, players can communicate, negotiate contracts, and make joint strategies. In non-cooperative games, there is no negotiated and binding contracts. Players form self-enforcing relationships.

2. Two-player vs. Multi-player games

Prisoner's Dilemma is a two-player game. However, there are many games involving more players which can be independent or in teams. Multi-player games can be difficult to analyse because there are opportunities for some players to form coalitions against others.

3. Zero-sum vs. Non-zerosum games

In a zero-sum game, the sum of players' gains equals the sum of all losses of other players. For example, the rock-paper-scissors game between two players is a zero-sum game, because when one wins, the other loses. For the non-zerosum game, the sum of gains and losses does not always add up to zero. Prisoner's Dilemma is a non-zerosum game (you can check by summing the cells in the payoff table [2.1](#)).

4. Complete vs. Incomplete information

In a game of complete information, players know all actions and corresponding payoffs of other players, such as Prisoner's Dilemma or chess. If players are uncertain of payoffs

of their own or others, the game has incomplete information and is very difficult to analyse.

5. Static vs. Dynamic games

In static games, players make their moves *simultaneously* without knowing the moves of others. On the other hand, games with *sequential* moves are dynamic games. The Prisoner's Dilemma is a static game; while chess is a dynamic game.

6. Discrete vs. Continuous strategies

Games with discrete strategies have a limited number of moves players can make. For example, Prisoner's Dilemma has only two possible actions; thus it is a discrete strategy game. If the player can choose between limitless number of moves, it is a continuous strategies situation. One example is when bidders choose a price (a real number) to propose to the sellers.

7. One-shot vs. Repetitive games

In a one-shot game, the game is played once without repercussions. A repetitive game is where the game is played for a number of repetitions. In repetitive games, players have to take into considerations history and reputation of the opponents; thus the results can be very different compared to one-shot games.

In summary, Analytical Game Theory, originated in the field of Mathematics, has increased its popularity in Economics. It is a mathematical deviation of what players with different cognitive capabilities are likely to do in games. "Analytical game theory is highly mathematical and usually based on introspection and guesses rather than observation of how people actually play in the games" (Camerer, 2003). In order to perform these introspections and guesses, the Analytical Game Theory makes strong assumptions about players. These assumptions and predictions from the games have sparked criticism from other scientists. From the criticism, a new field of *Behavioural Game Theory* has emerged.

2.1.2 Behavioural Game Theory

As discussed in the previous section, the Analytical Game Theory makes strong assumptions about players. Gotts et al. (2003) listed two assumptions and expressed criticisms of them:

- *Instrumental rationality*: Players have consistent references and unlimited computational capacity.
- *Common knowledge of rationality*: All players are aware of other agents' rationality-related assumptions.

Firstly, the first assumption said that the rational players will choose an action that maximizes their expected payoff (referred to as *utility maximization theory*), and the payoff needs to be computed and expressed in numerical terms. There are experiments which show that in at least some circumstances, people behave incompatibly with this model. Secondly, utility maximization theory assumes that all players' concerns are taken into account when making decisions. Most cognitive scientists do not agree with this assumption since they view human action as goal-directed and human minds have limited cognitive capability.

In order to overcome with these problems, the Analytical Game Theory has been expanded by adding emotion, limited reasoning, mistakes, learning, etc. to create a field of *Behavioural Game Theory*. Behavioural Game Theory aims to predict how people *actually* behave by combining traditional economic theory with theories from other research areas such as psychology and neuroscience. For example, in the case of Prisoner's Dilemma, Analytical Game Theory predicts that both players should defect which means that it is not rational for human beings to cooperate. This has been criticized because people do cooperate in real life. In the analysis of Analytical Game Theory, rational players don't cooperate because the conditions necessary for rational cooperation are absent. That is why, in Behavioural Game Theory, there are a lot of variations of Prisoner's Dilemma to study more about cooperation (by adding elements of punishment, reward, reputation, etc.).

[Camerer \(2008\)](#) states four main components of Behavioural Game Theory:

1. *Representation*

This refers to how players perceive or mentally represented a game. Analytical Game Theory overlooks the fact that players often incompletely perceive a game, or ignore some elements of the game to reduce computational complexity.

2. *Social preferences over outcomes*

Aside from players' own payoffs, they also have preferences regarding payoffs of other players and the distribution of these payoffs.

3. *Initial conditions*

This relates to the players' beliefs in the game. Analytical Game Theory assumes players' beliefs will be correct immediately (i.e. understand all aspects of the game and play perfectly) without special conditions such as pre-play communication. Two types of theory have emerged: cognitive hierarchy theories which assume iterated reasoning is limited in the human mind, and theories which assume players make stochastic mistakes.

4. *Learning*

In repeated games, players can learn from their own payoffs, from strategies of other

players, and learn about what other players are likely to do. Many models have been proposed for these learning processes.

The foundation of Behavioural Game Theory lays in empirical observations, mostly experimental results, because human behaviours in experiments contradict with prediction from Analytical Game Theory (Goeree and Holt, 2001). In the next section, the sub-field of *Experimental Economics* is introduced.

2.1.3 Experimental Economics

Controlled laboratory experiments have provided a methodology for Economics to progress further from armchair theorizing (Weibull, 2004), and been a bridge between Analytical Game Theory and Behavioural Game Theory. These experiments also created a sub-field: Experimental Economics. Research with experiments follow a scheme: A simple game, for which Analytical Game Theory make a clear prediction about players' behaviour, is set up for human players to play in a controlled laboratory. A player makes a simple choice which combines with choices from others to produce a payoff. Players are rewarded afterwards, usually with cash, according to their performance. The game can be repeated to observe how players learn over time. Comparing the experimental results with predictions from Analytical Game Theory, behavioural game theorists hope to understand more about human behaviour and develop new applicable theories. For example, Camerer (2003, pp. 20-24) describes some interesting observations from the laboratory experiments of game theory: Participants do not play the Nash equilibrium immediately, probably because they do not know game theory and need time and/or experience to understand the game. However, over time, they can converge to the equilibrium (or quite close to it), but not always. With these observations and empirical data, economists can theorize the reason people do not converge to the equilibrium, or explore the learning process while playing economic games.

Historically, the first economic experiment can be traced as far back as Bernoulli's study on St. Petersburg paradox in 1738 (see Bernoulli, 1954 for the translation from Latin to English). In St. Petersburg paradox, a player plays a game where a coin is tossed at each stage. The initial stake is two euros and double each time a head appears. The game ends when a tail appears. How much should someone pay to participate in this game? The expected payout of

this game is infinity (Equation 2.1), which suggests that a player will pay any amount of money to participate.

$$\begin{aligned}
 E &= \frac{1}{2} \times 2 + \frac{1}{4} \times 4 + \frac{1}{8} \times 8 + \dots \\
 &= 1 + 1 + 1 + \dots \\
 &= \infty
 \end{aligned} \tag{2.1}$$

Bernoulli tested this hypothesis by surveying other people around him, and concluded that most people are not willing to pay to participate in this game. This experiment has shown that people's behaviours do not simply follow the generated theory. Therefore, the theory needs to be modified to better explain the observed behaviours. The interaction between theories and experiments has pushed Economics forward. The early experiments are in 1930-1960, then Experimental Economics grows steadily in 1960-1980, then exponentially from the 1980s. In early experiments, there are three trends of research that stand out: individual decision-making, analysis of game theory model, and organization of markets (Branas-Garza and Barreda, 2015; Kagel and Roth, 1995).

For the first trend, the early experiments were designed to test theories of individual choice. Thurstone (1931) performed an experiment with *ordinal utility theory*. Wallis and Friedman (1942) gave a critical review and criticized Thurstone's experiment because of the ill-specified and hypothetical choices. As a subsequent experiment to follow up to Thurstone and in reply to Wallis and Friedman, Rousseas and Hart (1951) constructed a more concrete and realistic situation by having participants choose from breakfast menus and must eat all of what they chose.

In 1944, Von Neumann and Morgenstern (1944) presented a theory of individual choice under uncertainty and risk (*expected utility theory*) that greatly influences game theory and Experimental Economics, and inspires a new wave of experiments of interactive behaviour. For instance, Allais (1953) systematically explored the violations of expected utility theory by observing the incoherence between choices in an original problem and an equivalent reformulated one.

The second trend of early experiments concern tests of game theoretic hypotheses. In 1950, at RAND corporation, Melvin Dresher and Merrill Flood developed a game later known as the famous Prisoner's Dilemma (Flood, 1958, see). The game was repeated 100 times and participants cooperated more than what predicted by the Nash's equilibrium. This had triggered a stream of experiments filled with variations of the prisoner's dilemma (different repetition, payoffs, conditions of information, communication, etc.).

The experimental design was also exploited by two early works: [Kalisch et al. \(1954\)](#) and [Schelling \(1957\)](#). [Kalisch et al. \(1954\)](#) conducted several multi-player games and indicated some issues of experimental designs, which substantially influenced the design of later experiments. The work of [Schelling \(1957\)](#) on coordination games pointed out some parameters of how experiments are conducted, which are not addressed by existing theories, has essential impacts on experimental studies.

Lastly, the third trend is the research area of analysis of markets. [Chamberlin \(1948\)](#) conducted an experiment of a decentralized market where buyers and sellers were free to negotiate for imaginary products. The results were that the quantity of goods sold was larger than the prediction and the price did not converge to the equilibrium. One student from Chamberlin's experiments later conducted two studies ([Smith, 1962, 1964](#)), and stated that if the information about the transactions was public and participants interacted repeatedly, quantities and prices converged rapidly to equilibrium. In addition, another important work of [Siegel and Fouraker](#) raised methodological considerations in Experimental Economics. [Siegel and Fouraker \(1960\)](#) performed a series of experiments in which participants bargained in pairs. They paid attention to the information available to the players about each other's payoffs, went to great length to ensure anonymity, and not only motivated subjects with monetary motivation but also changed the incentives by varying the payoff differences.

From the 1980s, Experimental Economics has grown explosively: the number of experiments increased; different experimental methodologies appeared; experiments investigated the same issues from different points of view and critically examined others. In the branch of Experimental Economics or related to this area, there are several Nobel Prizes awarded in 1988 to Maurice Allis, in 2002 to Vernon Smith, in 2009 to Elinor Ostrom, in 2012 to Alvin E. Roth and Lloyd S. Shapley.

In parallel with Experimental Economics, there is the development of *behavioural economics*, which is discussed in the next section.

2.1.4 Behavioural Economics

Behavioural Economics is a discipline which provides Economics with realistic psychological foundations to developing theoretical models of human behaviours. The goal is to better inform economic models with more realistic assumptions from what other disciplines know about human behaviour. Behavioural Economics improves the standard theory by testing the prediction of the neoclassical approach, then relaxing one or two assumptions to provide psychological realism, and suggesting new theories. To be noted is that, usually, these adjustments are not radical because the modified assumptions are not fundamental to the economics approach. For instance, assumptions can acknowledge that: people have limits on computational power,

willpower, and self-interest; they care about fairness; they do not weight risky outcomes in linear fashion or discount the future exponentially at a constant rate (Camerer et al., 2004). Conclusively, Behavioural Economics seeks to understand the hidden motivation behind human behaviour in order to improve the existing theory, generate theoretical insights, make more accurate predictions, and suggest better policies.

In terms of historical context, many works in the 1950s and 1960s have led to the emergence of Behavioural Economics in the 1970s (Wilkinson and Klaes, 2012). Simon (1955) introduced the concept of *bounded rationality* to describe the cognitive limitations of human ability in acquiring and processing information. Several papers (Allais, 1953; Ellsberg, 1961; Markowitz, 1952; Strotz, 1955) pointed out anomalies of the standard model in individual decision-making. In the 1970s and 1980s, there are two works that influentially contribute to the birth of Behavioural Economics. First, the paper “Prospect theory: An analysis of decision under risk” (Kahneman and Tversky, 1979), one of the most cited paper in the prestigious journal *Econometrica*, documented violations of expected utility and proposed *prospect theory* which describes how people decide between probabilistic alternatives that involve risk and uncertainty. Prospect theory states that people make decisions based on expected value of loss and gain rather than the absolute value final outcome, and individuals dislike loss over gain (*loss-aversion*). The second one is the work of (Thaler, 1985) on *mental accounting* concept, which describes how people code, categorize, and evaluate financial activities.

The early works have established a recipe for research activities of Behavioural Economics (Camerer et al., 2004, p. 7).

- Identify normative assumption or popular models such as expected utility, discounted utility.
- Identify clear violations of the assumption or model (anomalies).
- Inspired by the anomalies, generate alternative theories that generalize existing models.
- Use assumptions in the previous step to construct economic models, derive fresh implications and test them.

In terms of research methods, at first, Behavioural Economics relied on evidence generated by laboratory experiments. Nevertheless, there are many practices in Experimental Economics that behavioural economists find extremely restrictive (Camerer et al., 2004). For instance, experimental economists rarely collect data regarding demographics, response time, and other cognitive measures that are useful to behavioural economists. Recently, there are a wide range of methods beyond laboratory experiments such as field experiments or computer simulation.

In the next section (Section 2.2), the method of computer simulation, in particular Agent-based Modelling and Simulation, is introduced and discussed in detail.

2.2 Agent-Based Modelling and Simulation

This section introduces the concepts of modelling and simulation then focuses on Agent-based Modelling and Simulation (ABMS). It also presents and explains the relationship between the two fields: (1) the origin of ABMS from Computer Science, Multi-Agent System (MAS), and (2) the application of ABMS to Economics, Agent-based Computational Economics (ACE). The interactions between MAS, ABMS, and ACE is shown in Figure 2.2.

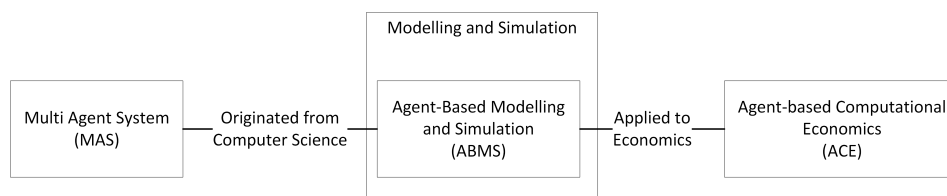


Fig. 2.2 The interactions between MAS, ABMS, ACE

2.2.1 Modelling and Simulation

Definitions

Firstly, we need to define three concepts: system, modelling, and simulation. Bertalanffy (1968) defines that “a system is an entity that maintains its existence through the mutual interaction of its parts”. A system can also be defined as a collection of interrelated parts that have a certain purpose. A system can exist in nature such as a forest containing many trees and animals. It can also be designed by humans, like traffic in a city or a mobile network. It can be consciously or unconsciously formed by human activities such as family or schools. A system can be modelled or simulated.

There are many definitions of a *model*. Shannon (1975) defines that “a model is a representation of an object, system, or idea in some form other than that of the entity itself” Banks (1998) states “a model is a representation of an actual system. The model should be complex enough to answer the question raised, but not too complex”. For Kelton et al. (2007), “a model is just a set of approximations and assumptions, both structural and quantitative about the way the system does or will work”. Garrido (2011) defines that “A model is an abstract representation of a real system”. So a model is created to answer a question or solve a problem. Thus with the same system, there can be many models depending on the questions asked and how people

approach solving that question. The process of abstracting the real world to construct a model is *modelling*.

[Robinson \(2004\)](#) defines a computational simulation as “an imitation (on a computer) of a system as it progresses through time”. Simulation usually involves designing the model and performing experiments to have better understanding of the system. While the model is abstract representation of the real system, the simulation represents the operation of the system. The nature of simulation is predicting what will happen to the system if certain actions are taken (what-if analysis). So with a simulation, users can vary the inputs, perform actions to see the changes in operations, thus get a better understanding of the system. In a way, a simulation is a decision support system because after gaining sufficient understanding, a user can decide on how to improve the real system.

The need of simulation

The reason we need simulation is that it provides a method to represent the increasingly complex world, to predict performance, or to perform what-if analysis. As an illustration, let's take an example of building a new bus terminal in a city. With such an expenditure, there are many crucial decisions to make when designing the terminal, such as the number of bus stops, information desks, seats for customers, the number of other services like toilets, food and drinks. Moreover, the number of staff and their shifts is also needed to be determined. So how do project managers decide the design of the terminal as well as the number of resources needed? One solution is inviting experts with experience of designing, building, and managing bus terminals. From their knowledge, along with many estimations, we build the bus terminal. But because of the high level of complexity of the situation, we can just hope that the solution will work.

But there is a better and more effective approach: simulation of the bus terminal. With the simulation model running on the computer, managers can see the flow of passengers and buses. Managers can also vary parameters, change the design and experiment with different situations without actually building the terminal. This approach is not only safer but also costs less. In fact, simulation has been used by many organizations and companies to improve their facilities, build new buildings, and simulate production lines or delivery networks. Simulation can support decision-making on various scales: retail, cargo screening, or future energy plan of a city.

Further, there are many examples of simulation every day without us noticing. For instance, the weather forecast is a simulation of the weather system. Simulations are also used for training purposes such as flight simulators that train pilots in different situations before going to control a real plane ([Paur, 2011](#)). Some examples are popular games: The Sims ([Electronic Arts](#),

2016) where we control people in a simulated environment, or World of Warcraft (Blizzard Entertainment, 2016) which is a simulation of a fantasy world. There is also great success in pandemic games. The Collapse (Ubisoft, 2016) where you play the patient zero infected with smallpox, and survive by navigating to the nearest hospital while minimizing the number of people you infect. Plague Inc. (Ndemic Creations, 2016) is a mobile-phone game where you play the role of a plague, evolve a pathogen, and try to wipe out the human population. The game has won several awards in 2012, and in 2013 the creator was invited to the Centers for Disease Control and Prevention to speak about the use of video games to raise public awareness of health issues (Sarkar, 2013).

Pros and Cons

Some advantages of simulation over experimentation with the real system are lower cost (since we do not build the real system), less experimental time (since virtual time can be speeded up), absolute control of the experimental conditions (we have less control over events in the real system).

Robinson (2004) lists the advantages of simulation modelling over other modelling approaches, such as the analytical approach in which the system is analysed using purely theoretical methods:

- Modelling variability: Some other approaches cannot model variability. For those approaches that are able to do so, modelling variability usually increases their complexity.
- Restrictive assumptions: Simulation requires a few simple assumptions to simplify the model. But many other modelling approaches require certain assumptions.
- Transparency: It can be difficult for a person to understand mathematical equations or spreadsheets. Simulation helps solve the problem by creating an animated display which even a non-expert can study and thus understand the system.
- Creating knowledge and understanding: When building the system, the experts require a lot of information about the system, as well as time to analyse and think about it. Just by building the model, the experts may have enough understanding to make good suggestions for improvements.

On the other hand, simulation has its drawbacks:

- Expensive: such as cost of software and consultants.
- Time consuming: takes time to collect data, perform different experiments.

- Expertise needed: requires domain experts and simulation developers.
- Data hungry: requires huge amount of data which need to be collected and may require pre-processing before use.
- Overconfidence: be careful and consider the validity of the model, assumption, and simplification when analysing the results.

Main paradigms in simulation

There are three main paradigms in simulation:

- System Dynamic Simulation
- Discrete Event Simulation
- Agent Based Simulation

The first paradigm System Dynamic was developed in the 1950s to help managers improve their understanding of industrial process by showing how organization structure, amplification and time delays interact ([Forrester, 1958](#)). The System Dynamic approach helps to understand the structure and behaviour of complex systems by representing the real world process with stocks, flows between stocks, and information which determines the value of stocks.

Second, Discrete Event Simulation approximates the real-world processes with a chronological sequence of events. Even the process in the real world is continuous, in many cases, when we analyse a real-world process, we only consider some significant moments, which are *events*. Each event in a simulation occurs at one point in time and makes changes to the system's state.

The final paradigm Agent Based Simulation is an individual-centric, decentralized approach. In Agent Based Simulation, a system is modelled as a collection of agents, which are autonomous decision-making units with diverse characteristics. The interaction of the individual behaviour of many agents results in the global behaviour of the system. When designing the model, we identify the agents (which can be not only people but also vehicles, cars, cities, products, etc.), define their behaviour, put them in an environment, establish connections, and run the simulation.

2.2.2 Agent Concept

Agent is a concept that has been applied to many disciplines. Before defining agent concept, let's clear out a confusion. In Economics, there is a term called *economic agent*. An economic agent is an economic decision maker such as a person, company, or organization, which has

an influence on the economy by actions like producing, buying, selling, investing, or taxation. Thus economists can refer to the people they study in Analytical Game Theory as economic agents. In this thesis and in the context of Agent Based Simulation, the *agent* concept refers to software agents or computational agents, which are decision makers in a computational simulation.

There are many definitions for agents with slight differences between fields. [Ferber \(1999\)](#) provides a comprehensive definition. An agent is a physical or virtual entity:

- (a) “which is capable of acting in an environment,
- (b) “which can communicate directly with other agents,
- (c) “which is driven by a set of tendencies (individual objectives or of a satisfaction/survival function which it tries to optimize),
- (d) “which possesses resources of its own,
- (e) “which is capable of perceiving its environment (but to a limited extent),
- (f) “which has only a partial representation of this environment (and perhaps none at all).
- (g) “which possesses skills and can offer services,
- (h) “which may be able to reproduce itself,
- (i) “whose behaviour tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representation and the communications it receives.” ([Ferber, 1999](#))

Also addressing the agent term, [Wooldridge and Jennings \(1995\)](#) distinguished two general usages of the term: *a weak notion of agency* with properties consisting of autonomy, social ability, reactivity, pro-activeness; and *a strong notion of agency* where, in addition to the previous properties, agent is conceptualized or implemented using human-like concepts: knowledge, belief, intention, emotion. Even though it is a weak notion, the four properties are commonly adopted for agent definition:

1. Autonomy: agents operate without human intervention and have a certain degree of control over their actions and internal states.
2. Social ability: agents can communicate with other agents via agent communication language.

3. Reactivity: agents perceive the environment and response to the perceived changes in a timely manner.
4. Pro-activeness: agents do not just react to external stimulus, but take initiatives according to internal goals.

The following sections look further into the software agent concept in the two fields of Computer Science and Economics to see how they vary.

2.2.3 Multi Agent System

Agent Based Simulation originates in Multi Agent System from Computer Science. In the 1990s, with the implementation of the computer networks (local area network and the Internet), it is the outset of *Distributed Artificial Intelligence* (DAI). DAI investigates knowledge modelling, communication and reasoning techniques that computational agents are needed to collaborate and solve problems (O'Hare and Jennings, 1996). One of the main research areas in DAI, *Multi Agent System* (MAS), concerns a collection of agents aiming to solve a given problem that is beyond their individual capabilities and/or knowledge. MAS can be viewed as a society of agents that interact by sending messages, exchanging knowledge, collaborating or negotiating to achieve either their own interest or a global goal (Oprea, 2004). MAS becomes a well-established and applied branch of Artificial Intelligence.

In the rise of computer networks where the resource and information required for a task can be distributed over many devices, O'Hare and Jennings (1996) explain the advantages of a MAS over a single centralized problem solver:

- Faster problem solving: utilize parallelism.
- Decrease communication: transmit partial solved solutions to other agents rather than raw data.
- More flexibility: agents with different abilities can dynamically collaborate.
- Increase reliability: allow agents to take on responsibility of failed agents.

Applications of MAS focus on solving a specific practical or engineering problem. For instance, Shehory et al. (1999) designed a MAS to aid aircraft maintenance processes. After inspecting an aircraft, mechanics fill in a form which later will be analysed by a form agent, then information will be retrieved by history agents and manual agents for the mechanics. MAS also has applications in many areas: resource management; manufacturing planning, scheduling and control; energy monitoring, diagnosis, and control; electronic commerce; virtual enterprise

Oprea (2004). Additionally, MAS was also applied to collective robotics. An example is RoboCup (Kitano et al., 1997), a robot soccer world cup, where two teams of robots controlled by software agents play soccer against each other in a highly dynamic environment.

Russell et al. (2003, pp. 46-54) categorize agents into five types based on elements that drive the decision-making.

1. Reflex agent: look up action based on condition-action rules.
2. Reflex agent with state: State is collection of values that partially describe the world. Agents in partially observable environments can keep track of unseen portions of the world (searching a maze, driving a car).
3. Goal-based agent: agent has goals and tries to predict effect of actions in the future.
4. Utility-based agent: adding a utility function to evaluate satisfaction on top of the goal.
5. Learning agent: agent can take feedback, make improvement, then suggest exploratory actions.

2.2.4 Agent-Based Modelling and Simulation

MAS gets widespread attention of the social sciences and Economics (Macal and North, 2009). After adapting MAS to other sciences, a new modelling approach has arisen. Even though there are many names for the methodology such as Agent Based Modelling, Agent Based Simulation, Multi-Agent Based Simulation, Individual-Based Modelling, ABMS will be used throughout this thesis. The fields applying ABMS include Economics (Tesfatsion, 2006), Ecology (Grimm and Railsback, 2013), Psychology (Smith and Conrey, 2007), Sociology (Macy and Willer, 2002), Archaeology (Romanowska, 2015). Some examples of diverse applications of ABMS are prediction of epidemics (Bagni et al., 2002), air traffic control (Conway, 2006), offshore wind energy analysis (Mast et al., 2007), predator-prey relationships (Mock and Testa, 2007), urban crime analysis (Malleson et al., 2010), fall of ancient civilization (Kohler et al., 2005), subway station evacuation (Li et al., 2016), and smart meter deployment planning (Zhang et al., 2016).

One of the reasons for the widespread application of ABMS is that systems investigated by social sciences and economics are usually *complex systems* and have *emergent behaviour*. Wilensky and Rand (2015, p. 5) argued that our need of understanding complex systems has resulted from the growth of human knowledge. In early history, we could survive with only understanding simple systems and local effects. But as technology and science have advanced, humans are more connected and become more affected by complex interactions. Thus, we

started to recognize these interactions in the systems. For example, a disease in one country can spread around the world within days and cause a worldwide epidemic; thoughtless financial decisions in one country can affect the economy of the rest of the world; cutting down rain forest in Brazil can have a negative impact on the climate of distant countries or even the entire Earth. One of the characteristics of a complex system is *emergence*. Goldstein (1999) defined emergence as “the arising of novel and coherent structures, patterns, and properties during the process of self-organization in complex systems”.

One example of emergence is the flocking of birds such as the classic V formation. Why do bird flocks fly in that formation? How does that beautiful pattern emerge? The V formation has intrigued many scientists (Hummel, 1983; Lissaman and Shollenberger, 1970; Weimerskirch et al., 2001). Science has established that the V formation saves energy of the whole flock, they take turns to lead the formation, and interestingly it is not centrally organized. However, it is not easy to explain why the formation occurs. People can guess that there is a leader in the front and the other birds follow him, or the biggest and strongest birds fly in front to deal with the air pressure. To explain the flocking behaviour, Reynolds et al. (1987) built a simulation which demonstrates that all the birds follow a set of rules and the flock emerges without a special leader. In his model, the birds follow three rules: collision avoidance (avoid collisions with nearby birds), velocity matching (attempt to match velocity of nearby birds), flock centring (attempt to stay close to nearby birds). So in the flocking example, an agent-based simulation can be constructed with each bird as an agent. Then using this simulation, different rules can be experimented with to see how and when the flocking behaviour emerges.

Additionally, ABMS is suitable for the simulation of human-centred systems (Siebers and Aickelin, 2011), in which social scientists and economists are usually interested. An agent is a very good representation for a human (even an animal) since both have many similar properties (Wooldridge and Jennings, 1995):

- Heterogeneous: with their own attributes, behaviours, goals.
- Autonomous: be able to adapt and modify their behaviour without direct intervention of others.
- Social ability: interact and communicate with others.
- Reactive: perceive and react to the changes in environment.
- Proactive: demonstrate goal-directed behaviour.

In terms of modelling principles, there are two influential approaches: KISS (Keep-It-Simple-Stupid) and KIDS (Keep-It-Descriptive-Stupid). *KISS approach*, as termed by Axelrod

(1997), starts with the simplest model and allows for more complex models if the simpler ones turn out to be adequate. *KIDS approach*, proposed by Edmonds and Moss (2005), suggests to start with a descriptive model as evidence and resources allow and then only simplifies it when this turns out to be justified.

In conclusion, ABMS can be used as a new methodology to gain understanding of the complex systems in Social Science and Economics. The increasingly popular application of ABMS in Economics created a sub-field, Agent-based Computational Economics.

2.2.5 Agent-based Computational Economics

Economies are complex dynamic and human-centred systems. There are many modelling methods that have appeared over time. Hamill and Gilbert (2016) briefly review the history of economic modelling with three models: (1) traditional macroeconomic models that comprise a set of equations but lacks a formal link with microeconomic analysis, (2) dynamic stochastic general equilibrium (DSGE) models that attempt to link macroeconomics with microeconomics but assume the economy can reach and sustain an equilibrium, and (3) ABMS that models the economy as a dynamic system of heterogeneous agents. The 2008 economic crisis prompted the need of re-evaluation of economic modelling and played an important role in the development of ABMS. In 2009, Farmer and Foley argued in *Nature* that ABMS is a great direction to go forward after the crisis. A year later, *The Economist* (2010) discussed whether ABMS can do better than conventional models, and pointed out the potential of ABMS.

Another work that has great influence on ACE is the work at Santa Fe Institute in New Mexico. The Institute fosters multi-disciplinary collaboration and focuses on *the sciences of complexity*. They view “the economy as an evolving complex system” and published three books in the same name (Anderson et al., 1988; Arthur et al., 1997; Blume and Durlauf, 2006). Their perspective, named the Santa Fe approach, has these main characteristics: “cognitive foundations, structural foundations, no global controller, and exhibiting continual adaptation, perpetual novelty and out-of-equilibrium dynamics” (Richiardi et al., 2012). So even before the global economic downturn in 2008, researchers had tried to overcome the restrictive assumptions in conventional models and viewed the economy as a non-linear, dynamic system.

Tesfatsion (2006) categorizes ACE research into four strands based on objectives:

- “*Empirical understanding*: Why have particular global regularities evolved and persisted, despite the absence of centralized planning and control?”
- “*Normative understanding*: How can agent-based models be used as laboratories for the discovery of good economic designs?”

- “*Qualitative insight and theory generation*: How can economic systems be more fully understood through a systematic examination of their potential dynamical behaviours under alternatively specified initial conditions?”
- “*Methodological advancement*: How best to provide ACE researchers with the methods and tools they need to undertake the rigorous study of economic systems through controlled computational experiments?” (Tsfatsion, 2006)

Chen (2012) reviews four economics origins (markets, cellular automata, automated tournaments, economics experiments) that motivate different concepts and designs of agents in ACE. Chen also summarizes the development of agents in ACE into three paradigms: simple-programmed agents, autonomous agents, and human-like agents.

- Simple-programmed agents: agents that follows very simple rules.
- Autonomous agents: agents whose behavioural rules are automatically generated by computers.
- Human-like agents: agents that try to mimic human intelligence.

2.2.6 Compare and Contrast between Multi Agent System and Agent-based Computational Economics

The differences between agent-based models in Computer Science and Social Science have been discussed in Gotts et al. (2003, p. 72) and Conte et al. (1998). This section compares and contrasts between MAS (from Computer Science) and ACE (from Economics). There are six differences: attention, development process, driven by data or theory, self-interest agent, complex dynamics, strong or weak notion of agent.

1. Attention

MAS is application oriented and focuses on solving software engineering problems. On the other hand, ACE investigates theoretical hypotheses and seeks understanding of human behaviour in economic context.

2. The development process

Figure 2.3 is created to describe the differences between modelling processes in Computer Science and Experimental Economics.

- Computer Science: From a real life’s problem, a scenario is created to build a conceptual model. Data can be collected by direct observation from real life,

or surveys. From the data, different behaviours of agents can also be derived to provide additional information for the conceptual model. Then a simulation is programmed from the conceptual model. Collected data can be used to parameterize the simulation. Finally, the simulation will help to provide solutions to the problem, or try to predict real life.

- **Experimental Economics:** From theory or a real life problem, economists formulate a question or a hypothesis. Then a conceptual model of behaviour, which is usually a simplified version of a real life situation, is created. From the conceptual model, a lab experiment is designed and conducted to collect data of human behaviour. Also a simulation is built from the conceptual model. The data collected from the lab experiment can be used to parameterize the simulation. Finally, using the simulation, economists gain better understanding, answer the formulated question or confirm the hypothesis, thus improve theory or solve the real life problem.

3. Data-driven vs. Theory-driven

The models of computer scientists are data-driven and the field has developed many experiences with data mining and visualization techniques. On the other hands, models of ACE are mostly theory-driven, i.e. agents' behaviours in the model are based on theories. But currently there are many efforts to use data very differently in modelling social and economic systems. "The social sciences are currently experiencing a transition from a data-poor to a data-rich situation. This allows one to falsify models, calibrate their parameters, and to move to data-driven modelling approaches" ([Helbing, 2012](#)). [Hassan et al. \(2008\)](#) proposes a flow diagram for the data-driven ABMS approach. Researchers can use not only data from lab experiment but also web experiment and massive multi-player games.

4. Self-interest

Agents in MAS are self-interested since they need to be stable and behave in a desired manner. In contrast, ACE is interested in real-world human behaviour, thus ACE's agents are not always self-interested (for example human-like agents can make mistakes).

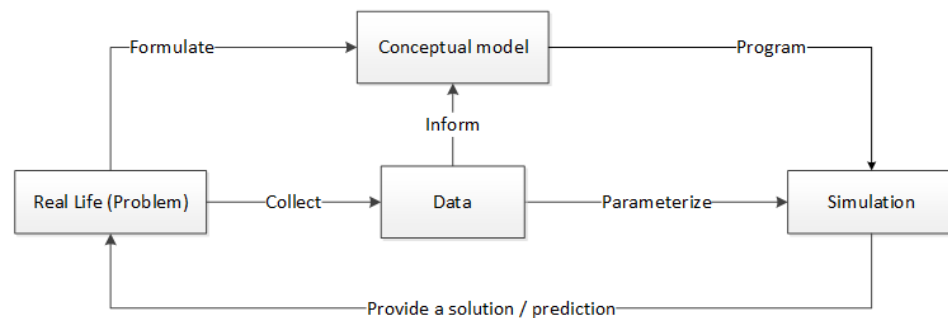
5. Complex dynamics

Research in ACE involves elements that raise system dynamics; while for MAS, complex dynamics is difficult to control and should be avoided.

6. Strong vs. Weak notion of agent

Agents in ACE are used as a "metaphor" or a "conceptual support", and "do not possess any of the properties generally assigned to the agents used in MAS" ([Drogoul et al.](#),

Computer Science



Experimental Economics

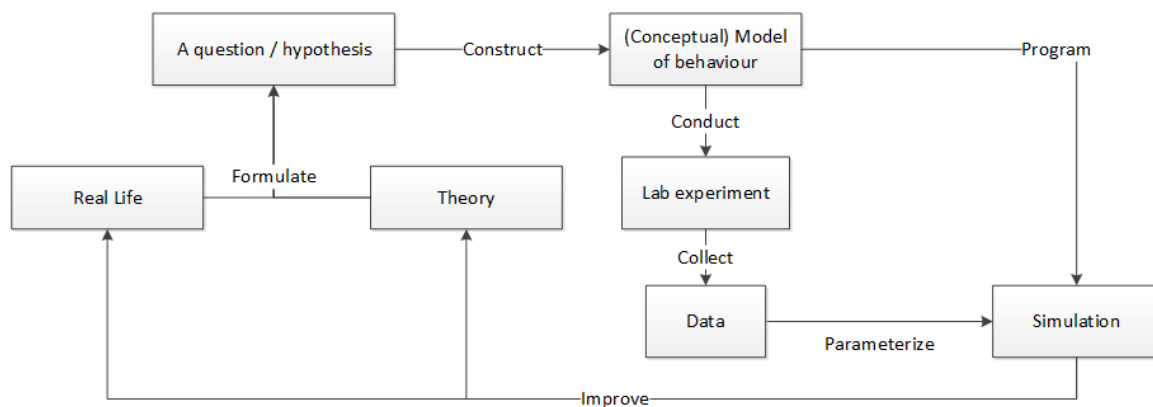


Fig. 2.3 Simulation processes used in Computer Science vs. Experimental Economics

2003). For example, in [Chen \(2012\)](#), there is a type of ACE agent named *zero-intelligence agent* which technically is a *randomly behaving agent*. This will not be classified as agent according to the definition in MAS. However, this is not the only type of ACE agent; there are many ACE agents with properties like agents in MAS: autonomy, proactive, learning capability.

2.2.7 Basic Software Engineering Concepts

Since a simulation is software, its development involves many software engineering techniques. When programming a simulation, in general, there are two approaches: programming from scratch or using a simulation software package. Both approaches use an *object-oriented programming language* because object-orientation is the mainstream development paradigm. *Object-orientation* is about organizing your software around objects. To understand the topic, some key terminologies will be introduced. To cope with the complexity of the real-world,

humans use an *abstraction* process and think in terms of objects. For example, looking at traffic, humans can decompose that real-world system into different objects: cars, bikes, traffic lights. An *object* is a basic unit of construction, that has some *attributes*, and can perform actions (*operations* or *methods*). So even though there are many cars on the road, they all have attributes such as colour, size, brand, and they can perform actions like accelerate, brake, turn. Objects of the same type are grouped into a *class*. For instance, there can be hundreds of cars on the road with different attributes but they all belong to a class of car, which have the same characteristics.

Object oriented programming (OOP) is an implementation method in which software is organized as a collection of objects, each of which is an instance of a class. To programming a complex system, it is inefficient to just start coding right away. To have a good system, one should analyse the requirements of the project, then develop a design from an object-oriented point of view that satisfies those requirements. This process is called *Object-oriented Analysis and Design (OOAD)*. The fundamental idea of OOAD is to break a complex system into its various objects (Booch, 2007). One of the most important design principles in software engineering is *separation of concerns*, which is the idea of separating computer software into sections such that each section addresses a separate concern and overlaps in functionality are reduced to a minimum.

There are many OOAD processes in software engineering, but they all use one graphical language to communicate their results. The name of this language is *Unified Modelling Language (UML)* developed in the 1990s by Grady Booch, James Rumbaugh, and Ivar Jacobson (Deitel and Deitel, 2011). UML is widely used because it is flexible, extendible, and independent of OOAD processes. It provides a feature-rich graphical representation scheme for the modeller to develop and express their object-oriented systems.

Lastly, there is a field named *Agent-Oriented Software Engineering (AOSE)* that researched techniques for engineering agent systems, including development methodology, formal specification and verification, and implementation of agent specifications (Wooldridge and Ciancarini, 2001). Many AOSE methodologies have been proposed. Each has several stages in the development process and was designed for different types of agent-based systems. See Cossentino et al. (2014) for a gathered documentation of 16 popular methodologies. The field is still developing, thus its use is not popular in ABMS field, and will not be discussed in detail in this thesis.

2.3 The Public Goods Game

2.3.1 Public Goods and the Free Rider Problem

Public goods are goods that people can consume without reducing their availability for others, such as broadcast television, radio signals, street lights. *Pure public goods* are:

- Non-excludable: a person cannot be prevented from enjoying their benefits.
- Non-rival: benefits are the same for everyone.

Additionally, based on the two properties of excludability and rivalry, public goods belong to an array of four other types of resources shown in Table 2.2. Within those four, “common pool resources” are very closely related to “public goods”. They are both non-excludable, thus can be used by many people. But in contrast to public goods, subtraction from the common pool resources decreases the availability for others. So the problem of Common Pool Resource game (CPR) is congestion or overuse; while for PGG, the problem is free riding. Both are collective action games: strategic situations that involves many players, of which collective actions affect the situation. Since the provision of public goods and management of common-pool resources are both important to the governance of social-ecological systems, they are usually mentioned and discussed together.

The dilemma of public goods is having individuals investing effort in creating a public good, while everybody, including those who do not contribute, benefit from it. People who do not contribute are called *free riders*; people who contribute are *cooperators*. PGG is a standard Experimental Economics approach to study human cooperation. In this game each player faces a dilemma: to invest into public goods and potentially get a higher return through cooperation in the future; or keep their endowment which essentially means free riding on other people’s contributions. Analytical Game Theory assumes that individuals maximize their own payoffs and the equilibrium predicts that the public good will not be provided because everyone will free ride. However, laboratory experiments of PGG has found that there is variability in behaviour: some individuals cooperate while others free ride (Fehr and Gächter, 2002; Fischbacher et al., 2001). Since cooperation is a fundamental feature of human society, it is important to understand why people choose to free ride, and what factors can decrease levels of free riding. Subsequently, we will discuss three approaches that can be used to analyse the PGG: Analytical Game Theory, Laboratory Experiment, and ABMS.

Table 2.2 Games categorized by rivalry and excludability

	Excludable	Non-excludable
Rivalrous	Private goods (food, clothes)	Common-pool resources (fish stocks, coal)
Non-rivalrous	Club goods (cinemas, private parking)	Public goods (air, radio signal)

2.3.2 The Analytical Game Theory Approach

In a simple case, PGG can be analyzed using the Prisoner's Dilemma. Suppose there are two players A and B and a public good which is worth the value of V . The cost of this good is C . The public good is provided if one or both contribute. The cost is shared between contributors. If both players contribute, each pays $C/2$. If only one does, then that person pays the full cost C . If no one contributes, the good is not provided. In summary, the payoff table is shown in Table 2.3.

Table 2.3 A payoff table of PGG ($C > V > C/2$)

		B	
		contribute	free ride
A	contribute	$V-C/2, V-C/2$	$V-C, V$
	free ride	$V, V-C$	$0, 0$

Note that for the PGG, the payoff when both contribute must be greater than 0 ($V - C/2 > 0$); and when a player is exploited, s/he will lose money ($V - C < 0$). Since Table 2.3 of PGG has the structure of Table 2.1b of the Prisoner's Dilemma, it is a version of the Prisoners' Dilemma. Therefore, with the same analysis as in Section 2.1.1, "free ride" strongly dominates "contribute". Two rational players will both choose to free ride. Therefore, the theoretical prediction of this game is non-cooperation².

If we desired to have both players contribute, penalties can be used to solve the problem. Suppose that free riding behaviour can be observed and a penalty P is enforced on free riders. The payoff matrix can be updated to include the punishment value of P . Table 2.4 show the payoffs when punishment is enforced. For the mutual condition to be the Nash equilibrium, conditions for "contribute" to strongly dominate "free ride" are $V - C/2 > V - P$ and $V - C > -P$. Thus the conditions of value P are $P > C/2$ and $P > C - V$.

The one-shot PGG (with no penalty) makes the following assumptions: players make a decision independently and simultaneously; players cannot communicate in any way; players

²The game theory analysis can be extended further to n-player PGG (Dixit and Skeath, 2015, see Ch. 11) but will not be covered in this thesis.

Table 2.4 A payoff table of PGG with punishment

		B	
		contribute	free ride
A	contribute	V-C/2, V-C/2	V-C, V-P
	free ride	V-P, V-C	-P, -P

have complete and common knowledge of the game structure and payoffs of all others under all combinations of strategies. If such a game is repeated for a finite number of times, the predicted outcome is still the Nash equilibrium of non-cooperation. This can be reasoned with backward reasoning: if one would not contribute in the last game, they would not cooperate in the previous game, the game before that, . . . all the way to the first game (Poteete et al., 2010, p. 145).

As discussed in Section 2.1, there are criticisms of these strong assumptions. Thus many experiments in Economics have been performed to evaluate the assumptions and predictions. The next section discusses the laboratory experiment approach for PGG.

2.3.3 The Laboratory Experiment Approach

Basic structure of a PGG experiment

The standard experiment of linear public goods provision is referred to as Voluntary Contribution Mechanism by Isaac and Walker (1988a,b). In the experiment, participants in a group of N people, are given an initial endowment ω each, and have to make an investment decision. If participant i contributes x_i of ω for the provision of public goods, his/her payoff (π_i) is:

$$\pi_i = \omega - x_i + MPCR \sum_{j=1}^N x_j \quad (2.2)$$

Marginal Per Capita Return (MPCR) is defined as “the ratio of benefits to costs for moving a single token from the individual to group account” (Isaac and Walker, 1988b). A linear PGG has a free rider problem when $MPCR < 1$ and $N \times MPCR > 1$.

For example, in an experiment by Fischbacher et al. (2001), individuals in a group of four were given 20 tokens, and decided how much they wanted to invest into a public good so called “project” ($N = 4$ and $\omega = 20$). The payoff function is $\pi_i = 20 - x_i + 0.4 \sum_{j=1}^4 x_j$. This means that if everyone invested all 20 tokens, the sum of invested tokens is 80. With the marginal payoff of 0.4, each will get the profit of $0.4 \times 80 = 32$. Thus the payoff for each person is $20 - 20 + 32 = 32$. On the other hand, if there are three cooperators and one free rider, the sum of tokens is 60 and the profit is $0.4 \times 60 = 24$. Each of the three cooperators has the payoff

of $20 - 20 + 24 = 24$, while the payoff of the free rider is $20 - 0 + 24 = 44$. The free rider receives more tokens compared to when everyone is contributing, but the payoff of the whole group decreases.

Insights from PGG experiments

A series of experiments were raised to explore the factors that improve cooperation. The experiments are computer-based. Participants are recruited, come to the laboratory, and make decisions using computers under different experimental settings. [Kagel and Roth \(1995, pp. 30-35\)](#) reviewed the early PGG experiments during the 1970-1980s. The PGG studies began with one-shot decisions and compared different decision mechanisms; however, they found little to no free riding ([Bohm, 1972](#); [Scherr and Babb, 1975](#); [Sweeney, 1973](#)). Afterwards, experiments had involved controlled artificial public goods and detected some degree of free riding behaviour ([Smith, 1979a,b, 1980](#)). Then repetition was introduced into experiments ([Banks et al., 1988](#); [Isaac et al., 1985](#); [Kim and Walker, 1984](#)); contribution generally decreases over time and there are results at odds with previous experiments.

[Ledyard \(1995\)](#) reviews PGG experiments and notes that even though there are many PGG experiments, most are not systematically approached, i.e. there are at least two factors are different between those experiments. Thus they are not truly comparable with others. However, he summarized 19 variables and their level of effect on the contribution in [Table 2.5](#). Even these variables were discussed separately, there are cross-over effects between factors. Ledyard also pointed out three major findings:

1. In a one-shot game and in the initial stages of finitely repeated games, people generally contribute halfway between free riding and total cooperation.
2. Contributions decline with repeated games.
3. Communication improves the rate of contribution.

[Janssen and Ostrom \(2006\)](#) compared the performance of several decision-making models that are constructed based on a probabilistic choice model with social preference, a learning model, and signalling behaviour. In the same study, Janssen presented three observations from linear PGG and illustrated with data from six experimental treatments of [Isaac and Walker \(1988b\)](#) and [Isaac et al. \(1994\)](#):

1. The average contribution at group level varies with group size and the MPCR.
2. There is a large variance of individual contribution.

Table 2.5 Variables and their effects on contribution, adapted from [Ledyard \(1995\)](#)

Type of variables	Variable	Effect on contribution
Environment	MPCR (marginal per capita return)	++
	Number of subjects	++
	Repetition	–
	Common knowledge	+
	Gender	0
	Homogeneity (symmetry)	+
	Thresholds	+
Systemic	Belief	+
	Economics training	–
	Experience	–
	Friendship/Group identification	+
	Learning	0
	Altruism, fairness	?
	Effort	?
	Risk aversion	?
Design	Communication	++
	Rebates	+
	Unanimity	–
	Moral suasion	?

Note: + means increase, 0 means no effect, - means decrease,

and ? means that this has not been measured yet.

A double symbol means strong effect. A single symbol means weak effect.

3. The changes of contribution levels between rounds vary across players, and can be dependent on experimental parameters and the number of remaining rounds.

Poteete et al. (2010, p. 229-231) constructs an overview of behavioural theory of collective-action dilemmas (both public goods and common-pool resources) on three levels of analysis: (1) individual behaviour, (2) microsituational variables, (3) the broader social-ecological context. Figure 2.4a shows the first level of analysis at the core (inside the rectangle) that is the internal association between learning and norm, trust, level of cooperation, and net benefits. The second level contains microsituational variables that affect the core of individual decision-making. Ten of them, which have significant impacts in experiments, are named in Figure 2.4a (outside the rectangles). The third level, shown in Figure 2.4b, is the broad social-ecological system that impacts the microsituations.

Within the ten microsituational variables, the first six have a positive impact on contribution; the next three have a diversity of outcomes; the last one has negative impact. The effects of each variable are summarized in the following:

1. *Marginal per capita return (MPCR)*. If MPCR is high, players' contribution makes more of a difference. This leads to more contribution.
2. *Security*. If there is security that their contribution will be returned if not enough others contribute, players are willing to contribute.
3. *The reputation of players are known*. When players know about the past history (reputation) of others' contribution, they will likely contribute if the reputation is positive.
4. *Longer time horizon*. Participants can reason that with a longer time horizon, early investment shows others the willingness to contribute to motivate other contributions, and leads to better individual return.
5. *Ability to enter and exit from a group*. The participants can leave when they are dissatisfied with the outcomes, stay or enter when cooperation is high.
6. *Communication*. Communication, either face-to-face or structured messages, allows participants to discuss, increase trust, and convince each other about an appropriate norm to follow.
7. *Size of group*. Participants are more likely to contribute in larger groups than in smaller groups.
8. *Information about average contribution*. In public goods, the average contribution tends to decrease over time. Seeing that, participants also tend to stop contributing.

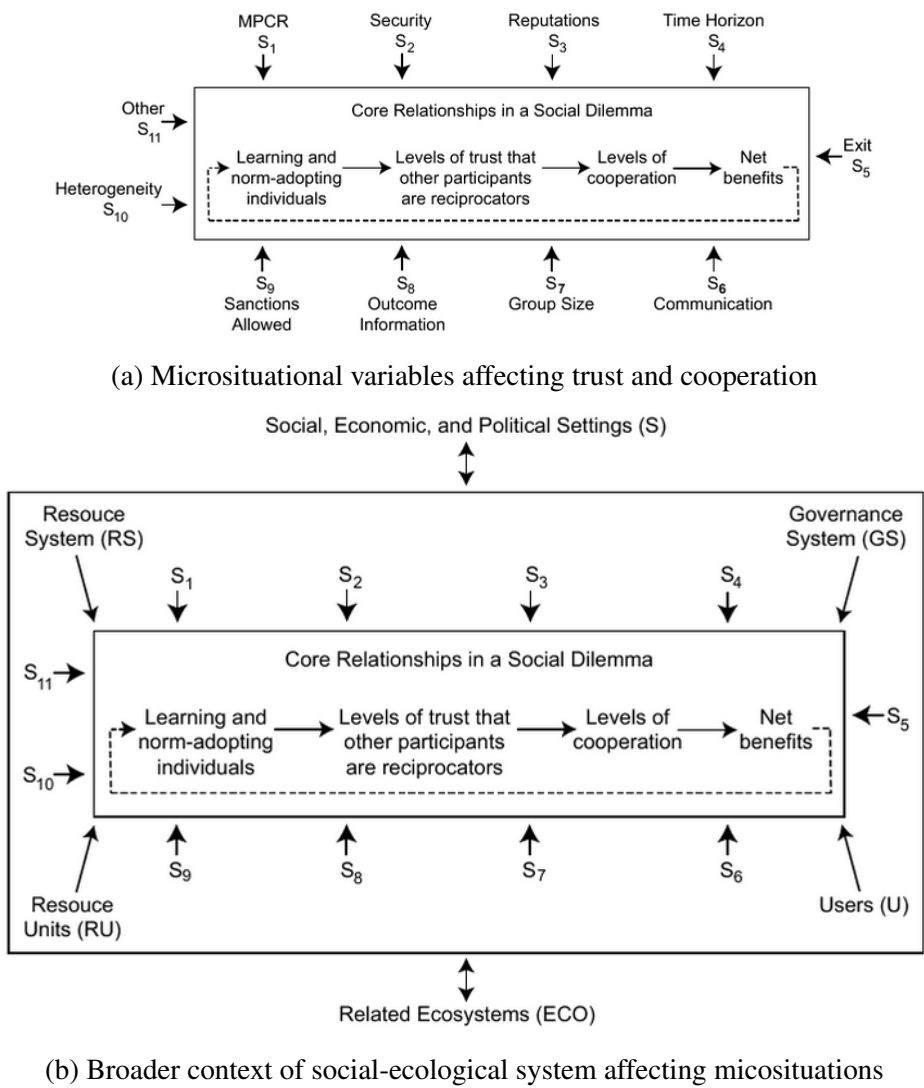


Fig. 2.4 Toward a more general behavioural theory of collective-action dilemmas (Poteete et al., 2010)

9. *Sanctioning capability*. Contribution varies depending on how a sanctioning capability is established, the cost of that ability, and other factors such as reward and communication.
10. *Heterogeneity in benefits and costs*. When a player recognizes that some receive more benefits without paying more cost, s/he may be unwilling to contribute. In a spatial PGG, heterogeneity in positions influences the access to resources and creates differences in power.

Laboratory experiments discovered people have *social preferences*, which is when one incorporates other's welfare into one's own utility function. [Lucas et al. \(2014\)](#) summarizes nine types of social preferences observed in laboratory experiments. The first five types are unconditional; the last four types are conditional.

1. Free Riders: Contribution is always zero for all rounds.
2. Low Cooperators: Contribution is set to a value that is randomly drawn from a lower half in the first round.
3. High Cooperators: Contribution is set to a value that is randomly drawn from an upper half in the first round.
4. Maximum Cooperators: Contribution is set to the maximum value for all periods.
5. Noisy Players: Contribution is randomly drawn from a uniform distribution of every possible value.
6. Conditional Cooperators: Contribution is equal to the belief of the contribution of other members.
7. Economic Altruists: They start at the maximum value and decrease linearly as other members give more.
8. Threshold Players: They do not contribute when others give less than their threshold level, and contribute fully when others give more than the threshold.
9. Hump (triangle) Players: They have a threshold halfway the maximum value. Below this threshold, the more others contribute, the more they contribute (like conditional cooperators). Above this threshold, the more others contribute, the less they contribute (like economic altruists).

2.3.4 The Agent-Based Modelling and Simulation Approach

There are many agent-based models of PGG investigating different aspects of human contribution behaviour under various settings. For example, [Janssen and Ostrom \(2006\)](#) developed agent-based models with different learning models, fitted with empirical data; [Wunder et al. \(2013\)](#) tested stochastic models of decision-making with different agent network; [Andrighetto et al. \(2013\)](#) experimented with punishment in combination with norm-signalling; [Lucas et al. \(2014\)](#) worked on the effects of group composition and social preference heterogeneity; [Farjam et al. \(2015\)](#) compared different punishment mechanisms. There are also models related to a specific real-life PGG such as income tax evasion ([Bloomquist, 2006](#)). A number of studies ([Brandt et al., 2003](#); [Hauert et al., 2002](#); [Helbing et al., 2010](#); [Santos et al., 2008](#)) investigated the spatial PGG where agents live on a grid and interact with local neighbours, and analyzed the spatial structure of interaction and its effects. ABMS can help to validate the results from laboratory experiments, test different hypotheses, or compare and contrast decision-making models. ABMS provides a methodology to gain insights and understanding, and develop theories about human behaviour in the context of public goods.

2.4 Research Gap

The potential and applicability of ABMS in their research is well known by social scientists. With the development of simulation platforms such as Netlogo, Repast Symphony, ABMS has gained popularity among social scientists. However, there are many concerns related to the use of software engineering (SE) methods in ABMS, such as the little focus of SE and the need for standards. [Siebers and Davidsson \(2015\)](#) represent a special section of the current state-of-the-art in using SE methods in ABMS. [Rossiter \(2015\)](#) addresses the structural design of simulations by outlining three best-practice properties of SE and mapping the architecture of these simulation toolkits. [Rossiter](#) also stresses the little focus of SE and cultural issues in adoption where people are “introducing everything by analogy and simulation-specific terminology, ‘hiding’ the generic SE origins and thus obscuring the interdisciplinary links.” [Collins et al. \(2015\)](#) discusses the use and need for standards for ABMS.

Knowing the gap between MAS and ACE (as discussed in Section 2.2.6) along with the little use of SE methods in ABMS, we believe that ABMS of PGG has not been used to its full potential and SE methods from Computer Science can provide a solution for this. There are two aspects that can be improved: development methodology and agent behaviour modelling.

2.4.1 Development Methodology Aspect

Programming a simulation requires knowledge from SE practice, which can be a great obstacle for domain experts (social scientists or economists). The obstacle is in not only programming skills but also the software-design mindset and SE best practices. One approach to overcome these obstacles is that domain experts can learn programming then implement the simulation themselves. As there are many dedicated agent-based environments, this is feasible. These software packages often use object-oriented programming, and ABMS researchers agree that the most natural way to design and implement their model is to adopt object-oriented practices. Nonetheless, such software packages often have a steep learning curve and also understanding the principles of SE will take a substantial time. This is a challenging obstacle for social scientists that never train or use object-oriented programming. Another solution is the collaboration between domain experts and programmers. Domain experts design the simulation, then it can be implemented by programmers. However, to achieve a good simulation with this process, domain experts must provide all the details required for implementation, and programmers must interpret the design correctly. This problem can be solved with a development methodology which incorporates SE methods. Using SE methods as a common medium, the development methodology can help improve communication and understanding between the two sides involved in the simulation model building process. It should require minimal SE knowledge so that it is easy for domain experts to use.

During the literature review, we have not found any suitable development methodology for the previously stated problem. General simulation development processes have been adapted to ABMS (Drogoul et al., 2003; Gilbert, 2007; Gilbert and Troitzsch, 2005; North and Macal, 2007). For example, the methodological process proposed by Drogoul et al. (2003) suggested three roles, two levels of knowledge, three level of agents, and link them together. Focusing on ACE, Li (2013) suggests a general modelling process that specifies a constructive perspective of the economic system, then constructing passive or active agents. Two works of Drogoul et al. (2003) and Li (2013) focus on the conceptual design phase, but do not specify the detail steps involved in system design and implementation. In addition, there are a few methodologies that have details about each process and focus on the SE aspects of the simulation building process. For example, Richiardi et al. (2006) identifies some methodological pitfalls and recommends using UML diagrams, with example of a class diagram and a sequence diagram. A notable suggested methodology is IODA (Interaction-Oriented Design of Agent simulations) by Kubera et al. (2011) that focuses on the design of an agent-independent specification of behaviours, called interactions. IODA is a good step forward but it is designed for large scale simulation and not suited for agents with a limited variety of interactions.

In summary, considering the underuse of SE and the existent methodologies, there is value in designing a new development framework that incorporates SE methods and provides detailed processes for simulation development. There are two noticeable benefits of SE: established lifecycle development and UML as an industry standard modelling language. Therefore, this framework should include a lifecycle for ABMS development and uses UML as the standard specification language.

2.4.2 Agent Behaviour Modelling Aspect

Before ABMS, mathematical equations have been the building blocks of many modelling approaches for social systems. Within the framework of game theory, economists model human behaviours with utility functions along with the assumptions of utility maximization and complete information (Gotts et al., 2003). Social psychologists utilize Variable-Based Modelling (VBM) that uses differential equations to describe changes in variable values over time, or use path diagrams to show causal flow among variables (Smith and Conrey, 2007). Using VBM, scientists search for causal explanations by seeking regular covariations between variables (Smith and Conrey, 2007); while ABMS offers a generative explanation which provides a deeper understanding of the phenomenon (Epstein, 2006). When shifting from the more traditional methods to ABMS, there is still heavy use of equations when it comes to defining a simulation model. However, modelling agent behaviour with solely equations has some disadvantages. An equation that captures all dynamic interactions can require complete information which may not be possible. The equations describing the social process can be totally intractable, for example, Hamilton et al. (1990) uses ABMS to model the sexual reproduction process as an adaptation to resist parasites because mathematical analysis for that problem is impossible. To model human behaviour, there are alternative options such as heuristics that are simple enough to be easily understood yet can express complex behaviours (Ebenhöh, 2006). Since human behaviour is complex and affected by many events and continuously changing factors (such as emotions), there is a need for a design approach that can model these characteristics in agent behaviour. Since MAS has a long research history on modelling and implementing agent behaviour, our development framework can address this issue during design of agent behaviour using SE techniques from MAS.

Laboratory experiment of PGG has involved complex factors of a dynamic environment. For example, communication is proven to increase contribution in experiments. However, to model communication in an agent-based model is not an easy task. We need to know what kind of messages people send around, and how they affect decision-making. Another example of complex factors is the mode of time. Most PGG experiments are in discrete time. People make decisions synchronously in one round, receive information such as average contribution, then

move on to the next one and repeat. However, most real-life PGG situations have a real-time aspect, which can alter the interaction nature in fundamental ways (Simon and Stinchcombe, 1989). There are game theoretic analyses of continuous-time games (Bergin and MacLeod, 1993; Laraki et al., 2005; Neyman, 2012; Sannikov, 2007), and some laboratory experiments (Dorsey, 1992; Friedman and Oprea, 2012; Oprea et al., 2014). For experiments of PGG, Dorsey (1992) employs continuous time in experiments where participants can change their contribution but only the final decisions of the periods are calculated for payment; Oprea et al. (2014) also considers continuous-time PGG but with continuous flow-payoff setting which is strategically different compared to Dorsey's work. A dynamic environment, created by the real-time aspect, allows complex interactions which can be difficult to model. Therefore, our framework supports the development of models that consider these complex dynamics in both discrete and continuous time.

2.5 Summary

ABMS is a relatively new methodology that has proven its capability in modelling complex human-centred systems. ACE is a sub-field of Economics that applies ABMS to model the economy and study human behaviours. This section presents intertwined relationships and the historical context of Analytical Game Theory, Behavioural Game Theory, Experimental Economics, Behavioural Economics, and ACE. PGG is also introduced and related to these research approaches. In addition, the section examines the MAS origin of ABMS, then compares and contrasts between MAS and ACE. Finally, the research gap is discussed and the aim of our thesis is to leverage the potential of ABMS of PGG in the aspects of development methodology and agent behaviours modelling. In the next chapter (Chapter 3), we revisit the aim, identify objectives, and then define our methodology.

Chapter 3

Research Methodology

This chapter presents the aim and objectives (Section 3.1), then develops the research methodology for the construction and evaluation of a new modelling framework. In Section 3.2, we explain the methodology choice. Section 3.3 discusses the strategy used to achieve the objectives. Consideration of simulation softwares is discussed in Section 3.4. Additionally, Section 3.5 describes the design of the initial framework v1. Lastly, Section 3.6 presents the summary of this chapter.

3.1 Aim and Objectives

Our research aim is to leverage the potential of ABMS for PGG focusing on the development methodology of ABMS and the modelling of agent behaviours.

In order to achieve our aim, the main objectives of our research are:

1. Construct a development framework that is tailored to ABMS of PGG by incorporating tools from software engineering.
2. In the development framework, specify a behaviour modelling approach for dynamic behaviours in discrete as well as continuous time.
3. Evaluate the framework for three aspects: modelling principles, embedding of psychological mechanisms, and discrete vs. continuous time settings.

In our aims, there are two aspects: constructing a development methodology which will be resolved by *the first objective*, and modelling agent behaviours which will be addressed by *the second objective*. Then in order to demonstrate how we leverage the potential of ABMS for PGG, the framework are evaluated three aspects of modelling by *the third objective*. For the first aspect, using two main modelling principles (KISS vs. KIDS), we can evaluate the

development lifecycle, establish how UML diagrams can be used for ABMS of PGG, and demonstrate the flexibility in development of the framework. Secondly, since using a sound basis in the discipline of Psychology is a popular way to extend traditional models, the ability to embed psychological mechanisms into agent decision-making is evaluated. Lastly, the framework should be implemented based on evidence from a continuous-time PGG experiment. The continuous-time setting is chosen because it is a real-life dynamic but rarely considered in both Experimental Economics and ABMS.

3.2 Methodology

The type of research was derived from the objectives, and then the research methodology was chosen. [Kumar \(2010\)](#) classifies the types of research on the basis of three different perspectives: applications of the findings of the study, mode of enquiry, and objectives of the study. From the perspective of applications, our research is *applied research* because we applied software engineering methods to construct a new modelling framework for agent-based models of PGG. From the perspective of enquiry mode employed, our research is *quantitative* because we perform a systematic empirical investigation, and the simulation results were measured in numerical form. From the perspective of objectives, we identify the type of research of each of the three objectives. The first and second objective are *descriptive* because the first describes the relevant components of the framework and the second specifies a behaviour modelling approach. The third objective is *exploratory* since they determine the feasibility of applying the framework to different scenarios. To achieve the above objectives, a case study methodology ([Yin, 2009](#)) was chosen because we want to cover contextual conditions: the application of the framework heavily depends on the context of the modelling problem. The case study methodology is also suitable given the time and resource constraints.

Three case studies were chosen in order to evaluate the framework in different contexts and fulfil the third objective. Firstly, evaluating the framework with two modelling principles was addressed in the first case study. Using one real-life situation as a problem, two agent-based models were developed using two different modelling principles: KISS and KIDS. Additionally, since the focus is to understand two modelling processes, the simulation is simple in terms of the structure of the PGG and agents' decision-making. Secondly, to incorporate a psychological model into an agent, the second case study utilized a laboratory experiment of PGG that investigates the correlation between Behavioural Inhibition System and punishment sensitivity. Lastly, the third case study was introduced with the focus on continuous-time PGG. The first case is a nested case with two nested units (i.e. two agent-based simulations were implemented). On the other hand, the second case and third case are single cases. After each case study, based

on the experience learnt from the case study's context, the framework was revised and improved if needed.

3.3 Research Activities

This section outlines research strategy and activities. Figure 3.1 shows an overview of research activities, and how the activities relate to the thesis outline (in red text). There are three different versions of the framework, represented by three rounded rectangles. The initial framework was designed by reviewing the literature, choosing, and incorporating relevant software engineering techniques to the modelling process of PGG. The framework then went through three development phases to evaluate three different aspects of modelling.

1. Modelling principles: The initial framework v1 was used to develop two models with different modelling principles. The first simulation is in discrete time, developed with KISS approach; while the second simulation is in continuous time, developed with KIDS approach. After development phase 1, the framework is revised and improved to the framework v2.
2. A psychological model of punishment sensitivity: Framework v2 was applied to create a simulation which is in discrete time and contains a psychological mechanism in agent decision-making. After this phase, the framework is revised and improved to the framework v3.
3. Continuous time setting: The framework v3 was used to develop a simulation to investigate dynamic behaviours in continuous-time PGG.

The three development phases are aligned with the three case studies described in Section 3.2. At the end of the lifecycle of each case study, the framework will be examined and revised for improvement before applying it to the next case study.

3.4 Choice of Simulation Software

In order to develop an agent-based simulation there are many softwares and toolkits available. They can be divided into three categories (Macal and North, 2014):

1. General (Object-Oriented) Programming Languages: Java, C++, Python
2. Desktop ABMS Application Development:

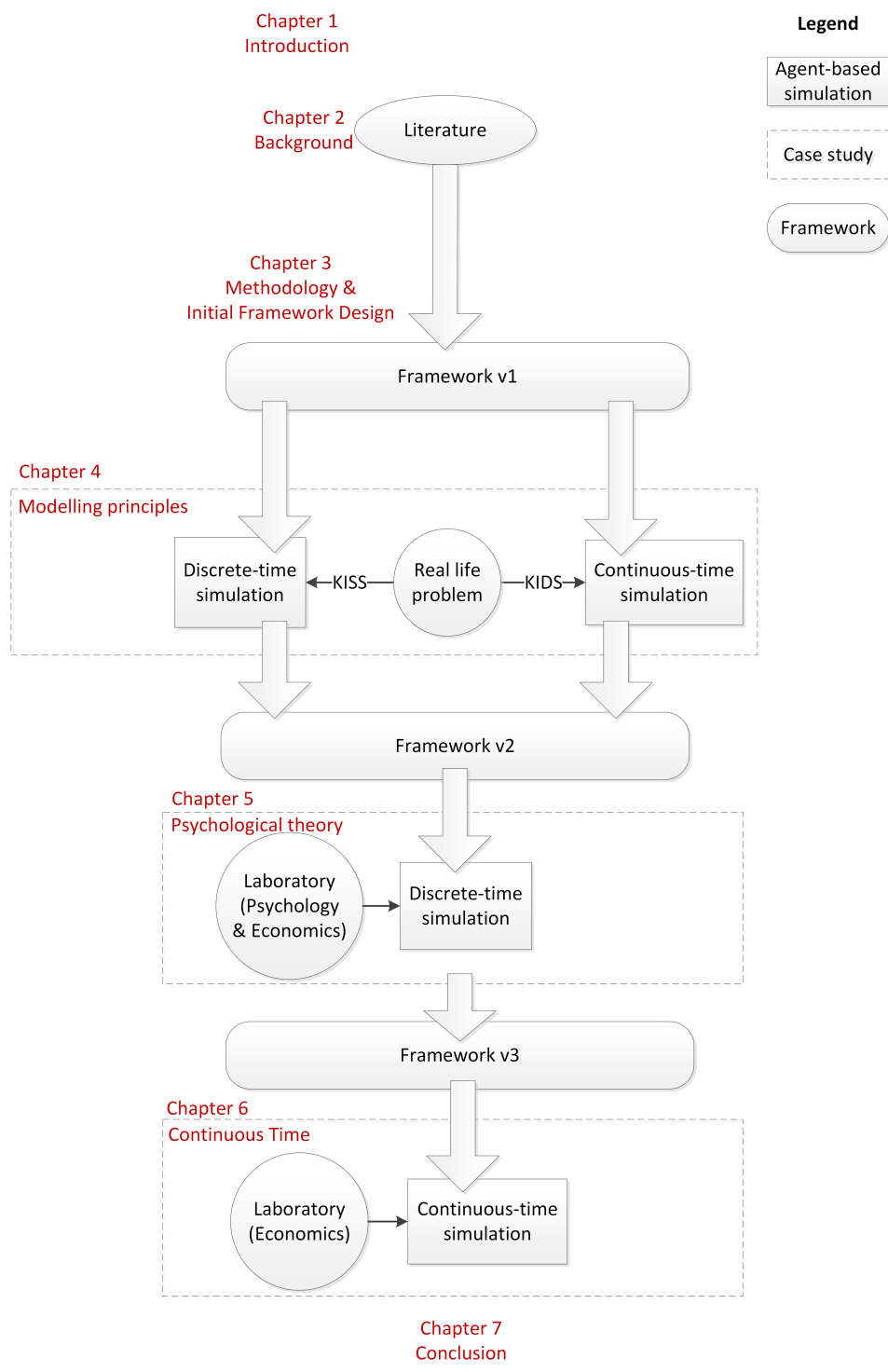


Fig. 3.1 Research activities

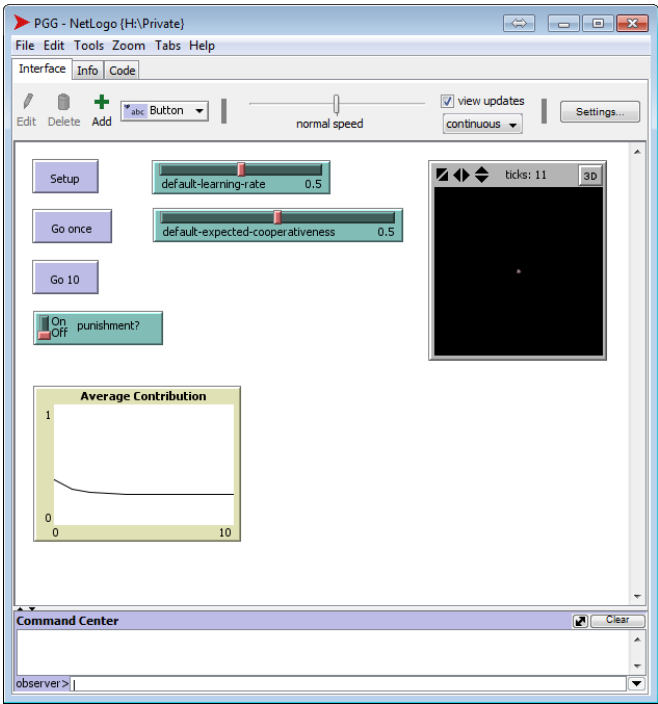
- Spreadsheets: Microsoft Excel using macro programming language VBA
- General Computational Mathematical Systems: MATLAB, Mathematica
- Dedicated Agent-based Environments: NetLogo, Repast Symphony

3. Large-scale (Scalable) Agent Development Environments: Repast, Swarm, MASON, AnyLogic, Simio

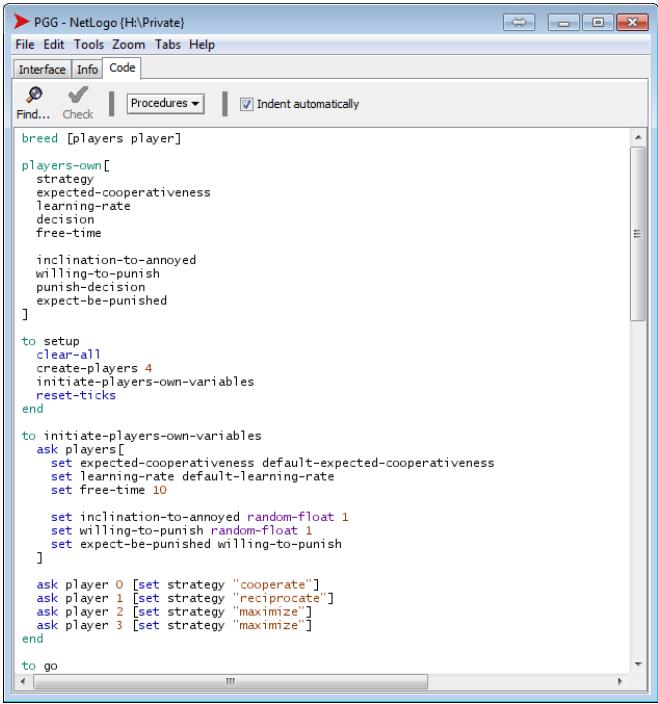
Apart from the requirement of object-oriented support, we want to save time during the development of the simulation because our focus will be on the development process. Thus, we eliminated the options of using a general programming language, and decided to use an agent-based development environment. Among economists, the NetLogo modelling environment ([Wilensky, 1999](#)) is a popular one because it is lightweight and easy to learn. Figure 3.2 shows the screenshots of a NetLogo model. The NetLogo software consists of three tabs: Interface, Info, and Code. The Interface tab defines the interface for users to interact and control the model (control buttons, fields for inputs, plots, etc.). The Info tab contains information related to the model such as model description, how to use, authors, contact information. The Code tab defines and controls behaviours of the models. However, in this project, NetLogo is not a suitable choice because it does not support object-oriented programming. In NetLogo, agent populations can be created with customized variables, but procedural programming is used to program agent behaviours. The programmer can create agents by defining agent types and their attributes, but can not define methods or functions for each agent type. Their behaviours are defined in a central location. Another disadvantage is that NetLogo does not provide extensive tools for continuous time ([Railsback and Grimm, 2011](#)).

After considering several simulation environments, we chose AnyLogic PLE edition ([XJTechnologies, 2014](#)), a multi-method simulation modelling tool developed by XJ Technologies. AnyLogic supports both object-oriented development and a continuous time environment. Figure 3.3 shows the integrated development environment (IDE) of AnyLogic. An IDE is a software that provides facilities (such as a source code editor, a compiler, a debugger, build automation tools) to maximize programmers' productivity during software development. In AnyLogic IDE, there are three sections:

- The left side is the project view that manages the currently opened projects in the workspace.
- The middle is the graphical editor that is the place used to define the structure and behavioural elements of agents.
- The right side is properties view that is used to view and modified the properties of an item selected from the graphical editor.



(a) Interface tab



(b) Code tab

Fig. 3.2 A screenshot of a NetLogo model

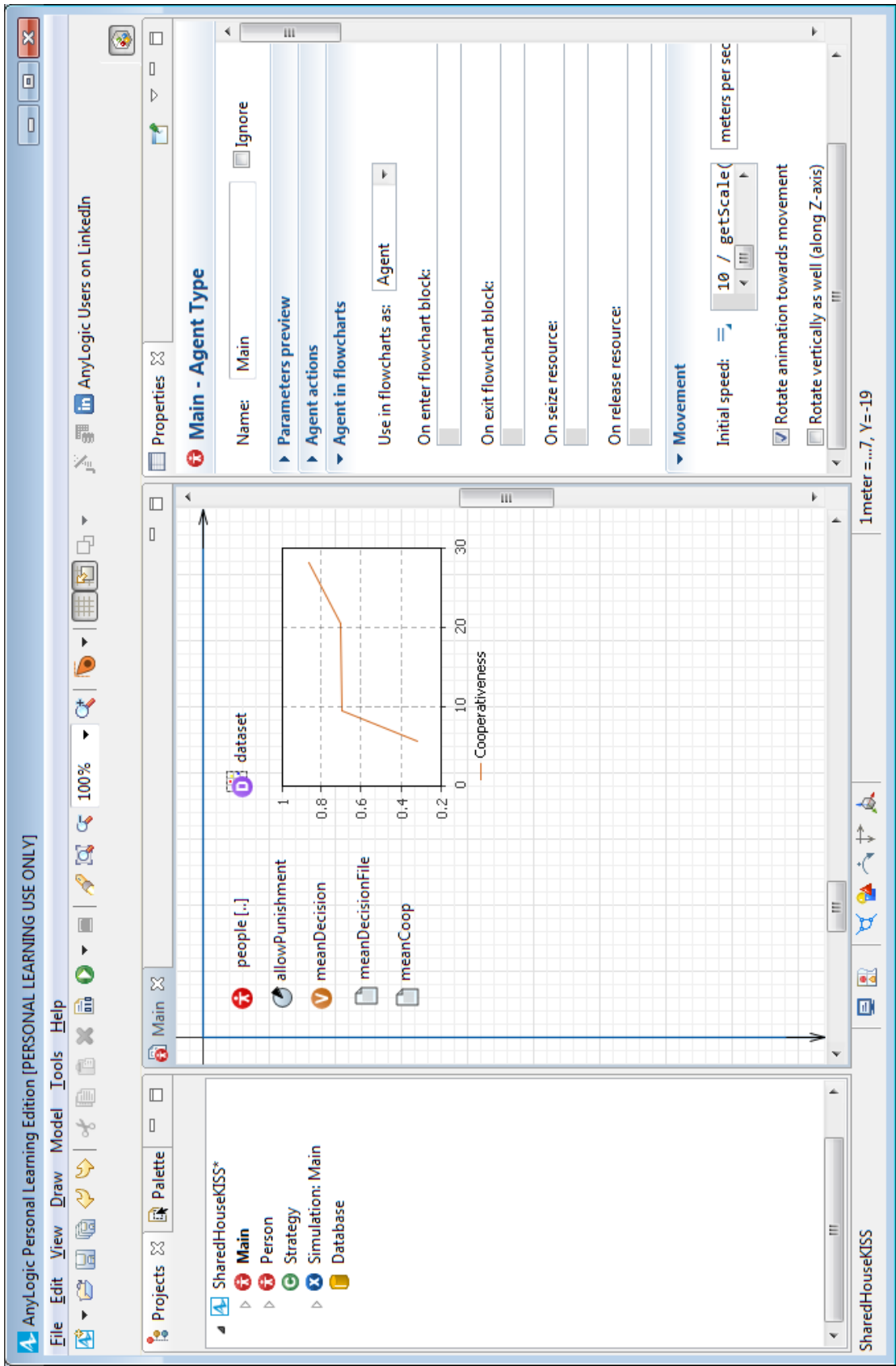


Fig. 3.3 AnyLogic IDE

3.5 Design of the Initial Framework

In this section, we discuss design decisions during the creation of the initial framework, namely v1, and decided to refer to the proposed framework using an acronym: ABOOMS (Agent-Based Object-Oriented Modelling and Simulation). Section 3.5.1 states the purpose of the ABOOMS framework and overview of the design process. The design is split into two sections: the macro process (Section 3.5.2) and the micro process (Section 3.5.3). Section 3.5.4 sums up the ABOOMS framework v1.

3.5.1 The ABOOMS Framework Overview: the Macro and Micro Process

The proposed framework has three purposes:

1. To provide an overview of major components and activities engaged in developing agent-based models in Economics.
2. To encourage economists and social scientists to practise the object-oriented mindset and UML diagrams.
3. Contribute to the development of the best practices for building social simulation models.

The ABOOMS framework approaches the design from two perspectives: the macro process (the development lifecycle) and the micro process (the analysis and design process). The macro process guides the overall development lifecycle of the simulation study, from analysing problem, identifying the requirements to the final simulation and experimentation. On the other hand, the micro process discusses the analysis and design techniques to get from the requirements to implementation.

3.5.2 Design of the Macro Process

In this section discusses the design of the macro process, which is a development lifecycle for the ABOOMS framework v1. It is an integration of three processes: the lifecycle of a simulation study, the process of laboratory experiment, and the development process of Object-Oriented Analysis and Design (OOAD).

The Lifecycle of a Simulation Study

Each case study is a simulation study, and since the framework is applied to all case studies, we use a lifecycle as a base to design the framework. Figure 3.4 shows the comprehensive

lifecycle of a simulation study (Robinson, 2004), composed of key stages (boxes) and processes (arrows). The key stages represent important deliverables:

- A conceptual model: a description of the model to be developed.
- A computer model: a simulation model implemented on a computer.
- Solutions and/or understanding: derived from the results of the experimentation.
- An improvement in the real world: obtained from implementing the “solutions and/or understanding” to the real world.

Using the generic lifecycle of a simulation study described by Robinson as a base, we focused on the four processes (conceptual modelling, model coding, experimentation, implementation) and discussed how they are relevant to PGG in the following.

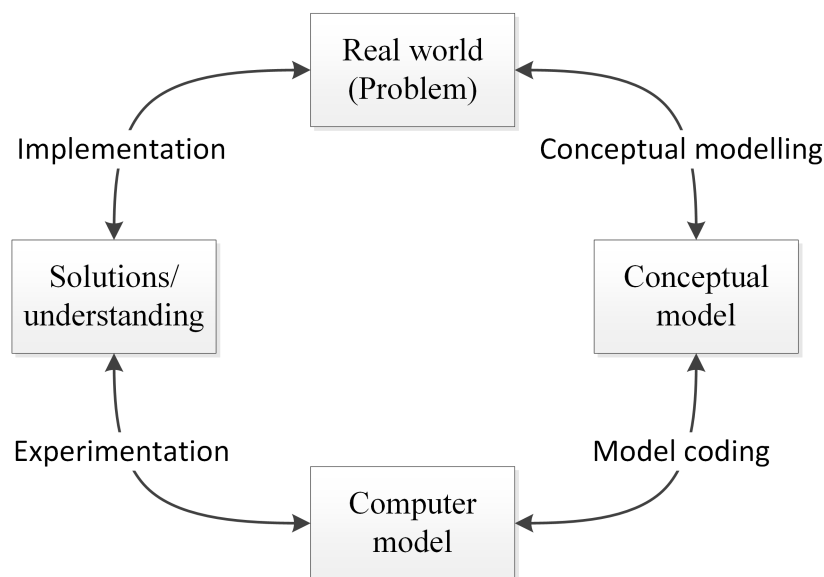


Fig. 3.4 The lifecycle of a simulation study (Robinson, 2004)

Conceptual modelling Robinson (2004) proposed a framework for conceptual modelling of simulation, which includes four key elements:

- Develop an understanding of the problem.
- Identify the objectives of the simulation study.
- Design the conceptual model: inputs and outputs.

- Design the conceptual model: the model content.

For the first element, modellers need to develop an understanding of the problem. For example, in real life, the problem arises when decision makers recognize it and want to find a solution. In the case of PGG, the problem can be social situations (such as management of public goods, saving of energy, etc.) which can be translated to PGGs. Understanding these problems correctly will result in a conceptual model that better captures the real-life situations. If the simulation is developed based on an economic laboratory experiment, then modellers need to understand the context, settings, and findings from the experiment.

In the next step, the objectives of the simulation study are defined. For example, in PGG related problems, the objectives can be investigating a theory or a model of behaviour, and/or perform “what-if” analysis without real-world restrictions (such as number of people, different ratio of demographics). The study can result from observations from laboratory experiments. Researchers can use a simulation to validate the phenomena from laboratory/real-life experiments by implementing a theoretical model of behaviour. Additionally, new research questions or hypothesis may be derived from the results of the simulation.

The third and fourth key elements are about designing the conceptual model that is suitable to tackle the problem. With the modelling objectives in mind, modellers design the conceptual model by defining:

- Inputs (Experimental factors): These are factors that can be changed to achieve the objectives. They can be quantitative (number of people per group) or qualitative (game settings or rules).
- Outputs (Responses): The responses can be used to analyse and understand the simulation system, and determine whether objectives are satisfied. For example, PGG is usually interested in the average contribution of all agents.
- Model contents: scope and level of details
 - *Scope of the model*: identify the model boundary, all the related components, and which components to be included/excluded.
 - For all the included components from the scope, identify *the level of details* of each component.
- Assumptions and simplifications:
 - Assumptions are made when there are uncertainties or beliefs that are accepted without proof.

- Simplification excludes unnecessary details for a simpler model and faster development.

Additionally, modellers need to collect and analyse:

- Existing theories and relevant models: It is common that economists base the simulation study on existing theories, or findings from lab experiments, or both. It can be a behavioural theory used as a base for our simulation study, or a behavioural model that will be implemented in our agents. In addition, it is also important to investigate the relevant agent-based models along with its objectives, hypothesis, findings, and limitations. Reviewing the existing models allows us to improve our study from what has already been established.
- Data: Recently, many ABMS studies have been driven by empirical data ([Hassan et al., 2008](#)); and the typical data source in micro-economics is laboratory experiment. However, data can also be obtained from other sources such as surveys or field experiments. In addition, there are also additional problems with data for ABMS such as longitudinal data, algorithmic data, etc. Since necessary data are not always available and even contradictory data can be found, we need to select the data sources, understand the collection process, and analyse data structure. Raw data are hard to use, modellers have to apply statistical techniques to filter away unnecessary data and extract useful information.

Model coding In this process, the conceptual model is converted to a computer model. A programmer can code the computer model from scratch using a general programming language (e.g. Java, C++). It can also be developed using simulation toolkits (e.g. NetLogo, Repast Symphony, AnyLogic) by researchers with basic knowledge of programming. During model coding, testing needs to be performed to ensure the model (most importantly agents) behaves as expected.

Experimentation In this step, experiments are performed with the computer model to gain better understanding of the real world system and propose solutions to real world problems. The modeller decides the input parameters for the simulation, and designs a set of experiments. The experiments can be different treatments on some experimental units. For PGGs, it can be different settings in game play, different models of agent behaviour, etc. Based on the experimental design, experiments are conducted. The experimentation can be multiple replications (e.g. run multiple times with different random number streams), single long run (e.g. run over a long time to collect a large sample of data), interactive experiment (e.g. modify to the model

after observing the result), or comparing experiments (e.g. comparing different experiment runs).

Data are collected during the experiment, then plotted, and analysed. The modellers show the data using suitable presentation, and interpret the results. After interpretation, the findings from simulated results need to be reported. It can be new insights, validation of observed phenomenon, verification of a theory, etc. In this process, we can update the experimental model to represent the current form of the system; changing it for the purpose of maintenance, to obtain other results, or for other use; or redefining a new system to model for studying an alternative solution to the problem.

Implementation There are three ways to think of implementation of the solutions and/or understanding. First is implementing the findings of a particular solution from the simulation to a specific real world problem. A second interpretation is the runnable implementation of the simulation model itself, that will be handed over to an organization or people with proper training for continuous use. The third interpretation is learning. The simulation helps decision makers to improve understanding of the real world system.

Verification and Validation Aside from the four processes above, there are two important tasks to be performed to ensure the accuracy and validity of the simulation. These two processes are deliberately omitted from Figure 3.4 by Robinson. The reason for this is that they are not single processes, but continuous ones which are performed throughout model development and use.

- Verification is the process of ensuring the conceptual model has been transformed into a computer model with sufficient accuracy (Robinson, 2004). There are many techniques that can be utilized to ensure the model matches specifications and assumptions: examining each possible action from logic flow diagrams, examining model output under a variety of different inputs, examining the model with domain experts, etc.
- Validation is the process of ensuring the model is sufficiently accurate for the purpose at hand (Robinson, 2004). A model can be validated by comparing the input-output transformations with the input-output transformations from the real systems or laboratory experiments. There are also problems that may arise when trying to validate the model: (1) there are no data to compare against; (2) the collected data are inaccurate; (3) there are little data.

In summary, the verification is concerned with the quality of the simulation: whether it is well-designed and error-free. On the other hand, the validation is concerned with checking

that the simulation meet the modeller's needs: how well the (proposed) simulation fulfils its intended use.

Laboratory Experiment Process of Public Goods Game

The laboratory experiment has a tight relationship with the PGG simulation study. Observations from laboratory experiment are inspirational resources for simulation studies. Data from laboratory experiment can be used to parameterize variables in the simulation, and/or validate the simulation's results. If the simulation study is based on a laboratory experiment, the structure of the experiment itself also affects the design of the simulation. Thus, in the framework, the process of a laboratory experiment will be included alongside the simulation lifecycle.

Let's discuss the structure of the PGG in laboratory experiments and how it affects the agent-based models of PGG. During the laboratory experiment, participants play the game in synchronization, which means that all players make their decision in each time step. In short, the process of playing the game can be summarized as a series of turns (or rounds), in which a player can contribute to the public good, receive payoff, and if applicable punish free-riders. Since the laboratory experiment has been performed in discrete time, the agent-based models of PGG in Economics are designed in a similar manner. This can be helpful since it would be easier to parametrize the simulation using previous data from the laboratory experiment. The software supporting lab experiments are also designed to play PGGs synchronously such as Z-Tree ([Fischbacher, 2007](#)) or NetLogo HubNet Public Good model ([Wilensky, 2003](#)). From the discussion above, the typical design of agent-based models of PGG has the following characteristics:

- Scheduling: synchronous.
- Steps in a game: give initial endowment, shuffle player to group, play a turn, reshuffle groups, play the next turn.
- Typical actions of players per turn: contribute, receive payoff, punish (if applicable).
- A payoff function required to calculate payoff in the game.

The process of laboratory experiment was discussed in Section 2.2.6 with Figure 2.3. Combining the laboratory experiment process with the lifecycle of a simulation study of [Robinson](#), an overview lifecycle of the ABOOMS framework v0 is shown in Figure 3.5. Figure 3.5 is split into three sections: real life is on the left, the laboratory experiment process is in the middle, the simulation development is on the right side. In the overview, there are four key stages from [Robinson \(2004\)](#) lifecycle: real world problem, conceptual model, computer model (simulation), and solutions/understanding. The OOAD is shown as a circular arrow on the right side because it is an iterative process, that will be explained in the next Section.

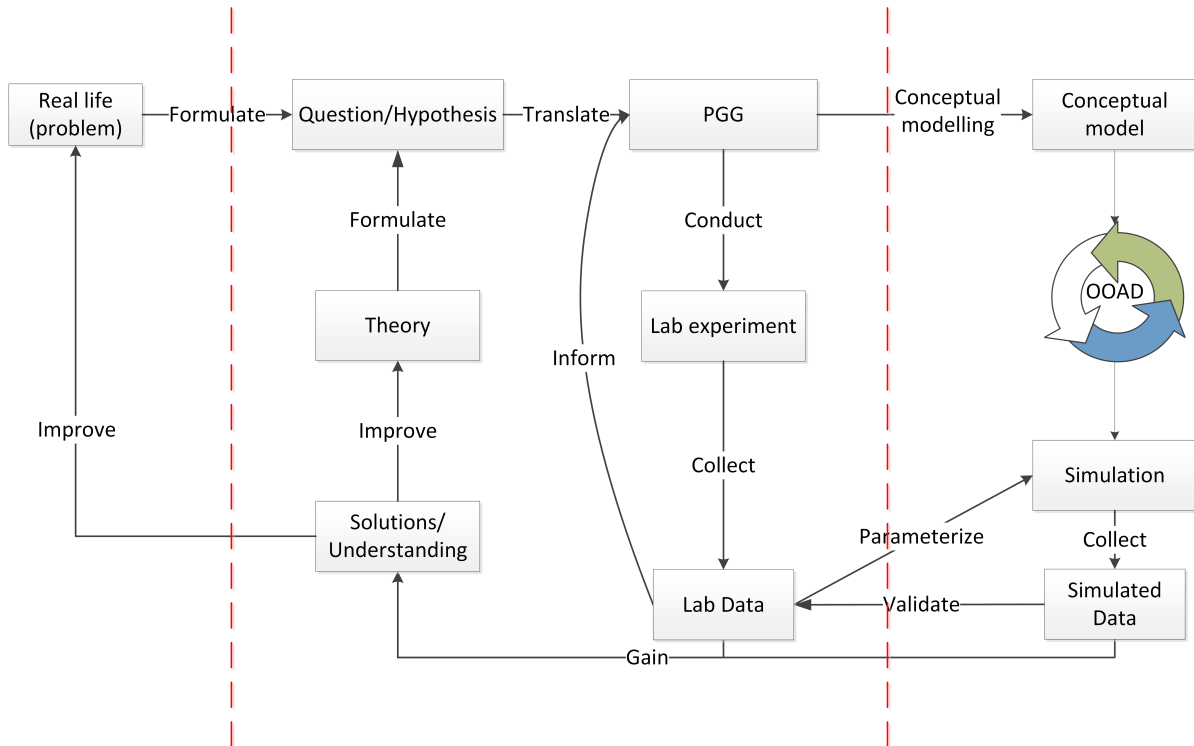


Fig. 3.5 Overview lifecycle of the ABOOMS framework v0

3.5.3 Design of the Micro Process

Object-Oriented Analysis and Design

This section discusses why and how OOAD can be incorporated to the framework. OOAD is employed because of the two key benefits: reusability and extensibility (Graham et al., 2000). Reusability is the use of existing assets in software development such as code, templates, components, frameworks, patterns, etc. Extensibility is the ability to extend the system with minimal effort and risk to the existing system. From its creation in the 1970s, OOAD has matured through the 1990s and become a firmly established methodology in software development (Graham et al., 2000). There are many established development processes in OOAD. Our framework adapts an iterative and incremental development process (Booch, 2007, p. 259) which includes these activities (Figure 3.6):

- Requirements engineering: The goal of this activity is to gather the requirements of the users. All requirements may not be gathered at once, but at least a few requirements are acquired.

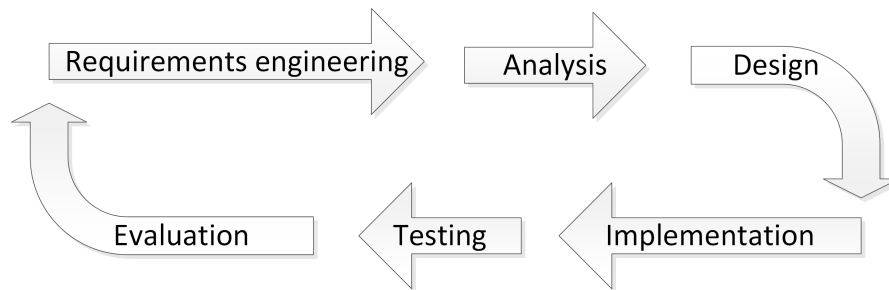


Fig. 3.6 An object-oriented development process

- **Object-oriented analysis (OOA):** The goal of OOA is to understand the system by organizing the requirements around objects. The primary task in OOA is to find the interactions between system and its environment (use cases).
- **Object-oriented design (OOD):** OOD determines the design of the system to solve the problem identified during OOA. OOD can define a software architecture by using design patterns (if applicable). The major tasks in OOD are:
 - Identify objects, along with their attributes and methods.
 - Establish the relationships between objects.
 - Establish the interfaces of each object and exception handling.
- **Implementation:** Develop the code (programming) that fulfils the system design, which resulted from OOD.
- **Testing:** Systematically test and debug to identify defects.
- **Evaluation:** Evaluate the system based on a list of criteria such as satisfying requirements, required features, speed, etc.

The development process is iterative (develop a system through repetitive cycles) and incremental (deliver a small portion or incremental release over time). Each activity in the development process is iterative, and interrelated with the activities before and after it. For example, implementation can reveal weaknesses in the design because the designer was unaware of a feature in the programming environment or the simulation toolkit, and did not take advantage of it.

The Unified Modeling Language

The Unified Modeling Language (UML) is widely accepted for the visual representation in object-oriented software development. It is a standard language for visualizing, specifying,

constructing, and documenting the artifacts of software systems as well as business modelling and other non-software systems. UML is flexible, extensible, and independent of any particular design process. With UML as one standard set of graphical notations, modellers, and developers can express their designs and communicate. Originally, UML was designed for object-oriented systems. Recently, there have been many efforts in applying UML to develop agent-oriented software systems. In Agent-Oriented Software Engineering (AOSE), [Bauer et al. \(2001a,b\)](#); [Bauer and Odell \(2005\)](#); [Odell et al. \(2000\)](#) provide guidelines for adapting UML for MASs, and propose *AGENT UML (AUML)* as an extension of the UML. In the field of ABMS, [Bersini \(2012\)](#) discusses UML in an ABMS context and presents four diagrams that, based on the author's experience, are useful. [Vermeir and Bersini \(2015\)](#) discuss the best practices in programming using ABMS, emphasize the planning phase with support of UML, and give an example making extensive use of simplified class diagrams. [Müller et al. \(2014\)](#) evaluate different types of model descriptions for agent-based models, and mention UML as a graphical representation that can “facilitate various purposes of model communication and understanding”.

In our framework, UML diagrams will be applied throughout for visual modelling of the system. Specifically, our case studies employ five UML diagrams¹. To understand how the UML diagrams can be employed, the next section introduces essential concepts of five types of UML diagrams (use case diagrams, class diagrams, state diagrams, sequence diagrams, activity diagrams) and applies them to an example of an ATM (Automated Teller Machine) system.

Use Case Diagrams Use Case Diagrams are a type of UML diagram that describes the interaction of actors with the system, and is used for OOA. A use case diagram consists of four elements: (1) the system; (2) actors which can be classed into people, organizations, devices, softwares, etc.; (3) use cases which represent actions that are performed by actors; (4) relationships between and among the actors and the use cases. Figure 3.7 is an example of a use case diagram of an ATM system. There are three actors (customer, technician, bank) and five use cases (check balance, deposit, withdraw, maintenance, repair). A use case is shown as an oval and placed inside the rectangle which represents the system. An actor is depicted as a stick figure. A relationship between a use case and an actor is represented by a straight line.

Class Diagrams Class diagrams specify an object's internal data and representation, and defines the operations the object can perform. This diagram is used for the structural design of the system. In class diagrams, a class is represented with a box, which consists of three parts.

¹For a more in depth introduction to UML, the reader is advised to refer to [Fowler \(2004\)](#). Here we only present the aspects relevant for our case studies.

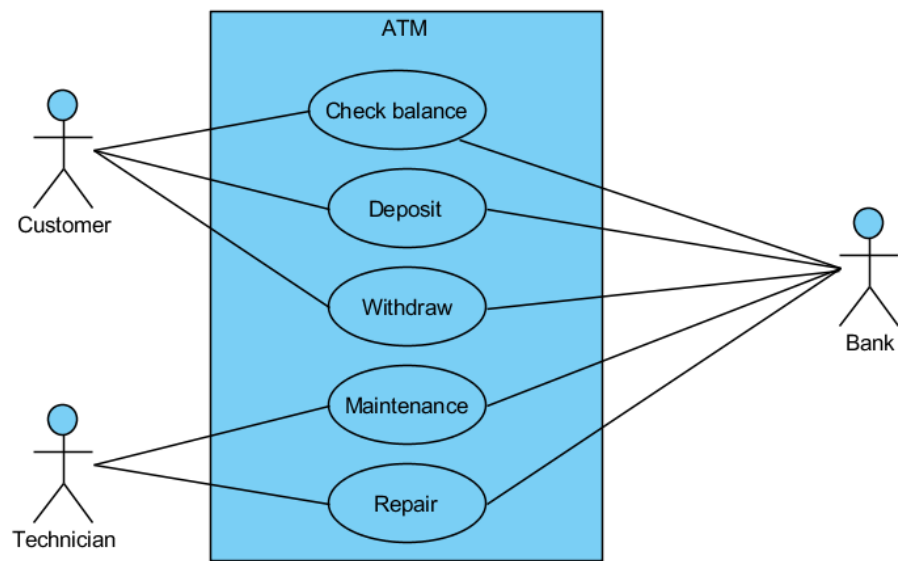


Fig. 3.7 Use case diagram of an ATM system

The first part contains the name of the class; the second part the attributes of the class; and the third part the methods or operations that the class can perform. Figure 3.8 is an example of Customer class of the Bank ATM example. The Customer class has three attributes (id, name, balance) and can perform three operations (checkBalance, deposit, withdraw). Each attribute has the type listed right after its name, separated by a colon (id, name, balance have types of String, String, double).

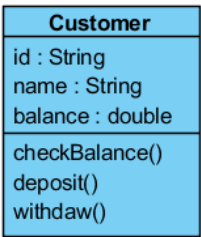


Fig. 3.8 Customer class diagram (which is part of an ATM system)

Usually in a system, there are many classes and they have relationships with each other. Figure 3.9 shows four essential types of class relationships: association, generalization, aggregation, and composition.

- *Association* declares that there is a link between two classes. It is indicated by a solid line. In Figure 3.9, Customer and Bank Account have an (bi-directional) association

relationship, in which the Customer is aware of which Bank Account he has and the Bank Account has information on which Customer it belongs to.

- *Generalization* expresses inheritance relationship between a more general class (superclass) and a more specific class (subclass). The subclass inherits structure and behaviour of its superclass, but can still have its own attributes and behaviour. It is indicated by a solid line with a closed arrowhead. The arrowhead points to the superclass and the opposite end connects to the subclass. For example, Figure 3.9 shows Bank Account is the superclass of Current Account and Saving Account subclasses.
- *Aggregation* and *Composition* express “whole to its part” relationship, which means one class is a part of another class. Aggregation is presented by a line with a hollow diamond shape; while composition is indicated by a line with a filled diamond shape. The diamond shape points to parent class; on the other end is the child class. The difference between aggregation and composition is that the child class can exist without the parent class in an aggregation relationship. In composition, if the parent class is destroyed, the child class is removed as well. For example, Figure 3.9 shows an aggregation relationship between Branch and Bank Account and a composition relationship between Bank and Branch. So a Branch has many Bank Accounts, and the Bank Accounts can still exist after the Branch is closed. On the other hand, the Bank consists of many Branches, but Branches must be deleted if the Bank does not exist.

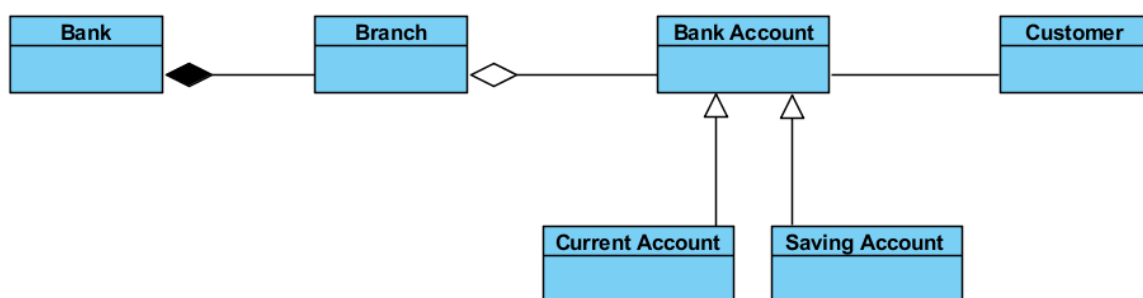


Fig. 3.9 Class diagram of an ATM system

State Machine Diagrams State machine diagrams describe different states of a system and the transitions between them, and can be used to visualize and model the reaction of objects by internal or external factors. This diagram is used for behavioural design of objects within the system. It has four elements:

- An initial state: a filled circle.
- A final state: a filled circle inside a slightly larger unfilled circle.
- States: rounded rectangles.
- Transitions: directed arrows between states.

Figure 3.10 shows the state machine of an example Customer class, which has two main states: “Idle”, and “Using ATM”. In the “Using ATM” state, there are four sub-states: “Authentication”, “Checking balance”, “Deposit”, “Withdraw”. At the top left is the initial state, which initializes this diagram to be “Idle”. When a user starts to use the ATM, transition from “Idle” to “Using ATM” will be triggered. When in “Using ATM”, the initial sub-state is “Authentication”. If the user fails to authenticate, a self-transition will be triggered so that the user can re-enter a password. If the authentication succeeded, then depending on the action the customer is going to take, the state can be either “Checking balance”, “Deposit”, or “Withdraw”. When the user finishes using the ATM, a transition will be triggered and the system goes to “Idle” state. Lastly, there is a transition from “Idle” state to the final state on the top right, which represents when the ATM is shut down.

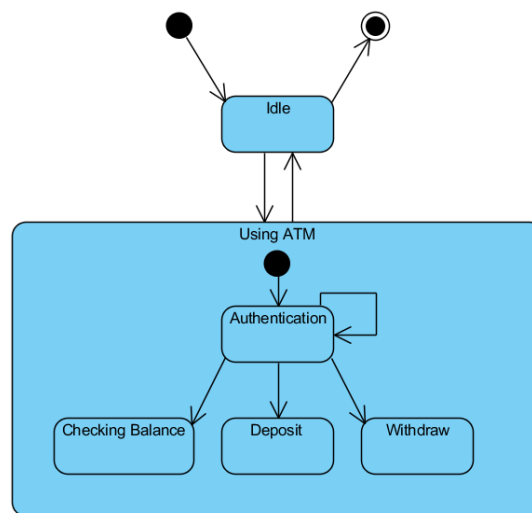


Fig. 3.10 A state machine diagram of an ATM system

Sequence Diagrams Sequence diagrams show the message interchanges between entities that are represented as lifelines in a sequential order. The diagram is used for behavioural OOD of parts of the system or the system as a whole (depending on the complexity of the system). The *entities* are written on the top of the diagrams. Below each of them is a dashed vertical

line, called a *lifeline*. *Messages* are drawn as a horizontal arrow from a sender to a receiver. Figure 3.11 shows a sequence diagram when a customer uses an ATM to check the balance of his bank account. There are three entities: customer, bank system, and database. After inserting the card into the ATM, the customer enters the PIN. The bank system will check the PIN against the database. If the PIN is correct, the customer will be granted access, then can request his balance. The bank system will get the balance from the database, then inform the customer. If the PIN is wrong, the customer cannot gain access, and has to enter the PIN again. To represent the loop (keep entering PIN) and if-then-else logic (correct or wrong PIN), two kinds of *combined fragments* (*loop* and *alt*) are used. Each is represented as a box. A loop combined fragment executes multiple times. An alt (alternatives) combined fragment shows conditional flow by modelling if-then-else logic. In Figure 3.11, the alt combined fragment is split into two sections by a dashed line: the first section shows the interaction when wrong PIN is enter, while the second section shows the messages when the PIN is correct and the user requests the balance.

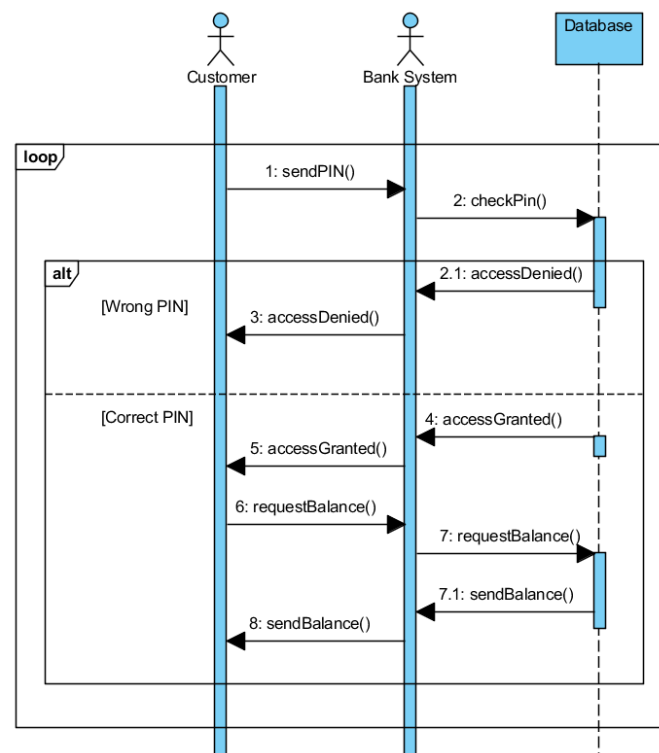


Fig. 3.11 A sequence diagram of an ATM system

Activity Diagrams Activity diagrams describe flow of control, focus on the activities to be performed and the parties responsible for the performance of those activities. This type of

diagram is used for behavioural OOD of parts of the system or the system as a whole (depending on the complexity of the system). Basic elements of activities diagrams include:

- Actions: rounded rectangles.
- A initial node: a filled circle.
- A final node: a filled circle inside a slightly larger unfilled circle.
- Decision nodes: diamond shapes with incoming and outgoing arrows.
- Partitions (Swim lanes): a frame with a name that indicates the responsible parties for a group of elements.

Figure 3.12 shows an activity diagram of an ATM system. The diagram has two partitions: ATM and Bank System. In the ATM partition, there are an initial node on the top and a final node at the bottom. After the customer inserts a card into the ATM, the first action is “Check the card”. Afterwards, the ATM goes to a decision node, where it checks the condition of the inserted card. If the card is damaged or wrong, it is ejected. If the card is normal, the ATM goes to “Enter PIN number” to receive the PIN from the customer. Then “Check PIN number” will be carried out by the Bank System. If the PIN is wrong, the card is ejected. Otherwise, the ATM will “Display Options” for customer to choose. After that, the Bank System will “Perform a transaction” that the customer chose, such as checking balance or cash withdrawal. When the transaction is completed, the card is ejected and the ATM goes to the final node.

Use Case Analysis for Agent-based Modelling and Simulation

Even though OOAD offers many benefits (established development lifecycle, UML as an industry standard modelling language), it is not directly suitable for agent-based systems because “There is a fundamental mismatch between the concepts used by object-oriented developers (and indeed, by other mainstream software engineering paradigms) and the agent-oriented view” (Wooldridge et al., 1999). In our case, the *use case analysis* technique, which identifies the requirements of the system, need to be adapted from object-oriented application to agent-based application. In object-oriented applications, a use case defines a typical interaction between actors and a software system. Actors in these applications are external entities (human or other systems). In an agent-based applications, agents are internal entities in the system and have capabilities similar to human actors (autonomy, social ability). Heinze et al. (2000) proposed two extensions to the UML Use Case Diagram for agent applications: an *agent* is an actor that is confined inside the system boundary, has behaviour defined by the agent use case,

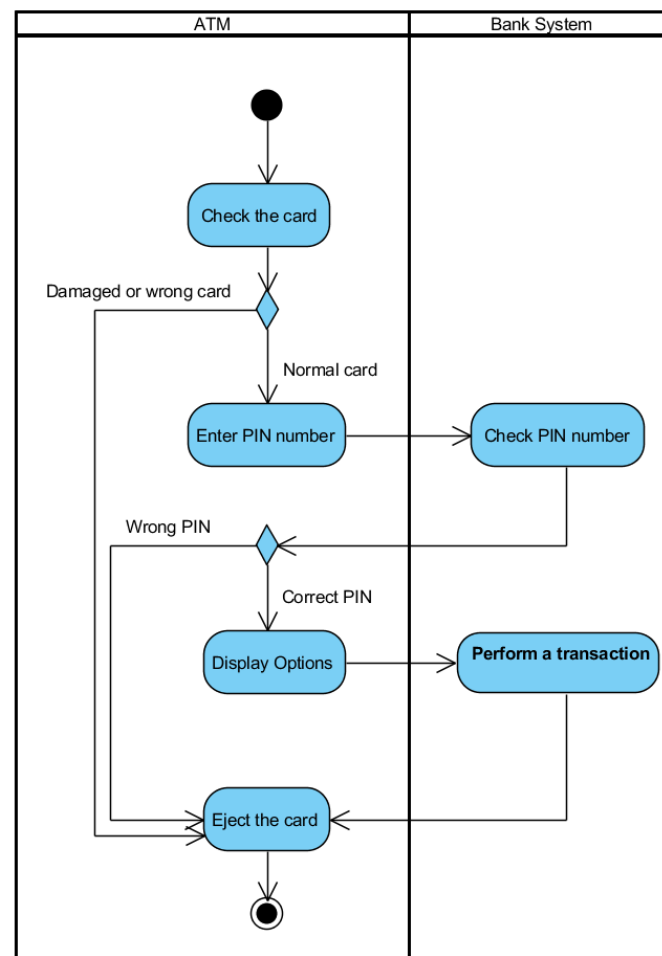


Fig. 3.12 An activity diagram of an ATM system

and can interact with other agents and actors; an *agent use case* is a collection of scenarios about the agent's interaction with the system.

Heinze et al. (2000) also suggests a three step process for specifying agent systems:

1. Identify agents and actors:
 - (a) Actors: External entities such as human and other systems
 - (b) Agents: Internal entities that have properties of agents
2. Identify use cases for agents and actors.
 - (a) A use case can be a behaviour to be exhibited by agents.
 - (b) A use case can also be an interaction between agents and external actors, or between agents.

- (c) Refine the use case model by adding lower level functionalities and behaviours.
3. Document each use case with the following structure:
- (a) Use case name: The name of the use case
 - (b) Description: Describe what the use case involves, agents' behaviour in context of environment, and interactions with other actors.
 - (c) Actors: A list of actors involved in the use case.
 - (d) Use case associations: A list of other use cases associated with this use case (via uses or extends).
 - (e) Pre-conditions: A list of conditions that must be true before initialization of this use case.
 - (f) Post-conditions: A list of conditions that must be true after this use case is completed.
 - (g) Flow of events: A list of activities or events that defines agents behaviour in this use case. This section can include a UML activity diagram to describe the flow of events.
 - (h) Alternative flows: A list of activities or events which agents can perform in exceptional circumstances, which are not handled by the main flow of events.

3.5.4 The ABOOMS Framework v1

The ABOOMS framework v1 consists of two sections: the macro process which guides the overall development lifecycle and the micro process in which the OOAD is applied.

The Macro Process

The development lifecycle of the ABOOMS framework v1 is an integration of three processes: the lifecycle of a simulation study, the process of laboratory experiment, and the development process of OOAD. Figure 3.13 shows the overview of the modelling development lifecycle of the framework, and highlights the interaction between ABMS and laboratory experiments in Economics. The boxes depict the key development stages, while the arrows represent the processes to get from one stage to another. In the figure, the OOAD is applied in the area marked by a dashed rounded rectangle. The processes (represented by arrows) with bold labels will be discussed in details in “the micro process” section.

The lifecycle can start with a real life problem or a theory. From the problem or theory, economists formulate the questions or hypothesis that can be related to an existing theory.

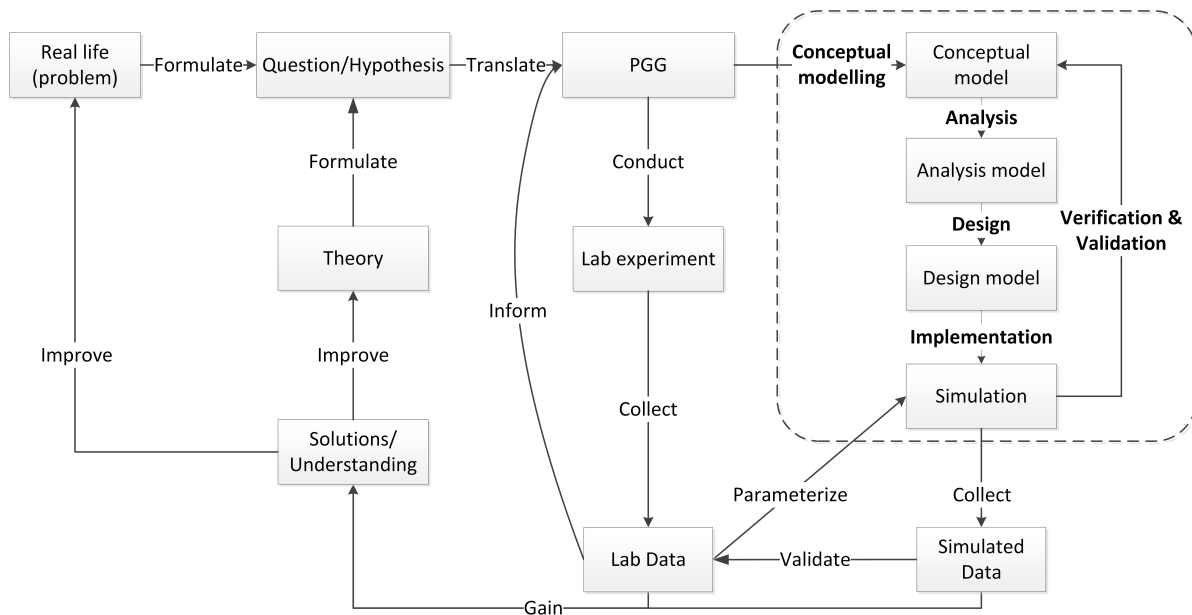


Fig. 3.13 Overview lifecycle of the ABOOMS framework v1

Based on real life and/or theories, economists can formulate the problem as a PGG, and conduct a laboratory experiment of that PGG to collect data. Before creating a simulation, the PGG is investigated to create a conceptual model. The next step is to perform analysis to create an analysis model of the model to be implemented. Based on the analysis model, the simulation is designed and implemented. The simulation can be parameterized using laboratory data. Afterwards the simulation is tested and evaluated. After evaluation, the process is started over from analysis if needed. This shows the iterative nature of the development process. After running and experimenting with the simulation, the simulated data can be validated against the laboratory data. After the interpretation of data, economists gain understanding of the real life problems or theory, and can find answers to the initial questions or evidence to support the hypothesis.

The Micro Process

The micro process focuses on the five important processes, which are represented by arrows with italic labels in Figure 3.13, to arrive from a *real life problem* at a *simulation*. The steps of the five processes are described in details below. In each process, there are multiple tasks.

Process 1 - Conceptual modelling

- Understand the problem situation.

- Determine objectives of the simulation study.
- Collect and analyse data and related theories.
- Design the conceptual model.
 - Identify the model inputs (experimental factors).
 - Identify the model outputs (responses).
 - Determine model content: scope and level of details.
 - Identify assumptions and simplifications.

Process 2 - Analysis

- Identify agents and actors.
 - Agents: Internal entities that have properties of agents.
 - Actors: External entities such as human and other systems.
- Identify use cases.
- Document use cases.
- Draw Use Case Diagram.

Process 3 - Design

- Define the structure (internal data, operation) with Class Diagram.
- Define behaviour: Define behaviour with behavioural diagrams, such as Statechart or Activity Diagram.

Process 4 - Implementation

- Choose a programming language and/or a software: The simulation can be programmed from scratch or with a specialist simulation toolkit (NetLogo, AnyLogic, Repast Simphony).
- Programming: Writing the code.
- Testing and debug: When coding, it is important to perform testing in order to ensure the correctness of the model.

Process 5 - Verification and Validation

- Verify whether the simulation correctly implemented the conceptual model.
- Validate and tune agent behaviour based on the information from the lab or literature.

Since the development process of OOAD is *iterative*, all phases are *interrelated*. This means that during development, if an error is spotted or there are changes in requirements or design, developers should go back to the appropriate phase and go through all the steps. For example, during the Implementation process, if there is a need for an object that is not specified in the initial design, the developers have to go back to the Design process to update the design document then continue on to Implementation.

3.6 Summary

This chapter described the methodological choice of using a case study strategy to achieve research objectives. To evaluate the ABOOMS framework, we considered three different aspects of modelling including modelling principles, application of a psychological model, and continuous-time setting. Three case studies were also selected and explained. They will be described in details in the subsequent chapters (Chapter 4, 5, 6). In this chapter, we also have presented an adapted lifecycle of a simulation study that can be used for investigating PGG problems. The initial ABOOMS framework v1 was created by combining the Robinson's lifecycle of a simulation study with a laboratory experiment process, then incorporate the development process of OOAD. The framework consists of an overview lifecycle and a detailed step-by-step description of five important processes to develop a simulation. In the next chapter, the ABOOMS framework v1 will be evaluated through a case study.

Chapter 4

Case Study 1: Keep-It-Simple-Stupid vs. Keep-It-Descriptive-Stupid

In this chapter, we examine the broader contextual level of the ABOOMS framework by applying it to the same context but using two different modelling principles: Keep-It-Simple-Stupid (KISS) and Keep-It-Descriptive-Stupid (KIDS). In order to compare and contrast the two principles, a problem of washing dishes in a shared house, which is described in Section 4.1, is addressed by both approaches. The development of agent-based models is described in Section 4.3 for KISS and Section 4.4 for KIDS. Afterwards, a simple experiment was performed for validation and simulation results were discussed (Section 4.5). Additionally, we discuss the flexibility of the framework as well as other properties including reusability and extensibility in Section 4.6. Lastly, the ABOOMS framework is revised and improved to a new version, namely v2, in Section 4.7. Section 4.8 summarizes this chapter.

4.1 Case Study of Washing Dishes in a Shared House

We model a real-life multi-player dilemma: washing dishes in a shared house. There are four people in a shared house. An individual can choose to wash dishes or not. One can contribute time to wash dishes. If one chooses not to wash dishes, one gains by saving time. However, if everyone does so, there will be no clean dishes for anyone.

The two modelling approaches (KISS and KIDS) are treated as two different simulation studies: Section 4.3 for the KISS approach and Section 4.4 for the KIDS approach. For the KISS simulation study, we developed a model as simple as possible with a high level of abstraction. On the other hand, in the KIDS approach, we focus on realism resulting in a more complex model with a lower level of abstraction. The framework v1 is applied to both

approaches. In the subsequent sections, the first four processes of the framework (conceptual modelling, analysis, design, implementation) are presented in detail; while the fifth process (testing and evaluation) is carried out but not written as a section. However, since the same problem was utilized, in the conceptual modelling process of the framework, there were three tasks that are the same for both studies and presented in Section 4.2. The last task of the conceptual modelling, i.e. designing the conceptual model, is presented separately for each approach.

4.2 Conceptual Modelling - Three Common tasks

4.2.1 Understand the problem situation

We can understand the problem from the problem description in Section 4.1. After understanding the description, the dish washing scenario is conceptualized as a PGG. Four people in the house are the players. The cleanness of the kitchen is the public good.

4.2.2 Collect and analyse data and related theories

Our simulation design is based on a PGG laboratory experiment from the literature. We decided to utilize settings from a PGG experiment by [Fehr and Gächter \(2002\)](#). In the experiments, there were 240 participants, divided into groups of four members. They played PGG with two different treatments: no punishment and punishment. In each game, every participant received 20 money units (MUs), could use them for contribution to a group project, and keep the rest of the money. For every invested MU, every member in the group, earned 0.4 MUs. Participants did not know the identities of other group members. They made a decision simultaneously then were informed of the investments of the others. In the games where punishment was allowed, participants could invest from 0 to 10 points to punish a member. Each point costs the punished members 3 MUs and the punishing member 1 MU. This type of punishment is called *altruistic punishment* because it is costly to punishing members and yields no material gain. The finding of this experiment is that cooperation thrives if altruistic punishment is allowed and breaks down if it is excluded. As an attempt to understand the finding of [Fehr and Gächter](#)'s experiment, [Pahl-Wostl and Ebenhoh \(2004\)](#) implemented an agent-based model in which agents have three strategies and can change their strategies over time based on cues. The work also utilized a simple learning function for updating expected cooperativeness and a punishment heuristic. These can be employed in our simulation and will be discussed in detail during the development processes.

4.2.3 Determine objectives of the simulation study

The objective of our simulation study is to predict the cooperation level in the shared house in two settings: no punishment and altruistic punishment. Based on [Fehr and Gächter](#)'s experiment, our hypothesis is that we will replicate the phenomena they observed: the decline of cooperation level when there is no punishment, and the rise of cooperation when altruistic punishment is allowed.

4.3 Modelling using KISS Approach

4.3.1 Conceptual Modelling - Design the conceptual model

With the KISS approach, the model has a high level of abstraction since most of the details are discarded to focus on the essence of decision making in the PGG. We followed the structure of PGG in [Fehr and Gächter](#)'s laboratory experiment. There were four agents representing four people in a shared house. The game was repeated with many rounds. This model used *synchronous* scheduling, in which agents perform actions simultaneously in a round. The money units in this case were minutes. At every round, all agents had 20 minutes of free time and made a decision how many minutes were to be spent on washing dishes. The invested minutes of all agents were increased by 60% and divided evenly among them as profit ($profit = 1.60 \div 4 \times totalContribution = 0.4 \times totalContribution$). It can be said that invested time doing the cleaning resulted in more free time (as profit) for the agents. When punishment is allowed, an agent can decide to spend a maximum 10 minutes for punishment. For every minute invested to punish, the punished agent has to lose 3 minutes of free time.

Based on the work of [Pahl-Wostl and Ebenhoh \(2004\)](#), there are three different strategies available for agents:

- Cooperative strategy: the agent always cooperates. In this model, the cooperative agent will cooperate by spending all of its time washing dishes, or 75% of its time if the contribution of others drops below a certain level.
- Reciprocal strategy: the agent changes their contribution according to recent experiences. In the case of the reciprocal strategy, the agent will adjust the contribution to be the same as the mean expected contribution of the previous round.
- Maximizing strategy: the agent always tries to defect by maximizing its own profit. The agent will calculate expected outcome from a list of possible decisions. Then the decision with the highest expected outcome is chosen.

In summary, the settings of the PGG are:

- Number of participants: 4.
- Group structure: one group of four people.
- One-shot or repeated games: repeated game with 15 rounds.
- Treatments: no punishment, altruistic punishment.
- Players' action: contribute, update belief, punish (if applicable).
- Game loop: Give initial endowment; Play a turn by letting agents choose cooperation level; Update belief; Punish (if applicable).
- Scheduling: discrete time and synchronous decision making.
- Initial endowment: 20 minutes.
- Payoff function: $outcome = freeTime - timeSpentWashing + 0.4 \times totalGroupWashingTime$
- Information: players were informed of the average cooperativeness of the group.

The framework required the following details for the conceptual model:

- Inputs (Experimental factors):
 - Punishment treatments: no punishment, altruistic punishment
- Outputs (Responses):
 - Average cooperativeness of the group over time
 - Cooperativeness of each agent over time
- Model Content:
 - Scope in Table 4.1
 - Level of details in Table 4.2
- Assumption:
 - People make decisions simultaneously.
 - People have complete and common knowledge of the game structure and payoffs.
- Simplification:
 - People cannot communicate in any way.
 - People use the same strategy in all rounds of the game.
 - People consider temporal payoff for decision making (the time to washing dishes instead of number of dishes).
 - Do not model day-to-day activities and schedule (such as eat, bathe, sleep).

Table 4.1 Model Scope of Case Study 1 - KISS approach

Component	Include/Exclude	Justification
Person	Include	Required for decision making.
House	Exclude	Do not model physical space.
Dish	Exclude	Replaced by temporal payoff.

Table 4.2 Model Level of Detail of Case Study 1 - KISS approach

Component	Detail	Include/Exclude	Comment
Person	Contribution	Include	The result of decision making.
	Others' contribution	Include	Information required as inputs of decision making.
	Strategy	Include	Required for decision making.
	Schedule	Exclude	Not explicitly modelled, but could be represented by rounds of PGG.
	Time takes to wash a dish	Exclude	Not required since schedule is not included.

4.3.2 Analysis

Identify agents and actors

During the simulation run, there is no user interacting with the simulation. Therefore, there is no actor. There is one type of Person agent that makes investment decisions during the game.

Identify use cases

Each round, a Person agent can perform three actions which correspond to three use cases: Contribute, Update Belief, and Punish.

Document use cases

1. Use case name: Contribute
 - (a) Description: Person makes decision on how many MUs to contribute.
 - (b) Agents: Person.
 - (c) Pre-conditions: A round is started.
 - (d) Post-conditions: Contribution is decided.

- (e) Flow of events: Person makes contribution based on the strategy.
- 2. Use case name: Update Belief
 - (a) Description: Person updates their belief about contribution based on others contributions.
 - (b) Agents: Person.
 - (c) Pre-conditions: Person is informed about the average cooperativeness of the group (not the individual contributions).
 - (d) Post-conditions: Belief is updated.
 - (e) Flow of events: Person updates belief.
- 3. Use case name: Punish
 - (a) Description: Person punishes the free riders.
 - (b) Agents: Person.
 - (c) Pre-conditions: Punishment is allowed.
 - (d) Post-conditions: Punishment decision is carried out.
 - (e) Flow of events: Person compares contribution of other agents in the group with the average cooperativeness. If it is lower, that agent is classified as a free rider and will be punished.

Draw use case diagram

Figure 4.1 shows the Use Case Diagram for this case study. It is noted that this use case diagram does not include the system represented by a box because Person agent is an internal entity. In Figure 3.7 of the ATM example, the ATM system box is drawn because the three actors are external entities. However, if this is an interactive simulation, i.e. the experimenter can interact with the simulation during the runtime, then there can be an actor named Experimenter. For instance, in a simulation study where Experimenter can affect Person's Contribute action (such as force a player to change the strategy from cooperation to defection), the use case diagram (Figure 4.2) has a box to represent the system and the Experimenter is placed outside and connected to the relevant use cases.

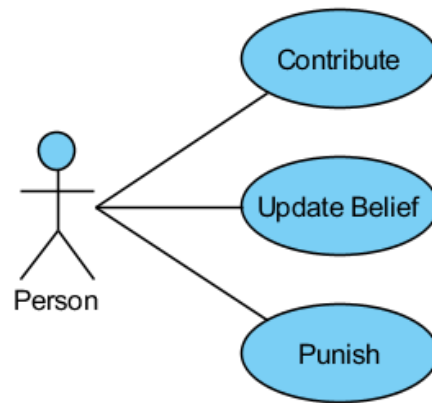


Fig. 4.1 Use case diagram - KISS approach

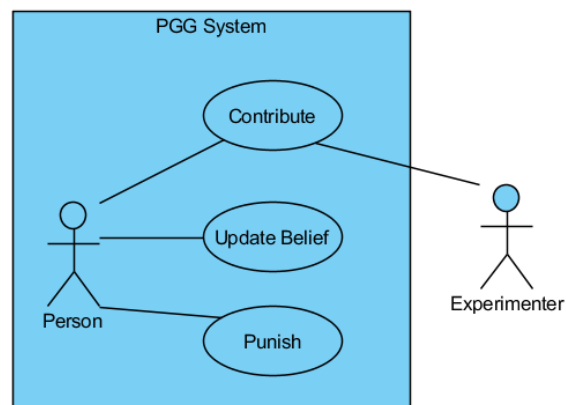


Fig. 4.2 Use case diagram - KISS approach with Experimenter

4.3.3 Design

Define structure with Class Diagrams

Figure 4.3 shows the class diagram for this study. There is a Person class that represents Person actors. Since the Person has three strategies to choose from, we need a *strategy* variable to present the strategy this Person is using. And since there are only three strategies, an enumerated type *Strategy* is defined and used to set up the strategy variable. An enumerated type, also called enum type or enumeration, is a special data type that enables a variable to be set to a value from a predefined constant. With enumeration, we can give names to the strategies which make the code easier to read. Additionally, a class Main is needed to initialize agent population, control the steps in the game, and collect the global value of individual agents' contributions to the public good. The relationship between two classes Main and Person is a uni-directional association, i.e. the two classes are related but only Main knows the relationship exists.

Because Main manages and controls all Person agents, Main class needs a *people* variable to store the list of Person agents and has operations such as *initialize()* to initialize the agent list and *play()* to control steps in the game. It is noted that these two operations of Main are not mentioned in the use case even though they involve the Person agents. This is because they are related to simulation procedures, not agent behaviours. Main also has a *meanDecision* variable to store the mean contribution, and an *allowPunishment* boolean variable that represents the experimental treatments by being set the value of true if punishment is allowed. In Main, there is a variable *people* with the type List<Person> that represents a list of Person agents. After creating and initializing a Person agent, Main will add that Person into this list.

Based on each use case, the attributes and variables of Person class is defined. Person class has four operations: *makeDecision()* for the Contribute use case, *updateBelief()* for the Update Belief use case, *punish()* and *receivePunish()* for the Punish use case. For each operation, there can be required variables listed inside the parentheses. For example, the *updateBelief()* operation requires the value of *othersMeanDecision* to be carried out.

In the Contribute use case, the cooperativeness of a Person agent is modelled as a variable named *decision*, ranged from 0 to 1. The time spent washing dishes is equal to *decision* $\times$ *maxWashingTimes*. When the *decision* variable is 1, the agent spends all 20 minutes washing dishes. In the Update Belief use case, to express the expected contributions from other agents, each Person agent has a variable *expectedCooperativeness* (ranged from 0 to 1) which is the average level of cooperativeness the Person agent expects the other three agents to adopt. There are variables that are related to the operations which will be discussed in more detail in the next section: *learningRate* that is a constant used in *updateBelief()* operation, and four variables

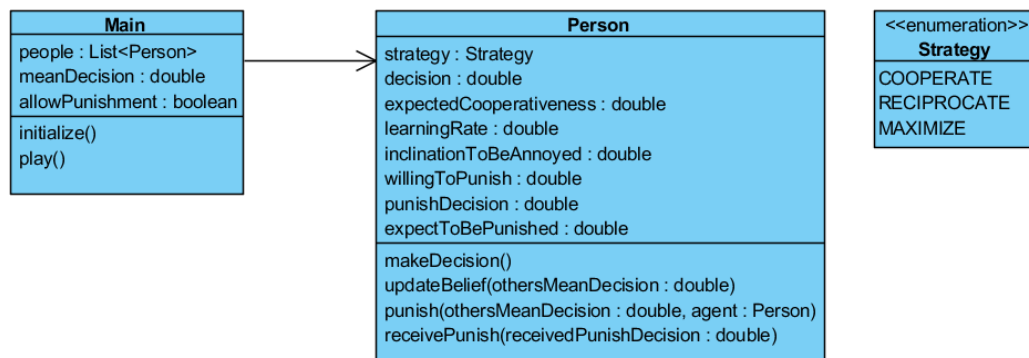


Fig. 4.3 Class diagram - KISS approach

inclinationToBeAnnoyed, *willingToPunish*, *punishDecision*, *expectToBePunished* required for the *punish()* operation.

Define behaviour with Activity Diagrams and Sequence Diagrams

Firstly, we need to define the basic flow of actions between Main and Person. Figure 4.4 shows the activity diagram. Main creates all Person agents and initializes their strategies. In a round, Main commands all agents to make a decision depending on the stages of the game. After all agents have made decisions, Main commands them to update belief. If punishment is allowed, Main commands agents to punish others in their group. If there is still a round to play, Main starts over again. If all ten rounds have been played, the simulation stops. It is noticed that even though Main commands agents to perform an action, the decisions are made by the agents. Figure 4.5 is the sequence diagram that describes three different kinds of messages sent from Main to Person: decide, update belief, and punish. In addition, Person can send a receivePunish message to the PunishedPerson (the person that is being punished).

For the Contribute use case, the contribution decision is made by *decide()* operation based on the strategy of each agent. Based on the conceptual model described in Section 4.3.1, three different strategies are defined as follow:

- Cooperative strategy: the agent always cooperates. If the *expectedCooperativeness* > 0.4, *decision* = 1. Otherwise, *decision* = 0.75.
- Reciprocal strategy: the agent changes their contribution according to recent experiences. In the case of reciprocal strategy, the agent will adjust the contribution to be the same as the mean expected contribution of the previous round (*decision* = *expectedCooperativeness*).

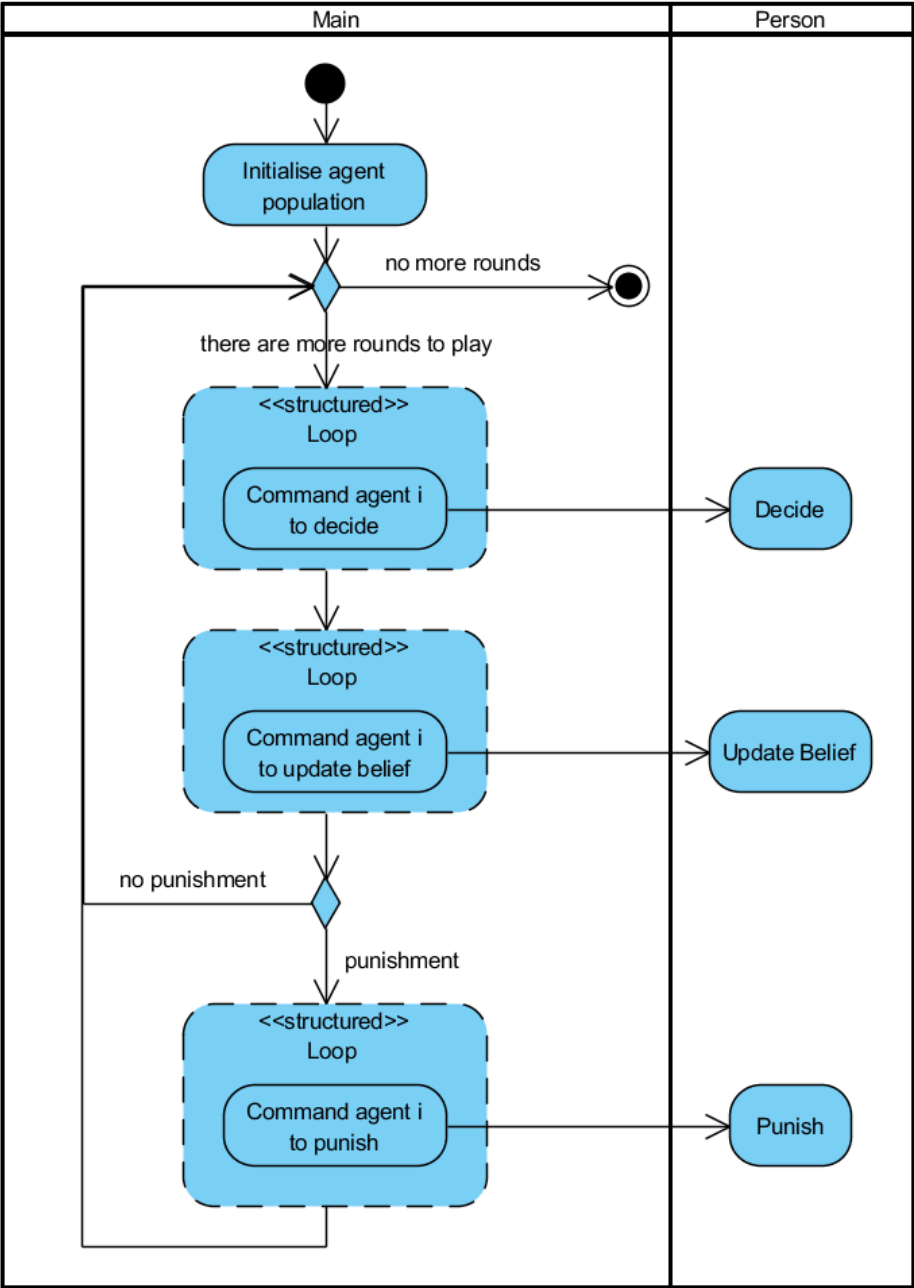


Fig. 4.4 Activity diagram - KISS approach

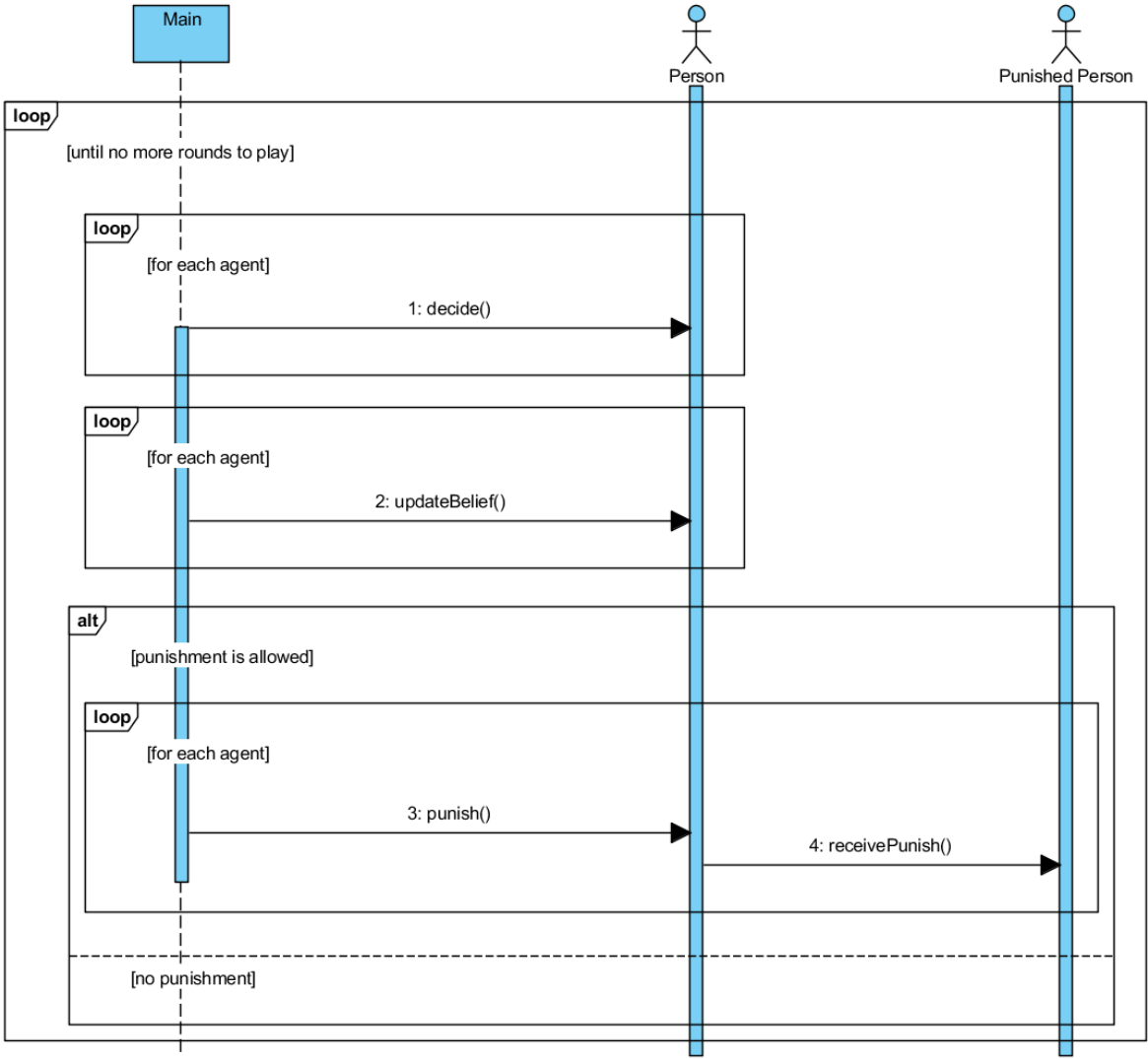


Fig. 4.5 Sequence diagram - KISS approach

- Maximizing strategy: the agent always tries to defect by maximizing its own profit. The *decision* is tried out from 0 to 1 with a predefined step of 0.1 (*temporaryDecision* = 0, 0.1, 0.2, ..., 1). With every temporary decision, the expected outcome is calculated using Equation 4.2. Then the decision with highest expected outcome is chosen.

In the Update Belief use case: after every round, the expected cooperativeness is recalculated using Equation 4.1 to improve the agent's general belief about others.

$$\text{expectedCooperativeness} = (1 - \text{learningRate}) \times \text{expectedCooperativeness} + \text{learningRate} \times \text{meanDecisionOthers} \quad (4.1)$$

where:

expectedCooperativeness is the level of expected contribution from other agents.

learningRate is the rate that the *expectedCooperativeness* will change.

meanDecisionOthers is the mean value of *decision* of other agents in previous round.

An agent also needs to have a utility function to calculate the outcome after each round: $\text{outcome} = \text{freeTime} - \text{timeSpentWashing} + 0.4 \times \text{totalGroupWashingTime}$. Since the *decision* (ranged from 0 to 1) represents the percentage of time used from the maximum time available for washing, $\text{timeSpentWashing} = \text{decision} \times \text{maxWashingTime}$. Similarly, the total group washing time can be calculated by multiply *maxWashingTime* with the sum of decision of the whole group which is $\text{decision} + \text{expectedCooperativeness} \times 3$; thus $\text{totalGroupWashingTime} = (\text{decision} + \text{expectedCooperativeness} \times 3) \times \text{maxWashingTime}$. After substitution, the *outcome* can be calculated using Equation 4.2.

$$\text{outcome} = \text{freeTime} - \text{decision} \times \text{maxWashingTime} + (\text{decision} + \text{expectedCooperativeness} \times 3) \times \text{maxWashingTime} \times 0.4 \quad (4.2)$$

where:

freeTime is the free time (minutes) of an agent.

decision is the decision value of an agent.

maxWashingTime is a predetermined value of maximum washing time of an agent.

expectedCooperativeness is the level of cooperativeness that an agent expects from others.

An agent can decide to spend a maximum 10 minutes for punishment. For every minute invested to punish, the punished player has to lose 3 minutes of free time. To punish after the belief update, each Person agent will examine the decision of each of the other three agents and decide the punishment. Additionally, since punishment is allowed, the player has to take punishment into account when making a decision. So the outcome equation needs to be updated. After it is calculated with Equation 4.2, the outcome value will deduce the expected punishment from other agents: $outcome = outcome - (expectToBePunished \times 3) \times 3 \times maxPunishTime = outcome - expectToBePunished \times maxPunishTime \times 9$.

4.3.4 Implementation

Figure 4.6 shows the AnyLogic IDE with the project SharedHouseKISS opened. This is the implementation of the KISS approach. By default, a project in AnyLogic is in continuous time. To enable discrete time, the developer needs to check the checkbox “Enable steps” in the properties view of Main (red circle in Figure 4.6). Below the checkbox, there is a code section that defines the actions within each time step. Behaviours in each time step follow the activity diagram (Figure 4.4) in the Design process.

Main creates Person agents and adds them to a list. When a Person agent is created, three attributes will be randomized following the pseudocode in Pseudocode 4.1. The random number is following the standard uniform distribution, which is the continuous uniform distribution on the interval [0,1].

Pseudocode 4.1: Initialization of Person agent

inclinationToBeAnnoyed $\leftarrow$ a random value from the standard uniform distribution;
willingToPunish $\leftarrow$ a random value from the standard uniform distribution;
expectToBePunished $\leftarrow$ *willingToPunish*;

For the Contribute use case, the agent decides the value of the *decision* variable based on their strategies. The pseudocode for contribution decision is in Pseudocode 4.2.

For the Punishment use case, the agents follow a punishment heuristic that is given in Pseudocode 4.3 (p is the punisher; o is the agent in question for punishment). When punishment is allowed, the outcome is updated to include the expected punishment from others (Pseudocode 4.4).

In AnyLogic IDE, the simulation is run by going to the menu Model then choose Run. Figure 4.7 shows the resulting simulation of the KISS approach. Figure 4.7a shows the Main screen during a simulation run. Besides the variables in the Design, two files and a timeplot have been added to collect statistics. The timeplot is updated during the simulation run. During

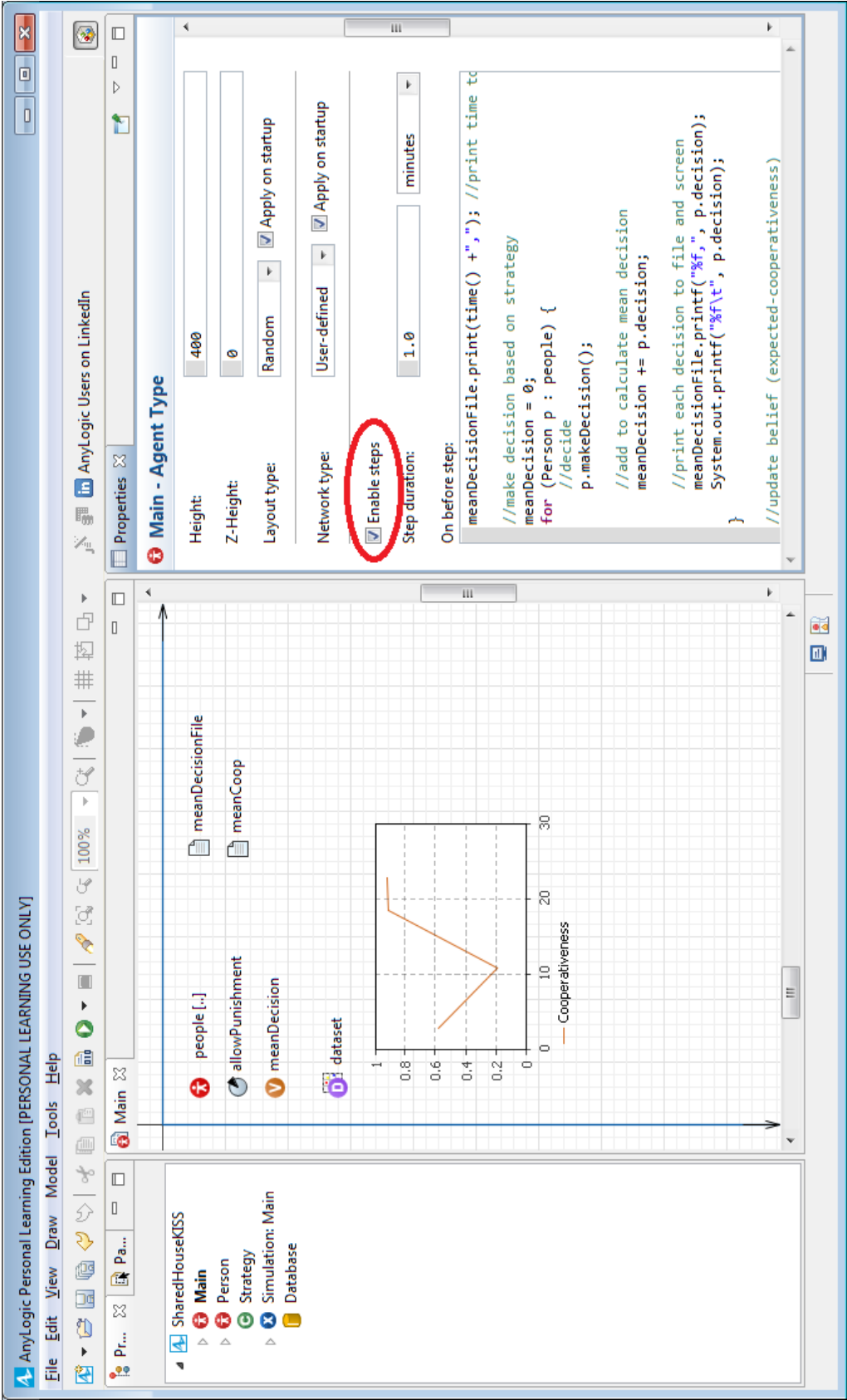


Fig. 4.6 AnyLogic IDE with the project of the KISS approach

Pseudocode 4.2: Contribution pseudocode

```

switch strategy do
  case COOPERATE
    if expectedCooperativeness > 0.4 then
      | decision  $\leftarrow$  1;
    else
      | decision  $\leftarrow$  0.75;
    break;
  case MAXIMIZE
    maxDecision  $\leftarrow$  0;
    maxOutcome  $\leftarrow$  0;
    outcome  $\leftarrow$  0;
    /* try different temporary decisions: 0, 0.1, 0.2, ..., 1 */
    for tempDecision=0 to 1 with step=0.1 do
      | outcome  $\leftarrow$  freeTime - tempDecision  $\times$  main.maxWashingTime +
      | (tempDecision + expectedCooperativeness  $\times$  3)  $\times$  maxWashingTime  $\times$  0.4;
      | if main.allowPunishment and expectedCooperativeness > tempDecision then
      | | outcome  $\leftarrow$  outcome - expectToBePunished  $\times$  maxPunishTime  $\times$  9;
      | if outcome > maxOutcome then
      | | maxOutcome  $\leftarrow$  outcome;
      | | maxDecision  $\leftarrow$  tempDecision;
    decision  $\leftarrow$  maxDecision;
    break;
  case RECIPROCATE reciprocal strategy
    | decision  $\leftarrow$  expectedCooperativeness;
    break;

```

Pseudocode 4.3: Punishment heuristic

```

if defection < 0 then defection  $\leftarrow$  0;
;
if defection > 0.1 then
  | angerLevel  $\leftarrow$  p.inclinationToBeAnnoyed + defection;
else
  | angerLevel  $\leftarrow$  p.inclinationToBeAnnoyed + defection - 0.8;
if (random value between 0 and 1) < (2  $\times$  angerLevel) then
  | p.punishDecision  $\leftarrow$  (p.willingToPunish + defection)/2;
else
  | p.punishDecision  $\leftarrow$  0;

```

Pseudocode 4.4: Calculate outcome with expected punishment

if *expectedCooperativeness* > *tempDecision* **then**
outcome $\leftarrow$ *outcome* – *expectToBePunished* $\times$ *maxPunishTime* $\times$ 9 ;

simulation runs, each Person agent and values of their variables can be observed, as shown in Figure 4.7b.

4.4 Modelling using KIDS Approach

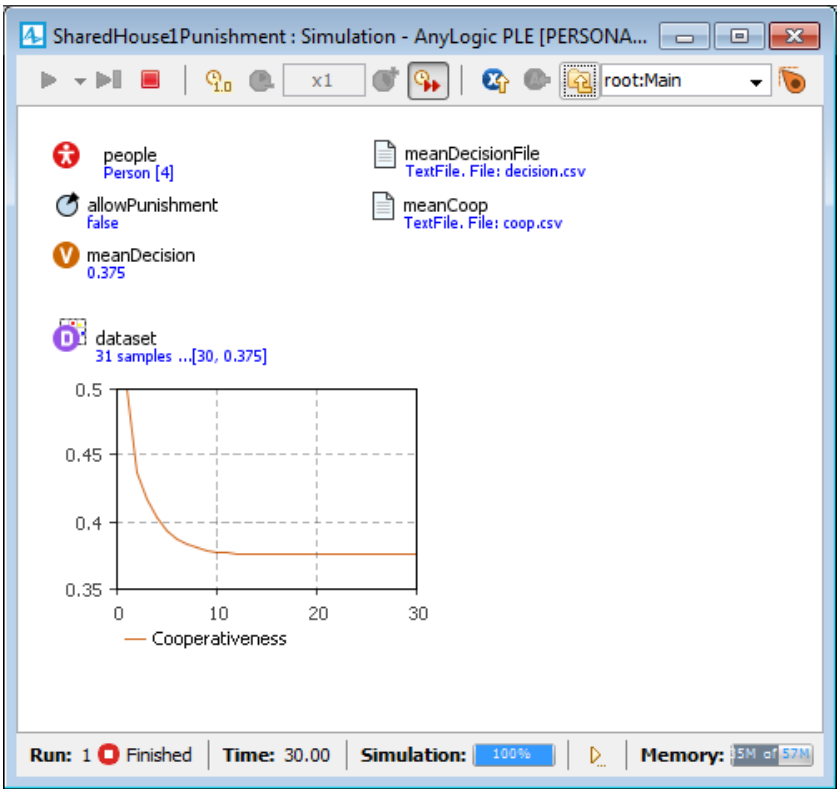
4.4.1 Conceptual Modelling - Design the conceptual model

The simulation in this chapter has the same purpose as the simulation of the KISS approach (Section 4.3) : comparing the cooperation level between normal PGG and PGG with punishment option. Many elements in the KIDS approach are the same as the KISS approach in Section 4.3. The PGG has the same payoff function. Agents in the shared house can use one of three strategies: cooperative strategy, maximizing strategy, and reciprocal strategy. However, in order to illustrate the capability of the framework and increase the realism of the simulation, rather than modelling the problem in a round-based fashion like in the KISS approach, the system is modelled in continuous time. The environment's time advances in equal increments, and can be returned as a combination of values of day, hour, minute, second.

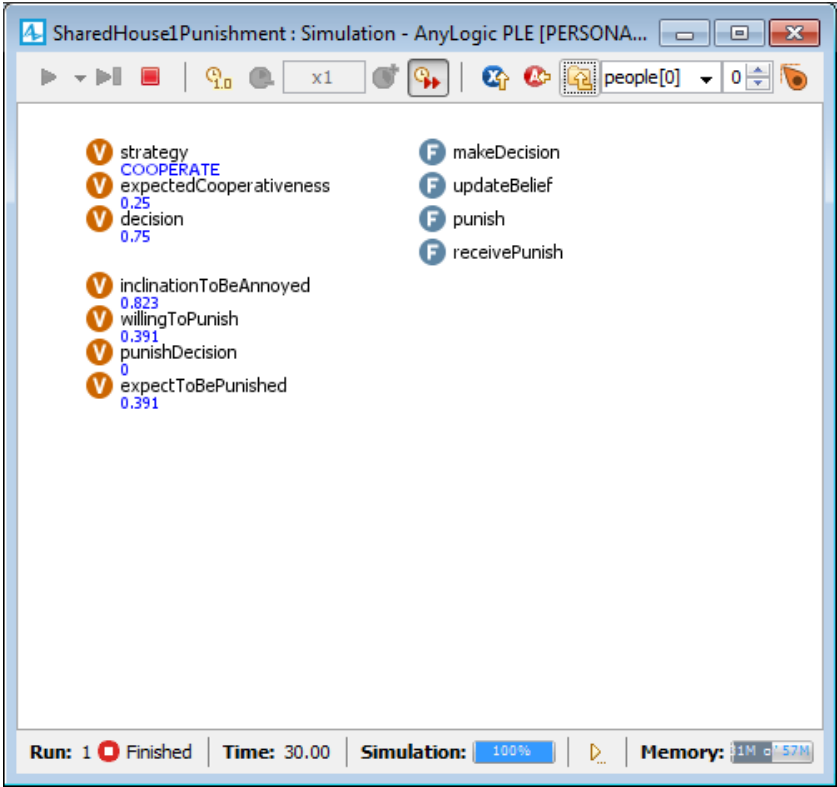
The system will be modelled with 24 hours a day. Each person has three meals a day at three random points of time (breakfast, lunch, and dinner). Each day the three meals can occur in three periods of time: 6:00-7:59, 12:00-13:59, and 19:00-20:59. And when one eats a meal, s/he will use the dishes. After using the dishes, s/he can choose to wash them or not. The meal time of each person is randomized and independent of others. Thus, the actions of people in this simulation are *asynchronous*. The dish washing situation in this simulation is still a PGG: time is contributed and everyone gets clean dishes. With the difference of asynchronous scheduling, the average cooperativeness of the group will be calculated by the most recent contributions. This means that during a meal period, if agent 1 makes a decision before agent 2, the contribution of agent 1 will be updated in the average cooperativeness given to agent 2.

In summary, the settings of the PGG are:

- Number of participants: 4.
- Group structure: one group of four people.
- One-shot or repeated games: repeated game with 15 meals.
- Treatments: no punishment, altruistic punishment.
- Players' action: contribute, update belief, punish (if applicable).



(a) Main screen



(b) Person screen

Fig. 4.7 AnyLogic implementation of the KISS approach

- Game loop: Give initial endowment; When the meal time arrives, the agent chooses cooperation level, updates belief, and punishes (if applicable).
- Scheduling: continuous time and asynchronous decision making.
- Initial endowment: 20 minutes.
- Payoff function: $outcome = freeTime - timeSpentWashing + 0.4 \times totalGroupWashingTime$
- Information: players were informed of the average cooperativeness of the group (with the most recent contribution of individuals).

The framework required the following details for the conceptual model:

- Inputs (Experimental factors):
 - Punishment treatment: no punishment, altruistic punishment
- Outputs (Responses):
 - Average cooperativeness of the group
 - Cooperativeness of each agent over time
- Model Content:
 - Scope in Table 4.3
 - Level of details in Table 4.4
- Assumption:
 - People make decisions simultaneously.
 - People have complete and common knowledge of the game structure and payoffs.
- Simplification:
 - People cannot communicate in any way.
 - People use the same strategy in the whole game.
 - People consider temporal payoff for decision making (instead of number of dishes, the time to washing dishes).
 - People have three meals per day: breakfast (6:00-7:59), lunch (12:00-13:59), and dinner (19:00-20:59).

4.4.2 Analysis

Identify agents and actors

People are modelled as Person agents, who can make contribution decisions during the simulation.

Table 4.3 Model Scope of Case Study 2 - KIDS approach

Component	Include/Exclude	Justification
Person	Include	Required for decision making.
House	Exclude	Do not model physical space.
Dish	Exclude	Replaced by temporal payoff.

Table 4.4 Model Level of Detail of Case Study 2 - KIDS approach

Component	Detail	Include/Exclude	Comment
Person	Contribution	Include	The result of decision making.
	Others' contribution	Include	Information required as inputs of decision making.
	Strategy	Include	Required for decision making.
	Schedule	Include	Meal schedule, simplified to three meals per day.
	Time takes to wash a dish	Include	Scale the washing time with the contribution decision.

Identify use cases

Each meal, a Person agent can perform three actions which correspond to three use cases: Contribute, Update Belief, and Punish.

Document use cases

1. Use case name: Contribute

- (a) Description: Person makes decision on how many MUs to contribute.
- (b) Agents: Person
- (c) Pre-conditions: When it is a meal time.
- (d) Post-conditions: Contribution is decided.
- (e) Flow of events: Person makes contribution based on the strategy.

2. Use case name: Update Belief

- (a) Description: Person updates their belief about contribution based on others contributions.
- (b) Agents: Person

- (c) Pre-conditions: Person is informed about the average cooperativeness of the group.
- (d) Post-conditions: Belief is updated.
- (e) Flow of events: Person updates belief.

3. Use case name: Punish

- (a) Description: Person punishes the free riders.
- (b) Agents: Person
- (c) Pre-conditions: Punishment is allowed.
- (d) Post-conditions: Punishment decision is carried out.
- (e) Flow of events: Person compares contribution of other agents in the group with the average cooperativeness. If it is lower, that agent is classified as a free rider and will be punished.

Draw use case diagram

The Use Case Diagram is the same as the KISS approach (Figure 4.1).

4.4.3 Design

The structure is described by the class diagram (Figure 4.8) and Person's behaviour is defined with statecharts (Figure 4.9). Using the statechart, agents only act when an event is triggered (a condition is met or a message is received). This makes asynchronous scheduling more computationally efficient and reflects the real world more accurately.

There are many elements in the class diagram similar to the KISS approach. There are two classes: Person and Main. The relationship between them is association. The Main class initializes Person agents but does not command them to perform any tasks. So in Main class, there are *people* variable that is a list to manage Person agents, *allowPunishment* boolean for experimental treatment, *meanDecisionDuringMeal* and *meanDecisionAfterMeal* to save the mean decision during meal (after each decision is made) and after meal (after all decisions are made), and an *initialize()* operation to initialize Person agents. Person has similar variables to the KISS approach (the first eight variables in the class diagram) and the same four operations. At the beginning of every day, Person agents randomize three points of time for three meals, that are stored in six variables (*hourBreakfast*, *minuteBreakfast*, *hourLunch*, *minuteLunch*, *hourDinner*, *minuteDinner*). During simulation run, Person agents observe time in the system, and make a decision when the meal times come.

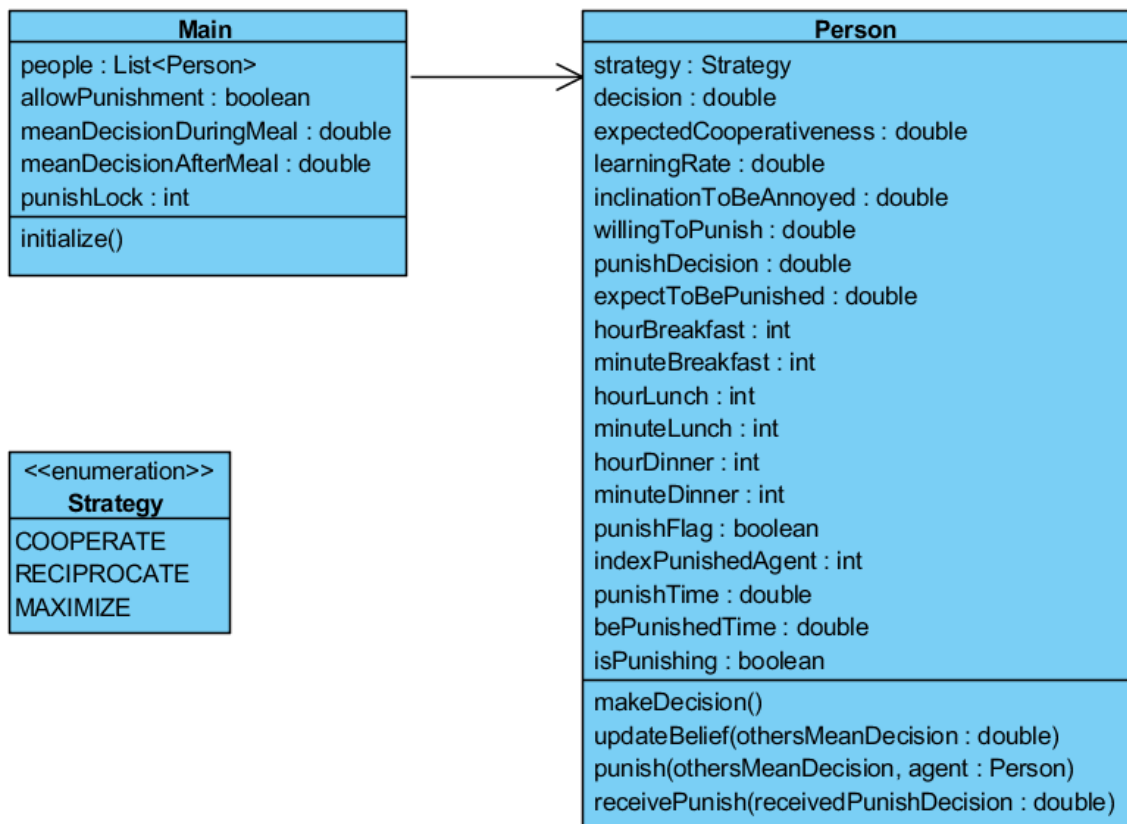


Fig. 4.8 Class diagram - KIDS approach

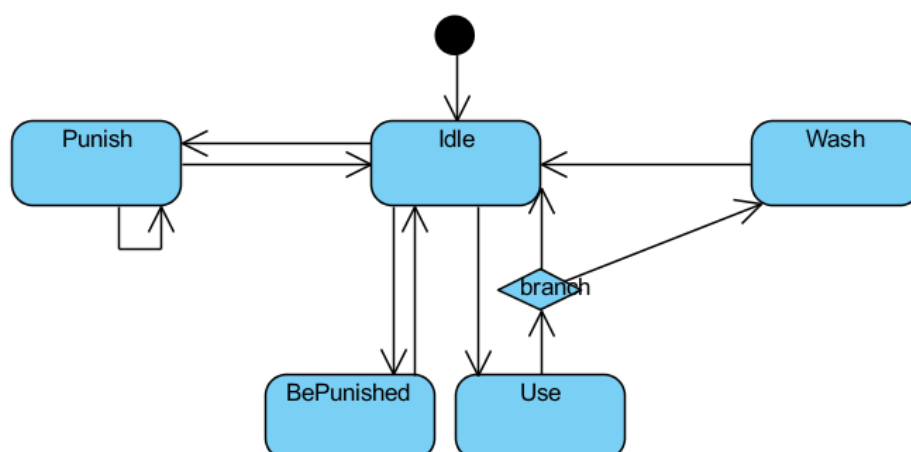


Fig. 4.9 Statechart - KIDS approach

The statechart of Person agent has five states: the state Idle for when the agent is doing nothing, two states Wash and Use to represent two use cases Contribute and Update Belief, and two states Punish and BePunished for the Punish use case. The statechart is shown in Figure 4.9. In the beginning, the agent is in state Idle. When it is meal time, the transition is triggered from Idle to Use. Each agent has three meals a day which are randomized to three different periods at the beginning of each day. After the agent has been in the Use state for a period, the outgoing transition will be triggered. When the agent leaves the Use state, the makeDecision function will be called to update the decision variable. The next state will be decided based on the decision variable. If the variable is 0, the agent will be in Idle state. Otherwise, it will go to Wash state. The greater the decision value is, the more time the agent stays in Wash state. After making the decision to use or wash dishes, the agent updates its belief by recalculating the expected cooperativeness using Equation 4.1.

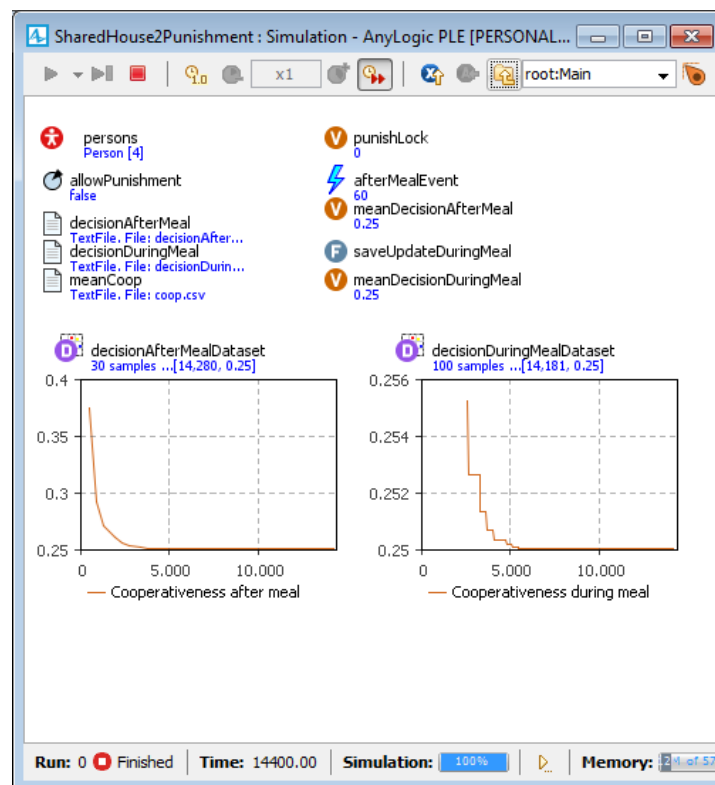
If punishment is allowed, the agent can move from Idle to Punish state. To avoid the conflict between using activities and punishing activities, the transition from Idle to Punish is triggered after the meal periods: 9:00, 15:00, and 22:00. In the Punish state, the agent will consider each of the other three agents for punishment. The punishment heuristic is the same as in KISS approach. When a punishment is decided, the punisher sends a message to the punished agent and this agent's state changes to BePunished for the period of punished time. The punishing agent also stays in Punish state during punishing time. After all three agents are considered, the punishing agent is back to Idle state.

4.4.4 Implementation

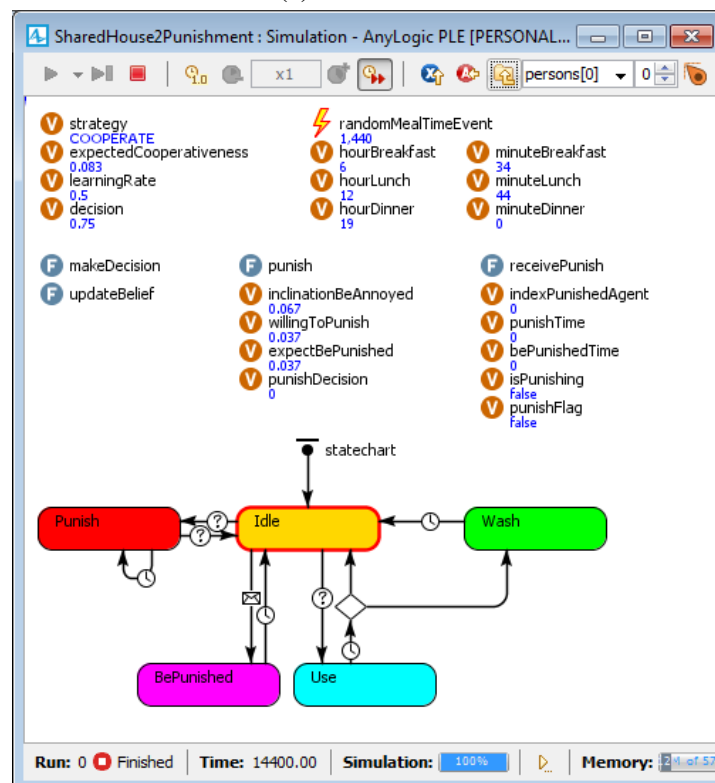
Figure 4.10 shows the implementation of the KIDS approach. Figure 4.10a is the Main screen showing the results of a simulation run without punishment. There are two timeplots of decision during and after meal. A screenshot of Person agent can be viewed in Figure 4.10b. The coding for Contribute and Punishment use cases of Person agent is the same as the KISS approach (Listing 4.1 and 4.3).

Statecharts can be implemented from scratch by adopting the state design pattern. Fortunately, modellers of all levels of expertise can take advantage of recent visual support of statecharts in simulation softwares such as AnyLogic, or Repast Symphony. In Figure 4.10b, the statechart is implemented as a graphical element in AnyLogic. More details on the transitions of the statechart and their triggered conditions are described in Table 4.5.

There are three variables to support the punishment mechanism. First is the *punishFlag* boolean variable with default value of false, that ensures an agent only goes to Punish state once per meal. The *punishFlag* is set to true when an agent moves from Idle to Punish state, and is reset to false when an agent changes from Idle to Use state. Secondly, the *isPunishing* boolean



(a) Main screen



(b) Person screen

Fig. 4.10 AnyLogic implementation of KIDS approach

Table 4.5 Transitions in the statechart of Person agent

From state	To state	Triggered by	Condition	Guard	Action
Idle	Use	Condition	(h==h1 && m==m1) (h==h2 && m==m2) (h==h3 && m==m3)	—	punishFlag=false;
Use	branch	Timeout	15 minutes	—	makeDecision();
branch	Wash	Condition	decision >0	—	
branch	Idle	Default	is taken if all other transitions are false	—	updateBelief();
Wash	Idle	Timeout	decision*maxWashingTime (minutes)	—	updateBelief();
Idle	Punish	Condition	main.allowPunishment && main.punishLock && !punishFlag && strategy!=MAXIMIZING && (h==9 h==15 h==22)	—	main.punishLock++; punishFlag=true;
Punish	Punish	Timeout	punishTime (minutes)	isPunishing	
Punish	Idle	Condition	indexPunishedAgent==main.persons.size()	—	main.punishLock--;
Idle	BePunished	Message	"BePunished"	—	main.punishLock++;
BePunished	Idle	Timeout	BePunished	—	main.punishLock--;

variable (default value of false) is the guard for the self-transition of Punish state and ensures that this transition is only triggered after a punishing decision has been made. *isPunishing* is set to false every time when entering the Punish state, and set to true after the punishing decision has been made in the *punish()* operation. The third variable is *punishLock*. During punishment, both the punishing agent and the punished agent will spend their time in Punish and BePunished state. For every minute the punishing agent is in Punish state, the punished agent spends 3 minutes in BePunished state. There can be the case that the punished agent will receive multiple punishments and the punished time will overlap. To avoid this situation, a synchronization mechanism using a lock *punishLock* has been designed so that there can be only one punishing agent at a time. The *punishLock* is an integer variable in Main with the default value of 0. The trigger of the transition from Idle to Punish state requires the lock to be 0. When the punishing agent enters Punish state, it increases the lock by 1. So when one agent is in Punish state, the other agents cannot move from Idle to Punish state. The lock is decreased by 1 when the punishing agent moves from Punish to Idle. In addition, the punished agent also increases the lock by 1 when it moves from Idle to BePunished and decreases the lock by 1 when it changes back to Idle state. With this lock mechanism, one punishing agent can punish multiple punished agents, and the lock only resets to 0 when all punished agents are back to Idle. For example, in Figure 4.11, agent A spends 1 minute to punish agent B, i.e. agent B stays in BePunished state for 3 minutes. After a minute in Punish state, agent A spends another minute to punish agent C. Agent C has to stay in BePunished for 3 minutes. So after 2 minutes, agent A finishes punishing and moves to Idle state, but agent B and C are still in BePunished state. At that moment, the lock is equal to 2, i.e. other agents cannot punish. When agent C finishes being punished and changes to Idle state, the lock is back to 0.

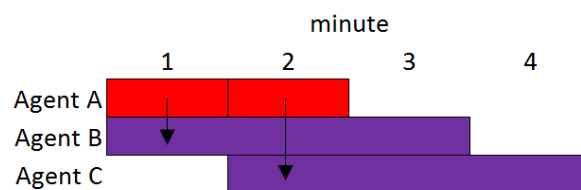


Fig. 4.11 Example of a punishing situation

4.5 Experiments and Results

Let us name agent-based simulations of KISS and KIDS approaches “Simulation 1” and “Simulation 2”. In this validation experiment, both simulations were set up with four agents: one cooperator, two maximizers, and one reciprocator. Simulation 1 runs for 15 rounds; while

Simulation 2 runs for 15 meals (5 days with 3 meals a day). The mean cooperativeness of all agents over time is collected in four different scenarios: Simulation 1 and 2 with or without punishment. In each scenario, the simulations are replicated 100 times and the average cooperativeness of each round (or meal) is calculated. Figure 4.12 illustrates the mean cooperativeness of four agents in the house over time. In the scenarios where punishment is allowed, the mean cooperativeness is increasing over time and becomes stable at 0.8. In the scenarios without punishment, the cooperativeness in the shared house decreases over time then remains stable at 0.33. Our hypothesis is confirmed: the trending of the cooperativeness in the simulation is the same as the observation in [Fehr and Gächter's](#) experiment.

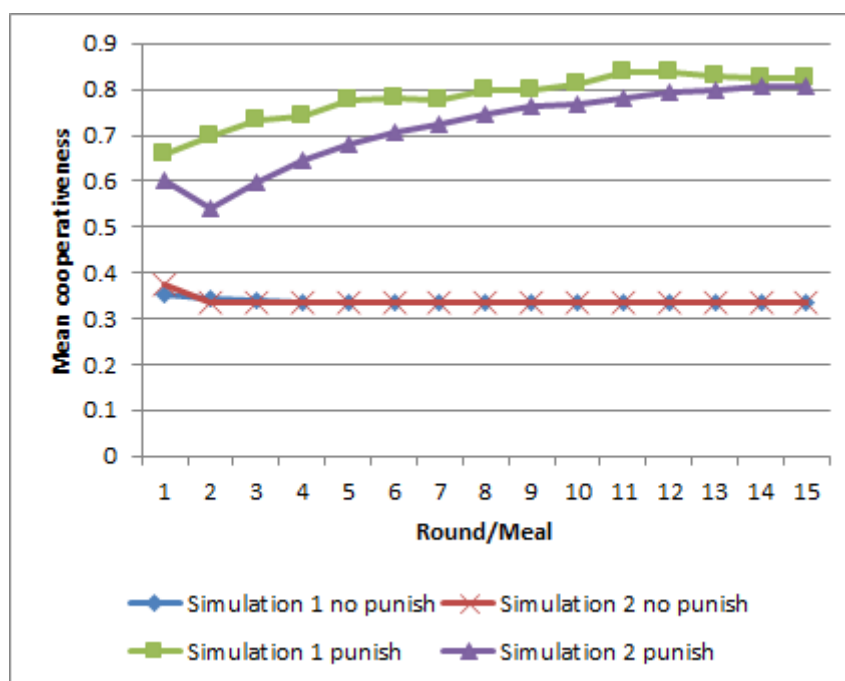


Fig. 4.12 Mean cooperativeness over time

There are two results to be discussed. Firstly, the initial value of mean cooperativeness is decided by the composition of different strategies and the punishment setting. The cooperativeness of cooperator is always 1. The initial mean cooperativeness of reciprocators is 0.5 because their initial expected cooperativeness is uniformly distributed from 0 to 1. The initial mean cooperativeness of maximizers is 0 during no punishment and increases to 0.5 when the punishment is enabled in their outcome calculation. Knowing the initial mean cooperativeness of the three strategies, the initial value of group cooperativeness can be calculated. Secondly, the reason for the decreased trend of contribution is that the two maximizers cause the mean cooperativeness of the house to be lower than the expectation of the reciprocator. Therefore, the reciprocator keeps lowering the decision variable until its expectation is equal to the mean

cooperativeness of the house. In short, the initial mean cooperativeness is decided by the composition of the strategies the agents use, and the trend of cooperativeness is driven by the reciprocator.

Without punishment, the mean cooperativeness of Simulation 1 is similar to Simulation 2. But when punishment is allowed, Simulation 2 has a sudden drop in the second round before it raises back to 0.8. The difference between Simulation 1 and 2 occurs because of the dissimilarity in the order of decision making and the belief update between the two simulations. In Simulation 1, in every round, all agents make the decision; and when every agent is finished with that, they will update the expected cooperativeness. In Simulation 2, all agents make a decision during a period of a meal time, and update the expected cooperativeness before going back to Idle state. Therefore, when the reciprocator makes a decision after the contributor, the reciprocator knows there are already contributions and raises his expected cooperativeness. But when the reciprocator makes a decision before the contributor, the reciprocator does not see any contribution; so his expected cooperativeness of other agents is lower than the previous case. This causes a significant drop in the reciprocator's cooperativeness in the next meal which leads to the sudden drop in Simulation 2. Simulation 2 has discovered some dynamics of how the order of agent decision making affects the cooperativeness, which is not present in Simulation 1. This is a result of making the agent behaviour more realistic with asynchronous updates in Simulation 2.

4.6 Discussion

4.6.1 The simulation studies

KISS vs. KIDS

In the experimental results, both KISS and KIDS approaches replicate the cooperativeness trending in [Fehr and Gächter](#)'s experiment; but KIDS has discovered some different behavioural dynamics. The two factors that allow these dynamics to appear are that (1) agents make decisions asynchronously during a meal and (2) when a decision is made during a meal, it is updated into the mean cooperativeness of that meal for the decision of the next agent. Since KIDS approach suggests starting with a descriptive model and then only simplifies it when this turns out to be justified, we can try to simplify the simulation model of KIDS approach. A discrete-time simulation can exhibit the same behaviour by (1) randomizing the order of decision making in every round, and (2) updating the mean cooperativeness of the group as soon as an agent decides. However, this simplified simulation will not be able to collect the states of the agents in a specific minute during the simulation run. This is the extra information

that the simulation of the KIDS approach provides with the support of the statechart compared to the one of the KISS approach.

Neither KISS nor KIDS is the best one in every situation. But in the areas dominated by complex phenomena (typically in Social Sciences and Economics), the balance is shifted towards KIDS. The ABOOMS framework can be used with both KISS and KIDS. Especially in the case of KIDS where the initial descriptive model can be complex, the framework provides a structured approach to design, document, and implement the simulation model.

Continuous-time setting

The model of KIDS approach has shown that the continuous-time setting can trigger a different mode of strategic reasoning compared to the discrete-time setting in KISS approach. The game theory and laboratory experiments usually design the game in discrete periods and strictly defined orderings of moves and information sets. Real social dilemmas do not have this structure. And the model of KIDS approach has shown it is interesting to investigate more about agent behaviours in continuous-time setting.

4.6.2 OOAD and UML

Object-oriented principles in the ABOOMS framework

To understand the capabilities of the ABOOMS framework, we examine the object-oriented principles in the two simulations. [Booch \(2007\)](#) defines four major elements of the object model:

1. “*Abstraction* denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.
2. “*Encapsulation* is the process of compartmentalizing the elements of an abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.
3. “*Modularity* is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.
4. “*Hierarchy* is a ranking or ordering of abstractions.” ([Booch, 2007](#))

Abstraction is a process that shows the essential information and hides away irrelevant information. In our framework, the abstraction is represented in two ways: data abstraction and

control abstraction. *Data abstraction* is when multiple small data types are combined together to create a complex data type. For instance, a Person class has attributes such as strategy and contribution which are meaningful in the PGG perspective, and ignores the irrelevant information such as height and weight. *Control abstraction* allows hiding a sequence of actions of a complex task. For example, a Person object performs the punish task, but the steps (get the mean cooperativeness of the group, compare with cooperativeness of another Person, decide whether to punish) are hidden. In short, the abstraction represents the important details and hides away the implementation.

Encapsulation is the process of information binding, i.e. binding both attributes and operations together within a class, and hiding all the details that are not related to the essential characteristics. For example, a Person class has makeDecision() operation that can be called by another class. But the logic to perform this task is hidden away and only known by the Person class. Additionally, when discussing the object-oriented capability of NetLogo and AnyLogic in Section 3.4, we mentioned that NetLogo can declare agent type with attributes but does not allow the programmer to define operations for the agent type. This means attributes are bound to the agent type, while operations are not. For comparison with the AnyLogic implementation of the KISS approach, the code for a NetLogo implementation of KISS approach is listed in Appendix A.

Modularity means dividing a program into many modules that are “cohesive (by grouping logically related abstractions) and loosely coupled (by minimizing the dependencies among modules)” (Booch, 2007). If we develop a simulation from scratch, it can be divided into three modules: one module for the interface, one for running the simulation, one for agents. But since we used AnyLogic, these modules are already developed, we only have to develop the agents and link them with the existing system.

Hierarchy allows a system to be made up of subsystems, and these subsystems can be made up of other subsystems. The design of both simulation models in this case study is quite simple so there is no clear hierarchy. However, if we think Main as a house that contains four Person agents, this is an aggregation relationship (“part-of” hierarchy).

With these four elements, our simulation models can follow the software engineering principle of *separation of concerns*. Since each section addresses a separate concern, it is easier to be reused in other systems (*reusability*). It is also easier to understand and manage the system. In addition, the functionality overlaps are also minimal, that means it is easier to extend the functionality with minimal changes to the current system (*extensibility*).

UML diagrams in ABMS

Although ABMS research has increasingly agreed to adopt OOAD to design and implement their models, UML is not mentioned much in their publications (Bersini, 2012). Bersini (2012) has made an effort to introduce some UML diagrams (class, sequence, state and activity diagrams) to the ABMS community. Our work is not to compete but to extend this effort by demonstrating the application of OOAD mindsets and UML specification. In our framework, OOAD mindset along with UML diagrams are recommended to use as the formal specification to ease the modelling process at all stages, from analysis to design and implementation. Not only does this help the communication between a wide range of research disciplines, it can also facilitate a good development methodology for better agent-related researches. Another advantage of UML specification is that it is independent of the development lifecycle, which means once learnt social scientists can also apply it for many development frameworks or methodologies.

4.6.3 Can everything be agents?

Section 2.2.6 discussed the different views on the notion of agent between MAS and ACE. In the development process of this case study, from the MAS perspective, there is an entity that does not fit in the notion of agent: *Main*. *Main* initializes agents' strategies, manages the simulation, and collects statistics. *Main* provides services and functions to support agents working together; however, it lacks agent properties such as goal-directed behaviour. From the ACE perspective, it is acceptable to consider *Main* as an agent. The different perspective on this matter is also reflected in terminologies of AnyLogic software. In AnyLogic 6, *Main* is classified as an *active object*. However, a new scheme in AnyLogic 7 has merged AnyLogic 6 definitions of agents, active objects, entities, and resource units into a single agent definition. Therefore, under the new perspective of AnyLogic 7, *Main* is classified as an agent. In the case study above, *Person* is identified as agent in the Analysis process and *Main* is added as a class in the Design process. However, if *Main* is classified as an agent, it would be included in the use case diagram in the Analysis process as well. From this, we learnt that the capability and features of simulation software can affect the design. Even though the design can be developed with any simulation software, knowing the software beforehand can benefit the design process. Additionally, since this can cause confusion, an approach is needed to better classify agents when identifying entities in the system.

There are many proposals on the different typologies related to agents. Bradshaw (1997) defines "agents are objects with attitude". Ferber (1999) classifies agents and objects: agents are the only one to act, while objects are passive or inactive. Omicini et al. (2008) define A&A

(Agents and Artifacts) meta-model. Agents are autonomous, proactive entities that encapsulate control and are in charge of goals or tasks that altogether determine the whole system behaviour. Artifacts are passive, reactive entities that are in charge of required services and functions that make agents work together and shape the agent environment. [Kubera et al. \(2010\)](#) argue that it is not easy and intuitive to identify types of entities in ABMS, and proposes a unification of all notions “everything can be agent”. To link the unified agent concept with other works, Kubera characterizes three properties of agent action: activity (can initiate action), passivity (can undergo action), and lability (change states without acting nor undergoing action), then frames concepts in other works using these properties. For example, artifacts in [Omicini et al. \(2008\)](#) are passive entities; objects in [Ferber \(1999\)](#) have either passivity or none of the three properties.

The essence of the Analysis process is to understand the system by organising requirements around agents. So it is important to only include entities that are significant to use cases of agents. After consideration, the A&A meta-model has been chosen since it provides a clear distinction between agents as important decision makers and other entities that support it. Thus the A&A meta-model will be added to the step “Identify agents and actors” in the Analysis process.

4.7 Framework Revision and the ABOOMS Framework v2

4.7.1 Revision

There are three changes: one is in the lifecycle in the macro process, two are in two processes in the micro process. Afterwards, the ABOOMS framework v2 is summarized with all changes in red.

The Macro Process

The lifecycle of the framework was revised and one improvement was made. Besides laboratory experiments, the data used for the simulation can also come from other sources such as field experiment or interview. For generalisation, these sources will be added to the framework. These sources provide qualitative evidences such as observation of people’s behaviour in the field or text gained from transcribing someone talking or being interviewed. These evidences are a natural sources to inform the specification of agent behaviours. It may not give a complete picture but can provide explanation about reasoning or behavioural rules, thus contribute to the understanding of behaviours that people have. For example, [Bharwani \(2004\)](#) and [Taylor \(2003\)](#) tried to systematically analysed transcripts from interviews to extract clues about the

in-context behaviour of individuals and inform their simulation specification. Since the process of analysing natural language data depends upon the understanding of the observers and/or analysts, the data should be made available so that other researchers can analyse and spot aspects that the original analyst might have missed. The process of analysing and applying qualitative data can be summarised as follows:

- Conduct interviews or experiments to collect qualitative data.
- Process (such as transcribing from interviews) and analyse the qualitative data to extract information about individual behaviour.
- Use the information to inform the micro-level specification of an agent-based simulation.
- Implement the simulation, then perform experiments to validate and understand the connection between micro-level behaviours and macro-level outcomes.

The Micro Process - Analysis

In the micro process, there are two improvements in Analysis and Conceptual Modelling processes. For the third improvement, we updated the steps in Analysis so that the A&A meta-model can be used to properly distinguish agents and other artifacts. However, for this improvement, we decided to change some terminologies to avoid confusion created by the word “actor”.

In Framework v1, the first step in Analysis is “Identify agents and actors”:

- Agents: Internal entities that have properties of agents.
- Actors: External entities such as human and other systems.

As agents can also be identified as internal actors, it can be confusing if the name of the step includes the two words: “agents” and “actors”. Since actors here are external entities, it can be called “external actors”. Therefore, the new name of first step in Analysis is “Identify agents, artifacts, and external actors”. More details on each terms are as follows:

- Agents (internal actors): autonomous, proactive entities that encapsulate control and are in charge of goals/tasks that altogether determine the system behaviour. For example, in PGG, players are agents.
- Artifacts (internal actors): passive, reactive entities that are in charge of required services and functions that make agents work together and shape the agent environment. For example, an artifact in PGG can be the “game master” who informs players of the current stage of the game.

- External actors: external entities such as human and other systems.

The Micro Process - Conceptual Modelling

Through developing the two simulations, we discovered that it can be very hard for developers to understand the PGG since there are many settings and available options. However, during development, there are similar elements in PGGs of both simulations. Thus, we propose to add one step in Conceptual Modelling that suggests basic settings of PGG for better documentation. The settings of the PGG can be:

- Group structure: how many people in one group? How many groups there are in total?
- One-shot or repeated games: The game is one-shot or repetitive. How many rounds of experiment are there?
- Players' action: Action that the players can perform such as contribute, punish.
- Game loop: The main flow of the game that Players have to follow. For example: Give initial endowment; Play a turn by letting agents choose cooperation level; Punish.
- Scheduling: discrete-time/continuous-time and synchronous/asynchronous decision making.
- Payoff function: $\pi_i = \omega - x_i + MPRC \sum_{j=1}^N x_j$, in which ω is the initial endowment, x_i is the contribution of participant i , N is the number of players in a group, MPRC is marginal private capital return ($MPRC < 1$).
- Information: Incomplete or complete. Which information is available for the participants, and when?

4.7.2 The ABOOMS Framework v2

The Macro Process

Figure 4.13 is the lifecycle of the ABOOMS framework v2.

The Micro Process

The steps of five processes used to develop the simulation are described in detail below:

Process 1 - Conceptual modelling

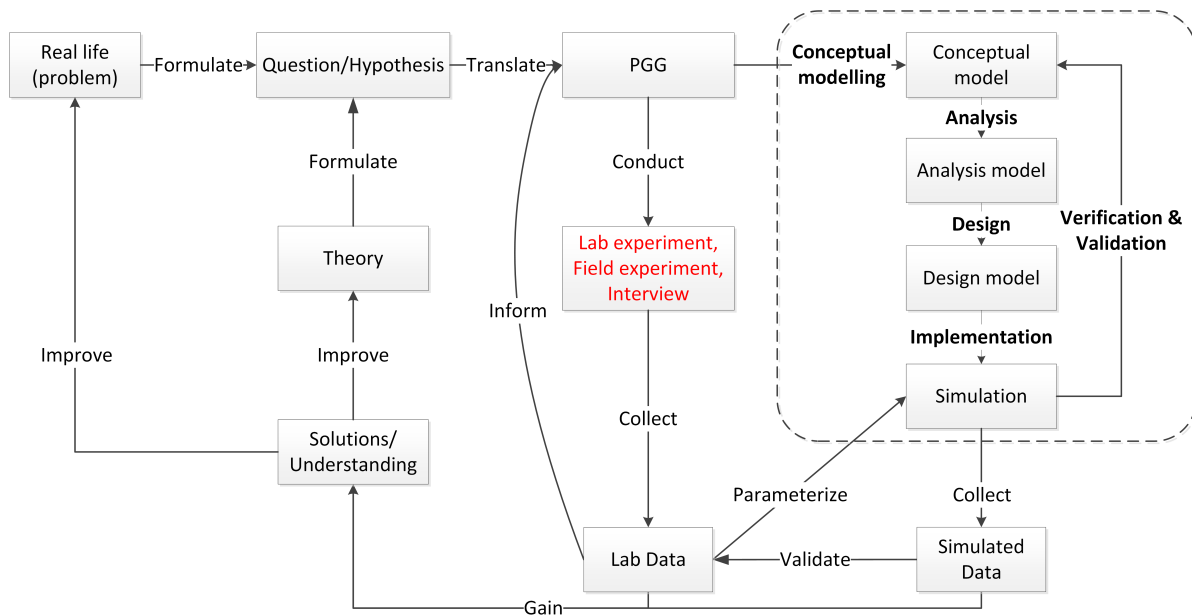


Fig. 4.13 Overview lifecycle of the ABOOMS framework v2

- Understand the problem situation.
- Determine objectives of the simulation study.
- Collect and analyse data and related theories.
- **Identify the settings of the PGG.**
- Design the conceptual model.
 - Identify the model inputs (experimental factors).
 - Identify the model outputs (responses).
 - Determine model content: scope and level of details.
 - Identify assumptions and simplification.

Process 2 - Analysis

- **Identify agents, artifacts, and external actors:**
 - **Agents (internal actors):** autonomous, proactive entities that encapsulate control and are in charge of goals/tasks that altogether determine the system behaviour. For example, in PGG, players are agents.

- Artifacts (internal actors): passive, reactive entities that are in charge of required services and functions that make agents work together and shape the agent environment. For example, an artifact in PGG can be the “game master” who informs players the current stage of the game.
 - External actors: external entities such as human and other systems.
- Identify use cases.
 - Document use cases.
 - Draw Use Case Diagram.

Process 3 - Design

- Define the structure (internal data, operation) with Class Diagram.
- Define behaviour: Define behaviour with behavioural diagrams, such as Statechart, Sequence Diagram, Activity Diagram.

Process 4 - Implementation

- Choose a programming language and/or a software: The simulation can be programmed from scratch or with a specialist simulation toolkit (NetLogo, AnyLogic, Repast Simphony).
- Programming: Writing the code.
- Testing and debug: It is important to perform testing in order to ensure the correctness of the model.

Process 5 - Verification and Validation

- Verify whether the simulation correctly implemented the conceptual model.
- Validate and tune agent behaviour based on the information from the lab or literature.

4.8 Summary

This chapter presents how OOAD can be used to design social agents. Using a problem of washing dishes in a shared house, two simulations were developed using two modelling principles: KISS and KIDS. In the development, use case diagrams were used to identify the interaction

of agents with the system; class diagrams were created to describe the structure of agents; and statecharts were applied to model agents' behaviour. In the experiment, two simulations replicate the trend of cooperativeness from [Fehr and Gächter](#)'s laboratory experiment. However, in the simulation of the KIDS approach, the order of agent decision making affects the cooperativeness in a way that is not presented in the KISS approach. Afterwards, the ABOOMS framework v1 was revised and improved to v2. In the framework v2, two refinements were made in the development lifecycle, a task for the settings of PGG is added into the Conceptual Modelling process, and the A&A meta-model is introduced to the Analysis process.

The ABOOMS framework acts as a general guidance that is sufficient for applying OOAD to modelling social agents in PGG. OOAD helps to increase the reusability and extensibility of agent-based models in the sense that existing models can be quickly adjusted by other researchers to perform different experiments. It is also easier to assemble multiple parts from multiple models to develop a new one. Using OOAD as a complementary approach not only enhances the design of agents but also improves the communication between the different stakeholders (e.g. economists, ecologists, psychologists, and computer scientists) working jointly on a Behavioural Economics case study. This work is meant to be the first step to overcome the communication barrier between the different stakeholders to achieve a standard agent technology with integrated interdisciplinary foundation.

Chapter 5

Case Study 2: A Psychological Model of Punishment Sensitivity

This chapter examines the micro-situational level of the ABOOMS framework, specifically how it supports complex decision making. The case study is based on an experiment about the effect of individual differences in punishment sensitivity on contribution levels in a public goods game (Section 5.1). These individual differences are modelled based on experiments on the Behavioural Inhibition Scale from Psychology, which is explained more in the Conceptual Modelling process in Section 5.2. The processes of Analysis, Design, and Implementation in the ABOOMS framework are described in Section 5.3, 5.4, and 5.5. Then Section 5.6 uses simulation experiments to examine the correlation between punishment sensitivity and variability of free riding behaviour under different punishment conditions. Section 5.7 discusses the simulation results and aspects related to design of statechart-based agents. Lastly, the summary of this chapter is presented in Section 5.9.

5.1 Case Study of Punishment Sensitivity in the Public Goods Game

The PGG is a standard experimental economics approach to study human cooperation. In this game, each player faces a dilemma: to invest into public goods and potentially get a higher return through cooperation in the future; or keep their endowment which essentially means free riding on other people's contributions. A standard finding using this paradigm is that there is variability in behaviour: some individuals cooperate while others free ride (Fehr and Gächter, 2002; Fischbacher et al., 2001). One mechanism to promote cooperation is monetary punishment for free riding. The work of Skatova and Ferguson (2013) demonstrates that the

way people are affected by the punishment in the PGG differs between individuals and the differences are explained by the trait of punishment sensitivity.

In their experiment, participants played a series of standard PGGs with varying punishment conditions. Before playing the game, individual punishment sensitivity of participants was assessed through the Behavioural Inhibition Scale (BIS) ([Carver and White, 1994](#)). One of the experimental treatments is that punishment is expected but sometimes not implemented, which is a twist compared to common punishment settings in other experiments. The results demonstrate that participants contribute more under threats of punishment compared to no threat of punishment, and that people with higher punishment sensitivity provide higher contributions even when punishment is not certain. This research suggests that varying probability of punishment could affect contribution levels of groups depending on individuals' levels of punishment sensitivity.

The lab-based design of the experiment with real participants limits opportunities to test how different levels of punishment threat, in combination with different levels of sensitivity to punishment of group members, affects contribution levels of different groups. This gap can be filled by modelling an experimental game with ABMS, which allows capturing group dynamics through varying parameters of punishment and sensitivity to punishment in a series of artificial experiments. The aim of this chapter is to develop a proof-of-concept agent-based simulation that can replicate the contribution levels in different experimental treatments in the lab.

5.2 Conceptual Modelling

5.2.1 Understand the problem situation

Since the simulation study is based on the laboratory experiment of [Skatova and Ferguson \(2013\)](#), we need to understand the structure and settings of the laboratory experiment. In the experiments, participants played in a group of four. The game comprises four blocks with punishment conditions in the following order:

1. A non-punishment block (a standard PGG).
2. An implemented punishment block.
3. A non-implemented punishment block.
4. A non-punishment block.

Each block consists of 10 rounds. After each round, participants are shuffled and put into a new group of four players. Each participant receives the initial endowment of 20 money units

(MUs), which can be divided between private and public accounts. After everyone makes their investment decision, the payoff is then calculated based on the Function 5.1.

$$\pi_i = 20 - g_i + 0.5 \sum_{j=1}^4 g_j \quad (5.1)$$

where a payoff (π) for a participant i is defined by their contribution (g_i) and the sum of contributions of all players in the group.

The environment in this game is modelled as discrete time. Each trial is a time step in the environment. In the first block, participants are only informed about conditions and settings in that block. After the first block of a standard PGG, participants receive additional instructions for the next three blocks, which introduce a punishment rule. In the non-implemented punishment block, the punishment never occurs. In the implemented punishment block, the punishment actually occurs in *two out of ten trials*.

5.2.2 Collect and analyse data and related theories

In the 1970s, Jeffrey Gray proposed the biopsychological theory of personality. He hypothesized two systems that control and motivate human behaviour: behavioural activation system (BAS) and behavioural inhibition system (BIS). BAS is activated upon receiving cues from the environment that the goal can be met or a reward can be received when a behaviour is performed. Thus, that behaviour is encouraged by BAS. On the other hand, BIS is activated upon receiving cues from the environment that punishment or negative responses may occur when performing a behaviour. Therefore, that behaviour is avoided (inhibited). BAS is related to positivity and rewards, while BIS is related to negativity and punishment. When an event occurs to a person, one of the two systems will activate accordingly to examine the events and motivate behaviour of that person. Since Skatova's experiment focuses on punishment sensitivity, BIS is concerned. Skatova hypothesized that the heterogeneity in BIS can explain variation in participants' responses in PGGs with punishment. Carver and White (1994) proposed a scale to assess BIS psychometrically using a questionnaire. Using this questionnaire to measure the BIS of participants before the laboratory experiment, the relationship between BIS sensitivity and behaviours in PGGs with possibility of punishment was investigated.

5.2.3 Determine objectives of the simulation study

The objective of the simulation study is to create a proof-of-concept simulation to:

- Validate the way BIS-based punishment sensitivity affects contribution under PGG with different punishment conditions.

- Experiment with different punishment conditions such as a different number of punishments occurring in the implemented punishment block (originally two out of ten rounds in the laboratory experiment).

5.2.4 Identify the settings of the PGG

The settings of the PGG are:

- Group structure: groups of four people.
- One-shot or repeated games: Repeated games in 4 blocks of different punishment conditions. Each block has 10 rounds.
- Players' action: contribute, update belief.
- Game loop: Give initial endowment; Play a turn by letting agents choose cooperation level; Update belief.
- Scheduling: discrete-time and synchronous decision making.
- Payoff function: Function [5.1](#)
- Information: Players know the average contribution of the group.
- Punishment conditions: Three different conditions are non-punishment, implemented and non-implemented punishment.

5.2.5 Design the conceptual model

The framework required the following details for the conceptual model:

- Inputs (Experimental factors):
 - Punishment treatments: non-punishment, implemented and non-implemented punishment.
 - Different ratio of properties in agent population.
 - Ratio of occurred punishment in implemented punishment block.
- Outputs (Responses):
 - Average contribution of the group per round, per block.
 - Contribution of each agents over time.
- Model Content:
 - Scope in Table [5.1](#)
 - Level of details in Table [5.2](#)

Table 5.1 Model Scope

Component	Include/Exclude	Justification
Person	Include	Required for decision making.

Table 5.2 Model Level of Detail

Component	Detail	Include/Exclude	Comment
Person	Contribution	Include	The result of decision making.
	Others' contribution	Include	Information required as inputs of decision making.
	Strategy	Include	Required for decision making.
	Punishment sensitivity	Include	To model the sensitivity of different people to others.

- Assumption:
 - People make decisions independently and simultaneously.
 - People have complete and common knowledge of the game structure and payoffs.
 - High-anxiety participants play more cooperative strategies.
- Simplification:
 - People cannot communicate in any way.

5.3 Analysis

5.3.1 Identify agents, artifacts, and external actors

There is one type of *agent*: Person. A Person agent represents a participant in the game. Person agents can make investment decisions. There is also a Main *artifact*, which acts as a game master, controls the game stages, and lets Person agents know about the current stage and punishment condition of the game. At the beginning of each game, each Person agent is assigned to a Group *artifact*. The Group artifact manages the contribution and punishment of the group. There are no external actors in this game. In summary, there are three types of entities: Person agent, Main artifact, and Group artifact.

5.3.2 Identify use cases

The next step is to identify use cases which can be performed by the actors. For the game in this case study, there are five use cases:

1. Prepare before a round: Main randomly shuffles Persons into groups of four, then informs Persons and Groups of the punishment condition.
2. Play a round: Persons play a round, which involves other use cases: invest, punish, and learn.
3. Invest: Persons decide how many MUs to invest.
4. Learn: Persons learn from the experience.
5. Punish: Groups determine which player is to be punished. This use case is optional and only performed if punishment is implemented in that round.

5.3.3 Document use cases

1. Use case name: Prepare before a round
 - (a) Description: Main performs activities before a round.
 - (b) Agents & artifacts: Person, Main, Group
 - (c) Pre-conditions: A previous round is ended.
 - (d) Post-conditions: Persons are shuffled into different groups. Punishment conditions are informed.
 - (e) Flow of events: Shuffle players into groups of four. Inform all Persons and Groups of the punishment conditions.
2. Use case name: Play a round
 - (a) Description: Persons play the game in a round.
 - (b) Agents & artifacts: Person
 - (c) Pre-conditions: A new round is started.
 - (d) Post-conditions: None.
 - (e) Main flow of events: Persons make invest decision [Invest use case]. Persons update their belief [Learn use case]. If punishment is implemented in this round, the free riders get punished [Punish use case].

3. Use case name: Invest

- (a) Description: A Person makes a decision on how many MUs to invest.
- (b) Agents: Person
- (c) Pre-conditions: A round is started.
- (d) Post-conditions: Contribution is decided.
- (e) Flow of events: A Person makes an investment based on the strategy.

4. Use case name: Learn

- (a) Description: A Person updates their belief about contribution based on other contributions.
- (b) Agents: Person, Group
- (c) Pre-conditions: A Person is informed about the average contribution of the group.
- (d) Post-conditions: Belief is updated.
- (e) Flow of events: A Person requests average contribution from the Group. A Person updates belief.

5. Use case name: Punish

- (a) Description: A Person punishes the free riders.
- (b) Agents: Person
- (c) Pre-conditions: Punishment is allowed.
- (d) Post-conditions: Punishment decision is carried out.
- (e) Flow of events: A Person compares the contribution of other agents in the group with the average contribution. If it is lower, that agent is classified as a free rider and will be punished.

5.3.4 Draw use case diagram

In the final step of the Analysis process, a use case diagram of agents and artifacts in the system is created (see Figure 5.1). In this use case diagram, there are two advanced concepts: «include» and «extend» relationships. For example, in the case of «include», the “Play a round” use case includes the “Invest” use case, which means that when “Play a round” is executed, “Invest” must be executed. In the flow of the including use case (“Play a round”), there are inclusion points to specify where the included use cases (“Invest” and “Learn”) are to be executed. For

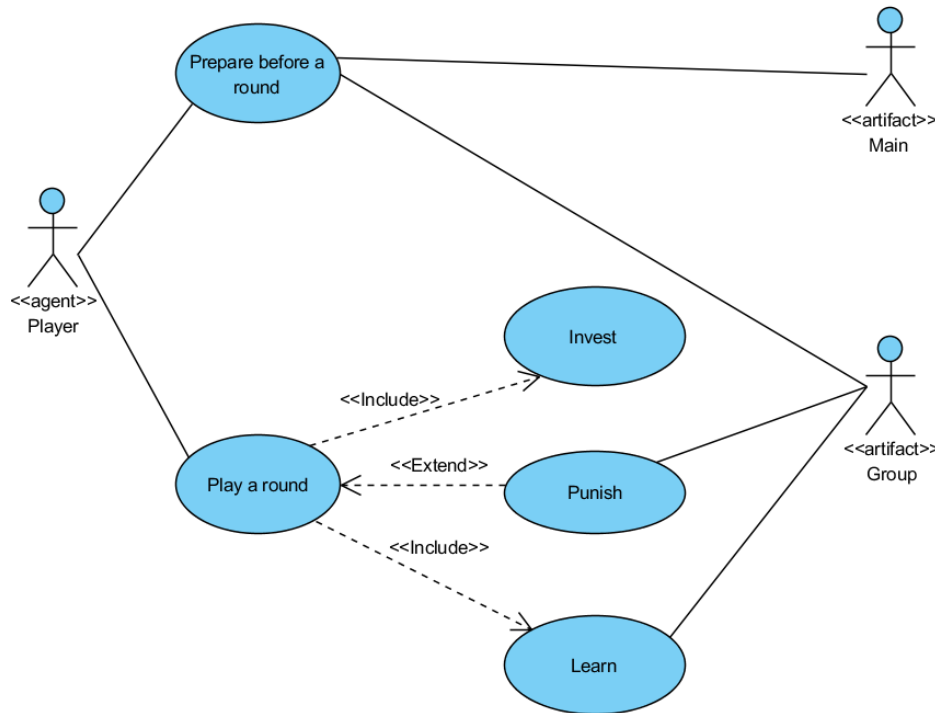


Fig. 5.1 Use case diagram

the «extend» relationship, the extending use case, “Punish”, might be performed as part of the “Play a round” use case but is not mandatory. The “Play a round” use case can execute successfully without invoking “Punish”. There are extension points in the flow to specify where the extending use case is executed. In the use case documentation in Section 5.3.3, the inclusion and extension points are mentioned with use case names in square brackets.

5.4 Design

In the Design process, we discuss the structural and behavioural designs. However, for better understanding, this section is split into three subsections: one on structural design, and two on different aspects of behavioural design (gameplay and punishment sensitivity).

5.4.1 Structural design

In terms of structural design, three classes are needed to structurally represent Main, Person, and Group. Figure 5.2 displays a class diagram for our system. In the class diagram, the attributes and operations of each class, and the relationships between the classes are described. Class Main is related to the Person and Group classes by *association* because Main has instance

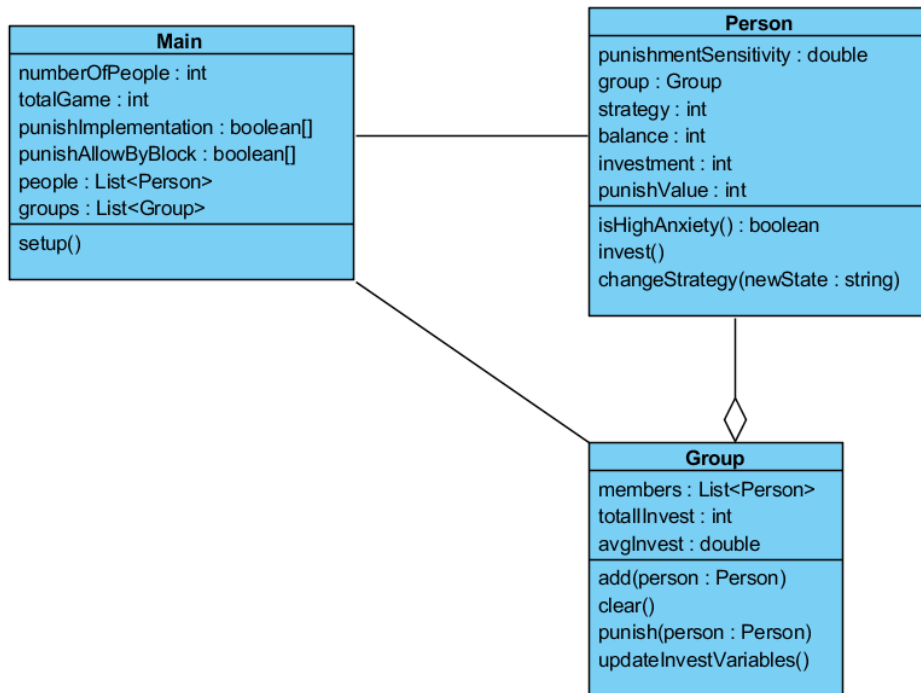


Fig. 5.2 Class diagram

variables of two lists of Persons and Groups. Additionally, Main declares and initializes them. On the other hand, class Person is related to Group by *aggregation*. A Group contains four Persons, and a Person can only belong to one Group.

For every Person, there is a variable *punishmentSensitivity*, which represents the punishment sensitivity value measured by the BIS-anxiety score, a subscale of BIS (Skatova and Ferguson, 2013). *punishmentSensitivity* ranges from 1 to 4. Based on Skatova and Ferguson's work, there are differences in behaviour of people with high and low punishment sensitivity (High sensitivity if their BIS-anxiety score is greater than 3.13). Accordingly, we categorize Person agents based on their *punishmentSensitivity*. Agents are categorized as "high-anxiety" if *punishmentSensitivity* of a Person agent is greater than 3.13, and "low-anxiety otherwise. People with differences in anxiety have the tendency to use different strategies. There are five available strategies for Person agents:

1. Full Cooperation (FC): always contribute 20 MUs.
2. Strong Conditional Cooperation (SCC): contribute 3-4 MUs more than average group investment in previous round.
3. Normal Conditional Cooperation (NCC): contribute the same or difference of 1 MU with average group investment in previous round.

Table 5.3 Person agent initialization

Percentage of agents	Strategy	Anxiety
5%	FC	100% High
20%	SCC	80% High, 20% Low
50%	NCC	50% High, 50% Low
15%	WCC	20% High, 80% Low
10%	FD	100% Low

4. Weak Conditional Cooperation (WCC): contribute 3-4 MUs less than average group investment in previous round.
5. Full Defection (FD): always contribute 0 MU.

High-anxiety agents tends to contribute more; while low-anxiety agents tends to contribute less. Therefore, at the beginning, each Person agent is assigned with a strategy and a *punishmentSensitivity* value following the distribution in Table 5.3. This table is created based on the assumption that high-anxiety participants play more cooperative strategies.

5.4.2 Behavioural design: gameplay

For the gameplay, the Main artifact acts as a mediator, controls the game stages, and coordinates Person agents and Group artifacts. In terms of internal data, Main has variables for total number of rounds, punishment condition of each round, and index of the current round. In term of behaviour, Main controls the game stages using a statechart (Figure 5.3a). The game has several stages including invest, payoff, and punish. Each stage of the game is represented by a state in the statechart of the Main artifact. Based on the current state of the statechart, the Main artifact sends messages to all Person agents to inform them of the current stage of the game, and its condition. The Person agents also have a statechart for the gameplay, as shown in Figure 5.3b. Based on the received message from Main, the Person agents also make a transition to the corresponding state.

In order to better represent the collaboration between actors, Figure 5.4 displays a sequence diagram, which describes the messages and interaction between three actors within one round of the PGG. There are three lifelines: Main on the left; Person in the middle, Group on the right. At each stage of the game, Main sends messages to the Person agents. There are three occasions where Main sends a message: Invest, Receive Payoff, Punish. For each message sent to a Person, Person performs actions then sends a reply message to inform Main that the required task has been done. For Receiving Payoff task, Person needs to performs actions and then sends a message to the Group to ask for the payoff value. For punish messages, there is a guard,

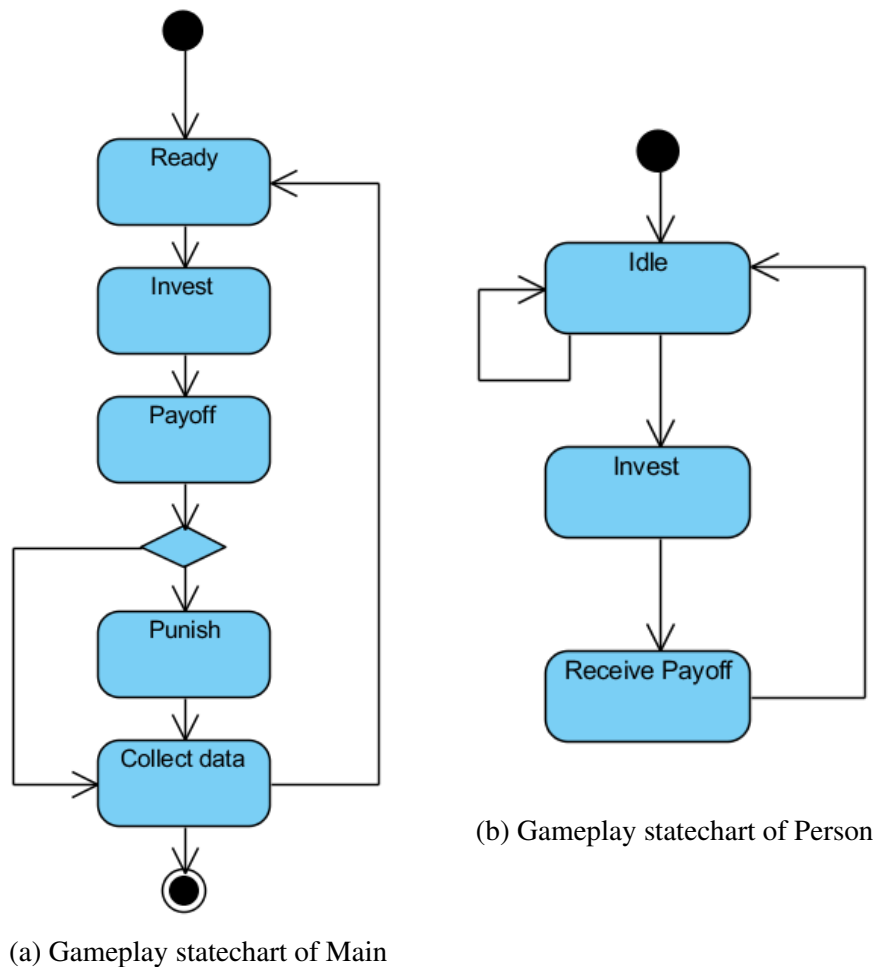


Fig. 5.3 Gameplay statecharts of a Main artifact and a Person agent

which is a condition that must be met before a message is sent. The guard is that punishment is allowed during that round of the game for the message to be sent. When receiving a punish message, the Person agent sends a message to the Group to ask whether it is punished. Upon receiving the message, the Group replies to the sender with a message containing the amount of punished MUs.

Combining the statecharts and the sequence diagram, the interaction in each round of a game is described in the following. At the beginning of every round, when in “Ready” state, the Main artifact prepares for the game by setting up variables, shuffling Person agents, and assigning them to groups. Then the Main artifact changes state to “Invest”, in which messages are sent to all Person agents. Person agents who are in “Idle” state would change to “Invest” state when they receive the message, and make a decision on how much to invest based on their strategy. After all Person agents make investment decisions, the Main artifact goes to “Payoff” state and sends messages to all Person agents. The Person agents change to “ReceivePayoff” and ask the Group artifact to calculate the payoff and the average investment of the group. After all Person agents inform the Main artifact that they have received payoff, Main sends messages to all Person agents so that they go to “Idle” state. In the Main artifact, if punishment is implemented during that game round, Main goes to “Punish” state, and sends messages to Person agents. There is a self-transition in “Idle” state of Person agents, which is triggered when they receive message about punishment. When triggered, Person agents ask their Group artifact whether they got punished. If punishment is not implemented, Main artifact changes state from “Payoff” to “CollectData”, and then goes back to “Ready” state. If this was the last round of the PGG, Main stops the simulation. Otherwise, the next round is played.

5.4.3 Behavioural design: individual differences in punishment sensitivity

A representation of punishment sensitivity is implemented in each Person agent to represent individual differences related to their BIS-anxiety value. People with higher punishment sensitivity would be more cautious and avoid free riding behaviour in response to signals of punishment. In addition, people are only cautious for a period of time, and then the effect of punishment deteriorates over time. Therefore, the agents are contributing more when there is a threat of punishment, and only being cautious for several rounds after. The behaviours are modelled with a PunishmentSensitivity statechart shown in Figure 5.5, which has two states: “Normal” and “Cautious”. The state change is controlled by two transitions:

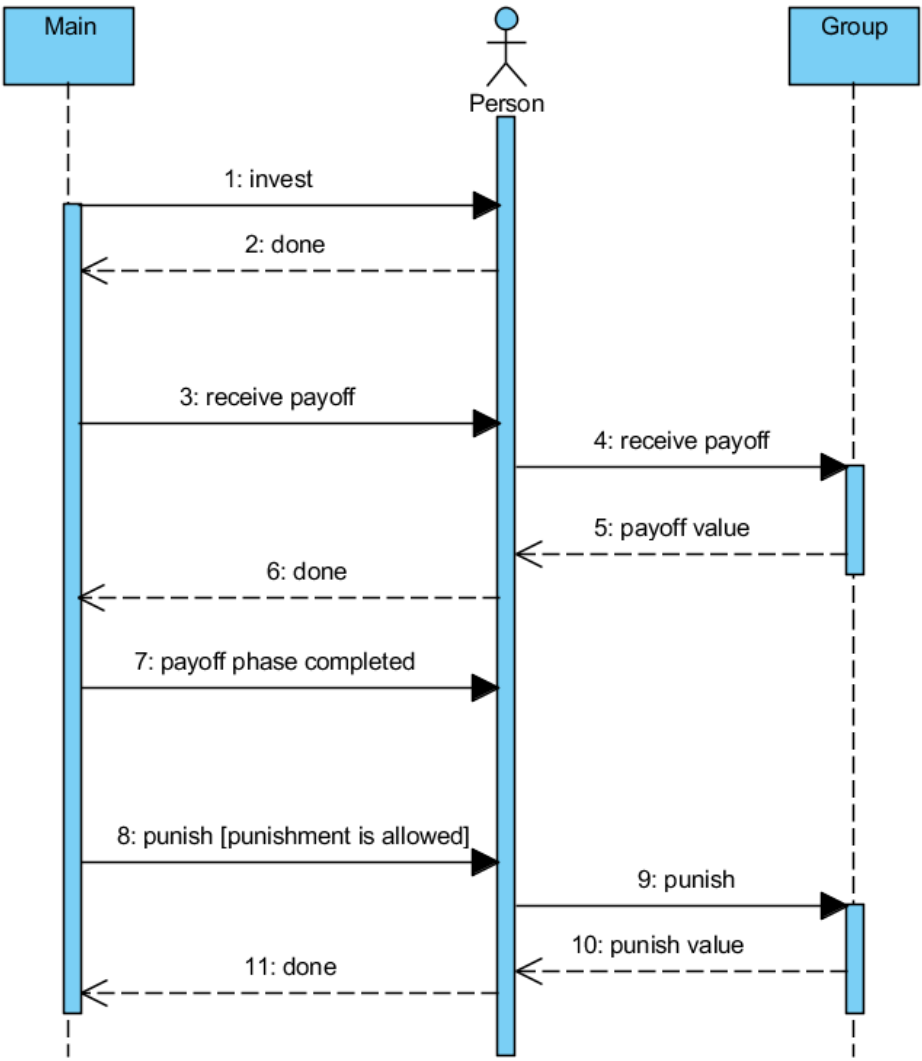


Fig. 5.4 Sequence diagram

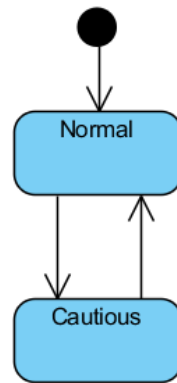


Fig. 5.5 PunishmentSensitivity statechart of Person agent

- From “Normal” to “Cautious”: This transition is triggered if (the agent is high-anxiety AND there is threat of punishment AND the agent has not been in cautious state) OR (the agent has just been punished).
- From “Cautious” to “Normal”: This transition is triggered if (there was no threat of punishment) OR (a low-anxiety agent has been in “Cautious” state for 3 rounds) OR (a high-anxiety agent has been in “Cautious” state for 10 rounds).

When a Person agent changes between Normal and Cautious state, the strategy of that agent would be changed as well. Agents who initially use SCC, NCC, WCC are called Conditional Cooperators. Agents with FC and FD strategies are called Full Cooperators and Full Defectors. To model the change between strategies, we use the probability of transitioning from one strategy to another. For example, when a condition cooperator (using NCC) with high anxiety is punished, s/he will be cautious and there is 20% probability of changing from NCC to SCC strategy (more cooperative strategies). Because we don’t have enough data, the design of the strategy change is assumed based on the following principles: when one person is cautious, s/he changes to a more cooperative strategy; when one is back to normal, s/he changes back to a less cooperative strategy.

For the agents who have conditional cooperation strategies (SCC, NCC and WCC), the strategy change is as described in Figure 5.6. When changing to “Cautious” state, agents would avoid free riding and stop using WCC strategy. Because agents with high anxiety contribute more under threat of punishment, when agents change to “Cautious” state, high-anxiety agents are more likely to change to SCC than low-anxiety ones. For example, in Figure 5.6a, 80% of high-anxiety agents using WCC strategy change to NCC strategy, and the rest (20%) change to SCC strategy. When agents change to the “Normal” state, low-anxiety agents are more likely to use WCC strategy than the high-anxiety ones.

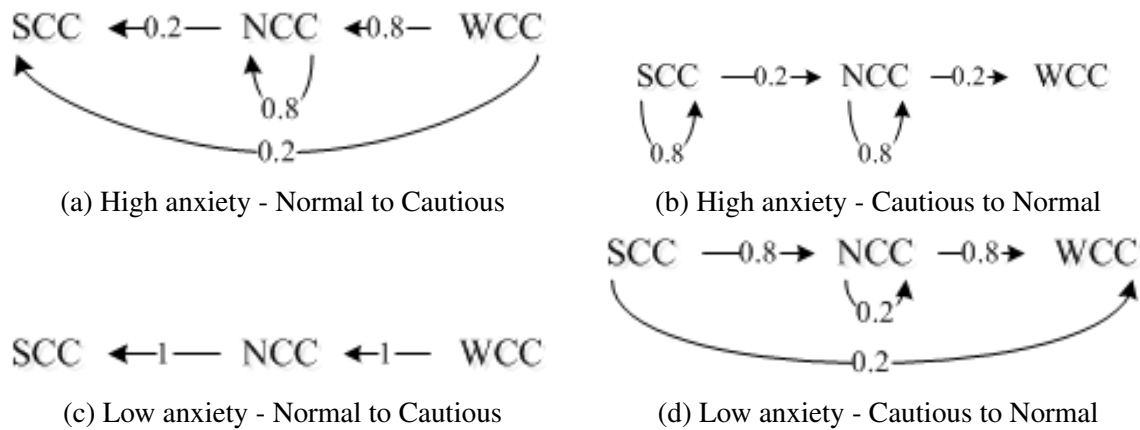


Fig. 5.6 Strategy change of Conditional Cooperators

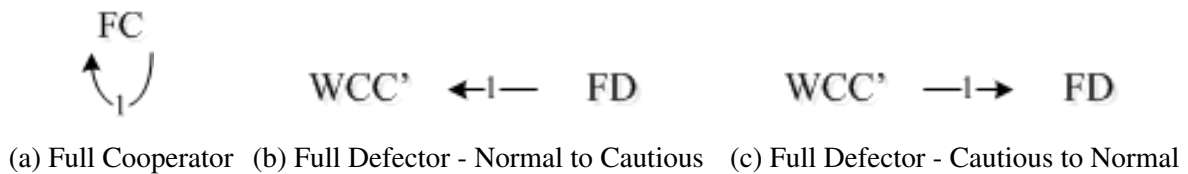


Fig. 5.7 Strategy change of Full Cooperator and Full Defector

The strategy change of Full Cooperators and Full Defectors is illustrated in Figure 5.7. The full cooperators do not change their strategy. The agents that use FD strategy would change to WCC strategy when changing to “Cautious” state, and change back to FD strategy when changing to “Normal” state. WCC’ is the same strategy as WCC. The reason for a different name is that WCC’ is a dummy-state that supports the full defector to switch back to FD without being affected by the strategy change of conditional cooperators.

5.5 Implementation

The agent-based model was implemented in AnyLogic 7, a multi-method simulation modelling tool. In this section, the implementation is split into structural and behavioural subsections.

5.5.1 Structural implementation

The three actors were implemented as follows: Person and Main are implemented using Agent Type; Group is implemented as a Java class. In AnyLogic, Main screen is the screen that the user will see first when running the simulation. Figure 5.8 shows a screenshot of Main screen in the implementation. On the top is a list of parameters and variables. The statechart of Main

is on the right side. There are some graphical representations of outputs including a histogram, time series plots, and time stacked charts. The screenshot shows many variables which are not covered in the class diagram (Figure 5.2) in the Design section. These variables were added as needed during implementation. This is expected because the development process in the framework is iterative and incremental. As the variables were added during implementation, the developer went back and updated the class diagram. This is a good practice to keep an up-to-date documentation of the simulation.

5.5.2 Behavioural implementation

When Main starts up, it initializes a population of Person agents and their properties, and creates a list of Group objects. For gameplay in the simulation, gameplay statecharts of Main and Person coordinated the actions between Main and Person. Figure 5.9 shows three statecharts in the implementation. Based on the description in the Design process, the transitions and their trigger conditions are defined in Table 5.4. After entering Ready state, Main initializes variables, and collects statistics before the round begins. Main also keep track of the number of played rounds by using the gameNo variable. If all rounds are played, the simulation is stopped. Next, after Main enters one of Invest, Payoff, or Punish states, it sends messages to all Person agents by using this command `deliverToAllAgentsInside (message);`. Main also sets `countActionPerGame` variable to 0, then uses this variable to count how many agents have completed their action. When the count is equal to the total number of agents, Main moves to the next state. In `CollectData` state, Main collects statistics such as total investment of this round or this block, and updates the plot. The details on gameplay behaviour of Person and Group are described in Section 5.4.2.

Moreover, because at any one time agents can only use one strategy out of the five predefined strategies, the strategies and strategy change can be implemented using a statechart. Each strategy can be represented by a state. The trigger of transition needs to be probability-based (following the design in Section 5.4.3). AnyLogic supports probabilistic-based transitions; however, they need to be carefully designed to express the desired strategy change. For example, there is an agent with three states A, B and C as in Figure 5.10a. Initially, the agent is in state A. There are two transitions from A to B and from A to C with probabilities of 0.8 and 0.2. The expected behaviour is that there is 80% chance it will change to B and 20% chance the new state is C. However, if we implement the AnyLogic statechart as in Figure 5.10b with the conditions of the two transition `randomTrue(0.8)` and `randomTrue(0.2)`, there is a chance that it does not change to either B or C. The reason for this is that the probabilities in the two transitions are calculated separately. If one condition of two transitions gets the value of true while the other gets the value of false, the transition with the value of true will be triggered. If

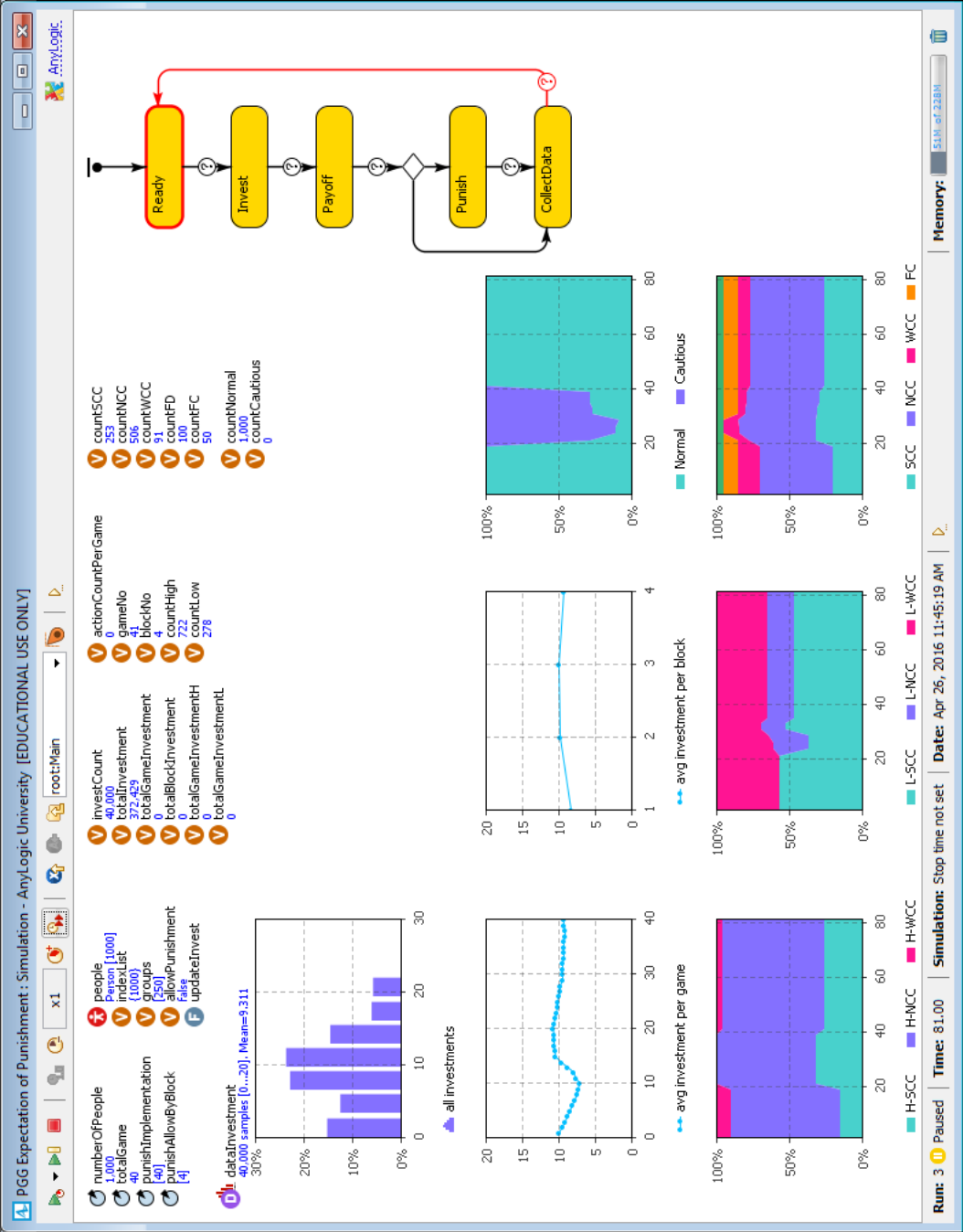


Fig. 5.8 An AnyLogic screenshot of the simulation of case study 2

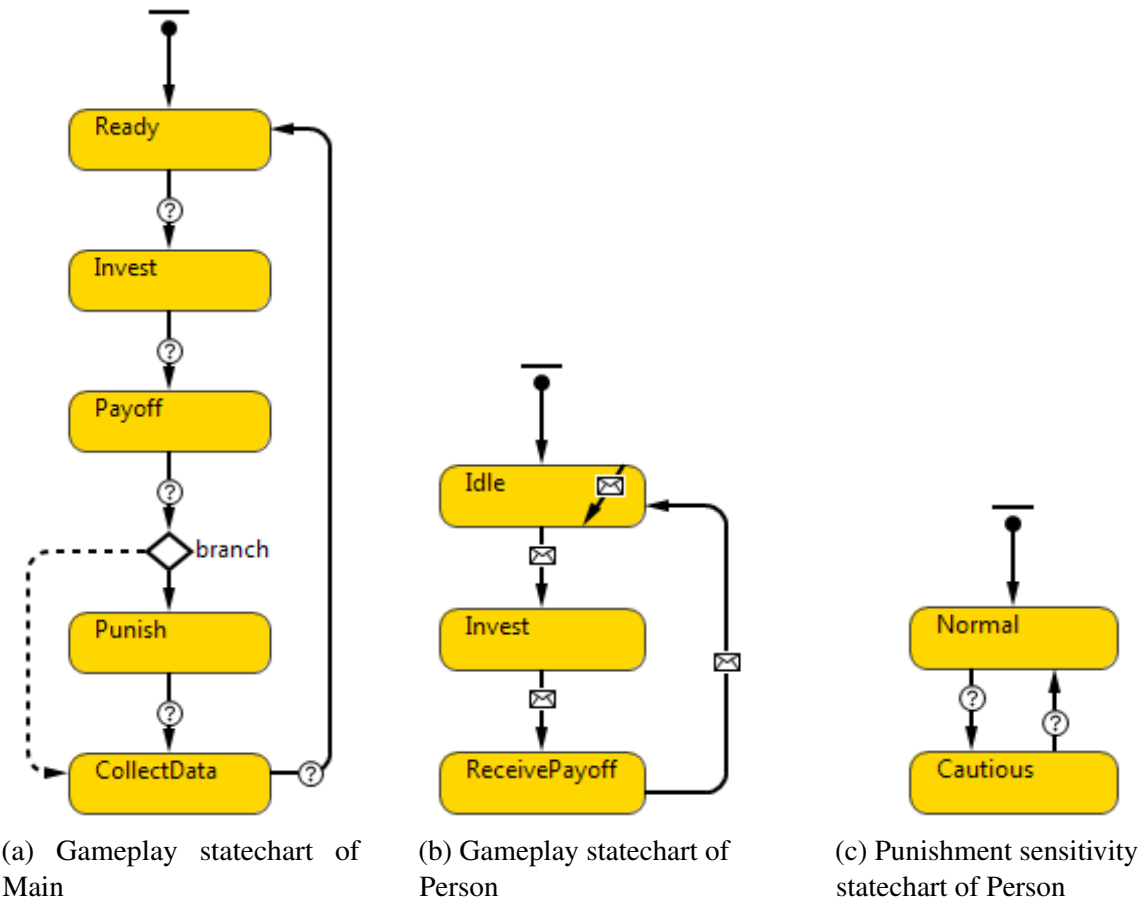


Fig. 5.9 Implementation of statecharts

Table 5.4 Transitions in statecharts of Main and Person

Agent	From state	To state	Triggered by	Condition
Main	Ready	Invest	Condition	gameNo >= 1 && gameNo <= totalGame
	Invest	Payoff	Condition	actionCountPerGame == people.size()
	Payoff	branch	Condition	actionCountPerGame == people.size()
	branch	Punish	Condition	punishImplementation [gameNo-1]
	branch	CollectData	Default	is taken if all other conditions are false.
	Punish	CollectData	Condition	actionCountPerGame == people.size()
	CollectData	Ready	Condition	true
Person	Idle	Invest	Message	"invest"
	Invest	ReceivePayoff	Message	"payoff"
	ReceivePayoff	Idle	Message	"payoffDone"
	Idle	Idle	Message	"punish"
	Normal	Cautious	Condition	(isHighAnxiety() && get_Main().allowPunishment && cautiousRoundCount==0) (tempPunishCount>0)
	Cautious	Normal	Condition	(!get_Main().allowPunishment) (!isHighAnxiety() && cautiousRoundCount==3) (isHighAnxiety() && cautiousRoundCount==10)

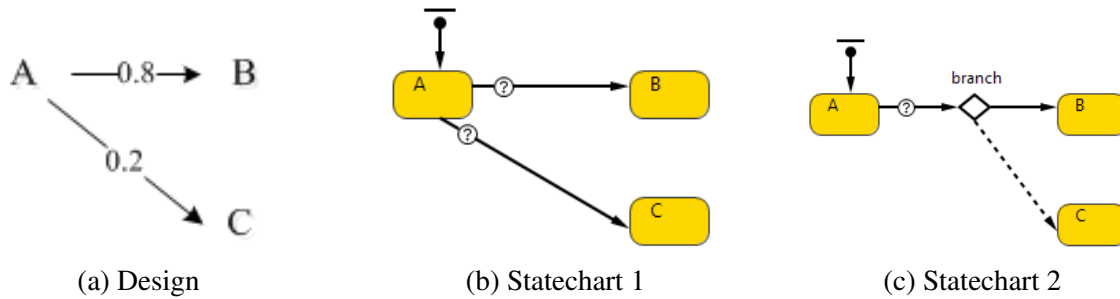


Fig. 5.10 An example of statechart with probability transitions

both of them get the value of true, the triggered transition will be chosen randomly. And there is a chance that both of them get the value of false, which means neither of the two transitions is triggered, and the agent remains in state A. This is not the desired behaviour.

There are two solutions for this problem. The first solution is using one variable for the probabilities of all out-going transitions of each state. The variable can be used in a Roulette wheel fashion to choose which transition will be triggered. For instance, a variable p is randomized from 0 to 1. Then the transitions are set as follows: if $0 < p < 0.8$, then trigger the transition from A to B. If $0.8 < p < 1$, then trigger the transition from A to C. The second solution is using the decision node supported by AnyLogic statecharts as shown in Figure 5.10c. First, the transition from A to branch is triggered by a message or other events. Then from the branch, there are two transitions: one to state B with condition `randomTrue(0.8)`, and one to state C as a default transition. There is an 80% chance that the state will change to B. If the transition to B is not triggered, the default one will (which has the probability of 20%). Therefore, the state will always change from A to either B or C. For the first solution, one variable is needed for each state to control probability transitions. While for the second solution, branches are used to guide the transitions. We decided to use the second solution for the strategy change of this case study. Figure 5.11 demonstrates the statechart for strategy. The transitions in the strategy statechart are derived from the description in Section 5.4.3, and described in Table 5.5.

In the strategy statechart, there are seven states:

- NoStrategy: a dummy-state during initialization of the strategy.
- FC, SCC, NCC, WCC, FD: states are for the five strategies.
- WWC2: a dummy-state which supports strategy change of full defectors.

After creation, the Person agent is in NoStrategy state. Then Person receives an initialization message from Main to inform its strategy, and changes to either one of five strategies states. Full cooperators do not change strategy so there is no outgoing transition. When the punishment

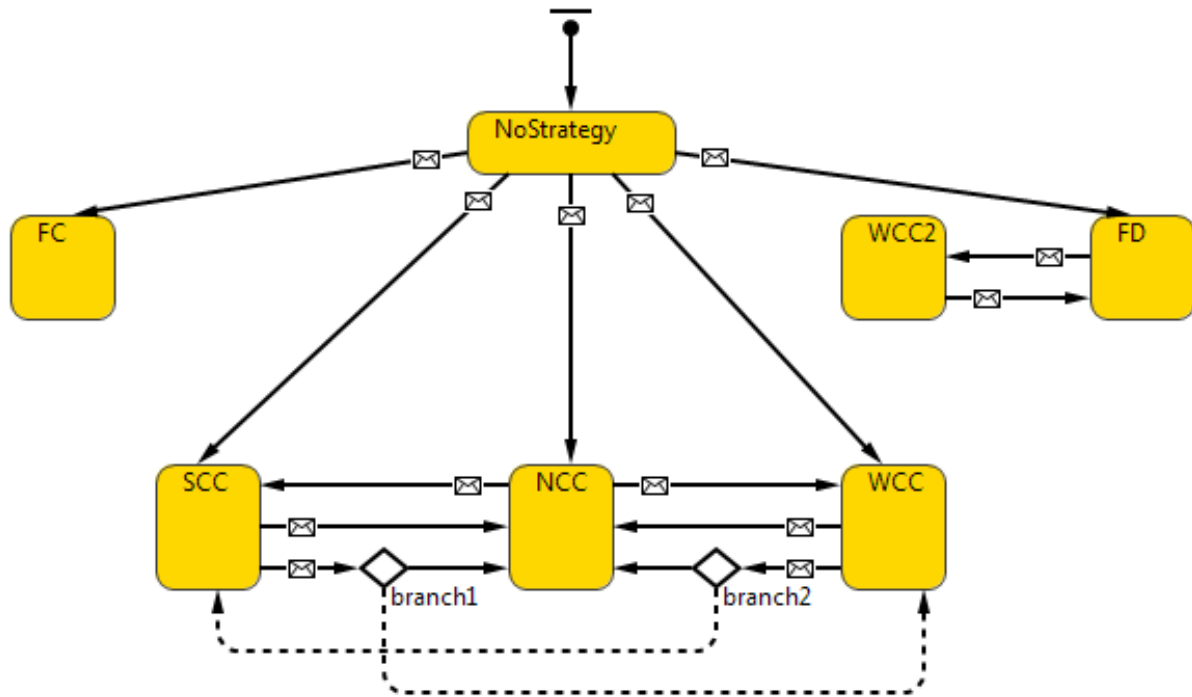


Fig. 5.11 Strategy statechart

sensitivity statechart (Figure 5.9c) changes to a new state, it sends a message to strategy statechart, and a strategy change might be triggered. The message contained a String of either "normal" or "cautious" to inform the new state of punishment sensitivity statechart. Although transitions receive this message, there are still conditions, such as level of anxiety and probability, that need to be satisfied before transitions can be triggered. Therefore, there are *guard* constraints, which will be evaluated before triggering a transition.

Even though strategy change is implemented using a statechart, there is another approach in representing strategy and its change. This approach, which uses a strategy variable and conditional statements, will be discussed further in the next subsection.

5.5.3 Alternative implementation of punishment sensitivity

For implementation of the strategy of Person agents, a variable `strategy` can be used to represent the current strategy an agent is using. The variable `strategy` encodes categorical information by having one of five possible values in corresponding to five strategies: FC, SCC, NCC, WCC, and FD. But in order to represent strategy WCC' of a full defector, we need to add other possible values; thus, there are six possible values for five strategies. This can be implemented by setting `strategy` as an integer, and encoding the strategies as follows: let FC=1, SCC=2, NCC=3, WCC=4, WCC2=5, FD=6. However, there are several problems with

Table 5.5 Transitions in Person's strategy statechart

From state	To state	Triggered by	Condition	Guard
NoStrategy	FC	Message	"FC"	
NoStrategy	SCC	Message	"SCC"	
NoStrategy	NCC	Message	"NCC"	
NoStrategy	WCC	Message	"WCC"	
NoStrategy	FD	Message	"FD"	
WCC2	FD	Message	"normal"	
FD	WCC2	Message	"cautious"	
SCC	NCC	Message	"normal"	isHighAnxiety() && randomTrue(0.2)
NCC	WCC	Message	"normal"	(isHighAnxiety() && randomTrue(0.2)) (!isHighAnxiety() && randomTrue(0.8))
SCC	branch1	Message	"normal"	!isHighAnxiety()
branch1	NCC	Condition	!isHighAnxiety() && randomTrue(0.8)	
branch1	WCC	Default	if all other conditions are false	
WCC	NCC	Message	"cautious"	!isHighAnxiety()
NCC	SCC	Message	"cautious"	(!isHighAnxiety()) (isHighAnxiety() && randomTrue(0.2))
WCC	branch2	Message	"cautious"	isHighAnxiety()
branch2	NCC	Condition	isHighAnxiety() && randomTrue(0.2)	
branch2	SCC	Default	if all other conditions are false	

this approach. It is easy to make a mistake between the value of one strategy and another. Or one can set a value that is not in the intended encoding. A solution to this challenge is by using enumeration, which is supported by Java via *enum* types. An enum type enables a way to define a set of predefined constants. And an enum type variable must be equal to one of the values that have been set in the enum type. The following code defines Strategy as enum type: `enum Strategy {FC, SCC, NCC, WCC, FD, WCC2};`. Then a `strategy` variable is declared with Strategy type. With the `strategy` variable, a Person agent chooses an amount of MUs to invest based on the current strategy with the pseudocode in Pseudocode 5.1.

Pseudocode 5.1: Investment decision

```

switch strategy do
  case SCC
    investment  $\leftarrow$  roundToInt(groupAvgInvest + uniformDiscrete(3,4));
    if investment > 20 then investment = 20;
    break;
  case NCC
    investment  $\leftarrow$  roundToInt(groupAvgInvest + uniformDiscrete(-1,+1));
    if investment > 20 then investment = 20;
    if investment < 0 then investment = 0;
    break;
  case WCC
  case WCC2
    investment  $\leftarrow$  roundToInt(groupAvgInvest - uniformDiscrete(3,4));
    if investment < 0 then investment = 0;
    break;
  case FD
    investment  $\leftarrow$  0;
    break;
  case FC
    investment  $\leftarrow$  20;
    break;

```

In order to implement the strategy change, nested if statements are used to decide which strategy to change to, based on anxiety level, the current strategy, and the new state of the punishment sensitivity statechart. Pseudocode 5.2 is the pseudocode from `changeStrategy()` function, which is called when a transition in punishment sensitivity statechart is triggered.

- `isHighAnxiety()` function returns `true` if a Person is high-anxiety, `false` if the Person is low-anxiety.
- `newState` variable is given as a parameter of `changeStrategy()` function.

- `strategy` is a variable for the current strategy.
- `randomTrue(double p)` generates `true` value with the given probability `p`.

5.6 Experiments and Results

After developing the simulation, a validation experiment was performed to compare our simulation results with those of [Skatova and Ferguson \(2013\)](#). In addition, we explored situations which have not been explored in the lab experiments. In these additional experiments, we focus on three different factors: punishment conditions, ratio of strategy, and ratio of anxiety.

5.6.1 Validation experiment

In this experiment, the model was set up using the settings of the experiment described in [Skatova and Ferguson \(2013\)](#) to validate the simulation results and examine whether the model replicates the contribution level over different blocks as well as the correlation between punishment sensitivity and free riding behaviour of a real experiment. The simulation was set up with 1000 Person agents, initialized with different punishment sensitivity values and strategy based on Table 5.3. Person agents played four blocks (10 rounds each block):

- A non-punishment block.
- An implemented punishment block.
- A non-implemented punishment block.
- A non-punishment block.

Only the Main artifact knows about the punishment conditions of the blocks. The Main artifact just informs Person agents that there would be a threat of punishment in the second and third block. In the implemented punishment block, punishment was implemented in *two out of ten* rounds. In order to observe the effect of punishment in an implemented punishment block (block 2), we fixed the two punishment rounds to be round 12 and 14 (the same in the paper).

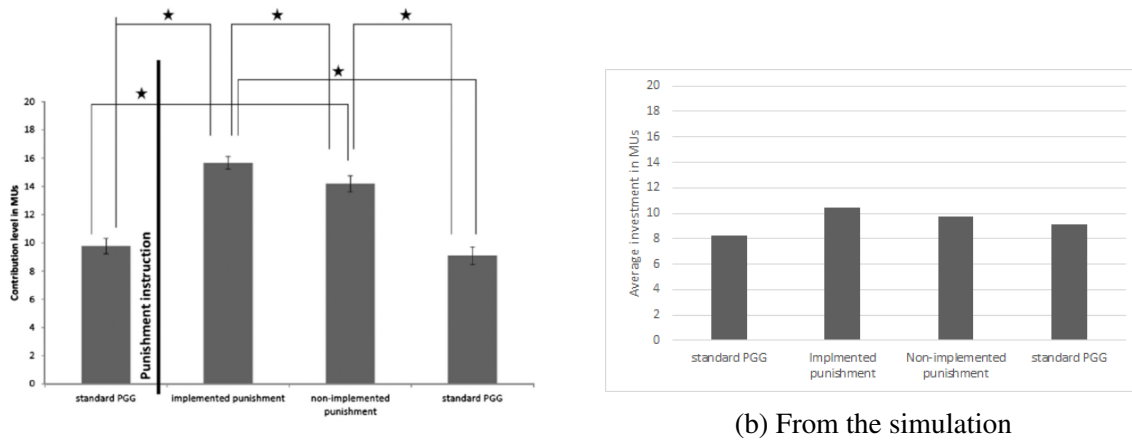
The average group investments over four blocks, shown in Figure 5.12, replicates the trend in the [Skatova and Ferguson](#) laboratory experiment. The average investment in the first block of standard PGG was 8.25. In the second block where punishment was implemented in two out of ten rounds, the group investment increased to 9.77. In the third block where the punishment was expected but not implemented, the group investment decreased to 8.64. In the last block of standard PGG, the group investment dropped to 7.34.

Pseudocode 5.2: Strategy change

```

if isHighAnxiety() then
  if newState = Cautious then
    if strategy = WCC then
      if randomTrue(0.8) then
        strategy  $\leftarrow$  NCC;
      else
        strategy  $\leftarrow$  SCC;
    else if strategy = NCC then
      if randomTrue(0.2) then strategy  $\leftarrow$  SCC;
  else if newState = Normal then
    if strategy = SCC then
      if randomTrue(0.2) then strategy  $\leftarrow$  NCC;
    else if strategy = NCC then
      if randomTrue(0.5) then strategy  $\leftarrow$  WCC;
  else
    if newState = Cautious then
      if strategy = WCC then
        strategy  $\leftarrow$  NCC;
      else if strategy = NCC then
        strategy  $\leftarrow$  SCC;
      else if strategy = FD then
        strategy  $\leftarrow$  WCC2;
    else if newState is Normal then
      if strategy = SCC then
        if randomTrue(0.8) then
          strategy  $\leftarrow$  NCC;
        else
          strategy  $\leftarrow$  WCC;
      else if strategy = NCC then
        if randomTrue(0.8) then strategy  $\leftarrow$  WCC;
      else if strategy == WCC2 then
        strategy  $\leftarrow$  FD;

```



(a) From the laboratory experiment (Skatova and Ferguson, 2013)

Fig. 5.12 Average investment over four blocks

Because in the second block of this particular experiment, the punishment was implemented at rounds 12 and 14, Figure 5.13 shows that the average investment has a sharp rise after rounds 12 and 14. In those two rounds, it appeared that free riding agents, especially low-anxiety agents, got punished, switched to more cooperative strategies, and contributed more in the next round. If this is a lab experiment, we would not be able to investigate much further. But since we use an agent-based model, we can analyse the decision making process of agents better.

Figure 5.14 shows the states of punishment sensitivity statechart of all agents. In the first 10 rounds, all agents were in Normal state. When there was a threat of punishment, high-anxiety agents changed to Cautious state. In rounds 12 and 14, when the punishment was implemented, free riding agents got punished and changed to Cautious state. The more agents were in Cautious state, the more contribution there was overall. The strategy change of conditional cooperators, who are the majority, plays a crucial role in the contribution in the system. Figure 5.15 shows the strategy change of high and low anxiety agents. High-anxiety agents changed to a more cooperative strategy when there was a threat of punishment then changed the strategy again after being cautious for 10 rounds. Low-anxiety agents still used the same strategy even when there was a threat of punishment. They only changed when punishment was implemented (rounds 12 and 14): the more agents used SCC, the less agents used WCC. After 3 rounds of being cautious, low-anxiety agents changed back to Normal state, and used less cooperative strategies.

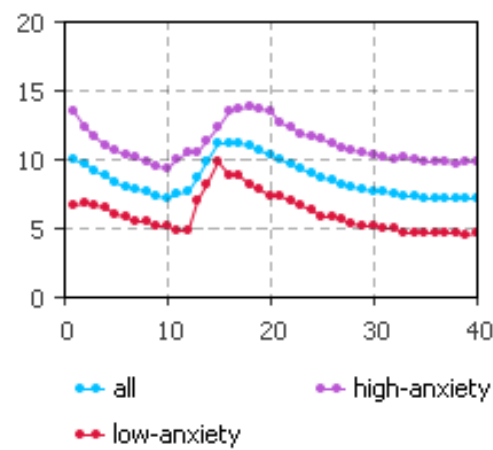


Fig. 5.13 Average investment over 40 rounds

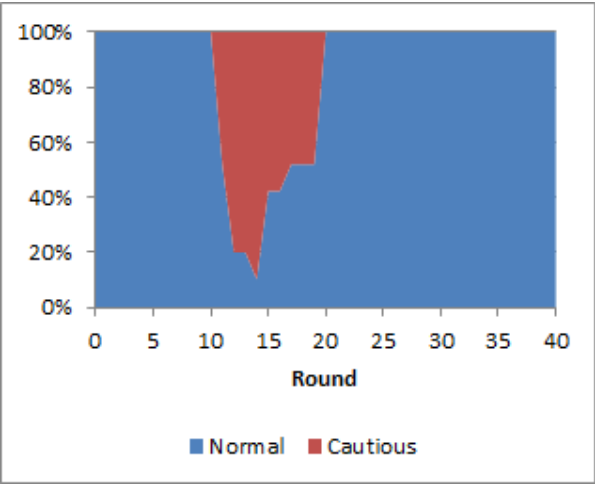


Fig. 5.14 Agent states of punishment sensitivity statechart

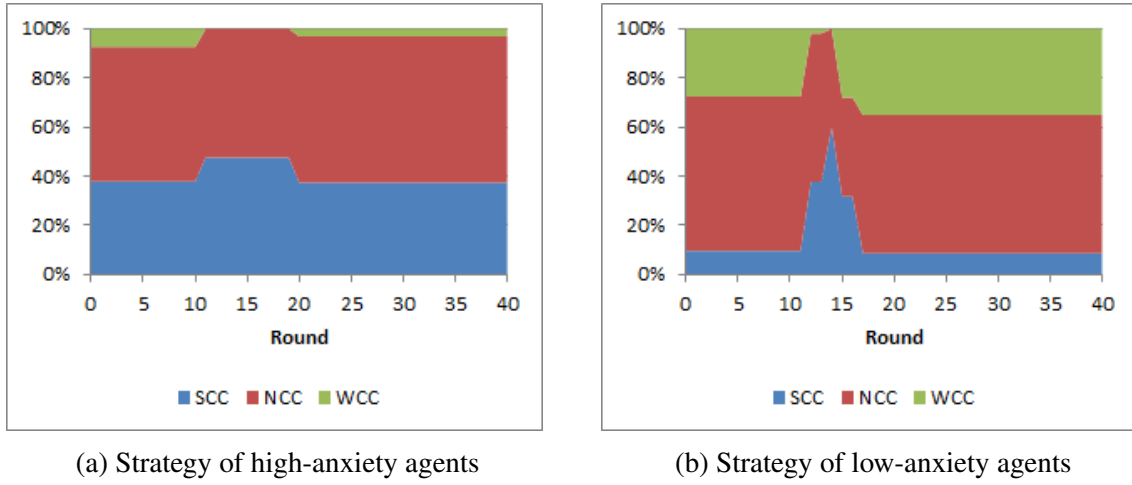


Fig. 5.15 Strategy of agents over time

5.6.2 Experiments with different punishment conditions

One of the applications of this model is to use it for examination of the contribution levels under different levels of punishment. In this experiment, the model was set up with the same ratio of agents but under different punishment conditions. We then analysed the simulation results to understand more about free riding behaviours in various punishment conditions.

Figure 5.16 shows an experiment where in every round, punishment was expected but only implemented periodically. If the punishment was implemented every round (red line), the contribution level increased gradually before stabilizing. The same trend occurred for implemented punishment every 3 rounds (green line), but the contribution level was lower. If the punishment was implemented every 5, 10 or 15 rounds, the contribution levels oscillated, which means contributions decreased over time and only increased in rounds with implemented punishment. The greater the period between two implemented punishment rounds was, the lower the contribution level was.

Figures 5.17 and 5.18 show another experiment in which the *implemented punishment blocks* occurred periodically in a series of blocks of 10 rounds. In the implemented punishment blocks of Figure 5.17, randomly on two out of ten rounds individuals were punished. While in Figure 5.18, randomly on five out of ten rounds punishment was implemented. The contribution level decreased and for a certain number of rounds became stable, to only increase when punishment was implemented. Comparing Figures 5.17 and 5.18, the contribution level (overall as well as the peaks) was higher in Figure 5.18. This was because there were more rounds where punishment was implemented in Figure 5.18 than in Figure 5.17.

The above experiments show that contribution levels had different dynamics in different punishment conditions. Using the agent-based models as a decision support tool, policy

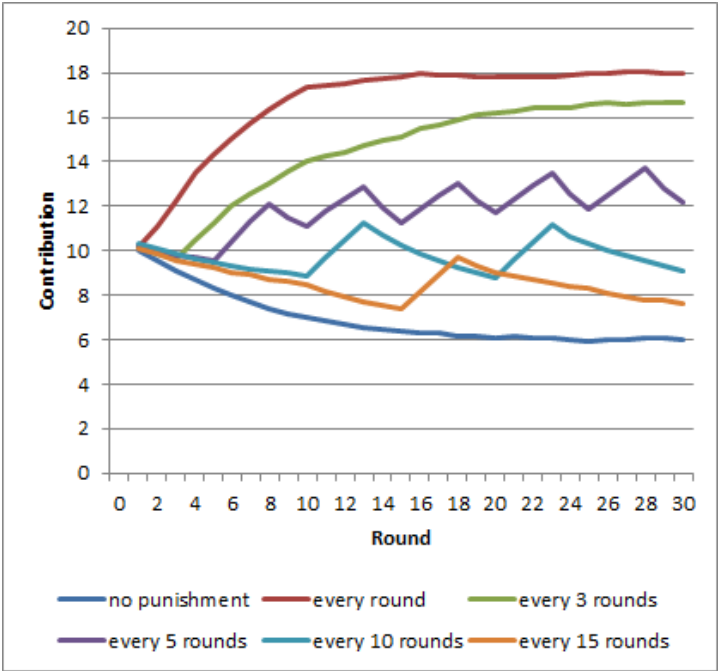


Fig. 5.16 Punishment was periodically implemented on rounds in an expected punishment condition

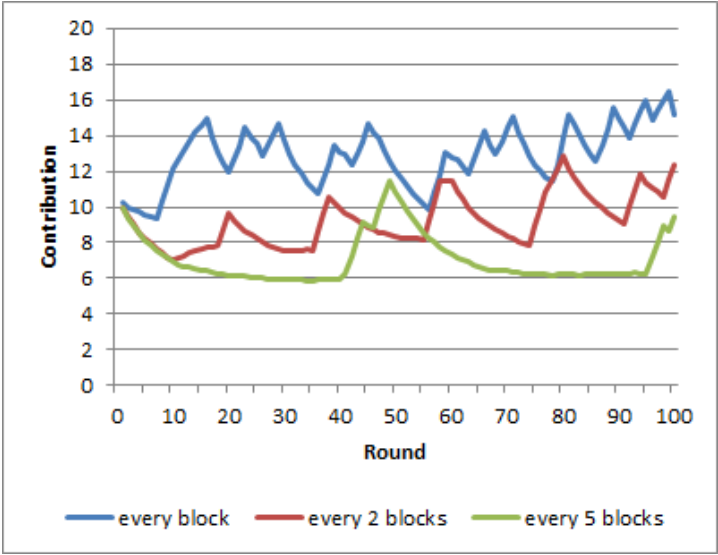


Fig. 5.17 Implemented punishment blocks occurred periodically with randomly *two out of ten* punishment games per block

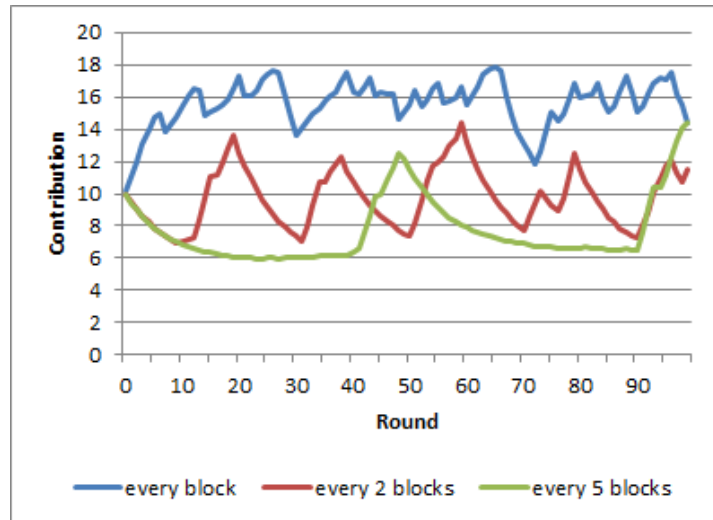


Fig. 5.18 Implemented punishment blocks occurred periodically with randomly *five out of ten* punishment games per block

makers can decide on how to implement punishment in order to achieve a desired pattern of contributions in the real world public good scenario.

5.6.3 Experiments with different ratios of strategies

We investigated how the ratios of strategies used by agents affect the total investment of four blocks. Figure 5.19 shows the results of six experiments. The total percentages of five strategies (FC, SCC, NCC, WCC, FD) are always 100%. In all six experiments, the percentage of FC was fixed at 5%. There were three experiments with 10% of FD (solid line), and three experiments with 30% of FD (dotted line). So in each of the six experiments, the percentages of FC, FD, and SCC were fixed, then percentage of NCC was varied from 0% to the maximum possible percentage, and the rest would be percentage of WCC. The percentage of NCC was represented by the x-axis, while the total investment was represented by the y-axis. For example, the red solid line had 5% FC, 10% FD, 20% SCC; and as the percentage of NCC increased, the total investment increased as well.

How the ratios of strategy affect the total investment can be concluded by comparing these experiments:

- Looking at one experiment, we noted the larger the percentage of NCC was, the more investment there was into PG.
- Comparing the blue, red, green lines, we saw that the larger the percentage of SCC was, the more investment there was into PG.

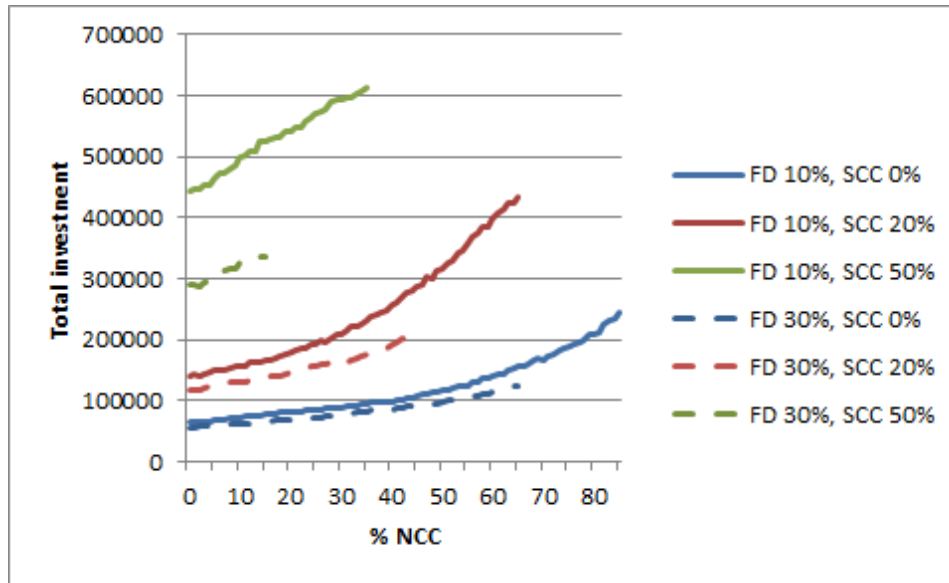


Fig. 5.19 Total investment in different ratios of strategy

Table 5.6 Experiments with different ratios of anxiety

		Anxiety		
		Experiment 1	Experiment 2	Experiment 3
Strategy	FC	100% High	100% High	100% High
	SCC	80% High, 20% Low	80% High, 20% Low	20% High, 80% Low
	NCC	50% High, 50% Low	80% High, 20% Low	20% High, 80% Low
	WCC	20% High, 80% Low	80% High, 20% Low	20% High, 80% Low
	FD	100% Low	100% Low	100% Low

- Comparing the solid line (FD 10%) and dotted line (FD 30%), the smaller the percentage FD was, the more investment there was into PG.
- With FD of 10%, the increase in percentage of SCC (from 0% to 20% to 50%) resulted in a bigger rise of total investment compared with FD of 30%.

5.6.4 Experiments with different ratios of anxiety

The final experiments were to change the ratios of anxiety and examined the investment over time. Using the validation experiment as the baseline, two more experiments were set up by changing the ratios of high and low anxiety of conditional cooperators (CSS, NCC, WCC). In the second experiment, 80% of conditional cooperators were high anxiety and 20% were low anxiety. In the third experiment, 20% of conditional cooperators were high anxiety and 80%

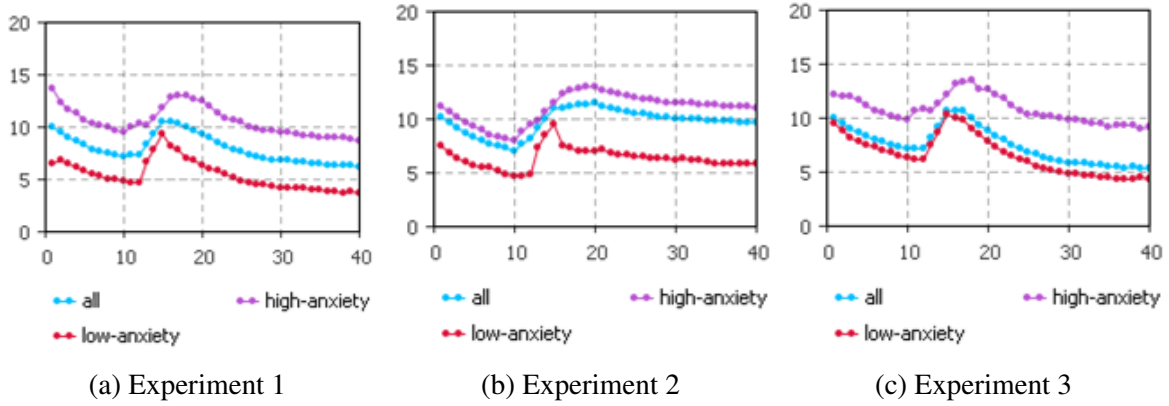


Fig. 5.20 Investment over time in different ratios of anxiety

were low anxiety. Table 5.6 shows the three experiments and corresponding percentages of anxiety levels.

The results of the three experiments are shown in Figure 5.20. In the first block (first 10 rounds) the investment trend was the same for the three experiments. This is because the investment in the standard PGG is only affected by the ratio of strategies used by agents, not by their anxiety. For the last three blocks:

- In the second experiment, because there were more high-anxiety agents, the investment was increasing faster than the first experiment, and became stable at a higher value.
- In the third experiment, because there were more low-anxiety agents, the investment was increasing to approximately the same value as the first experiment, but became stable at a lower value.

5.7 Discussion

5.7.1 Simulation and experiments

The validation experiment has shown that the simulation is able to replicate the contribution levels in different punishment policies. We further experiment with different settings to understand more about the system. Some results are expected and can be devised: increased contribution when there is higher frequency of punishment, or when there are more people using cooperative strategies. However, there are experiments that would be very hard or take a long time to perform without the agent-based simulation. For example, in Section 5.6.2, when we compare two punishment policies: (1) punishment is implemented every 5 rounds, and (2)

implemented punishment blocks occurred every block (randomly 2 out of 10 rounds in that block). Even though in both policies, the implemented-punishment rounds are about 20% of total rounds, the second policy (blue line in Figure 5.17) has higher contribution than the first policy (purple line in Figure 5.16). This means that the second policy should be chosen since the same number of punishments is carried out (on average 1 punishment round for every 5 rounds) but the second policy leads to more contribution. These experiment can be carried out easier in ABMS, as long as the model is carefully designed and implemented.

Behavioural Economics has been trying to extend the traditional approach by integrating other sciences (such as psychology and neuroscience) to add more layers into human decision-making models, thus increase the realism of the model. With this case study, the ABOOMS framework extends the agent-based models of PGG in Economics with BIS from Psychology. Modellers can develop agent-based models in which the overall decision making process of agents is affected by the combination of many decision-making factors derived from models in different disciplines. Theoretical models can be validated with the simulation approach demonstrated in this chapter. However, the approach demonstrated with this framework is data-oriented. It requires data and information to parameterize and initialize the simulation. One solution can be conducting a laboratory experiment which is designed to collect the data for the simulation. Another solution is making assumptions and using data from other simulations and research works.

5.7.2 Statechart for Behavioural Modelling

On the micro-behavioural level, there are many ways to design agents' behaviour. For example, rule-based agents have a list of rules which specify the action to perform when certain criteria are met. For simple and static systems, this approach works very well. However, for complex and dynamic systems, designers might miss a scenario when creating all these rules, or an enormous list of rules is required to cover all the scenarios, dynamics, etc. In order to better handle the dynamics, in the ABOOMS framework, statecharts are recommended for agent behaviour modelling. A statechart reduces the logical complexity by providing a compact way to represent a set of complex rules and conditions. Moreover, the state-based solution allows context information to be used locally in a particular state then discarded afterward. Statecharts are more robust at expressing behaviour in a dynamic environment.

As it reduces the logical complexity, for a simple case, it is not required to employ a statechart. For example, even though a boolean variable can be translated into a statechart, it is more efficient to just use a boolean variable. However, if there are many conditions that trigger the change of a true/false value, a statechart can be useful. Nonetheless, a statechart is a design pattern, thus domain experts can use a statechart in the Design process and do not have

to worry about how it is implemented. In term of implementation, there are many approaches. It can be a variable with many if-then-else statements, or a feature supported by a library, or a feature built into the programming languages or simulation softwares. In short, a statechart is recommended but not compulsory.

How do modellers design a statechart? The first step is to list all the elements that affect decision making. Then keep an eye out for a piece of information, a variable that can only take one value from a fixed set of pre-defined values. For example, the strategy of an agent is usually limited to a fixed set of strategies. Another example is the activity a player engages with during the game. Each of these elements can be designed using a statechart. When you decide to design a statechart, you must identify:

- States: What states are there?
- Transitions: Identify all the transitions required between states. From one state to another, there can be many transitions with different triggers.
- Triggered conditions: The conditions that trigger state transition.
- Guard: The guard condition of a state to be checked after triggered.

In summary, a statechart is a powerful tool because:

- A statechart reduces the logical complexity by providing a compact way to represent a set of complex rules and conditions.
- The state-based solution allows context information to be used locally in a particular state then discarded afterward.
- Statecharts are more robust at expressing behaviour in a dynamic environment.

5.8 Framework Revision and the ABOOMS Framework v3

5.8.1 Revision

There is no improvement needed for the development lifecycle and five micro processes. However, in order to make the ABOOMS framework easier to use and extend, some statechart suggestions for agents have been added:

- Action statechart: represent actions that agents can perform. It describes a sequence of actions in PGG. These actions are most likely to have tight relationships to the stages of the game. Agents with different social preferences can have different action statechart designs.

- Strategy statechart: represents strategies of agents in a game. This is suitable for the situation where agents change their strategies and there are many complicated conditions that trigger these. For example, in the second case study, the changes of punishment conditions trigger the change of social preferences.
- Mental statechart: represents mental states of agents. This can be used to represent the mental state of agents such as the punishment sensitivity in the second case study.
- Spatial statechart: represents location of agents. This can be used in a spatial PGG where the decision of agents is affected by their location.

Moreover, in the discussion of the second case study (Section 5.7.2), why, when, and how to use statecharts for agent behavioural modelling has been discussed. Combining this with the statechart suggestion above, a guide on using statecharts for PGG has been created and added into the ABOOMS framework v3.

In summary, the macro and micro process of the framework v3 is the same as the framework v2; and a guide for agent statecharts have been added to support the Design process in the framework v3.

5.8.2 The ABOOMS Framework v3

The macro process

Figure 5.21 is the lifecycle of the ABOOMS framework v3.

The micro process

The steps of five processes used to develop the simulation are described in detail below:

Process 1 - Conceptual modelling

- Understand the problem situation.
- Determine objectives of the simulation study.
- Collect and analyse data and related theories.
- Identify the settings of the PGG.
- Design the conceptual model.
 - Identify the model inputs (experimental factors).

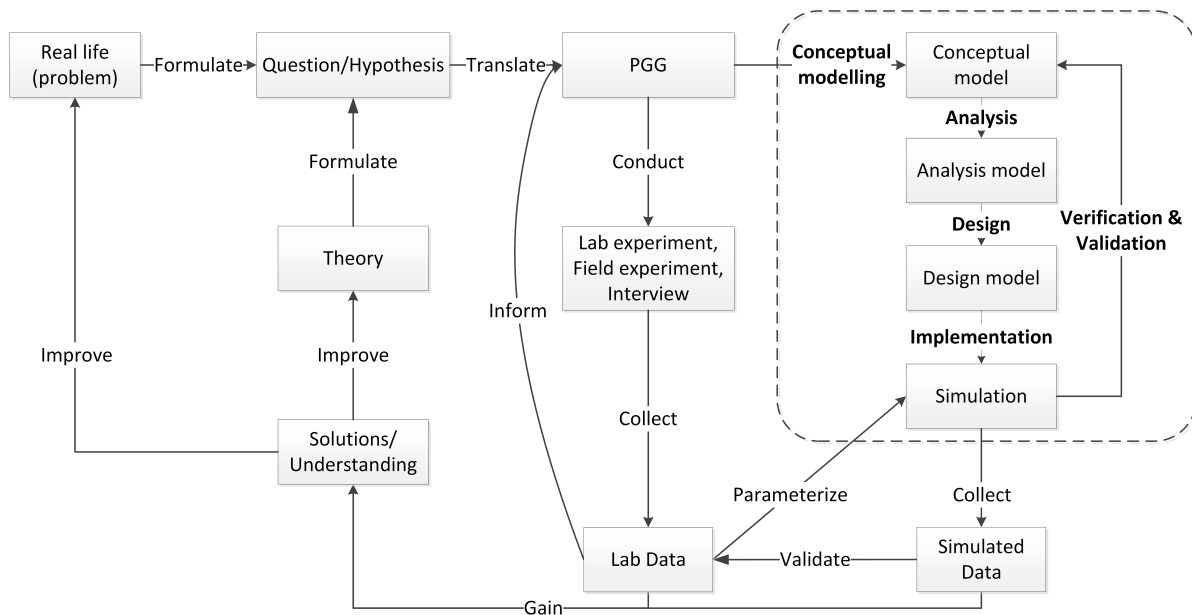


Fig. 5.21 Overview lifecycle of the ABOOMS framework v3

- Identify the model outputs (responses).
- Determine model content: scope and level of details.
- Identify assumptions and simplification.

Process 2 - Analysis

- Identify agents, artifacts, and external actors:
 - Agents (internal actors): autonomous, proactive entities that encapsulate control and in charge of goals/tasks that altogether determine the whole system behaviour. For example, in PGG, players are agents.
 - Artifacts (internal actors): passive, reactive entities that are in charge of required services and functions that make agents work together and shape the agent environment. For example, an artifact in PGG can be “game master” to inform players of the current stage of the game.
 - External actors: External entities such as human and other systems.
- Identify use cases.
- Document all use cases.
- Draw Use Case Diagram.

Process 3 - Design

- Define structure: Define the structure (internal data, operation) with Class Diagram.
- Define behaviour: Define behaviour with behavioural diagrams, such as Statechart, Sequence Diagram, Activity Diagram.

Process 4 - Implementation

- Choose a programming language and/or a software: The simulation can be programmed from scratch or with a specialist simulation toolkit (NetLogo, AnyLogic, Repast Simphony).
- Programming: Writing the code.
- Testing and debug: It is important to perform testing in order to ensure the correctness of the model.

Process 5 - Verification and Validation

- Verify whether the simulation correctly implemented the conceptual model.
- Validate and tune agent behaviour based on the information from the lab or literature.

Behavioural modelling with statechart

Steps to modelling behaviour with statecharts:

- List all the elements that affect agent decision making.
- Consider a piece of information or a variable that can only take one value from a fixed set of pre-defined values.
- Decide how many statecharts are needed.
- For each statechart, identify: states, transitions, triggered conditions, and guards.

The following are some useful tips on implementing statecharts:

- An agent can have multiple statecharts.
- From one state to another, there can be many transitions with different triggered conditions.

- Transitions in statecharts are the driving force behind statechart behaviours. Transitions can be triggered by:
 - Probability.
 - Conditions based on variables.
 - Messages from other agents.
 - Other statecharts: the current state of statecharts, or messages between statecharts.

Statecharts suggestion for agents in PGG:

- Action statechart: represent actions that agents can perform. It describes a sequence of actions in PGG. These actions are most likely to have tight relationships to the stages of the game.
- Strategy statechart: represents strategies of agents in a game. This is suitable for the situation where agents change their strategies and there are many complicated conditions that trigger these.
- Mental statechart: represents mental states of agents.
- Spatial statechart: represents location of agents. This can be used in a spatial PGG where the decision of agents is affected by their location.

5.9 Summary

Using ABMS, this chapter has modelled the effects of individual differences in punishment sensitivity in a Public Good Game. The simulation has validated the behaviours which were observed in laboratory experiments of [Skatova and Ferguson](#). We also demonstrated that ABMS can be used to investigate different aspects of human decision-making which do not integrate with traditional economic models of behaviour. Additionally, the implemented agent-based simulation can be used as a decision support tool for policy makers to examine the free riding behaviours in varying punishment conditions in a real world scenario which resembles a public goods game (e.g., recycling, littering, energy use at home, etc.). In this chapter, the usage of a statechart to model complex decision making is also discussed: when a statechart can be used; how to design states, its transitions, triggered conditions, and guards. From the framework v2, The detailed guide for statecharts is added to create the ABOOMS framework v3.

Chapter 6

Case Study 3: Continuous-time Setting

In this chapter, the ABOOMS framework is applied to simulate a PGG experiment played in continuous time where participants can change their contribution at any time. Although this setting is involved in most real life situations, to the best of our knowledge, there are only very few laboratory experiments and no agent-based simulation model with continuous time setting in Economics. The reasons might be the problem of traceability of such a complex system or difficulty in programming. The purpose of this chapter is to show how the ABOOMS framework can support modelling of agent behaviour in a continuous-time environment and what insights this agent-based simulation can contribute to Economics.

6.1 Case Study of Continuous-time Public Goods Game

In this chapter, the case study is based on an experimental work on continuous-time PGG (Oprea et al., 2014). The laboratory experiment varies two experimental treatments:

Timing protocol (discrete vs. continuous time) For a game in discrete time, the participants make a decision every minute for 10 rounds (10 minutes in total), and payoff for each round is calculated every minute. In continuous time, the participants make contributions in 10-minute intervals, can change their contribution at any time during this interval, and receive a *flow payoff* (payoff is divided into smaller portions for a shorter time period). For example, a payoff function f is used in discrete time to calculate the payoff every minute. In a continuous time game where the payoff is calculated every 100 ms, the payoff function will be $f' = f/600$ because $1 \text{ min} = 600 \times 100 \text{ ms}$.

Communication protocol (no-communication vs. free-form communication) The participants cannot communicate in any way in a no-communication protocol. On the other hand, in free-form communication, participants can communicate with other group

members by typing any text in a chat window. The messages are colour-coded for each player.

The experiment of [Oprea et al.](#) reported that continuous time slightly improves contribution compared to discrete time but cannot reach mutual cooperation, and free-form communication substantially raises the contribution comparing to no-communication (Figure 6.1). Since the free-form communication contains too much noise, it would be very hard to identify the elements that drive contribution in this setting. Therefore, this case study focuses on the no-communication setting. The experiment shows that in no-communication experiments, the participants start to use *pulses* in contribution as an effort to communicate. A pulse is a situation when one increases his contribution to a very high value (often the maximum value) then drops down to the previous level immediately. These pulses may be used as signals of cooperation, but we can not be certain that all the pulses are seen or how they are interpreted by other parties. The participants could miss the pulse, did not understand the signal, or saw it but still waited for responses from others. Without communication, the intention of the pulses is ambiguous.

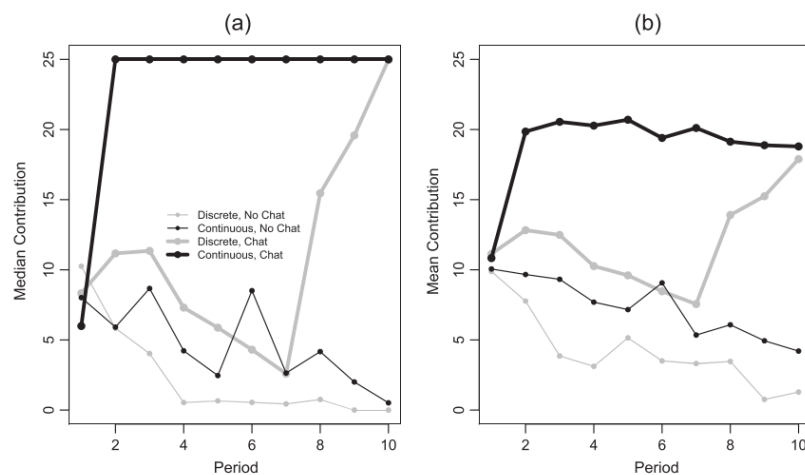


Fig. 6.1 (a) Median and (b) mean contributions by period and treatment ([Oprea et al., 2014](#))

The first question this study is aiming to answer is how much these pulses influence the contribution. When performing a pulse, people have different rates: some wait for a while before dropping down; others drop down immediately. In addition, we hypothesize that the slight increase of contribution in continuous time is not only caused by the ability to create a pulse as an attempt of communication, but also by the ability to evaluate and contribute asynchronously. With the asynchronous decision making, people can evaluate at different frequencies. Some evaluate and change their contribution frequently; while for others, there is a longer break between two evaluations. Therefore, the second hypothesis is that the evaluation rate (fast and slow) can also affect the group contribution. In summary, the simulation study

explores how the two behaviours (pulse and asynchronous evaluation) influence contributions in continuous time.

6.2 Conceptual Modelling

6.2.1 Understand the problem situation

In the experiment of [Oprea et al. \(2014\)](#), the participants were arranged into groups of four. They played the PGG in their group using a computer. Each received an endowment of 25 tokens. Each token invested into the public goods became 1.2 points and shared between group members. The payoff is calculated by Equation 6.1. Figure 6.2 contains the screens that the participants used during experiments. The screen on the left shows contributions in the group. Each player is represented by a dot with a different colour. The x-axis is the contribution, the y-axis is the payoff.

$$\pi_i = 25 - g_i + 0.3 \sum_{j=1}^4 g_j \quad (6.1)$$

where a pay-off (π) for a participant i is defined by their contribution (g_i) and the sum of contributions of all players in the group.

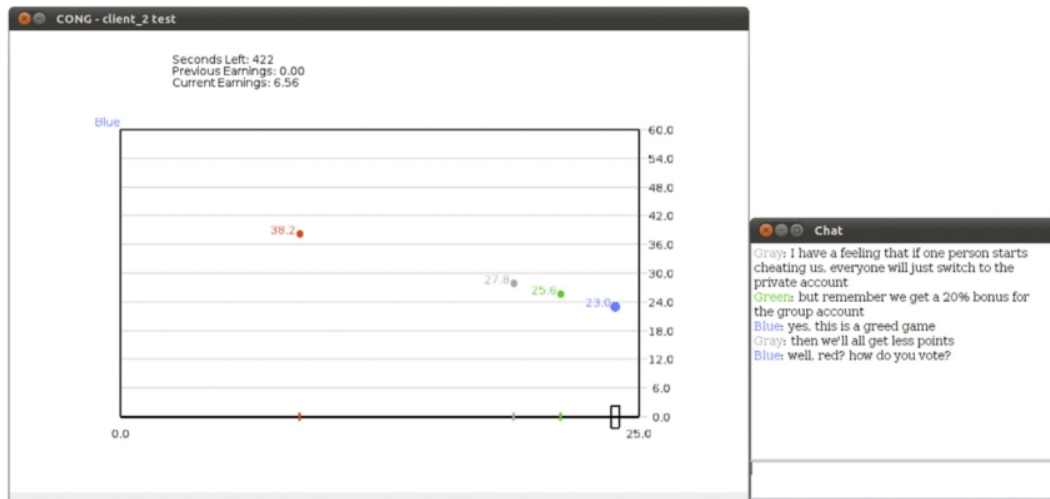


Fig. 6.2 Screenshot of a continuous-time laboratory experiment ([Oprea et al., 2014](#))

Participants can change their contribution at any time by using a slider at the bottom of the graph. Everyone can see contributions as well as payoffs of others in their group. For each game, participants play for 10 minutes and the payoffs are calculated by a utility flow function over time.

6.2.2 Collect and analyse data and related theories

To the best of our knowledge, there are only two laboratory experiments of continuous-time PGG: Dorsey (1992) and Oprea et al. (2014). In the experiments of Dorsey (1992), participants can change their contribution throughout the game but only the final decisions of the periods are calculated for payment. Oprea et al. (2014) performed continuous-time experiments but with continuous flow-payoff setting which is strategically different from Dorsey's work.

With the data from the experiments of Oprea et al. (2014), we can analyse the participants' behaviour in the no-communication continuous-time setting. There are five groups of four playing in that setting over ten minutes. Figure 6.3 shows the median and mean of contributions for every minute. Details on individual contribution in each group can be found in Appendix B. In Figure 6.3a, generally, the median of contribution decreases over time: Contribution of Groups 3 and 4 drops to near 0 in the second round; Group 2 starts at full contribution then decreases over time; Group 1 and 5 begin with low contribution then oscillate before diminishing. Although the contribution of the whole population decreases, there are different dynamics of behaviour in the five groups. Thus we decided to categorize the behaviour of agents into different types as an attempt to understand and explain the group dynamics. Based on the observation from the data, there are five types of behaviour:

1. Cooperator: People that contribute maximum value for a long time (up to 1-2 minutes), then drop to zero contribution (when dissatisfied). But later increase their contribution to maximum again. The total time they contribute is around a third of total game time.
2. Generous Conditional Cooperator (GCC): People who match their contribution to the average of other members in the group, and sometimes increase contribution for a short period of time (i.e. a pulse).
3. Conditional Cooperator (CC): People who match their contribution to the average of other members in the group.
4. Defector: People who contribute 0 most of the time (with noises).
5. Noisy players: People who change contributions frequently and to a wide range of values from 0 to 25.

Table 6.1 shows the classification of behavioural type of each participant. The table also shows the rate of change, that is calculated by the function $\sum_{t=1}^{600} |g_t - g_{t-1}|$ where g_t is the contribution at t second. For GCCs, the number of pulses (in 10 minutes) is also counted. During a pulse, the contribution increases to a high value, which is greater than 20, and stays at that level for as short as 1 second or up to 15 seconds. Table 6.2 is the count of each type and is

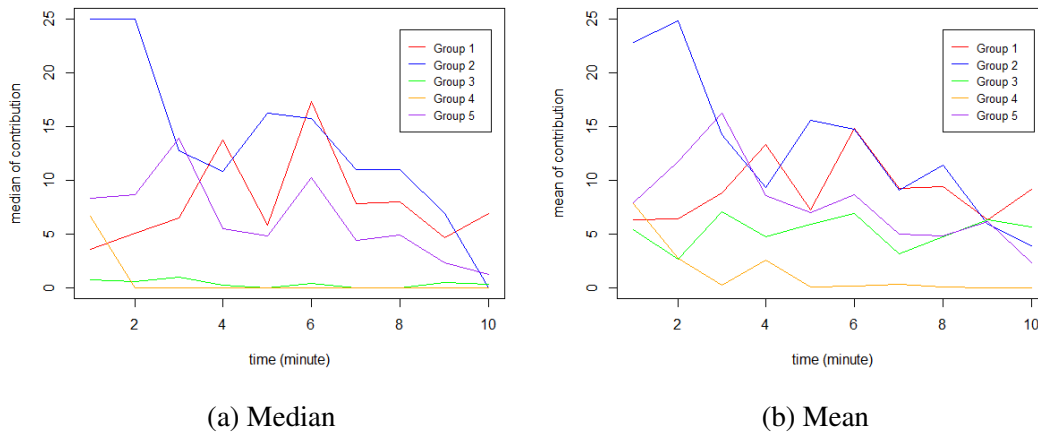


Fig. 6.3 Median and mean of contributions over 10 sub-periods of 1 minute (Oprea et al., 2014)

used to estimate the percentages of each types and use them to parameterize the simulation. For our simulation, the noisy player is excluded to eliminate unnecessary noise. The remains four types are estimated as follows:

- 10% of Cooperators.
- 40% of GCCs.
- 30% of CCs.
- 20% of Defectors.

6.2.3 Determine objectives of the simulation study

The objective of the simulation study is to investigate how the two following behaviours can influence the contribution in continuous-time PGG: pulse behaviour and evaluation rate.

6.2.4 Identify the settings of the PGG

The settings of the PGG are:

- Group structure: groups of four people.
- One-shot or repeated games: repeated games.
- Players' action: evaluate, contribute, pulse.
- Scheduling: continuous-time and asynchronous decision making.

Table 6.1 Individuals in no-communication continuous-time experiments

ID	Group	Type	Rate of change	No of pulses
102	1	noisy player	2.42	3
103	1	GCC	1.85	
107	1	GCC	1.30	
108	1	GCC	1.86	
111	2	CC	0.73	12
114	2	CC	0.42	
115	2	CC	0.91	
116	2	cooperator	1.42	
202	3	defector	1.04	3
203	3	defector	0.51	
204	3	GCC	1.36	
205	3	cooperator	0.99	
216	4	defector	0.43	25
218	4	defector	0.24	
2110	4	GCC	0.27	
2112	4	CC	0.48	
221	5	GCC	3.13	2
227	5	noisy player	5.31	
229	5	GCC	1.30	
231	5	CC	0.58	

Table 6.2 Number of each type in no-communication continuous-time experiments

Type	Count
cooperator	2
GCC	7
CC	5
defector	4
noisy player	2
Total	20

- Game loop: There is no control loop. The players change contribution any time during the game and receive flow payoff.
- Payoff function: Function [6.1](#).
- Information: players know the contribution of each player in the group.

6.2.5 Design the conceptual model

The framework required the following details for the conceptual model:

- Inputs (Experimental factors):
 - Different ratio of behavioural types.
 - Rate of evaluation.
 - Rate of staying generous, i.e. pulse (higher rate means more time staying at high contribution).
- Outputs (Responses):
 - Average contribution over time.
 - Contribution of each agent over time.
- Model Content:
 - Scope in Table [6.3](#)
 - Level of details in Table [6.4](#)
- Assumption:
 - People do not change strategy in all rounds of the game.
 - There are no noisy players.
 - Ratio for the base experiment: 10% of cooperators, 40% of GCCs and 30% of CCs, 20% of defectors.
 - GCCs perform pulse on average 7 times during 10 minutes, or 0.7 times per minute. (based on the median of number of pulses in Table [6.1](#)).
- Simplification:
 - People cannot communicate in any way.
 - People have complete and common knowledge of the game structure and payoffs.
 - Agents take 500 ms to evaluate and 500 ms to change their contribution.

Table 6.3 Model Scope

Component	Include/Exclude	Justification
Person	Include	Required for decision making.
Communication	Exclude	No-communication setting which means the agent can not send messages to communicate intention or negotiate.

Table 6.4 Model Level of Detail

Component	Detail	Include/Exclude	Comment
Person	Contribution	Include	The result of decision making.
	Others' contribution	Include	Information required as inputs of decision making.
	Strategy	Include	Required for decision making.

6.3 Analysis

6.3.1 Identify actors

There is one type of agent, a Person agent, that represents a participant in the game. Each Person agent is assigned to a Group, which is an artifact. The Group artifact manages the contribution of the group, calculates the contribution and the payoff for Person agents. There is also a Main artifact that initializes the agent population and collects statistics. In summary, there are three internal actor types: Person agents, Group artifacts, and a Main artifact.

6.3.2 Identify use cases

From the description of actors, the next step is to identify use cases that can be performed by the actors. For the game in this case study, there are five use cases:

1. Initialize: Main initializes the properties of Person agents, and randomly shuffles them into groups of four.
2. Make decision: Person agents evaluate [Evaluate use case], then contribute [Contribute use case].
3. Evaluate: Person agents update their belief (learn from experience).
4. Contribute: Person agents make a contribution based on their type.
5. Pulse: Person agents increase contribution to a high value then drop down.

6.3.3 Document use cases

1. Use case name: Initialize
 - (a) Description: Main performs initialization at the start of the game.
 - (b) Agents & artifacts: Person, Main, Group.
 - (c) Pre-conditions: Simulation starts.
 - (d) Post-conditions: Person agents are shuffled into different groups.
 - (e) Flow of events: Shuffle Person agents into groups of four.
2. Use case name: Make a decision
 - (a) Description: Person makes a decision.
 - (b) Agents & artifacts: Person
 - (c) Main flow of events: A Person evaluates [Evaluate use case]. A Person contributes [Contribute use case].
3. Use case name: Evaluate
 - (a) Description: Person updates their belief about contribution based on other contributions.
 - (b) Agents & artifacts: Person, Group
 - (c) Pre-conditions: None.
 - (d) Post-conditions: Belief is updated.
 - (e) Flow of events: A Person requests average contribution of others from Group. A Person updates belief.
4. Use case name: Contribute
 - (a) Description: A Person makes a contribution based on their type.
 - (b) Agents & artifacts: Person
 - (c) Pre-conditions: Evaluation is done.
 - (d) Post-conditions: Contribution is decided.
 - (e) Flow of events: A Person makes their contribution based on strategy.
5. Use case name: Pulse

- (a) Description: Person agents increase contribution to a high value then drop down.
- (b) Agents & artifacts: Person
- (c) Pre-conditions: Group contribution is less than 20.
- (d) Post-conditions: None.
- (e) Flow of events: A Person increases contribution to a high value (from 20 to 25), waits for a period, and makes decision [Make Decision use case].

6.3.4 Draw use case diagram

Figure 6.4 shows the use case diagram. There are «include» relationships between “Make a decision” (the including use case) and “Evaluate” and “Contribute” (the included use cases).

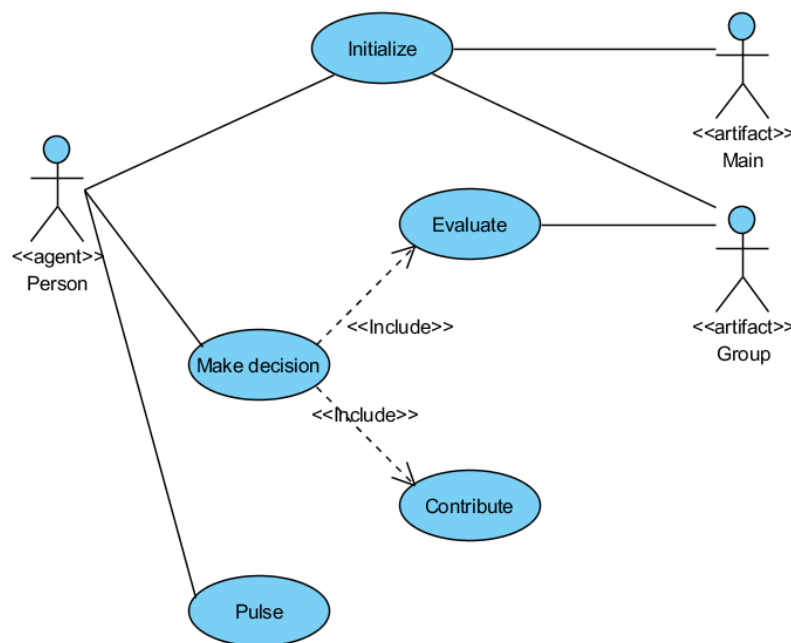


Fig. 6.4 Use case diagram

6.4 Design

6.4.1 Structural design

From the actors, there are three classes representing Person agent, Main artifact, and Group artifact. Main class links to the Person and Group classes by association relationship, allowing

the Main class to initialize and manage those two classes. There is an aggregation relationship between Person and Group classes: a Group contains four Persons, and a Person can only belong to one Group.

Because there are major differences in behaviour between types of Person agents, such as pulse behaviour of GCCs compared to dissatisfied behaviour of cooperators, inheritance is employed in the structural design to capture the semantics of the type hierarchy. Person is the superclass, and each of four types is a subclass (i.e. aggregation relationship between Person and Group class). So there are seven classes in total, shown in the class diagram Figure 6.5. The four subclasses inherit attributes and behaviours of Person superclass, but still have their own attributes and behaviours. For example, the *group* variable of Person class is inherited by subclasses, but *contribute()* operation is overridden by each subclass to implement their own behaviour.

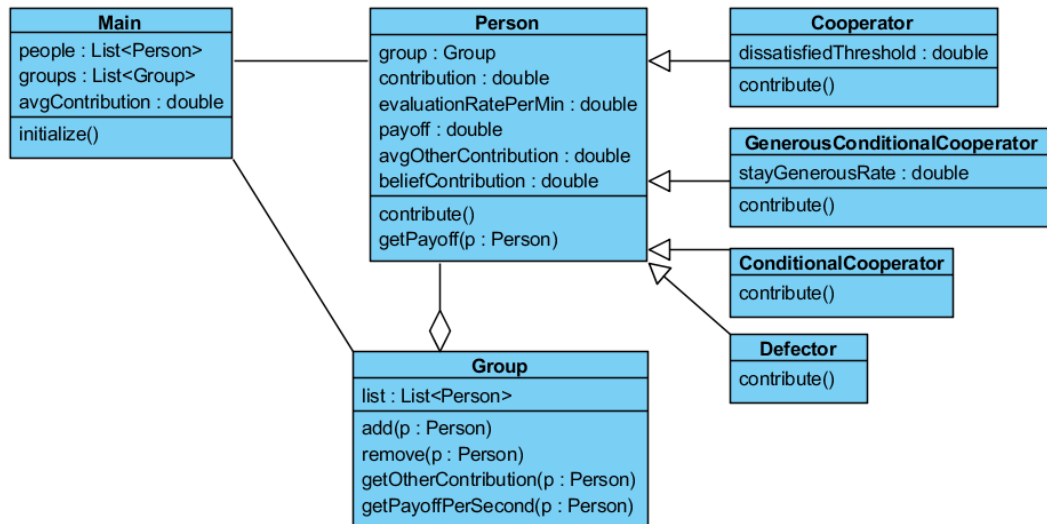


Fig. 6.5 Class diagram

6.4.2 Behavioural design

At the beginning of a simulation run, Main sets up the ratio of different behavioural types of Person agents, initializes their belief and contribution, and randomly adds Person agents into different groups of four. During the simulation run, Main collects statistics on contributions every second. Each simulation runs for 10 minutes, so there are 600 points of data per participant.

For the four types of agents (Cooperator, GCC, CC, Defector), each has a statechart to represent different contribution behaviours (Figure 6.6). The Person agent has two variables

that are frequently used in the statecharts: *evaluateRatePerMin* is the rate at which agents evaluate and make decisions; and *avgOtherContribution* is the average contribution of other group members. The detailed description of each statechart is as follows:

- The statechart of Cooperator agents, shown in Figure 6.6a, has two states: Contribute and Dissatisfied. When in Contribute state, cooperators contribute 25. The transition from Contribute to Dissatisfied is triggered by the rate *evaluateRatePerMin*, with a guard condition that *avgOtherContribution* is less than *dissatisfiedThreshold*. The transition from Dissatisfied to Contribute is triggered by a rate of 1 per minute. As a reminder, in AnyLogic, when a trigger is of type rate, the transition is executed with the timeout distributed exponentially. This means if the rate is 3, the transition will be executed on average 3 times per time unit, but the time between executions is distributed exponentially.
- Figure 6.6b is the statechart of CCs with three states: Idle, Evaluate, ChangeContribution. The transition from Idle to Evaluate is rate triggered with the rate *evaluateRatePerMin*. When entering Evaluate, an agent requests the *avgOtherContribution* from the Group artifact, and then updates its *beliefContribution*. The transitions from Evaluate to ChangeContribution and from ChangeContribution to Idle have a timeout of 500 ms. A simplification has been made here: agents take 500 ms to evaluate and 500 ms to change their contribution.
- The statechart of GCCs, illustrated in Figure 6.6c, is an extension from the statechart of CCs by adding a Generous state. The transition from Idle to Generous is rate triggered (0.7 per minute as explained in the assumption of conceptual modelling). The transition from Generous to Evaluate is triggered with a rate *stayGenerousRate*.
- Figure 6.6d is the statechart of Defectors with only one state. There is a self-transition with the rate of *evaluationRatePerMin*.

For the Evaluation use case, agents update their belief *beliefContribution* using an actual belief model (Model 3 in (Fischbacher and Gächter, 2010)): $beliefContribution = 0.415 \times avgOtherContribution + 0.569 \times beliefContribution + 0.118$. (Fischbacher and Gächter formed this equation from the lab experiment and used it in their simulation.)

For the Contribute use case, the agents use the following equations for making a contribution decision:

- Cooperator: $contribution = 25$
- Defector: $contribution = 0$

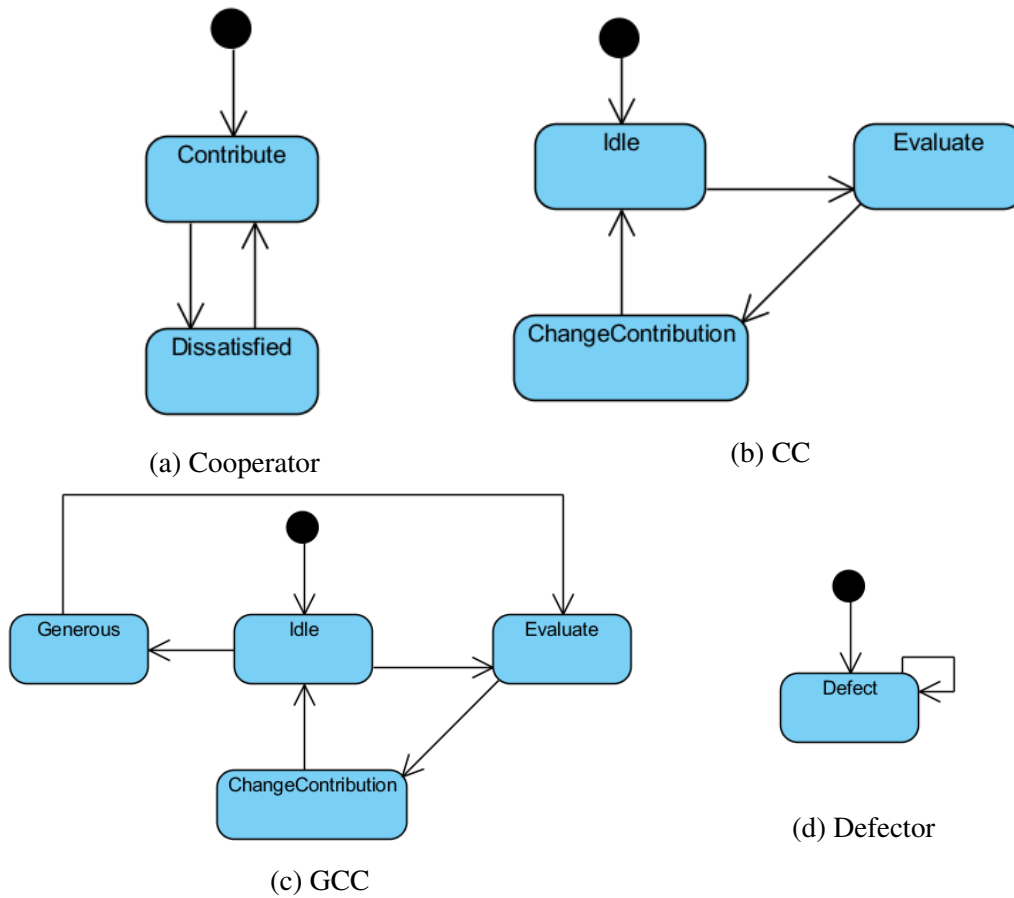


Fig. 6.6 Statecharts of the four types of agents

- GCC and CC: $contribution = beliefContribution$

6.5 Implementation

The simulation was implemented with AnyLogic 7 PLE. Figure 6.7a shows the Main screen during simulation experiments. For inheritance, AnyLogic IDE shows the superclass elements as transparent in the IDE view of the subclass. Figure 6.7b is the IDE view of the Cooperator agent. The transparent elements (7 variables, 1 function, and 1 time plot) are inherited from Person superclass. Other elements (*dissatisfiedThreshold* variable, *contribution* function, and a statechart) are its own attributes and behaviours.

The seven parameters (related to the inputs identified in the conceptual modelling process) shown on the left side of the Main screenshot (Figure 6.7a) can be varied for simulation experiments. For the first experimental factor (ratio of agent strategies), four parameters are required for percentages of four agent strategies: *numCoop*, *numGCC*, *numCC*, *numDefect*.

The sum of these four parameters is 1000 agents. The other two experimental factors, rate of staying generous and rate of evaluation, are ranged from 0 to 1. For the experiments to be manageable, evaluation rate is set to either *slow rate* (5 per minute) or *fast rate* (30 per minute). For example, if the evaluation rate of an agent is slow, it means that agent, on average, evaluates 5 times per minute, i.e. evaluates every 12 seconds. For the fast evaluation rate of 30 per minute, agents, on average, evaluate every 2 seconds. The rate of staying generous is set to one of the same two values as evaluation rate but with different names: longer stay generous (5 per minute) and shorter stay generous (30 per minute). Three parameters are introduced: *percentageSlowGCC*, *percentageLongerStayGenerous*, and *percentageSlowCC* to represent the percentages of different rates in the population of GCCs and CCs. The pseudocode for initialization of these two types is listed in Pseudocode 6.1.

Pseudocode 6.1: Initialization of GCCs and CCs

foreach *Person p in the set of Generous Conditional Cooperators* **do**

 beliefContribution $\leftarrow$ uniform(0,25);

 contribution $\leftarrow$ beliefContribution;

if randomTrue(*percentageSlowGCC*) **then**

 evaluationRatePerMin $\leftarrow$ 5;

else

 evaluationRatePerMin $\leftarrow$ 30;

if randomTrue(*percentageLongerStayGenerous*) **then**

 stayGenerousRate $\leftarrow$ 5;

else

 stayGenerousRate $\leftarrow$ 30;

foreach *Person p in the set of Conditional Cooperators* **do**

 beliefContribution $\leftarrow$ uniform(0,25);

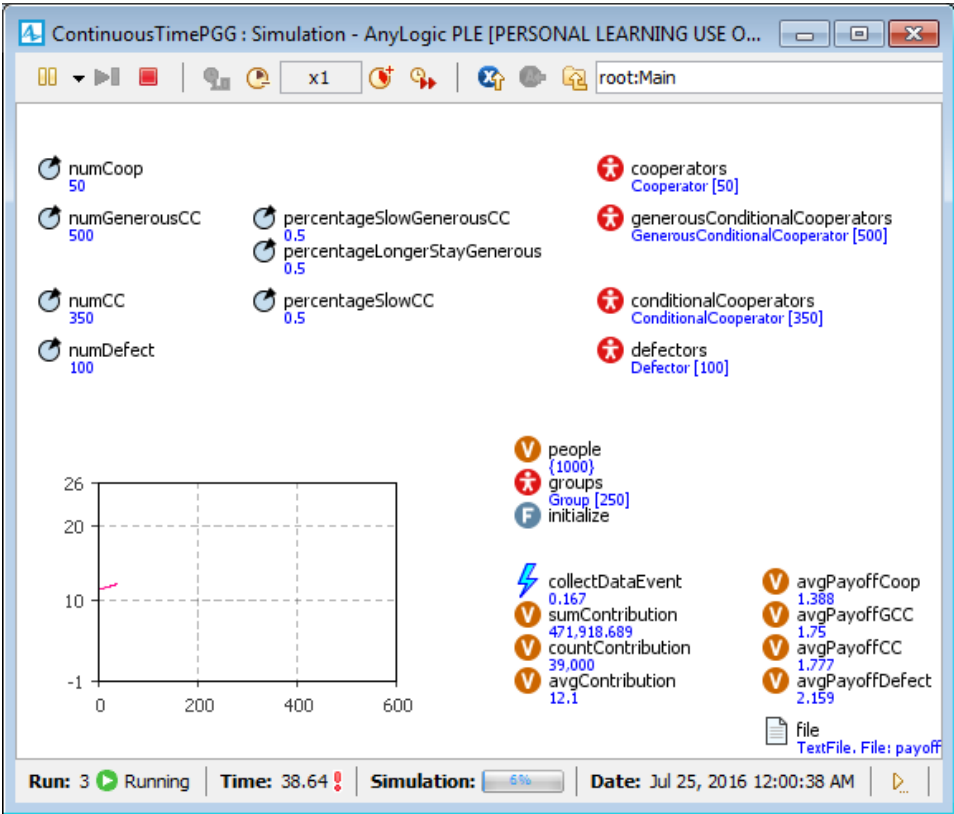
 contribution $\leftarrow$ beliefContribution;

if randomTrue(*percentageSlowCC*) **then**

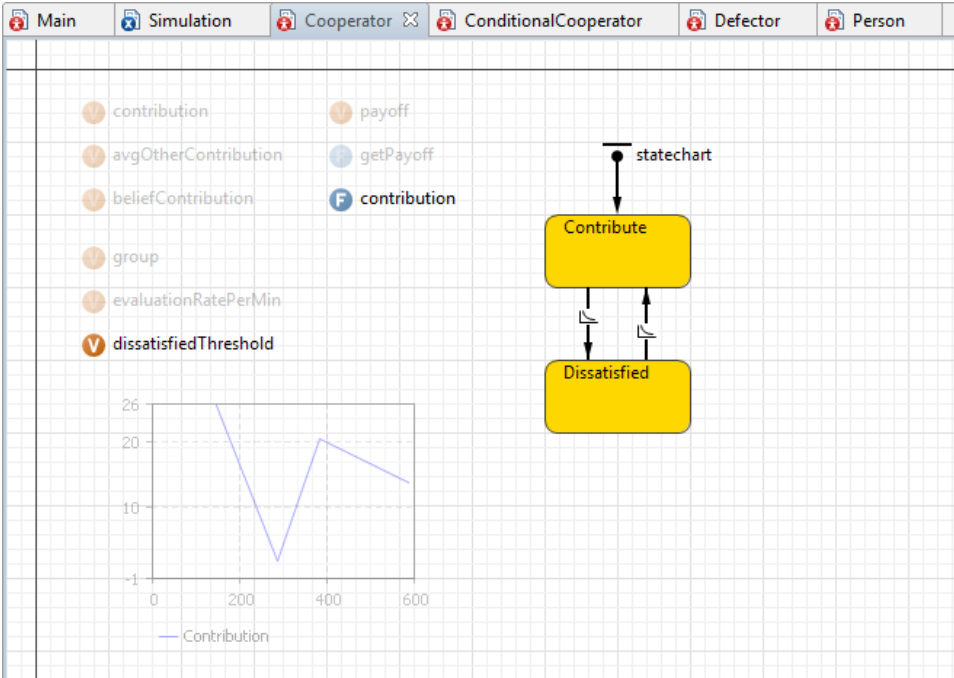
 evaluationRatePerMin $\leftarrow$ 5;

else

 evaluationRatePerMin $\leftarrow$ 30;



(a) Main Screenshot



(b) Cooperator Screenshot

Fig. 6.7 AnyLogic implementation screenshots

6.6 Experiments and Results

6.6.1 The presence of GCCs

The first three experiments were set up with the parameters in Table 6.5. For each experiment, there were 10 replications and the contribution was collected every second. Then the average contributions over time of 10 replications were calculated and shown in Figure 6.8. The first experiment has the ratio of strategies as suggested in the conceptual modelling, and the percentages are all set to 50% for a balance mix of different rates. In order to compare to the situation when GCCs are absent in the population, in the second experiment, 400 GCCs are changed to CCs. Thus there are 700 CCs in total. In Experiment 2, the contribution decreases and reach an equilibrium. The presence of GCCs in Experiment 1 has lead to a higher value of contribution. The contribution slightly decrease then stabilizes. This leads to our first result:

Result 1: The presence of GCCs increases group contribution.

Table 6.5 Parameters of Experiment 1, 2 ,and 3

Parameters	Experiment 1	Experiment 2	Experiment 3
numCoop	100	100	100
numGCC	400	0	0
percentageSlowGCC	0.5	0	0
percentageLongerStayGenerous	0.5	0	0
numCC	300	700	700
percentageSlowCC	0.5	0.5	0
numDefect	200	200	200

To explain the effect of GCCs, an analysis of the contribution dynamics within groups is needed. Figure 6.9 presents contribution dynamics within six groups that are chosen from the simulation runs. Groups 1 and 2 have no GCCs. Group 1 has more cooperators so the contribution is high; Group 2 has more defectors so the contribution is low. When GCCs are introduced into the game, we have Groups 3, 4, 5, and 6.

The GCC behaviour of pulsing causes the average contribution of the game to increase. For example in Groups 3 and 4, the contribution is increasing within the game but dropping down after GCCs get out of Generous state. Within the Generous state, other CCs and GCCs increase their contribution slowly (depending on the learning rate of the belief update function) but then drop down when GCCs go down. However, in some situations, the contribution keeps at a high level, such as in Groups 5 and 6. Especially when comparing Groups 4 and 6, they have the same type structure but in the end, the contribution of Group 6 is higher. This is because

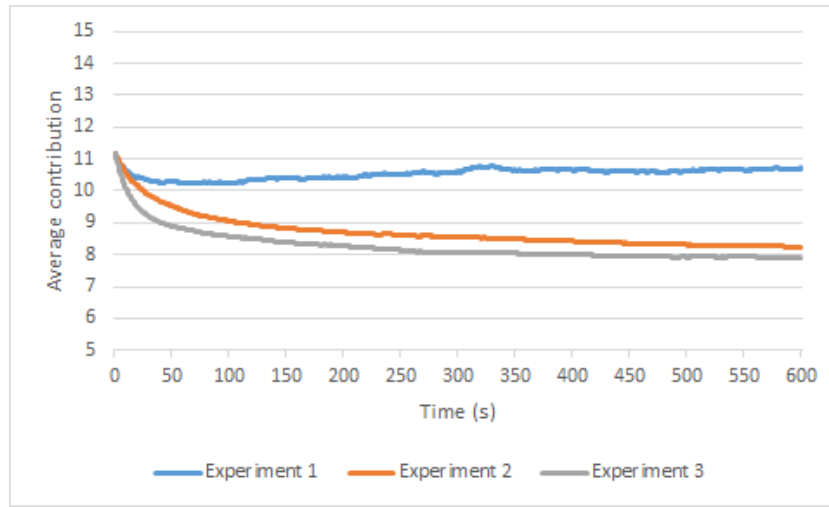


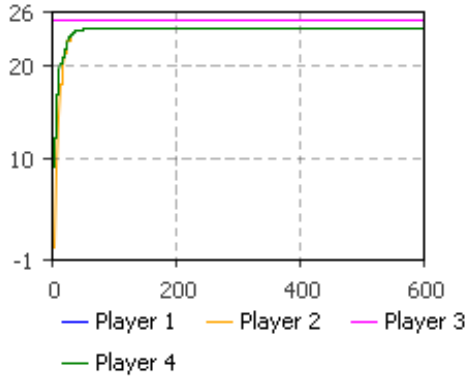
Fig. 6.8 Average contributions of Experiment 1, 2, and 3

the green player made two pulses in a short amount of time and the second pulse has a longer duration, which allows other GCCs and CCs to increase their contribution to a higher level.

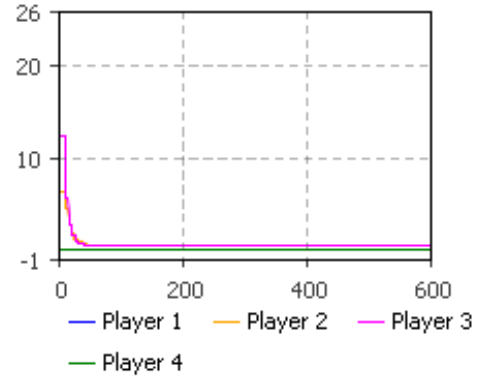
6.6.2 Evaluation rate of CCs in a simulation without GCC

To see how evaluation rates affect the contributions, the third experiment is a variation of the second experiment with all CCs having a fast evaluation rate. So without GCCs, contribution breaks down faster when there are more agents with fast evaluation rate (Experiments 2 and 3 in Figure 6.8). When we set the simulation run for a longer period, both experiments reached the same equilibrium, but the third experiment approached it at a faster rate. **Result 2:** Without GCCs, a faster evaluation rate of CCs leads to the situation in which the average of group contributions reach the equilibrium faster.

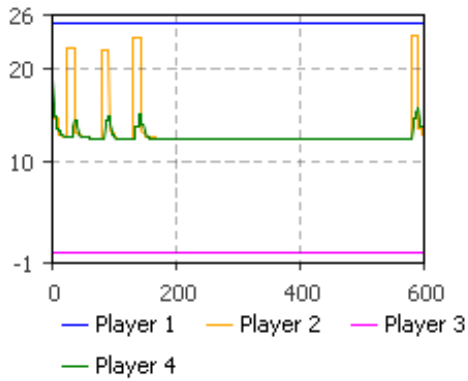
Experiments 2 and 3 contain only three types of players (Cooperators, CCs and Defectors). The defectors never change contribution. Since the average group contribution is around 10 and does not drop below 5, cooperators contribute 25. Thus the CCs are the drivers behind the dynamics of group contribution. Since CCs match their contribution with the average contribution, the average contribution will converge toward an equilibrium based on the number of cooperators and defectors. In Experiments 2 and 3, the average contribution in the simulation starts at about 11.8 and converges toward about 8.5. Since CCs are the only types of agents that change their behaviour over time, the faster they evaluate and make a decision, the faster the average contribution reaches equilibrium.



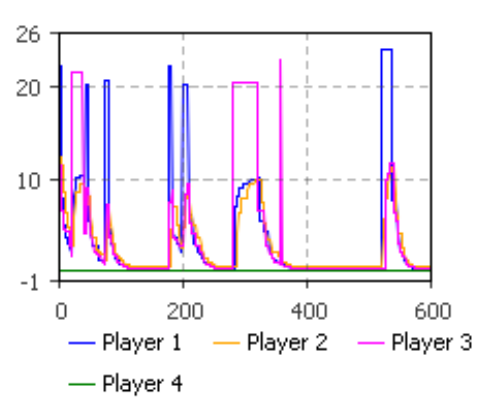
(a) Group 1 (2 Cooperators, 2 CCs)



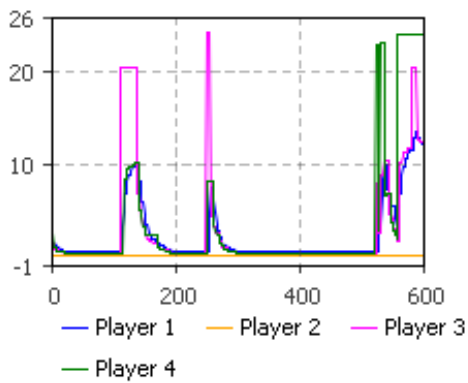
(b) Group 2 (2 Defectors, 2 CCs)



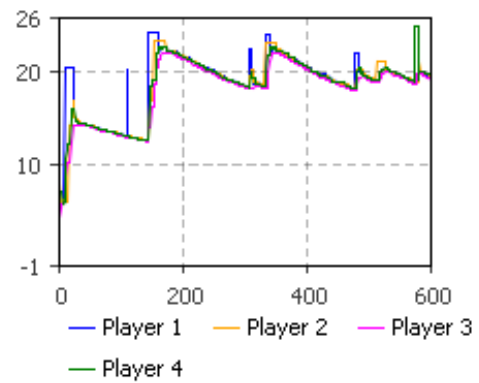
(c) Group 3 (1 Cooperator, 1 Defector, 1 GCC, 1 CC)



(d) Group 4 (1 Defector, 2 GCCs, 1 CC)



(e) Group 5 (1 Defector, 2 GCCs, 1 CC)



(f) Group 6 (3 GCCs, 1 CC)

Fig. 6.9 Contribution in different groups

6.6.3 Evaluation rate and generosity rate of GCCs

After experimenting with the evaluation rate of CCs in the third experiment, the fourth experiment is introduced to investigate the evaluation rate and generosity rate of GCCs. The parameters set-up for the fourth experiment are shown in Table 6.6. For each combination of the two percentages, the simulation was run for 10 replications, and the average contribution of each replication was collected. Then the average contributions of all 10 replications was averaged to create Table 6.7. The contribution is increasing as both percentages increase from 0% to 100%. It is highest when both percentages are 100%, i.e. all GCCs have slow evaluation rate (evaluateRate=5) and stay generous for a longer period (stayGenerousRate=5). **Result 3:** For GCCs, slow evaluation and staying generous longer leads to higher group contribution.

Table 6.6 Parameters of Experiment 4

Parameters	Experiment 4
numCoop	100
numGCC	400
percentageSlowGCC	Varied in 0, 0.5, 1
percentageLongerStayGenerous	Varied in 0, 0.5, 1
numCC	300
percentageSlowCC	0.5
numDefect	200

Table 6.7 Average contribution with different rates of GCCs

		percentageLongerStayGenerous		
		0	0.5	1
percentageSlowGCC	0	9.64	10.28	10.90
	0.5	9.73	10.63	11.17
	1	10.01	10.83	11.43

The slow evaluation reduces the break down of contribution (from explanation of Result 2). In addition, staying generous longer allows other CCs and GCCs to increase their contribution to a higher level (from explanation of Result 1). Thus, these two factors together boost the group contribution.

To have a closer look at generosity rate of GCCs, we performed a fifth experiment to explore how the duration GCCs stay generous affects group contribution. In the fifth experiment, we set all GCCs to have the same generosity rate by removing the *percentageLongerStayGenerous*.

The parameter *generosityRate* is introduced and varied from 5 to 60 (with the step of 5). Table 6.8 describes all parameters of this experiment. The average contribution of the population is collected for each value of generosity and used to plot Figure 6.10. In the figure, the average group contribution drops down when the *generosityRate* increases (higher *generosityRate* means less time staying in Generous state). This is consistent with Result 3.

Table 6.8 Parameters of Experiment 5

Parameters	Experiment 5
numCoop	100
numGCC	400
percentageSlowGCC	0.5
<i>generosityRate</i>	Varied from 5 to 60 with the step of 5
numCC	300
percentageSlowCC	0.5
numDefect	200

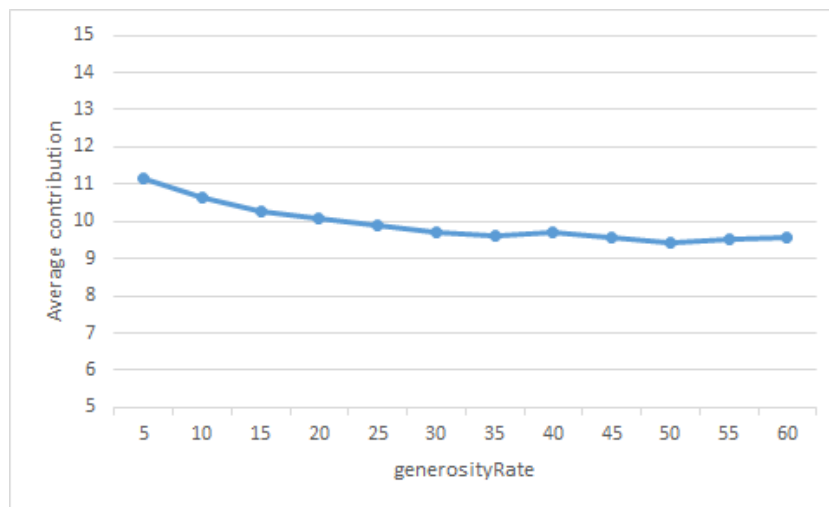


Fig. 6.10 Experiment 5 - Average contribution of different generosityRate

6.7 Discussion

The new preference type GCC has explained some group dynamics in the continuous-time setting. Even though GCCs boost contribution in overall, there are some group compositions that break down the contribution such as when there are many defectors in a group. One of the group compositions that will lead to high contribution is a combination of Cooperators

and CCs/GCCs. But since Cooperators are few in the population, there are not many of these groups. Another case where contribution can be high is when GCCs pulse at the same time or together. This requires the group to have at least two GCCs and their pulses are the same at some points of time. If the communication is limited, achieving mutual cooperation is hard in continuous-time PGG. Nevertheless, these interesting interactions have shown that the continuous time PGG with flow payoff can trigger a different mode of reasoning compared to discrete time.

The reasons for a different reasoning mode can be that continuous time allows participants to change their contribution immediately when an event happens and express their intention. For example, in a group where everyone is contributing, when a participant free rides, other members can react immediately and drop their contribution to 0 in order to show that they are not happy with the free riding behaviour. However, afterwards, even though the participant that free rides increases contribution, the group can still fail to reach full cooperation because of trust issues. Another possible reason for different reasoning is that it is relatively cheap to pulse because of the flow payoff. The pulse can be seen as the participant pays a small amount to signal their goodwill to contribute. Pulse does not strongly affect the contribution because its intention is unclear to others and people may be waiting for others to act first.

Lastly, since the number of participants playing the no-communication continuous-time setting is only 20 in the experiment of [Oprea et al. \(2014\)](#), many assumptions have been made during the data analysis of this case study. For example, the pulse rate is estimated to be 0.7 per minute. Another example would be that the ratio of agent behavioural types is estimated from only 20 people. Future experiments should be performed with more participants to collect more data on behaviour and increase the accuracy of data analysis.

6.8 Summary

This chapter presents an agent-based simulation study of PGG in continuous-time setting with flow payoff, which is rarely considered in Economics. A new preference type (Generous Conditional Cooperator) along with asynchronous evaluation has been used to explain group dynamics in continuous time. The simulation study also experiments with different group compositions, rates of evaluation, and rates of staying generous to understand more about their effects to the group contribution over time. The ABOOMS framework, especially with the support of statecharts, has demonstrated its capability in modelling dynamic behaviour in continuous-time PGG. A documentation of the final version of the ABOOMS framework is available in [Appendix C](#).

Chapter 7

Conclusion and Future Work

This chapter summarizes the thesis in Section 7.1 and reviews the uses of the ABOOMS framework in Section 7.2. Then Section 7.3 evaluates the fulfillment of aim and objectives. Next, the chapter reviews contributions made throughout this research to fulfil the research gap (Section 7.4), and identifies limitations and proposes directions for future researches (Section 7.5).

7.1 Summary of Research

The research in this thesis leverages the potential of ABMS with a focus on PGG. The main motivation of the thesis is provided by the mismatch between ABMS in Computer Science and its adaptation to Economics. As an effort to leverage ABMS's potential, software engineering techniques such as OOAD and UML were integrated into a novel development framework, namely ABOOMS. This ABOOMS framework was then used to investigate three aspects economists encounter when modelling PGG: (1) modelling principles: KISS vs. KIDS, (2) modelling of psychological mechanisms, and (3) continuous-time setting. After evaluating and revising with three case studies, the final version (v3) of the ABOOMS framework includes a development lifecycle of the macro process and a step-by-step development guideline of the micro process, and a short guide to model agent behaviour with statecharts.

Additionally, the simulations of each case study provide understanding and insights to the behavioural dynamics in the PGG:

- Social preferences (i.e. people can have different preferences or strategies: cooperators, conditional cooperators, defectors) can explain behaviours observed in experiments of PGG. Also, the conditional cooperators have large impacts on the dynamics of group contribution.

- People behave differently with different information and experimental treatments.
- Cooperation is hard to maintain in a standard PGG. Even when a treatment improves cooperation, it can be difficult to explain and model the effect. For example, punishment has been modelled successfully while communication is too noisy for modelling.
- Continuous time setting allows participants to express more complex behaviours. Thus, it is more difficult to experiment, to analyse, and to model.
- Modelling of PGG requires data from experiments to better capture and model human behaviour.
- Modelling of PGG requires theory from the social sciences, since it provides the foundation to design agent behaviours and the goal of a simulation study is to understand why people behave that way.

7.2 Uses of the ABOOMS Framework

7.2.1 Purposes and General Uses

The main purpose of the ABOOMS framework is to provide a unified development methodology for the ABMS community. The central idea is that by following activities in the framework modellers can practise the object-oriented mindset. Even though the framework focuses on the development process of agent-based models, the simulation model that is designed with the framework will have a good documentation as a co-product since UML diagrams are used throughout the framework. With good documentation, the framework can facilitate better communication and serve as a bridge between disciplines.

Indeed, for better communication, there are other works focus on the documentation of agent-based models. One example is the ODD (Overview, Design concepts, and Details) protocol that is first developed by [Grimm and Railsback \(2006\)](#) then improved by [Grimm et al. \(2010\)](#). The primary objective of the ODD protocol is to standardize the descriptions of agent-based models by making a common format that is complete, understandable, and reproducible. To use the ODD protocol, modellers fill in information following a given sequence of seven elements, grouped in three categories (Overview, Design concepts, Details). The ODD protocol is not designed to support the development process; however, it is a good tool to describe an implemented agent-based simulation. Then if we compare the documentation between the ODD protocol and the ABOOMS framework, the ODD protocol ensures consistency using a specified order of elements. But the information provided can be in various formats such as

text, graph, or flow chart. On the other hand, the ABOOMS framework ensures consistency using UML diagrams but does not require a fixed order when describing the elements. It can be said that the ABOOMS framework suggests the use of a common modelling language (UML) but it is flexible in the describing order of design elements; thus modellers can follow any order they prefer to describe the final simulation. In addition, it is a good idea to use the framework in combination with the ODD protocol: the modellers can use the ABOOMS framework for development, then integrate the documentation from the ABOOMS with the ODD protocol for model description.

So in term of flexibility, the ABOOMS framework provides a loose structure which leaves rooms for other practices and tools to be incorporated. For example, during the Analysis process, modellers can apply other proposed system structures when identifying agents, such as the structure consisting of active and passive agents by [Li \(2013\)](#). Another example is that in the Design process, the modellers have the flexibility of using other design concepts such as different classes for emotions, learning, etc., because the two tasks in the Design process (define structure and define behaviour) are generic. Modellers can also design different classes for various methods of reasoning such as fuzzy system, genetic algorithm, and neural network. In addition, the use of UML diagrams, which is independent of the implementation software, means that the design can be implemented in any object-oriented programming language or simulation software.

Lastly, the framework is designed to be easy to use by providing details on both macro and micro level of the development process. And modellers can also practise the object-oriented mindset just by following the detailed tasks listed in the micro process. However, the framework does require modellers to have an understanding of UML, not all the diagrams, but at least the five essential ones that are introduced and mentioned in the framework documentation (use case diagram, class diagram, statechart, sequence diagram, and activity diagram). Beginners can use the case studies in this thesis as examples. When having enough experience, people can start to apply advanced software engineering concept such as design patterns when utilising the framework.

7.2.2 The Influence of Software Engineering Methods

This section evaluates the influence of using SE methods in the framework. The spread of ABMS to various disciplines and the ongoing increase in size and complexities of the simulation models have forced the need of standard definitions and development processes. Using SE methods can satisfy this need because it has been developed by computer scientists and software engineers for many years to fulfil similar needs for software projects. A big benefit is the better documentation using UML diagrams which has been discussed in the previous section

(Section 7.2.1). Furthermore, in term of project management, SE methods can also provide a better control over the development process, whether the modellers working individually or in multi-disciplinary team; thus it leads to the improvement of quality of the simulations.

Additionally, practising the object-oriented mindset from conceptual modelling to implementation ensures that the key software engineering principle of *separation of concerns* is properly applied. The general idea is that the system is broken into layers and components that each has specific concerns and responsibilities with as little overlap as possible. This reduces the complex system into a series of manageable components. Therefore, the system is *easy to change*, since a change of a feature are usually isolated to a single component (or a few components directly associated with that feature), instead of interlaced throughout a large and complex code base.

Furthermore, the minimal functionality overlaps also improve the *extensibility* of the system. For example, in the third case study, when a new preference type (Generous Conditional Cooperator) is introduced, only one class needs to be created to implement the behaviour for that new agent type. This class inherits from the Person class, which means the programmer does not have to worry about other actions carried out by the Person class (such as getting payoff) or about logic in main game loop.

Moreover, since each component addresses a separate concern, it also improves the *reusability* of the system. For example, a modeller wants to explore how agents in one of our PGG simulations behave in another economic game that has a slightly different set of rules. Since the game structure is similar with PGG, the modeller can start with designing the game loop and then reuse the Person class and related subclasses in the new simulation. We can reuse not only the whole agents, but also parts of them such as their behaviours. For instance, we assume that there is a PGG simulation in which contribution behaviours use complex optimization algorithms. Therefore, each contribution behaviour is designed as a class. Each class has a *contribute()* function that takes inputs, then uses an algorithm to calculate and returns the amount of money units to contribute to the game. If a modeller want to use one of the algorithms in a new simulation, s/he only needs to reuse relevant classes of that algorithm, as long as the new system can provide it with the relevant inputs.

In summary, in the context of increased complexity of simulation models, it can be difficult to control and manage projects without SE methods. Impromptu development approaches can result in simulations with inflexible design, insufficient testing, and ambiguous documentation. Using SE methods leads to the benefits of identifying correct requirements to implement the simulation that meets the modellers' needs. By identifying the necessary features and designing a good model, the development cost is optimised and it also reduces the post-development cost (maintenance or continuous improvement).

7.2.3 Use with Procedural Programming

As stated in Section 3.4, the AnyLogic environment was selected to be used in the thesis because it supports object-oriented programming very well. However, the NetLogo modelling environment, which uses procedural programming, is a popular choice for ABMS beginners because it is easy to learn. Because of its popularity, this section addresses how the ABOOMS framework can still be utilised when using a procedural programming environment. After the OOAD process, the object-oriented design model needs to be converted to a new design model for procedural programming by following these proposed steps:

1. List actions that agents can perform. These actions can involve a single agent (such as when an agent updates its internal variables) or multiple agents (when agents have to communicate or make decision based on other actions).
2. Identify the main loop, which is a series of actions, in the simulation. In the PGG scenario, it is the game loop with actions such as receiving endowment, then contributing, then receiving payoff, and repeat.
3. Arrange the actions in Step 1 along the main loop in Step 2.
4. When coding, each action is coded as a procedure and the main loop will invoke these procedures as needed.

So the resulted design model has a flow-based or event-based thinking and is suited for procedural programming environments such as NetLogo. For documentation, the activity diagrams in the object-oriented design model are useful in the design model for procedural programming because they describe the flow of control. To understand more about the differences between the two design approaches, one can study and compare between the design of KISS model in the first case study (Section 4.3) and its NetLogo implementation (Appendix A).

In short, it is possible to implement a NetLogo simulation with a design model from the ABOOMS framework by converting it to a flow-based design model. The conversion process can work with simple models; however, as the models get more complex, the implementation becomes exponentially more difficult. Thus, it is in the best interest of the modellers to use an object-oriented environment to fully utilise the ABOOMS framework.

7.3 Achievement of Aim and Objectives

Our research aim is to leverage the potential of ABMS for PGG focusing on the development methodology of ABMS and the modelling of agent behaviours. To achieve this aim, three research objectives are identified in Section 3.1. The objectives are evaluated as follows:

Objective 1: Construct a development framework that is tailored to ABMS of PGG by incorporating tools from software engineering. The first objective was fulfilled by the design of Framework v1 in Section 3.5. The design decisions and relevant knowledge were also discussed. Moreover, the framework was evaluated, revised, improved from v1 to v2 in Chapter 4, and from v2 to v3 in Chapter 5.

Objective 2: In the development framework, specify a behaviour modelling approach for dynamic behaviours in discrete as well as continuous time. The second objective was achieved in the framework v3 by using the sequence diagrams in combination with statecharts. The sequence diagrams help define the interaction between agents over time, which can be difficult to define with other methods. On the other hand, statecharts have proved their capability and flexibility in modelling dynamic behaviour of agents in three case studies, and has thus been suggested in the framework. The statechart approach was used for both discrete and continuous time. In the last case study, it was successful at modelling complex behaviours which are enabled by the continuous-time environment.

Objective 3: Evaluate the framework in three aspects: modelling principles, embedding of psychological mechanisms, and discrete vs. continuous time settings. To meet the third objective, three case studies, corresponding to three aspects, were carried out. The case study demonstrated the ability of the framework to work well in various situations. The evaluation of the first case study focused on the macro process (the development lifecycle); while the second and third case studies evaluated the micro process (focus on analysis and design processes). The three case studies also provided insights to improve the framework from v1 to v3.

7.4 Contribution to Knowledge

The main contribution that this thesis makes to the body of knowledge is a methodology of developing agent-based models of PGGs using well established tools from SE. The methodology is manifested in form of a framework that has been developed throughout the thesis and tested in several case studies. The framework utilizes OOAD to provide a structured approach to identify agent interactions, and design simulation architecture and agent behaviour. It includes a development lifecycle, a step-by-step development guideline, and a guide to model agent behaviour with statecharts. These statecharts allow us to better capture the realism of human decision making behaviour.

Another contribution that this thesis makes is an agent-based model of a PGG in continuous-time setting, based on the lab experiments of Oprea et al. (2014). To the best of our knowledge such agent-based models do not exist. In this context we also introduced a new preference type.

To better explain the dynamics in non-communication continuous-time setting, a preference type “Generous Conditional Cooperator” was introduced, based on the observations of the data from [Oprea et al.](#)’s lab experiments. In the end we were able to show with our agent-based model experiments that continuous time PGGs generate dynamics that are not present in discrete time PGGs. Therefore, it is important to study discrete as well as continuous time games, either as laboratory or as agent based model experiments. Our new framework allows one to do the latter in a structured way.

7.5 Limitations and Future Works

Several directions could be followed from the research presented in this thesis. In terms of strategic interactions, the framework could be tested with other variations of the PGG. Some ideas would include spatial PGG in which agent locations affect its access to resources, or PGG with variant income. It could also be tested for related games. Some candidates would be Prisoner’s Dilemma and Common Pool Resources. Furthermore, the application of the framework could be tested for related disciplines which also use the PGG as a research tool. These include Ecology ([Grimm and Railsback, 2013](#)) and Sociology ([Macy and Willer, 2002](#)). Since ABMS is utilized by many disciplines, it is beneficial to use a unified development framework, and our framework is a step toward that goal.

In addition, the simulation can also be implemented with other simulation packages or programming languages to validate the framework’s independence of simulation softwares. AnyLogic is not a favourite choice in Economics. Even though it supports object-orientation very well and works well with complex statecharts, it is a commercial package and has a steep learning curve. It would be good to test the implementability of models developed with this framework in other packages. This would allow everyone to use his/her preferred software package and would support the spread of the framework. One recommendation is Repast Symphony since it has good support for object-orientation and graphical statecharts. Another approach is to implement using a general programming language such as Java or C/C++. With a general programming language, there are less restrictions compared to a simulation package, but some supported features (statechart support, statistical presentations) have to be implemented from scratch. The other problem is that most non computer scientists are not professional programmers, thus the learning curve for a general programming language is steeper than that of a simulation package.

Lastly, for the continuous-time PGG, more laboratory experiments are required to confirm the existence of the proposed social preference type Generous Conditional Cooperators. Further

investigation on other behavioural aspects such as learning or sanction behaviours in continuous time compared to discrete time is needed as well.

References

- Allais, M. (1953). L'extension des theories de l'équilibre economique general et du rendement social au cas du risque. *Econometrica*, 21(2):269–290.
- Anderson, P. W., Arrow, K. J., and Pines, D. (1988). *The economy as an evolving complex system: the proceedings of the evolutionary paths of the global economy workshop, held September, 1987, in Santa Fe, New Mexico*. Addison - Wesley.
- Andrighetto, G., Brandts, J., Conte, R., Sabater-Mir, J., Solaz, H., and Villatoro, D. (2013). Punish and Voice: Punishment Enhances Cooperation when Combined with Norm-Signalling. *PLoS ONE*, 8(6):e64941.
- Arthur, W. B., Durlauf, S. N., and Lane, D. A. (1997). *The economy as an evolving complex system II*. Addison-Wesley, Reading, MA.
- Axelrod, R. (1997). Advancing the art of simulation in the social sciences. *Simulating Social Phenomena*, pages 21–40.
- Bagni, R., Berchi, R., and Cariello, P. (2002). A comparison of simulation models applied to epidemics. *Journal of Artificial Societies and Social Simulation*, 5(3).
- Banks, J. (1998). *Handbook of simulation: principles, methodology, advances, applications, and practice*. John Wiley & Sons.
- Banks, J., Plott, C., and Porter, D. (1988). An experimental analysis of unanimity in public goods provision mechanisms. *The Review of Economic Studies*, 55:301–322.
- Bauer, B., Bergenti, F., Massonet, P., and Odell, J. J. (2001a). Agents and the UML: A Unified Notation for Agents and Multi-agent Systems? In *International Workshop on Agent-Oriented Software Engineering*, pages 148–150. Springer Berlin Heidelberg.
- Bauer, B., Müller, J. P., and Odell, J. (2001b). Agent UML: A Formalism for Specifying Multi-agent Interaction. *International journal of software engineering and knowledge engineering*, 11(03):207–230.
- Bauer, B. and Odell, J. (2005). UML 2.0 and agents: how to build agent-based systems with the new UML standard. *Engineering Applications of Artificial Intelligence*, 18(2):141–157.
- Bergin, J. and MacLeod, W. B. (1993). Continuous Time Repeated Games. *International Economic Review*, 34(1):21.
- Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk. *Econometrica*, 22(1):23–36.

- Bersini, H. (2012). UML for ABM. *Journal of Artificial Societies and Social Simulation*, 15(1):9.
- Bertalanffy, L. V. (1968). *General system theory: Foundations, development, applications*. George Braziller Inc.
- Bharwani, S. (2004). *Adaptive Knowledge Dynamics and Emergent Artificial Societies: Ethnographically Based Multi-Agent Simulations of Behavioural Adaptation in Agro-Climatic Systems*. PhD thesis, University of Kent, Canterbury, UK.
- Binmore, K. G. (2012). *Playing for Real: a Text on Game Theory*. Oxford University Press, coursepack edition.
- Blizzard Entertainment (2016). World of Warcraft.
- Bloomquist, K. M. (2006). A Comparison of Agent-Based Models of Income Tax Evasion. *Social Science Computer Review*, 24(4):411–425.
- Blume, L. E. and Durlauf, S. N. (2006). *The economy as an evolving complex system, III: current perspectives and future directions*. Oxford University Press.
- Bohm, P. (1972). Estimating demand for public goods: An experiment. *European Economic Review*, 3(2):111–130.
- Booch, G. (2007). *Objected-Oriented Analysis and Design with Applications*. Addison-Wesley.
- Borel, E. (1921). La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes rendus de l'Académie des Sciences*, 173(1304-1308):58.
- Bradshaw, J. M. (1997). *Software agents*. MIT Press.
- Branas-Garza, P. and Barreda, I. (2015). Experiments in Economics. In *Experimental Economics*, pages 1–16. Palgrave Macmillan UK.
- Brandt, H., Hauert, C., and Sigmund, K. (2003). Punishment and reputation in spatial public goods games. *Proceedings. Biological sciences / The Royal Society*, 270(1519):1099–1104.
- Camerer, C. F. (2003). *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press.
- Camerer, C. F. (2008). Behavioral game theory and the neural basis of strategic choice. *Neuroeconomics. Decision-Making and the Brain*, pages 193–206.
- Camerer, C. F., Loewenstein, G., and Rabin, M. (2004). *Advances in Behavioral Economics*. Princeton University Press.
- Carver, C. S. and White, T. L. (1994). Behavioral inhibition, behavioral activation, and affective responses to impending reward and punishment: The BIS/BAS Scales. *Journal of Personality and Social Psychology*, 67(2):319–333.
- Chamberlin, E. H. (1948). An experimental imperfect market. *The Journal of Political Economy*, pages 95–108.

- Chen, S. H. (2012). Varieties of agents in agent-based computational economics: A historical and an interdisciplinary perspective. *Journal of Economic Dynamics and Control*, 36(1):1–25.
- Collins, A., Petty, M., Vernon-Bido, D., and Sherfey, S. (2015). A Call to Arms: Standards for Agent-Based Modeling and Simulation. *Journal of Artificial Societies and Social Simulation*, 18(3):12.
- Conte, R., Gilbert, N., and Sichman, J. S. (1998). MAS and Social Simulation: A Suitable Commitment. In *Multi-Agent Systems and Agent-Based Simulation*, pages 1–9. Springer.
- Conway, S. R. (2006). An Agent-Based Model for Analyzing Control Policies and the Dynamic Service-Time Performance of a Capacity-Constrained Air Traffic Management Facility. In *ICAS 2006 - 25th Congress of the International Council of the Aeronautical Sciences*, Hamburg, Germany.
- Cossentino, M., Hilaire, V., Molesini, A., and Seidita, V. (2014). *Handbook on Agent-Oriented Design Processes*. Springer Science & Business Media.
- Deitel, P. and Deitel, H. (2011). *Java How to Program*. Prentice Hall Press.
- Dinther, C. V. (2008). Agent-based simulation for research in economics. *Handbook on Information Technology in Finance*.
- Dixit, A. K. and Skeath, S. (2015). *Games of Strategy: Fourth International Student Edition*. WW Norton & Company.
- Dorsey, R. E. (1992). The voluntary contributions mechanism with real time revisions. *Public Choice*, 73(3):261–282.
- Drogoul, A., Vanbergue, D., and Meurisse, T. (2003). Multi-agent Based Simulation: Where Are the Agents? *Multi-Agent-Based Simulation II. Third International Workshop, MABS 2002 Bologna, Italy, July 15–16, 2002 Revised Papers*, 2581:1–15.
- Duffy, J. (2006). Agent-Based Models and Human Subject Experiments. *Handbook of Computational Economics*, 2(05):949–1011.
- Ebenhöh, E. (2006). *Modelling human behaviour in social dilemmas using attributes and heuristics*. PhD thesis, University of Osnabrück.
- Economist (2010). Agents of change: Conventional economic models failed to foresee the financial crisis. Could agent-based modelling do better?
- Edmonds, B. and Moss, S. (2005). From KISS to KIDS—an ‘anti-simplistic’ modelling approach. *Multi-Agent and Multi-Agent-Based Simulation*, pages 130–144.
- Electronic Arts (2016). The Sims - Official Site.
- Ellsberg, D. (1961). Risk, Ambiguity, and the Savage Axioms. *The Quarterly Journal of Economics*, 75(4):643.
- Elsenbroich, C. and Gilbert, N. (2014). Agent-Based Modelling. In *Modelling Norms*, pages 65–84. Springer Netherlands, Dordrecht.

- Epstein, J. M. (2006). Remarks on the Foundations of Agent-Based Generative Social Science.
- Farjam, M., Faillo, M., Sprinkhuizen-Kuyper, I., and Haselager, P. (2015). Punishment mechanisms and their effect on cooperation: A simulation study. *JASSS*, 18(1).
- Farmer, J. D. and Foley, D. (2009). The economy needs agent-based modelling. *Nature*, 460(7256):685–6.
- Fehr, E. and Gächter, S. (2002). Altruistic punishment in humans. *Nature*, 415(6868):137–40.
- Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison Wesley.
- Fischbacher, U. (2007). z-Tree: Zurich toolbox for ready-made economic experiments. *Experimental Economics*, 10(2):171–178.
- Fischbacher, U. and Gächter, S. (2010). Social preferences, beliefs, and the dynamics of free riding in public goods experiments. *American Economic Review*, 100(1):541–556.
- Fischbacher, U., Gächter, S., and Fehr, E. (2001). Are people conditionally cooperative? Evidence from a public goods experiment. *Economics Letters*, 71(3):397–404.
- Flood, M. M. (1958). Some Experimental Games. *Management Science*, 5(1):5–26.
- Forrester, J. W. (1958). Industrial dynamics: a major breakthrough for decision makers. *Harvard business review*, 36(4):37–66.
- Fowler, M. (2004). *UML distilled: a brief guide to the standard object modeling language*. Addison-Wesley Professional.
- Friedman, D. and Oprea, R. (2012). A continuous dilemma. *American Economic Review*, 102(1):337–363.
- Garrido, J. (2011). *Introduction to Elementary Computational Modeling: Essential Concepts, Principles, and Problem Solving*. CRC Press.
- Gilbert, N. (2007). *Agent-Based Models (Quantitative Applications in the Social Sciences)*. Sage Publications Inc.
- Gilbert, N. and Troitzsch, K. (2005). *Simulation for the social scientist*. McGraw-Hill Education (UK).
- Goeree, J. and Holt, C. (2001). Ten little treasures of game theory and ten intuitive contradictions. *American Economic Review*.
- Goldstein, J. (1999). Emergence as a construct: History and issues. *Emergence*.
- Gotts, N., Polhill, J., and Law, A. (2003). Agent-based simulation in the study of social dilemmas. *Artificial Intelligence Review*, 19(1):3–92.
- Graham, I., O’Callaghan, A., and Wills, A. C. (2000). *Object-oriented methods: principles & practice*. Addison-Wesley.

- Greene, W. H. (2012). *Econometric Analysis*. Pearson.
- Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., and Railsback, S. F. (2010). The ODD protocol: A review and first update. *Ecological Modelling*, 221(23):2760–2768.
- Grimm, V. and Railsback, S. (2013). *Individual-based modeling and ecology*. Princeton University Press.
- Grimm, V. and Railsback, S. F. (2006). Agent-Based Models in Ecology: Patterns and Alternative Theories of Adaptive Behaviour. *Agent-based computational modelling*, pages 139–152.
- Hamill, L. and Gilbert, N. (2016). *Agent-Based Modelling in Economics*. John Wiley & Sons.
- Hamilton, W. D., Axelrod, R., and Tanese, R. (1990). Sexual reproduction as an adaptation to resist parasites (a review). *Proceedings of the National Academy of Sciences of the United States of America*, 87(9):3566–73.
- Hassan, S., Antunes, L., Pavon, J., and Gilbert, N. (2008). Stepping on Earth: A Roadmap for Data-driven Agent-Based Modelling. *Proceedings of the 5th Conference of the European Social Simulation Association (ESSA08)*.
- Hauert, C., Monte, S. D., Hofbauer, J., and Sigmund, K. (2002). Volunteering as red queen mechanism for cooperation in public goods games. *Science*.
- Heinze, C., Papasimeon, M., and Goss, S. (2000). Specifying agent behaviour with use cases. In *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, volume 1881, pages 128–142.
- Helbing, D. (2012). *Social self-organization : agent-based simulations and experiments to study emergent social behavior*. Springer.
- Helbing, D., Szolnoki, A., and Perc, M. (2010). Punish, but not too hard: how costly punishment spreads in the spatial public goods game. *New Journal of Physics*.
- Hummel, D. (1983). Aerodynamic aspects of formation flight in birds. *J. Theor. Biol.*, 104.
- Isaac, M. and Walker, J. (1988a). Communication and Free-Riding Behavior: The Voluntary Contribution Mechanism. *Economic Inquiry*, 26(4):585–608.
- Isaac, R., Walker, J. M., and Williams, A. W. (1994). Group size and the voluntary provision of public goods: Experimental evidence utilizing large groups. *Journal of Public Economics*, 54(1):1–36.
- Isaac, R. M., McCue, K. F., and Plott, C. R. (1985). Public goods provision in an experimental environment. *Journal of Public Economics*, 26:51–74.
- Isaac, R. M. and Walker, J. A. (1988b). Group Size Effects in Public Goods Provision: The Voluntary Contributions. *The Quarterly Journal of Economics*, 103(1):179–199.
- Janssen, M. a. and Ostrom, E. (2006). Empirically based, agent-based models. *Ecol. Soc.*, 11(2).

- Kagel, J. H. and Roth, A. E. (1995). *The Handbook of Experimental Economics*. Princeton University Press.
- Kahneman, D. and Tversky, A. (1979). Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2):263.
- Kalisch, G. K., Milnor, J. W., Nash, J. F., and Nering, E. D. (1954). Some experimental n-person games. *Decision Process*, pages 301–327.
- Kelton, W. D., Sadowski, R. P., and Sturrock, D. T. (2007). *Simulation with Arena*. McGraw-Hill Higher Education.
- Kim, O. and Walker, M. (1984). The free rider problem: Experimental evidence. *Public Choice*.
- Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997). Robocup: The robot world cup initiative. *Proceedings of the first international conference on Autonomous agents*, pages 340–347.
- Kohler, T. A., Gumerman, G. J., and Reynolds, R. G. (2005). Simulating Ancient Societies. *Scientific American*, 293(1):76–84.
- Kubera, Y., Mathieu, P., and Picault, S. (2010). Everything can be Agent! In *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1547–1548, Toronto, Canada. International Foundation for Autonomous Agents and Multiagent Systems.
- Kubera, Y., Mathieu, P., and Picault, S. (2011). IODA: An interaction-oriented approach for multi-agent based simulations. *Autonomous Agents and Multi-Agent Systems*, 23(3):303–343.
- Kumar, R. (2010). *Research Methodology: A Step-by-Step Guide for Beginners*. SAGE Publications.
- Laraki, R., Solan, E., and Vieille, N. (2005). Continuous-time games of timing. *Journal of Economic Theory*, 120(2):206–238.
- Ledyard, J. O. (1995). Public goods: A survey of experimental research. In *Handbook of Experimental Economics*, pages 111–194. Princeton University Press.
- Li, X. (2013). Standardization for Agent-based Modeling in Economics.
- Li, Z.-y., Tang, M., Liang, D., and Zhao, Z. (2016). Numerical Simulation of Evacuation in a Subway Station. *Procedia Engineering*, 135:616–621.
- Lissaman, P. B. S. and Shollenberger, C. A. (1970). Formation Flight of Birds. *Science*, 168(3934).
- Lucas, P., De Oliveira, A., and Banuri, S. (2014). The Effects of Group Composition and Social Preference Heterogeneity in a Public Goods Game: An Agent-Based Simulation. *Journal of Artificial Societies and Social Simulation*, 17(3):5.
- Macal, C. and North, M. (2014). Introductory tutorial: agent-based modeling and simulation.

- Macal, C. M. (2016). Everything You Need to Know about Agent-based Modelling and Simulation. *Journal of Simulation*, 10(2):144–156.
- Macal, C. M. and North, M. J. (2009). Agent-based modeling and simulation. *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 86–98.
- Macal, C. M. and North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3):151–162.
- Macy, M. W. and Willer, R. (2002). From Factors to Factors: Computational Sociology and Agent-Based Modeling. *Annual Review of Sociology*, 28(1):143–166.
- Malleson, N., Heppenstall, A., and See, L. (2010). Crime reduction through simulation: An agent-based model of burglary. *Computers, Environment and Urban Systems*, 34(3):236–250.
- Markowitz, H. (1952). The Utility of Wealth.
- Mast, E., van Kuik, G., and van Bussel, G. (2007). Agent-based modelling for scenario development of offshore wind energy. Technical report, Delft University of Technology, The Netherlands.
- Mock, K. and Testa, J. (2007). An agent-based model of predator-prey relationships between transient killer whales and other marine mammals. Technical report, University of Alaska Anchorage, Anchorage, AK, Tech. Rep.
- Müller, B., Balbi, S., Buchmann, C. M., de Sousa, L., Dressler, G., Groeneveld, J., Klassert, C. J., Le, Q. B., Millington, J. D., Nolzen, H., Parker, D. C., Polhill, J. G., Schlüter, M., Schulze, J., Schwarz, N., Sun, Z., Taillandier, P., and Weise, H. (2014). Standardised and transparent model descriptions for agent-based models: Current status and prospects. *Environmental Modelling & Software*, 55:156–163.
- Ndemic Creations (2016). Homepage - Ndemic Creations.
- Neyman, A. (2012). *Continuous-time Stochastic Games*. Center for the Study of Rationality.
- North, M. J. and Macal, C. M. (2007). *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*. Oxford University Press.
- Odell, J., Parunak, H., Bauer, B., and Arbor, A. (2000). Extending UML for Agents. In *Proceedings of the Agent Oriented Information Systems*. Springer.
- O’Hare, G. M. P. and Jennings, N. (1996). *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons.
- Omicini, A., Ricci, A., and Viroli, M. (2008). Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456.
- Oprea, M. (2004). Applications of multi-agent systems. *Information Technology: Selected Tutorials*, 157(39):239–270.
- Oprea, R., Charness, G., and Friedman, D. (2014). Continuous time and communication in a public-goods experiment. *Journal of Economic Behavior and Organization*, 108:212–223.

- Pahl-Wostl, C. and Ebenhoh, E. (2004). An Adaptive Toolbox Model: a pluralistic modelling approach for human behaviour based. *Jasss-the Journal of Artificial Societies and Social Simulation*, 7(1).
- Paur, J. (2011). Try Before You Fly: How Dreamliner Pilots Train Without Lifting Off.
- Poteete, A. R., Janssen, M. A., and Ostrom, E. (2010). *Working together: collective action, the commons, and multiple methods in practice*. Princeton University Press.
- Railsback, S. F. and Grimm, V. (2011). *Agent-based and individual-based modeling: a practical introduction*. Princeton University Press.
- Rasmusen, E. and Blackwell, B. (1994). Games and information. *Cambridge, MA*.
- Reynolds, C. W., W., C., Reynolds, and W., C. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34.
- Richiardi, M., Leombruni, R., Saam, N., and Sonnessa, M. (2006). A Common Protocol for Agent-Based Social Simulation.
- Richiardi, M. G., Chu, D., Strand, R., Fjelland, R., Epstein, J. M., Fagiolo, G., Moneta, A., Windrum, P., Farmer, J. D., Foley, D., Gouriéroux, C., Monfort, A., Hodgson, G. M., Horgan, J., Kleijnen, J. P. C., Leombruni, R., Richiardi, M., Marks, R. E., Ostrom, T. M., Phelan, S. E., Rosser, J. B., Schelling, T. C., Schumpeter, J., Smith, V. L., Vriend, N. J., Winker, P., Gilli, M., Jeleskovic, V., and Grazzini, J. (2012). Agent-based computational economics: a short introduction. *The Knowledge Engineering Review*, 27(02):137–149.
- Robinson, S. (2004). *Simulation: The Practice of Model Development and Use*. Hoboken, NJ: John Wiley & Sons.
- Romanowska, I. (2015). So You Think You Can Model ? A Guide to Building and Evaluating Archaeological Simulation Models of Dispersals. *Human Biology*, 87(3).
- Rossiter, S. (2015). Simulation Design: Trans-Paradigm Best-Practice from Software Engineering.
- Rousseas, S. W. and Hart, A. G. (1951). Experimental verification of a composite indifference map. *The Journal of Political Economy*, pages 288–318.
- Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (2003). *Artificial intelligence: a modern approach*. Prentice Hall.
- Sannikov, Y. (2007). Games with Imperfectly Observable Actions in Continuous Time. *Econometrica*, 75(5):1285–1329.
- Santos, F., Santos, M., and Pacheco, J. (2008). Social diversity promotes the emergence of cooperation in public goods games. *Nature*.
- Sarkar, S. (2013). Plague Inc. creator invited to speak at CDC about spread of infectious disease.
- Schelling, T. C. (1957). Bargaining, communication, and limited war. *Conflict Resolution*, 1(1):19–36.

- Scherr, B. A. and Babb, E. M. (1975). Pricing public goods. *Public Choice*, 23(1):35–48.
- Shannon, R. E. (1975). *Systems simulation: the art and science*. Prentice-Hall.
- Shehory, O., Sycara, K., Sukthankar, G., and Mukherjee, V. (1999). Agent aided aircraft maintenance. *Proceedings of the third annual conference on Autonomous Agents - AGENTS '99*, pages 306–312.
- Siebers, P. O. and Aickelin, U. (2011). A first approach on modelling staff proactiveness in retail simulation models. *Journal of Artificial Societies and Social Simulation*, 14(2):2.
- Siebers, P.-O. and Davidsson, P. (2015). Engineering Agent-Based Social Simulations: An Introduction. *Journal of Artificial Societies and Social Simulation*, 18(3):13.
- Siegel, S. and Fouraker, L. E. (1960). *Bargaining and group decision making: Experiments in bilateral monopoly*, volume 19. McGraw-Hill.
- Simon, H. A. (1955). A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics*, 69(1):99–118.
- Simon, L. K. and Stinchcombe, M. B. (1989). Extensive Form Games in Continuous Time: Pure Strategies. *Econometrica*, 57(5):1171.
- Skatova, A. and Ferguson, E. (2013). Individual differences in behavioural inhibition explain free riding in public good games when punishment is expected but not implemented. *Behavioral and brain functions : BBF*, 9(1):3.
- Smith, E. R. and Conrey, F. R. (2007). Agent-based modeling: a new approach for theory building in social psychology. *Personality and social psychology review : an official journal of the Society for Personality and Social Psychology, Inc*, 11(1):87–104.
- Smith, V. L. (1962). An Experimental Study of Competitive Market Behavior. *Journal of Political Economy*, 70(2):111–137.
- Smith, V. L. (1964). Effect of Market Organization on Competitive Equilibrium. *The Quarterly Journal of Economics*, 78(2):181.
- Smith, V. L. (1979a). An Experimental Comparison of Three Public Good Decision Mechanisms. *The Scandinavian Journal of Economics*, 81(2):198–215.
- Smith, V. L. (1979b). Incentive Compatible Experimental Processes for the Provision of Public Goods. In *Research in Experimental Economics*. JAI Press.
- Smith, V. L. (1980). Experiments with a decentralized mechanism for public good decisions. *American Economic Review*, 70(4):584–599.
- Strotz, R. H. (1955). Myopia and Inconsistency in Dynamic Utility Maximization.
- Sweeney, J. W. (1973). An experimental investigation of the free-rider problem. *Social Science Research*, 2(3):277–292.

- Taylor, R. (2003). *Agent-Based Modelling Incorporating Qualitative and Quantitative Methods: A Case Study Investigating the Impact of E-commerce upon the Value Chain*. PhD thesis, Manchester Metropolitan University, Manchester, UK.
- Tesfatsion, L. (2006). Agent-based Computational Economics: A constructive approach to economic theory. In *Handbook of Computational Economics, Volume 2: Agent-Based Computational Economics*, pages 831–880. Elsevier.
- Thaler, R. (1985). Mental Accounting and Consumer Choice. *Marketing Science*, 4(3):199–214.
- Thurstone, L. L. (1931). The Indifference Function. *The Journal of Social Psychology*, 2(2):139–167.
- Ubisoft (2016). Collapse.
- Vermeir, A. and Bersini, H. (2015). Best Practices in Programming Agent-Based Models in Economics and Finance. In Amblard, F., Miguel, F. J., Blanchet, A., and Gaudou, B., editors, *Advances in Artificial Economics*, pages 57–68. Springer International Publishing, Cham.
- Von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320.
- Von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.
- Wallis, W. A. and Friedman, M. (1942). The empirical derivation of indifference functions. In *Studies in mathematical economics and econometrics: in memory of Henry Schultz*, pages 175–189. Chicago: University of Chicago Press.
- Weibull, J. W. (2004). Testing Game Theory. In *Advances in Understanding Strategic Behaviour*, pages 85–104. Palgrave Macmillan UK, London.
- Weimerskirch, H., Martin, J., Clerquin, Y., Alexandre, P., and Jiraskova, S. (2001). Energy saving in flight formation. *Nature*, 413(6857):697–8.
- Wickens, M. (2008). *Macroeconomic theory: a dynamic general equilibrium approach*. Princeton University Press.
- Wilensky, U. (1999). NetLogo.
- Wilensky, U. (2003). NetLogo HubNet Public Good model.
- Wilensky, U. and Rand, W. (2015). *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT Press.
- Wilkinson, N. and Klaes, M. (2012). *An introduction to behavioral economics*. Palgrave Macmillan.
- Wooldridge, M. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152.

- Wooldridge, M., Jennings, N. R., and Kinny, D. (1999). A Methodology for Agent-Oriented Analysis and Design. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 69–76, New York, NY, USA. ACM.
- Wooldridge, M. and Ciancarini, P. (2001). Agent-Oriented Software Engineering: The State of the Art. In *Agent-Oriented Software Engineering*, pages 1–28. Springer Berlin Heidelberg.
- Wunder, M., Suri, S., and Watts, D. (2013). Empirical Agent Based Models of Cooperation in Public Goods Games. *ACM 2013 June 16-20*, 1(212):891–908.
- XJTechnologies (2014). AnyLogic.
- Yin, R. K. (2009). *Case Study Research: Design and Methods*. SAGE Publications.
- Zhang, T., Siebers, P.-O., and Aickelin, U. (2016). Simulating user learning in authoritative technology adoption: An agent based model for council-led smart meter deployment planning in the UK. *Technological Forecasting and Social Change*, 106:74–84.

Appendix A

The NetLogo Implementation of Case Study 1 - KISS approach

```
breed [players player]
```

```
players-own[  
  strategy  
  expected-cooperativeness  
  learning-rate  
  decision  
  free-time  
  
  inclination-to-annoyed  
  willing-to-punish  
  punish-decision  
  expect-be-punished  
]
```

```
to setup  
  clear-all  
  create-players 4  
  initiate-players-own-variables  
  reset-ticks  
end
```

```
to initiate-players-own-variables  
  ask players[
```

```

    set expected-cooperativeness default-expected-cooperativeness
    set learning-rate default-learning-rate
    set free-time 10

    set inclination-to-annoyed random-float 1
    set willing-to-punish random-float 1
    set expect-be-punished willing-to-punish
]

ask player 0 [set strategy "cooperate"]
ask player 1 [set strategy "reciprocate"]
ask player 2 [set strategy "maximize"]
ask player 3 [set strategy "maximize"]
end

to go
  ask players [set free-time 10]
  decide-investment
  update-belief
  if punishment? = true [punish]
  tick
end

to go10
  foreach n-values 11 [?] [ go ]
end

to decide-investment
  ask players with [strategy = "cooperate"] [set decision
    decision-cooperator]
  ask players with [strategy = "reciprocate"] [set decision
    decision-reciprocator]
  ask players with [strategy = "maximize"] [set decision decision
    -maximizer]
end

to-report decision-cooperator
  let mydecision 0
  ifelse (expected-cooperativeness > 0.4)

```

```

    [set mydecision 1]
    [set mydecision 0.75]
    report mydecision
end

to-report decision-reciprocator
    let mydecision expected-cooperativeness
    report mydecision
end

to-report decision-maximizer
    let mydecision 0
    let highest-outcome -1000
    let decision-list n-values 11 [? / 10] ;list (0.1 0.2 0.3 ...
        1)

    foreach decision-list [
        let outcome (10 - ( ? * 10) + ( ? + expected-cooperativeness
            * 3) * 10 * 0.6 / 4)
        if punishment? = true AND expected-cooperativeness > ? [
            set outcome (outcome - (expect-be-punished * 3) * 3 * 5)
        ]
        if (outcome > highest-outcome) [
            set highest-outcome outcome
            set mydecision ?
        ]
    ]
    report mydecision
end

to update-belief
    ask players [
        let mean-decision-others (mean [decision] of other players)
        set expected-cooperativeness ((1 - 0.5) * expected-
            cooperativeness + 0.5 * mean-decision-others)
    ]
end

to punish

```

```

ask players with [strategy = "cooperate" OR strategy = "
  reciprocate"] [
  let mean-decision-others (mean [decision] of other players)
  let anger-level 0
  ask other players [
    let deflection (([mean-decision-others] of myself) -
      decision)
    if deflection < 0 [
      set deflection 0
    ]
    ifelse deflection > 0.1
      [ask myself [set anger-level (inclination-to-annoyed +
        deflection)]]
      [ask myself [set anger-level (inclination-to-annoyed +
        deflection - 0.8)]]
    ifelse (random-float 1) < (2 * [anger-level] of myself)
      [ask myself [set punish-decision (willing-to-punish +
        deflection) / 2]]
      [ask myself [set punish-decision 0]]
    if [punish-decision] of myself > 0
      [set expect-be-punished (1 - learning-rate) * expect-be-
        punished + learning-rate * [punish-decision] of myself
      ]
  ]
]
end

```

Appendix B

Contributions in Non-Communication Continuous-time PGG

This appendix presents the figures of contributions in non-communication continuous-time PGG ([Oprea et al., 2014](#)). There are five groups corresponding to five figures. In each figure, there are four colour-coded lines representing four players. The x axis is the time in seconds. The y axis is the contribution ranged from 0 to 25.

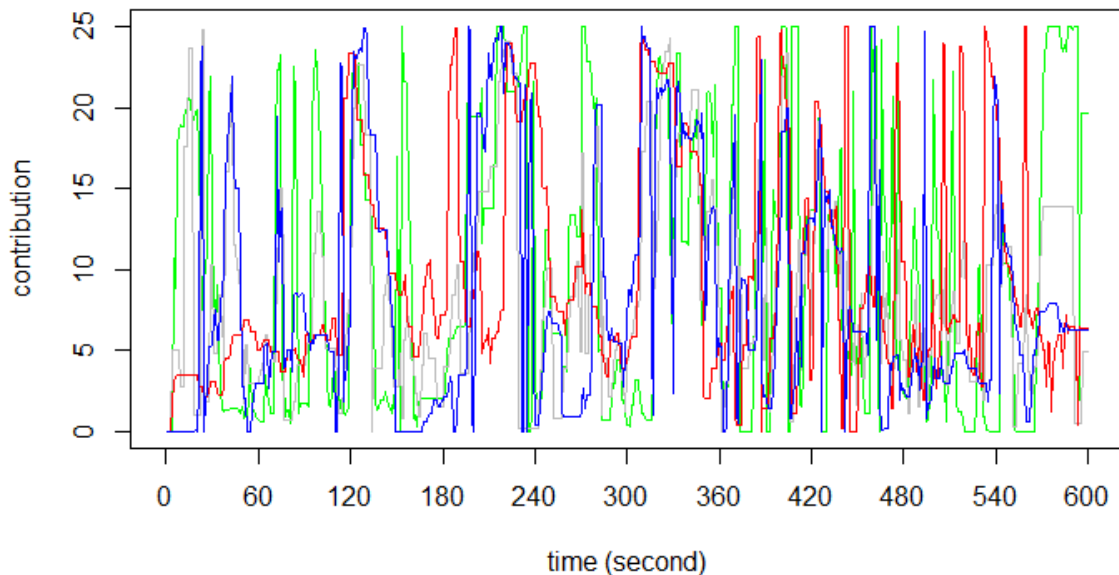


Fig. B.1 Group 1

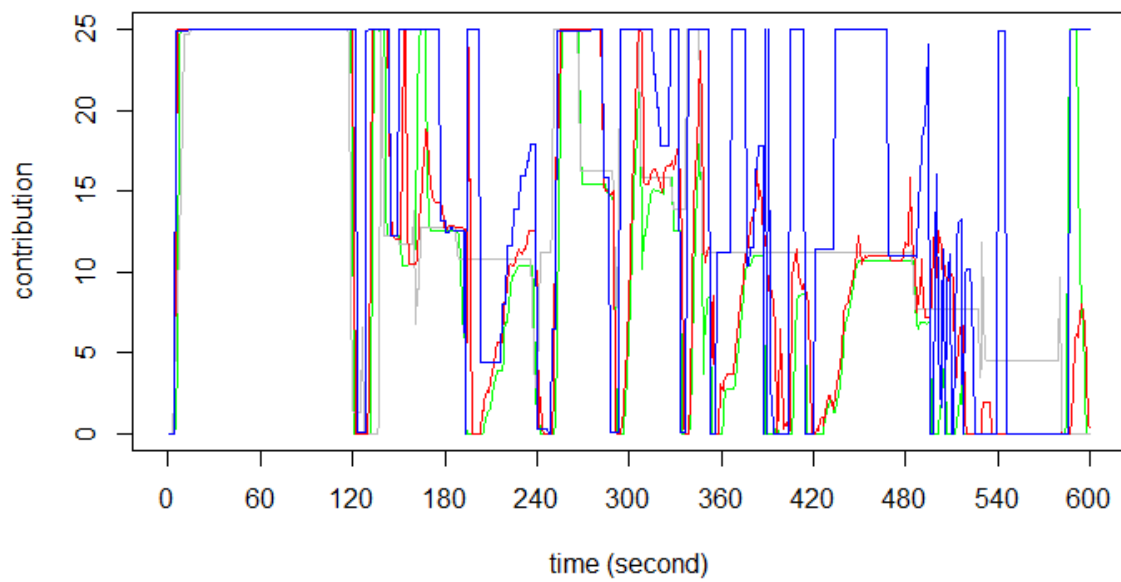


Fig. B.2 Group 2

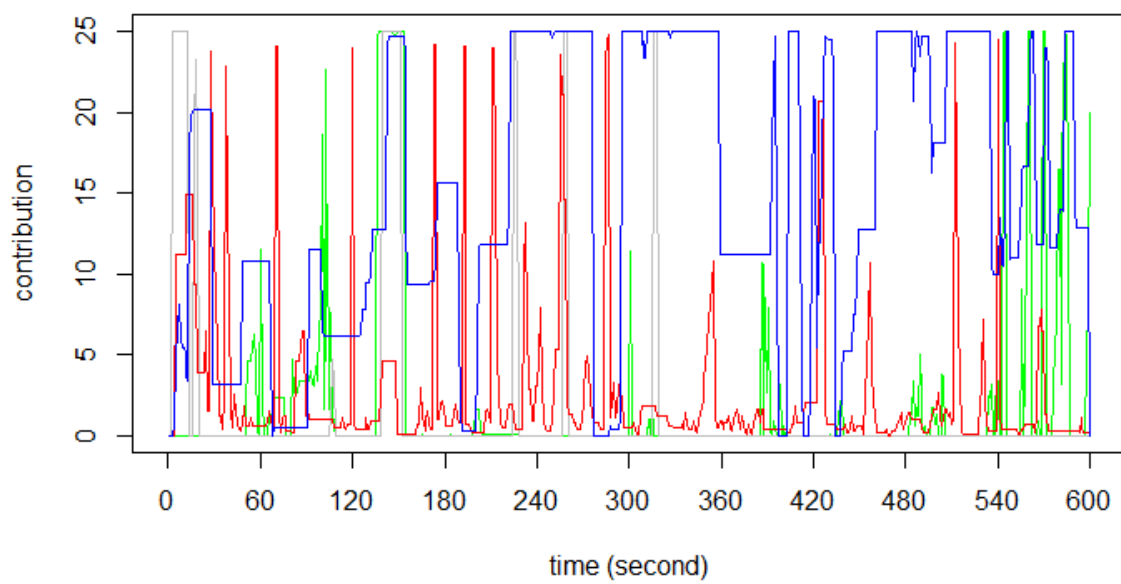


Fig. B.3 Group 3

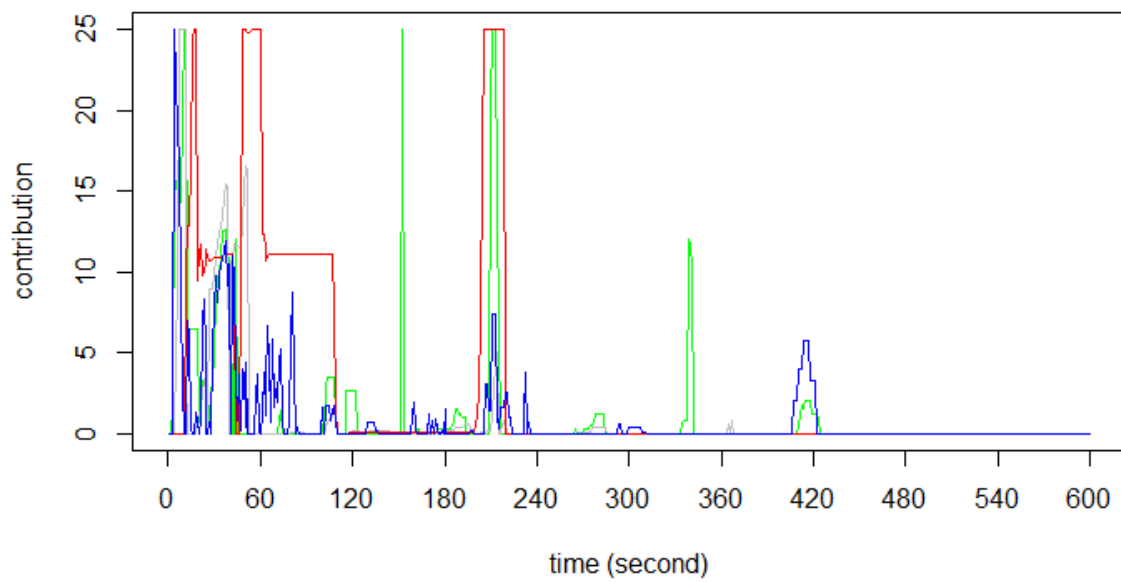


Fig. B.4 Group 4

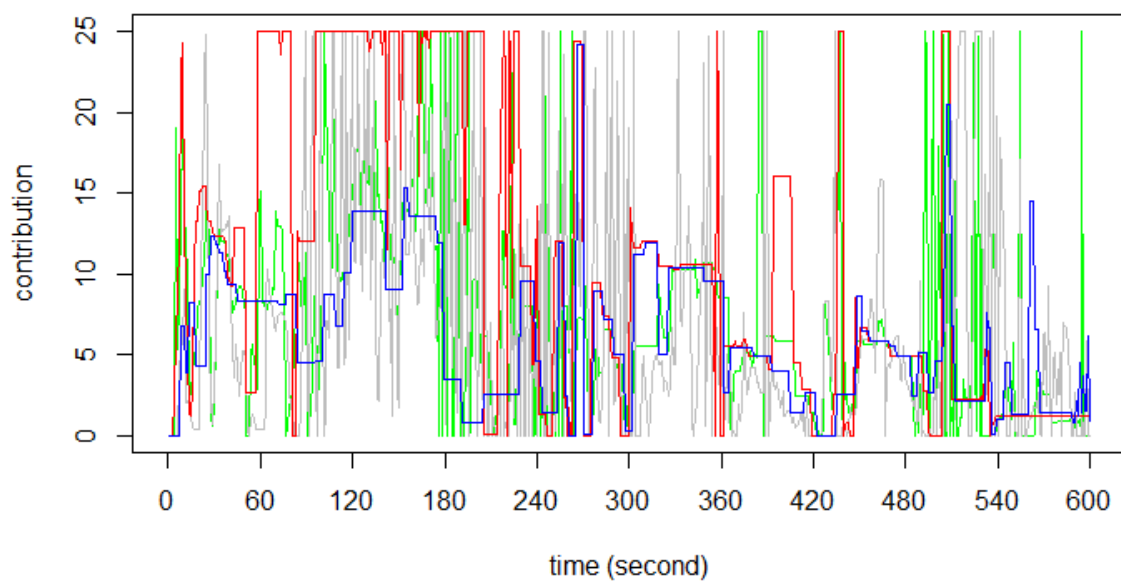


Fig. B.5 Group 5

Appendix C

Documentation of the ABOOMS Framework

Documentation of the ABOOMS framework consists of two sections: the macro process which guides the overall development lifecycle of the simulation study and the micro process in which the Object-Oriented Analysis and Design (OOAD) is applied.

C.1 The macro process

The overview lifecycle of the ABOOMS framework (Figure [C.1](#)) emphasizes the relationship between Public Goods Game (PGG) laboratory experiment and Agent-Based Modelling and Simulation (ABMS). The framework guides developers from a real life problem to an agent-based simulation. However, it is also designed to be applicable in the case that the study is based on theory and does not have a direct link to a real life problem. In that case, just remove the box of real life problem and continue to use the framework, start from the theory and hypothesis. In the framework, the OOAD is applied in the area of a dashed rounded rectangle. In addition, there are five processes (with bold labels) that will be described in detail in the next section.

C.2 The micro process

The steps of five processes used to develop the simulation are described in detail below:

Process 1 - Conceptual modelling

- Understand the problem situation.

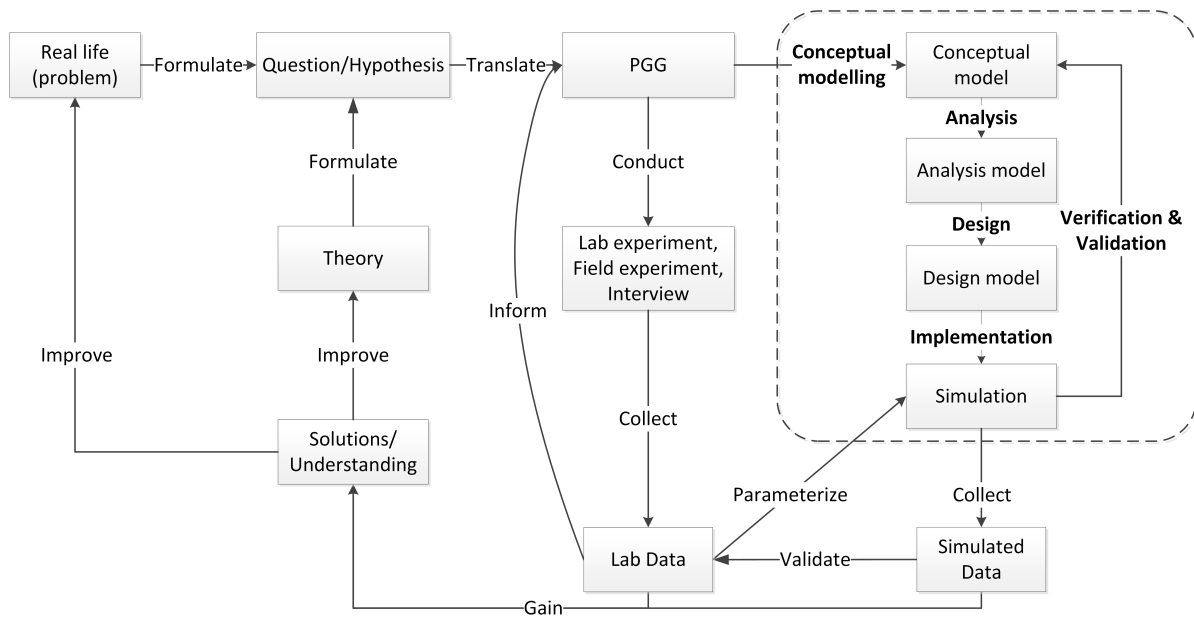


Fig. C.1 Overview lifecycle of the ABOOMS framework

- Determine objectives of the simulation study.
- Collect and analyse data and related theories.
- Identify the settings of the PGG.
 - Group structure: how many people in one group? How many groups there are in total?
 - One-shot or repeated games: The game is one-shot or repetitive. How many rounds of experiment are there?
 - Players' action: Action that the players can perform such as contribute, punish.
 - Game loop: The main flow of the game that Players have to follow. For example: Give initial endowment; Play a turn by letting agents choose cooperation level; Punish.
 - Scheduling: discrete-time/continuous-time and synchronous/asynchronous decision making.
 - Payoff function: $\pi_i = \omega - x_i + MPCR \sum_{j=1}^N x_j$, in which ω is the initial endowment, x_i is the contribution of participant i , N is the number of players in a group, MPCR is maginal private capital return ($MPCR < 1$).
 - Information: Incomplete or complete. Which information is available for the participants, and when?

- Design the conceptual model.
 - Identify the model inputs (experimental factors).
 - Identify the model outputs (responses).
 - Determine model content: scope and level of details.
 - Identify assumptions and simplification.

Process 2 - Analysis

- Identify agents, artifacts, and external actors:
 - Agents (internal actors): autonomous, proactive entities that encapsulate control and in charge of goals/tasks that altogether determine the whole system behaviour. For example, in PGG, players are agents.
 - Artifacts (internal actors): passive, reactive entities that are in charge of required services and functions that make agents work together and shape the agent environment. For example, an artifact in PGG can be “game master” to inform players of the current stage of the game.
 - External actors: External entities such as human and other systems.
- Identify use cases.
 1. A use case can be a behaviour to be exhibited by agents.
 2. A use case can also be an interaction between agents and external actors, or between agents.
 3. Refine the use case model by adding lower level functionalities and behaviours.
- Document all use cases.
 1. Use case name: The name of the use case
 2. Description: Describe what the use case involves, agents’ behaviour in context of environment, and interactions with other actors.
 3. Agents, artifacts, external actors: A list of actors involved in the use case.
 4. Use case associations: A list of other use cases associated with this use case (via uses or extends).
 5. Pre-conditions: A list of conditions that must be true before initialization of this use case.

6. Post-conditions: A list of conditions that must be true after this use case is completed.
 7. Flow of events: A list of activities or events that defines agents behaviour in this use case. This section can include a UML activity diagram to describe the flow of events.
 8. Alternative flows: A list of activities or events which agents can perform in exceptional circumstances, which are not handled by the main flow of events.
- Draw Use Case Diagram.

Process 3 - Design

- Define structure: Define the structure (internal data, operation) with Class Diagram.
- Define behaviour: Define behaviour with behavioural diagrams, such as Statechart, Sequence Diagram, Activity Diagram.

Process 4 - Implementation

- Choose a programming language and/or a software: The simulation can be programmed from scratch or with a specialist simulation toolkit (NetLogo, AnyLogic, Repast Symphony).
- Programming: Writing the code.
- Testing and debug: It is important to perform testing in order to ensure the correctness of the model.

Process 5 - Verification and Validation

- Verify whether the simulation correctly implemented the conceptual model.
- Validate and tune agent behaviour based on the information from the lab or literature.

C.3 Behavioural modelling with statechart

Steps to modelling behaviour with statecharts:

- List all the elements that affect agent decision making.
- Consider a piece of information or a variable that can only take one value from a fixed set of pre-defined values.

- Decide how many statecharts are needed.
- For each statechart, identify: states, transitions, triggered conditions, and guards.

The following are some useful tips on implementing statecharts:

- An agent can have multiple statecharts.
- From one state to another, there can be many transitions with different triggered conditions.
- Transitions in statecharts are the driving force behind statechart behaviours. Transitions can be triggered by:
 - Probability.
 - Conditions based on variables.
 - Messages from other agents.
 - Other statecharts: the current state of statecharts, or messages between statecharts.

Statecharts suggestion for agents in PGG:

- Action statechart: represent actions that agents can perform. It describes a sequence of actions in PGG. These actions are most likely to have tight relationships to the stages of the game. There are two designs for this statecharts: central design like Figure 4.9 of the first case study, or circular design like Figure 5.3 of the second case study. In Figure 4.9, there are a dummy state “Idle” in the middle and agents are flexible when changing state. On the other hand, in Figure 5.3, agents have to follow a sequence of states.
- Strategy statechart: represents strategies of agents in a game. This is suitable for the situation where agents change their strategies and there are many complicated conditions that trigger these. It can be complex by putting all strategies into one statechart such as Figure 5.11, or splitting into smaller statecharts as in Figure 6.6.
- Mental statechart: represents mental states of agents. For example, Figure 5.5 is simple but can model complex behaviour when combining with other statecharts.
- Spatial statechart: represents location of agents. This can be used in a spatial PGG where the decision of agents is affected by their location.