Simulating Social Relations in Multi-Agent Systems

Brendan J. Neville

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the
Diploma of Membership of the Imperial College

August 4, 2011

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

BRENDAN J. NEVILLE

Abstract

Open distributed systems are comprised of a large number of heterogeneous nodes with disparate requirements and objectives, a number of which may not conform to the system specification. This thesis argues that activity in such systems can be regulated by using distributed mechanisms inspired by social science theories regarding similarity/kinship, trust, reputation, recommendation and economics. This makes it possible to create scalable and robust agent societies which can adapt to overcome structural impediments and provide inherent defence against malicious and incompetent action, without detriment to system functionality and performance.

In particular this thesis describes:

- an agent based simulation and animation platform (PreSage), which offers the agent developer and society designer a suite of powerful tools for creating, simulating and visualising agent societies from both a local and global perspective.
- a social information dissemination system (SID) based on principles of self organisation which personalises recommendation and directs information dissemination.
- a computational socio-cognitive and economic framework (CS_CEF) which integrates and extends socio-cognitive theories of trust, reputation and recommendation with basic economic theory.
- results from two simulation studies investigating the performance of SID and the CS_CEF.

The results show the production of a generic, reusable and scalable platform for developing and animating agent societies, and its contribution to the community as an open source tool. Secondly specific results, regarding the application of SID and CS_CEF, show that revealing outcomes of using socio-technical mechanisms to condition agent interactions can be demonstrated and identified by using PreSage.

Acknowledgements

Let me kick off with the usual quantifiers, qualifiers and get out of trouble 'for forgetting you' statements. I'm going to keep my acknowledgements very succinct, so you're likely to be in the very large sub-set of the worlds population that I leave out. It may be cold comfort to you, but after all I'm less than six degrees separated from actor Kevin Bacon and he's not going to get a mention...(D'oh!). Its not personal either, we may be great friends, you might be one of the kindest people to have ever lived, but if I can't link you to this sodding Thesis you won't get a mention; except Mother Teresa of course. Now with that out of the way I'm going to get on with the serious business of insulting all the people in the subset I have included, by failing to sufficiently capture in a symbolic fashion their contribution to this thesis, my career and their personal importance to me.

I'd like to start with the kindest member of the group formerly known as Intelligent Information Systems (IIS), Shahareen Hilmy. Shahareen has always taken the time to help me even when I was an academic nobody, a young rebel in baggy trousers whipping around the halls on skates; and just last week as much older academic nobody with a beard. I'd also like to thank Professor Ebrahim Mamdani, the founding father of the IIS group, for whom my errands to the photocopier taught me about humility and paper jams, but more than that his dedication to science, technology and innovation created opportunities for all of us at IIS; we owe him a debt of gratitude and remembrance. Likewise Professor Patrick Purcell, who apart from helping me with my first papers and being a beacon of integrity, character and wit, to which we should all aspire to mimic; genuinely loved Vicky's baking. Professor Keith Clark deserves a mention simply for his sense of humour and infectious laugh, but I'll add that he gallantly stayed awake during my presentation at the PROMAS workshop long enough to impart on me his sage advice for my improvement. Rino Falcone and Cristiano Castelfranchi with whom my work on the Alfebiite project forms chapter five of this thesis, thank you for taking the time to answer my questions.

Moving onto to those who have had the (cough) pleasure of working with me over the

years, specifically, Alexander Artikis, Asimina Vasalou, Daniel Ramirez-Cano and Lloyd Kamara. I'd like to thank-you for bringing an air of professionalism to the proceedings, but mainly for your support and comradeship in the face of PTSD (PhD Thesis Stress Disorder). Lloyd unsurprisingly for those that know him gets an extra special mention, my Google-fu, my prowess with computing machines is down to him more than anyone, dude you're the man, wink, nod, fist bump, manly hug.

For putting a roof over my head last year I'd like to thank Dave and Christine Betts, I know at times it was tough for them to not change the locks. To my Parents Des and Joan Neville, thanks for bringing me up proper, and letting me shirk the washing up duties to do my homework, a genius bit of strategy on your part, as without that motivation I may never of made it to Imperial in the first place.

Dr Jeremy Pitt, my PhD supervisor, my line manager, my trusted advisor, my colleague, my friend. Thank you for the laughs, the experiences, and of course the travel. Most of all thank you for seeing in me the potential to finish this, even when I couldn't and after the viva for restricting yourself to double digit "I told you so's".

My last and most deeply felt gratitude goes to my wife Victoria Randall, I say wife but we aren't married, partner seems odd to me, girlfriend seems a tad informal and lodger she hates, mainly cause she pays my rent. Without her understanding (most of the time), patience (some of the time) and dedication to 'bringing home the bacon', this thesis would remain unfinished. Luckily beatification isn't run like the Oscars, otherwise Mother Teresa would be stuck with the lifetime achievement award.

Like I said I'm keeping it succinct. Do please continue reading it gets funnier towards page 147.

Contents

LIST OF FIGURES

List of Tables

INTRODUCTION 1

Networked computers and multi-agent systems (MAS) are commonly used as a platform for a new range of applications in for example manufacturing, health, transport, commerce, entertainment, education, and social interaction. Features of these applications include dynamic network infrastructures, heterogeneous components, unpredicted events, sub-ideal operation (failure to comply to specifications), incomplete and inconsistent information, absence of centralised control, and so on. Techniques from autonomic computing [82] and adaptive systems [41] have proved useful in addressing some of these features; for others the idea of an agent society has been proposed (e.g. [6, 112]) which has emphasised the need for conventional rules and social relationships between components.

There still remains though a requirement for system designers and software engineers to retain some understanding of the application under development, and especially of complex systems where random events, erratic behaviour, and self-modification can render the system behaviour difficult to a-priori analysis. In a complementary role alongside mathematical analysis, rapid prototyping has proved to be an extremely effective tool in helping to understand large-scale MAS, abstracting away from details in order to verify that certain desirable properties hold. However, in autonomic, large-scale MAS, there is an additional requirement not just to verify properties, but also to observe the global outcomes that are the consequence of local decision making with respect to conventional rules and social relations.

In this thesis, we propose a rapid prototyping tool PreSage whose emphasis is on the simulation of agent societies and the social relationships between agents, allowing the designer to study social behaviour of components, the evolution of network structures, and the adaptation of conventional rules. In this sense, it occupies a space distinct from powerful application development environments, like JADE [14]; agent based modelling and simulation tools [89, 91] where the primary purpose is to model and explain the behaviour of non-artificial agents; and other rapid prototyping environments for MAS (e.g. [90, 120]) whose principal function is, as stated above, to verify system-wide properties.

The agents forming these new applications will require the ability to reason and predict the behaviours of their peers in much the same way a conventional agent would reason about changes in the physical environment. However the actual intentions, beliefs etc of an agent's peer will likely be unknown, as such the agent's model of its peers must be solely based on their externally observable behaviours.

Given our analogy of complex systems as agent societies, we look to concepts from the social sciences that describe how human societies operate given such challenges: concepts relating to trust, peer selection and social networks. These concepts influence our design of "socio-cognitive" agents, whose modelling of their peers and formation of "social ties" enables them to perform in complex, dynamic applications. We demonstrate the effectiveness of this approach using PreSage to examine two prototype systems which utilise social relations.

1.1 Preliminary Definitions and Assumptions

This section provides a brief overview of a number of generally accepted concepts and approaches in order to put the methodology and contributions of this thesis into perspective. Namely, we introduce a definition for agent societies, we discuss the principles of social interaction, relationships and social capital and finally we introduce intelligent agents and socio-cognitive agents.

1.1.1 Agent Societies

The Open Agent Society (OAS) paradigm springs from the desire to create 'functioning' applications whereby the welfare and prosperity of the people participating in the system is assured. For example in an information trading scenario the system "should be regulated by (governed by, conditioned by) the kind of relations (contractual and normative) found in human business and social interactions." [33, 100]. This is a reference to Henri Bergson's Open Society [18] as well as Karl Popper's *Open Society and its enemies* [101], which called for a system of governance not based on a central authority or 'concrete' rules, but instead pushed for flexible and adaptable rules which conferred rights and where governance is transparent and open to criticism. In summary this argues for the following:

Accountable governance: In decentralized systems operating with partial local information, no benevolent dictator can exist. This is addressed via self-governance in the form of an agent democracy.

The Rule of law: In an open decentralised system no central arbiter can restrict the actions of agents, therefore an agent's capability to perform an action is a brute fact (the robot *can* fire a missile) but this is separate from the agent's right, be it moral or legal to do so (the robot *shouldn't* fire a missile at a school). An example of why this is important can be found in the domain of mobile agents where a host node may have the capability to destroy a mobile agent process but not the permission, due to the resource value of the agents accumulated knowledge [23].

A Market economy: Without making a statement about what kind of market economy, free or otherwise; merely indicating that an efficient mechanism for deciding what resources are produced and how they are allocated is required.

A Moral Framework: Moral issues are imbued in almost all aspects of our lives, some of which are more clearly apparent than others. Our morals, as individuals, affect our perception of the world and our behaviour towards others, as such a framework for decision making is required such that our actions reflect our moral and ethical stand-point. This is not a simple task as demonstrated by numerous thought experiments. Ultimately, the moral framework which we, as a nation agree upon are used to instantiate the laws that govern us, the social norms, and socio-cognitive relations (Trust, Forgiveness) that emerge in our society.

Mutability: An adaptable system which acknowledges that tomorrow can be different from today; as well as being non-deterministic. As Popper argued "no society can predict, scientifically, its own future states of knowledge"

The Open Agent Society aims to incorporate these principles into workable frameworks for developing open decentralised applications. Whereby we define an open system as one in which agents are heterogeneous and may be competing; they can possess conflicting goals, and have internals that are opaque to the system. As computer scientists it is useful to provide a formalization of the components of an Open Agent Society for which the following is proposed:

- a set of social constraints: physical capabilities, institutional powers, norms (permissions, obligations, and prohibitions), sanctions, and enforcement policies.
    - these can be specified in the form of action languages [86].
- a common communication language
    - this is from current research in agent communication languages (ACLs) [51, 53].
- a social structure, roles and the relationship between roles
    - see electronic institutions [3, 50, 73].
- social relations, interactions and socio-cognition
    - trust, reputation and forgiveness
    - similarity, relatedness and recommendation
    - information exchange, opinion formation and judgement aggregation.

1.1.2 Social Relations and Social Capital

From the above definition of an agent society, elements of social constraints are addressed by Sergot, Jones and Artikis in [5]. Agent internals to facilitate subjective reasoning about such concepts have been investigated by Kamara et al [80]. This thesis focuses on the final bullet point from above and views the system from the perspective of individual agents and their interactions. This covers both communicative and socio-cognitive aspects of agent interaction (e.g. [29, 49, 98]). In addition to being relevant to agent designers and implementers, this can be used to model, analyse and explain the behaviour of the agents in terms of social theories (e.g. [29, 49]). Specifically we address issues regarding interactions between agents based on peer modelling, social interactions and relationships.

Arguing for the importance of such relations Durkheim [45] states:

"men cannot live together...without tying themselves to one another with strong durable bonds".

By this Durkeim implies that in human societies the individual social interactions and relationships between people act to organise, inform and regulate human society as a whole. Further to this, Coleman [31] asserts that:

"...social capital is productive, making possible the achievement of certain ends that in its absence would not be possible. ... Unlike other forms of capital, social capital inheres in the structure of relations between actors and among actors.".

This implies that social relations have a function within a population which enables individual and collective goals to be realised; furthermore this 'value' is emergent from the underlying social network.

Resnick [104] defines the term 'sociotechnical capital', "...to refer to productive resources that inhere in patterns of social relations that are maintained with the support of information and communication technologies (ICTs).". Resnick [103] goes on to outline five categories in which social capital (and by analogy sociotechnical capital) functions to add value to a collective and to its constituents, namely: Information exchange, Resource exchange, Emotional support, Coordination and Collective action. While Resnick's work is motivated towards designing Human Computer Interfaces which support the development of social capital in human societies, it is clear that the challenges present in computational societies also align themselves with these categories.

In order to design agent systems that can leverage socio-technical capital, a framework for social relations is required. To this end, Jennings calls for the explicit representation of relationships between software agents: "it is important to explicitly represent the relationship. In many cases, relationships are subject to ongoing change: social interaction means existing relationships evolve and new relations are created."[76]. But what exactly do we mean by a social interaction or relationship in the context of multi-agent systems and how would they be manifest in a computational society?

Max Weber [121] claims "an Action is 'social' if the acting individual takes account of the behavior of others and is thereby oriented in its course". Indicating an a priori requirement for an individual to hold beliefs regarding a peer; before social action can occur, and that those beliefs should affect an agent's chosen action. Repeated social actions serve to multiply and enforce these beliefs forming subjective social relations between peers. This is how we define social interaction, as the repeated occurrences of social actions between peers or within a community which lead to social relations.

Therefore in building social capital an agent's actions must be directed by its social relations, relations built from repeated social interaction which it perceives exists between it, its peers, and its community. Of course the assumption of rational goal-directed peers may be seen to undermine the development of social capital. Why for instance would a programmer/owner wish its agent to be directed in its actions by social relations as opposed to the maximisation of their utility? Because each agent in the system is reliant upon each other to achieve their future goals. Therefore they must behave such that they secure future cooperation, this is the basis for social exchange theory. This of course means that for a system to function as a society there must exist interdependence between actors, which Conte [102] terms 'Social Dependence', whereby an agent is dependant on another's actions to achieve its goals.

1.1.3 Intelligent Agents and Multi-agent Systems

Within the computer science field there exists a plethora of definitions for the term Agent. Many are accompanied by a dissection of the key characteristics, and followed by an analysis of the strengths inherent in an agent based approach [55, 60, 78, 96, 125, 126]. Example definitions include:

- "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives." [126].

- "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" [107].
- "Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions." [69].

In essence this describes a paradigm, an approach to conceptualising and designing systems, as opposed to any concrete language or architectural plan. This flexibility allows individual researchers to adapt the definition to suit their perspective and requirements.

Multi-agent systems (MAS) are equally varied in their taxonomy, in the simplest terms MAS refers to any system comprised of more than one agent. This therefore encompasses everything from, several 'agents' operating on the same machine each coordinating and performing a task to aid a single user; to thousands of agents each representing a different owner be they an individual or a institution, and communicating across multiple networked environments. This later definition adds the premise of an agent system situated within a human society, and by combining the concept of agents described in anthropomorphic terms leads us to the possibility of designing open distributed applications based on the paradigm of Agent Societies.

Socio-Cognitive Agents

In this thesis we specifically address the concept of socio-cognitive agents, we categorise such computational agents as socio-cognitive in reference to Albert Bandura's agent perspective of social cognitive theory [11]. In this, Bandura describes the role that observing the behaviour of others (and to a lesser extent the environment) has on developing cognitive models and personality. Developing systems which benefit from the creation of social capital, requires us to design agents capable of 1) developing a model of their social relationship to a peer based on their externally observable behaviours, and 2) reason with this model to plan their future interactions. We will utilise socio-cognitive agents as the basis for implementing social relations and agent societies; these are then applied in-order to overcome challenges inherent in our case studies. Specifically our socio-cognitive agents will be capable of developing a cognitive model of their relationship to a peer based on experiential data and social discourse (for example recommendations).

1.2 Case Studies

This section briefly introduces our two case studies designed to investigate the effect of agent to agent social relations and the methods used to model them on the welfare (both social and monetary) gained by agents.

1.2.1 Self-Organising Social Recommendation Networks

In this case study we present an agent-oriented decision and reasoning framework for action selection. Which aims to encourage agents self select plans based on the recommendations of their peers. However instead of using a centralised collaborative filtering system, we utilise the application layer network itself, in connection with principles from self-organisation and information dissemination to determine the flow of information through the system. We demonstrate how an agent can utilise subjective knowledge of its relation to its peers, to self organise its social ties; and in doing so improve the quality of the information on which the agent makes its decisions and performs actions. While this helps to fulfil the individual agents' goal of maximising its utility, we also demonstrate how this cooperation between large numbers of self interested agents can create a "self-directing automatic system"(Friedrich von Hayek [68]) which enables agents to autonomously make utility maximising decisions and reduce the workload involved in the governance of such systems.

1.2.2 Agent Mediated E-commerce

Our second case study investigates the role of trust relationships and reputation in regulating the function of open distributed resource exchange markets. Given the open nature of such a scenario markets of this type would be susceptible to a number of threats. If the agent paradigm is going to be adopted it is necessary that mechanisms are in place which not only protect against malice and incompetence, but also allow the agents' performance outcomes to compliment the motivation of their owners. Hence a producer agent must be capable of the the long-term maximisation of its owner's profit and consumer agents must maximise their owner's utility given their budgetary restrictions. Essentially this means when improving the agents defences we must not make them economically sub-optimal, as this would inhibit the market's ability to allocate resources efficiently. Therefore this case study develops a computational formalisation of trust and reputation theories, it integrates these models with rational choice theory to develop a set of socio-cognitive agents capable of rational choice but directed by their social relations.

1.3 Thesis Summary and Structure

In summary, this thesis presents a new simulation platform PreSage which is designed to fill a gap between social simulation tools that provide extensive support for experimentation, and agent development tools that focus on developing individual standards compliant agents. Specifically, we provide a scientific tool with support for animating systems across a range of input parameters, that allows for computationally complex agents, flexible design of agent communication, and simulation of the context in which the agents operate such as the physical environment and dynamic networks. Our case studies support our claims for PreSage but also contribute to the community their own insights.

Our experiments with self-organising recommendation, show that established beliefs about the suitability and properties of centralised collaborative filtering algorithms do not directly translate to distributed platforms. That the performance of the system (in terms of decision making accuracy) is far from equal for all participants, and that it is highly dependant on the interaction of an agents initial requirements and its internal reasoning.

Agents that are endowed with the ability to reason with interpersonal concepts such as trust and reputation have long been acknowledged as integral to developing loosely coupled open distributed systems. Within this field we contribute a framework for trust based decision making which is scalable, protects against false testimony, and reduces computation by transitioning an agents decision making from time consuming reputation based trust to form relationships based on reliance trust. Our evaluation approach is also novel, following on from our experiments on social recommendation we argue that in simulating agent based trust we must ground our results in an accurate manner. As such we propose and develop a economic simulation, and demonstrate how to integrate our trust framework into the agents monetary decision making.

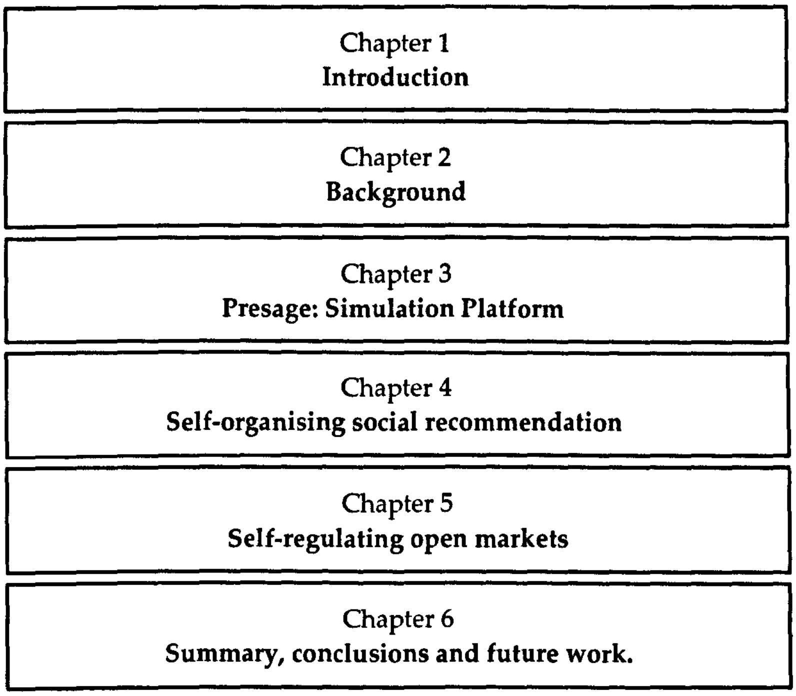

Chapter 1
Introduction
Chapter 2
Background
Chapter 3
Presage: Simulation Platform
Chapter 4
Self-organising social recommendation
Chapter 5
Self-regulating open markets
Chapter 6
Summary, conclusions and future work.

Figure 1.1: Thesis Structure

The remainder of this thesis is structured as follows. We continue in chapter 2 by surveying the state of the art with regards to the application of simulation, recommendation and trust in open distributed multi-agent systems. This provides context and motivation for subsequent chapters whereby we detail our theories, results and contributions to these domains. These later chapters can be summarised as:

- In Chapter 3 we present our modular and reusable simulation platform PreSage, describing its architecture and functionality and discussing some example applications and experiments which have been developed from it.
- In Chapter 4 we describe the first of our case studies using PreSage for simulating agent societies which are formed from social relations. Specifically we look at the effect of agent decision making on the formation and organisation of distributed recommender networks.
- In Chapter 5 we investigate trust relationships and the formation of subjective yet collective beliefs such as reputation, and their utilisation for the regulation and performance of open agent systems, in this case a resource trading market.
- Finally, Chapter 6 summarises the incremental conclusions of the earlier chapters specifically in defining the strengths and weaknesses of our approach and identifying the challenges which can be met through further development of our work.

The following table is included to clarify which elements of this thesis have been published else where.

Year	Authors	Title	Relation to Thesis
2011	Jeremy Pitt, Brendan Neville, Sam Macbeth, Hugo Carr	Animation of Open Multi-Agent Systems Agent-Directed Simulation 2011	Chapter 3: Detailed account of the experiments which have been performed using the platform.
2008	Brendan Neville, Jeremy Pitt	Simulating Agent Societies with PRESAGE (Demo Paper). Proceedings of Engineering Societies in the Agents World 2008 (ESAW)	Chapter 3: Brief account of the platform accompanied by a software demo of the experiments e.g. Coloured Trials, Voting in Manets etc.
2008	Brendan Neville, Jeremy Pitt	Presage: A programming environment for the simulation of agent societies.Proceedings AAMAS Workshop on Programming Multi-agent Systems (ProMAS)	Chapter 3: Detailed account of the simulation platform.
2004	Brendan Neville, Jeremy Pitt	A simulation study of social agents in agent mediated e-commerce. Proceedings of the Seventh International Workshop on Trust in Agent Societies.	Chapter 5: Brief account of the CS_CEF followed by experimental results.
2003	Brendan Neville, Jeremy Pitt	A computational framework for social agents in agent mediated e-commerce. In Engineering Societies in the Agents World IV 2003.	Chapter 5: Detailed account of the CS_CEF.
2003	Mark Witkowski, Brendan Neville and Jeremy Pitt	Agent mediated retailing in the connected local community. Interacting with Computers 15 pages 5–32.	Chapter 4, 2: Collaborative filtering and centralised recommender systems effect of algorithm on system dynamics.
2002	Lloyd Kamara, Alexander Artikis, Brendan Neville, and Jeremy Pitt	Simulating computational societies. In ESAW III Third International Workshop, pages 53 - 67. Springer.	Chapter 2: Initial outline of our methodology for simulating Agent Societies from three perspectives.

Table 1.1: Publications related to the Thesis

BACKGROUND 2

2.1 Introduction

The introduction defined the terminology and broad themes used throughout this thesis, in doing so we have addressed the prior literature and background specifically with reference to Software Agents, Agent societies and Social Relations. This chapter follows on from this, taking an in-depths look at the domains, concepts and definitions relating to the three main sub-sections of this thesis.

We begin in section 2.2 with aspects of social information exchange, namely its facilitation, regulation, and utilisation through social networks. We go on to discuss Trust and Reputation which form the basis for our investigations into regulation of agent mediated commerce in Section 2.3. Section 2.4 concludes the chapter by outlining theories of social simulation, its relevance to agent systems and requirements for a tool to investigate agent systems and specifically social relations between peers (i.e. our case studies on self-organising social recommendation, and trust in agent-mediated e-commerce).

2.2 Self-organising social recommendation

Our investigation into socially adapted information dissemination describes a self organising system for disseminating agent/robot plans in large scale distributed systems. It combines concepts from self-organisation, social relationships and recommendation to produce a collaborative filtering social network for the dissemination of information (the plans). This section reviews the current research in these domains.

2.2.1 Recommender Systems

First generation recommender systems were based on a centralised database architecture, they used one of three different approaches to make product suggestions to users, namely: content based recommendation, collaborative filtering and a combination of these two techniques. Content based recommendation is based on concepts from information retrieval [13]. An example would be NewsWeeder [87]. This method compares the quantifiable features of an item (e.g. name, genre, author, colour etc.) with those in a user's profile, the higher the correlation the higher the likelihood of suggesting the item. On the other hand, collaborative filtering compares user opinions about an item and matches peers into cliques based on the similarity of their ratings. Using these groupings the system can suggest items to a user on the basis of positive reviews from their clique.

Burke [26] provides an in depths comparison of the various methods for generating recommendations. Some of the recognised disadvantages of content based approaches include: that not all content is quantifiable by categorization alone, quality is not always definable and can be subjective, the approach is unable to suggest things outside of a given specification. On the other hand collaborative filtering requires a large number of overlapping opinions from each agent in order to accurately map the similarity of users and requires the solicitation of reviews from users. Due to the recognition that many (if not all) techniques for generating recommendations are in some way flawed, many researchers have opted to create systems utilizing a number of techniques combined, generally a mix of content based and collaborative. These are classified as hybrid recommender systems e.g. [97] and Entree (Burke [25]).

Collaborative Filtering

In this section we introduce the core concepts of shared preference modelling and collaborative filtering and their use in tailoring suggestions to an individual user. The methods rely on the observation that individuals share many of their preferences with others (formed into cliques). This gives rise to the possibility of creating recommender systems based on collaborative filtering methods (Alspector et al., [2]; Herlocker et al., [71]; Konstan et al., [85]). Such systems identify individuals or clusters with similar tastes and preferences across the web or from (typically large) databases and suggest items drawn from the purchases or recommendations made by the clique. These techniques are now becoming widespread, increasingly forming the basis of choice recommendation strategies for agent systems and web delivery alike, and are likely to have significant economic impact (Fink and Kobsa, [52]).

Systems developed within the academic community that leverage collaborative filtering techniques include GroupLens (Resnick et al., [105]) which investigates the use of collaborative filtering to provide interest filtering of internet news articles, others would include Tapestry (Goldberg et al. [62]) and Ringo (Shardanand et al, [111]). There are also several well known algorithms for doing collaborative filtering for example the cosine similarity [47, 116] and the Pearson correlation [47, 72].

What these systems have in common is a high degree of centralisation, relying on a large database of users, products and user opinions. This centralisation is not inherently scalable, even from a third party web service perspective. It is even less appropriate when we look to new distributed peer to peer architectures, where this dataset and the computation of peer similarities and recommendations would need to be replicated across large numbers of low specification peers.

We propose that this representation and its associated computational tasks can be replaced by a dynamic Social Network based on the individual relationships between peers. This will direct recommendation information to nodes which can benefit from it, whilst simultaneously filtering out the vast majority of either harmful or useless data. This will improve an agent's decision making and reduce the computational and network overhead.

2.2.2 Social Networks

A social network is, at its most basic interpretation, defined as a set of nodes (or vertices) with a set of interconnecting relationships defined as edges, whereby nodes can represent individual people, organisations or even nations. The edges between the nodes can denote any conceivable social relationship or dependency.

These are often modelled as friendships, marriage, kinship or where you went to school, but they can also be based on shared interests, desires or goals. Some relationships can be static (familial relations), others may change rapidly and unpredictably (Trust). In addition some edges can simply be a binary relation, connected or not, whereas others may be represented as a degree of connectedness.

Numerous ethnographic studies have demonstrated connectedness and the effect of real life social networks on the passing of information; how the nature of ties can affect the speed, accuracy or usefulness of the communication. Travers and Milgram [117] investigated the degree of connectivity and path length between people with their 'small world experiment'. Gavoneter [64] showed how weak ties are the most useful in obtaining novel information (as in finding something new or surprising) which he demonstrates with studies of job hunting. Whereas Friedkin [56] shows that strong ties are important for the speed and efficiency (probability) of information transfer.

From the perspective of agent systems, information search and matchmaking, a number of prototype systems have been developed which utilise social networks and referral chains. In [81], Kautz et al describes ReferralWeb which discovers users' existing social network and utilises this to search for documents in a targeted manner. Foner [54] developed Yenta to help users discover peers who possess similar interests. The aim being that users could form collaborations or find experts.

Foner discusses the advantages of distributing this matchmaking in comparison with centralised systems, two points are of particular relevance. Firstly, the inability of central matchmaking to scale to large numbers of users and secondly, the impact centralisation has on applications which require high availability.

In our introduction we argued that social relations between agents can have a profound influence on the function of agent systems. Friedkin, Gavoneter, Travers and Milgram have demonstrated that it is not simply pair-wise relationships that influence a population but the network of relationships. Kautz and Foner have leveraged users existing networks to improve performance in information and peer seeking tasks.

This clearly motivates further research into agents and social networks, what is needed is a platform which allows us to systematically analyse the formation, organised adaptation and performance of agent systems, especially where the network of social relations has a significant affect on the system behaviour.

As such the outputs of Chapter 4 in relation to social networks are two fold, firstly we take a bottom up approach to investigating how simple changes to the decision making of agents affects the emergent social network and how this network in-turn influences the prosperity of the individual agents. Secondly, we contribute a set of reusable modules for PreSage which enable future experimenters to quickly build agent systems which form, organise and utilise networks of social relations.

2.2.3 Information Dissemination

A popular approach to information dissemination in distributed computing utilises models and studies of infectious disease [10, 48]. For instance, Demers et al [42] discusses the use of algorithms which he refers to as analogous to epidemics; for replicating a database over a distributed network, the aim is to create copies of the database to ensure high-availability, scalability and consistency. Demers' specifically looks at three epidemic style strategies for propagating updates, namely: *Direct Mail*, where updates are immediately sent to all nodes; *Rumor Mongering* where nodes randomly spread an update until it starts encountering others already aware of the entry; *Anti-Entropy* where nodes periodically choose a peer to compare entries and resolve differences. Other examples include Grapevine's [20] message delivery service which utilises an epidemic approach to updating its distributed message database, which the authors note creates a high-availability but not an entirely consistent database.

Traditional studies into epidemic infection have focused on how rates of infection can be reduced and the spread of disease vectors stopped. Work in database replication and information dissemination has, rightly, taken the opposite point of view, specifically how to ensure consistency by defining protocols and parameters to ensure delivery of data(infection).

Our work addresses a new requirement in-between these extremes; we propose to model *Social Information Dissemination*, where the action (strategy) of passing on information is directed based on a node's beliefs about the receiving peer. The strategies listed above e.g. Anti-Entropy, Rumor Mongering, Direct Mail etc do not take into account the possibility that information which is disseminated can be harmful to the decision making of a receiving peer.

2.2.4 Self-organisation

Research into Self-organisation spans a number of domains, including but not limited to: Philosophy, Biology, Chemistry, Physics, Mathematics, Economics, Social Sciences and Computer Science. For the purposes of this thesis we will focus on work within the domain of computer science [44, 57, 92]. Specifically De Wolf et al [40], who provide an interesting and in depths review of the lineage of both terms, in addition to proposing their own working definitions and characteristics. He defines emergence as the following:

"A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel w.r.t. the individual parts of the system."

De Wolf uses the term emergents to indicate "properties, behaviours, structures and patterns" that are observed at a higher level of abstraction and are the result of the interaction between the parts of the system, themselves observed at a lower level. It is this Micro-Macro effect that is generally perceived as the fundamental characteristic of emergence, in conjunction with a number of other properties which De Wolf lists: radical novelty, coherence, interacting parts, dynamism, decentralised control, two-way link, robustness and flexibility. We will address a number of these below:

We would argue that *Radical novelty* is the property most widely misused, often believed that an emergent property should not be predictable or that the result should be 'surprising' or unexplainable. This is simply not the case, by radical novelty we mean there exists an observable globally emergent property, which is a qualitative and/or quantitative macro scale phenomenon; such that in terms of any behaviour or property of the discrete parts it is unique. De Wolf adds that this property cannot be studied by examining the individual parts, but can be studied (explained) by examining the system as a whole.

Decentralised control simply implies that no agent is coordinating the macro level phenomena;

Two-way link implies a bidirectional process of cause and effect between the emergent property and the micro level components, which some authors describe in terms of feedback mechanisms.

Self-organisation seems to attract a far less philosophical approach to definition, is defined by De wolf as: "a dynamical and adaptive process where systems acquire and maintain structure themselves, without external control." which happens to exactly coincide with our use of the term, whereby our agents adapt their communication channels over time in order to form the network structure. They do this autonomously and without external control, and the resultant structure serves the function of filtering and directing information.

This section has outlined concepts from self-organisation, social networks, information filtering and dissemination; these form the basis of the behaviours our agents and prototype system exhibit in chapter 4. While we make some theoretical contribution to the fields of information filtering and dissemination. In the most part our innovation is in demonstrating the convergence of these concepts, and by utilising the PreSage platform we can investigate their interaction holistically, we also build a prototype to form the basis of future research.

2.3 Trust

As discussed in our introduction the motivation for developing the concepts, techniques, models and software of multi-agent systems, is to address challenges in a new class of complex distributed software applications. Where these systems are designed as open multi-agent systems their global function relies upon the individual interactions of their constituent parts. In fact these interactions relate to the communication of agents though mechanisms such as negotiation[77], coordination [75] and cooperation/collaboration. However open system components may be written by developers with varying motivations, interests or moral stance. Therefore the outcome of these pairwise interactions is clouded with uncertainty. An agent cannot be certain of their peer's intentions nor can it predict with certainty their ability to perform an action which it is reliant upon. As a result of this, trust has been proposed as fundamental to creating agents which are capable of integrating into stable and efficient collectives.

Trust is generally accepted as key to the correct and efficient functioning of human society [22, 37, 59, 74]. Arrow [4] describes it as having *"a very important pragmatic value, if noting else. Trust is an important lubricant of a social system. It is extremely efficient; it saves a lot of trouble to have a fair degree of reliance on other people's word."*. In fact it is pervasive to many strata of human interaction and decision making, for example: interpersonal relationships, our trust in organisations (e.g. banks, governments and educational institutes) and even trust based on a role or occupation e.g. doctors [88]. These organisations and roles are also present in the conceptualisation of MAS; thus further motivating the study of trust.

Castelfranchi states: trust is *"critical for modelling and supporting groups and teams, organisations, co-ordination, negotiation, with the related trade-off between local/individual utility and global/collective interest; or in modelling distributed knowledge and its circulation."* [29]. Castelfranchi is identifying that trust enables us to rely on the actions of our peers even in circumstances where it is not in their best interest to behave in this manner. These circumstances are often referred to as the *"Tragedy of the commons"* [66]. As Turner [119] points out MAS often exhibit the properties of a shared finite resource.

2.3.1 Trust Definitions

"...trust is based on an individuals theory as to how another person will perform on some future occasion" [63]

Trust therefore is an important component, in an uncertain and dynamic environment, it is cognitively too complex (expensive) to formulate accurate expectations about the behaviour of others for every task and context. It is also arguable that regardless of the computational complexity of doing so, it is more likely the case that it is impossible to calculate a prediction without access to the agents internals in the form of its goals, intentions, plans etc; which a peer is unlikely to be privy to.

From the perspective of modelling trust as a belief of an agent, it is Gambetta that provides us with a succinct and workable definition. *"Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends"* [59]. By describing trust as a subjective probability, Gambetta opens the possibility of numerically reasoning with trust values and integrating it with other numerically decision making tools from economics. In this sense his definition can be linked to utility and in a monetary system currency. As such his definition can be read as Trust is a estimated belief of Agent A of the likelihood that the agent B will intend (be willing) and capable of performing an action which benefits financially (increases the utility) of the Agent A. This definition allows us to model the cognitive factors in an agent's beliefs, reasoning, and decision to trust.

Reputation is a key facet of trust and by association an important aspect promoting agent interaction. In particular, we note how reputation is related to the notion of scalability: in a system with a large number of agents, the likelihood of any individual agent having interacted with a specific peer is less. This leads to an overall reduction in the amount of direct experience available to an agent during its decision-making. This issue is countered by basing trust decisions on the combined testimony of third parties (a target's reputation). In addition reputation has a stabilising or healing affect on trust relationships. This is evident when an improbable series of failures occurs (though improbable this will eventually occur given sufficient simulation time) the result of this is for the trust between the two cooperating peers to be effectively destroyed. The participants then decide not to cooperate in future, however their peers continue interacting with both sides and it is the repeated recommendation of trusted peers which will rebuild the trust relationship. Thus it is important to include reputation as an important component in trust models which are resistant to minor aberrations in agent behaviour due to circumstances beyond their control.

2.3.2 Computational Trust and Reputation models

In [12] Barber et al present two reliability (trust) belief revision algorithms, one based on direct experience and the other utilising recommendations. From their experimental results they conclude that the combination of the algorithms would result in the most desired system performance. Singh et al [129] propose that when incorporating the testimony of peers to form a reputation belief the agent should consider the reputation of the testimonies source. They go on to test their model in experiments based on the iterated prisoners dilemma (IPD). The IPD and its many variants are used extensively in social simulations, specifically Axelrod [7] and Yao et al [128]. Witkowski [122] simulates a trading environment of supplier and consumer agents, the agents select partners to trade with on the basis of trust. This trust is based purely on the trusting agent's direct experiences, and is updated simply by a trust update function as analysed by Jonker and Treur in [79]. Sen et al [109, 110] show how in a group of self-interested agents, reciprocal behaviour can promote cooperation and increase global performance. Sabater and Sierra [108] propose the REGRET reputation management system. In addition, it provides an ontological dimension to trust, chaining a number of facets of a task (effectively sub tasks) into an overall trust belief. The REGRET system, focuses on the accurate calculation of the reputation belief and on the reactivity of that belief to changes in trustworthiness of the target agent.

These computational trust investigations are set up using abstract games such as the IPD, our scenario takes this further by combining the elements of defection from the IPD with a grounding in the economic interactions of online markets. From the perspective of online markets Kollock motivates the need for trust and reputation: "The motivations of those we interact with can be inferred but never known directly and the quality of goods and services we are offered is often unknown or known only approximately" [84]. While Kollock is in fact referring to trust in web based markets, the sentiment is equally valid when applied to future agent mediated markets; trading in services, ad-hoc networks or power generation.

The computational models outlined earlier derive trust as a probability calculated from a series of interactions and those of third parties. In this sense they in fact form highly accurate calculations of risk and reliance, however anecdotally our personal experience leads us to understanding that trust is not always the same as a calculated risk; it is a spectrum of nuanced concepts, e.g. blind trust, affective trust, institutional trust, personal trust, reliance trust. It is also cognitive short-cut to 'get things done' when operating under uncertainty.

From this perspective we propose to transition though different stages of trust depending on our certainty (or confidence) in the trustworthiness of our partners. Our Socio-cognitive model accounts for this by allowing an agent to begin with blind trust and move onto risk based trust and finally onto reliance based trust. Blind trust enforces action in situations of absolute uncertainty. During risk based trust, agents will determine their trust from their own experiences and will seek to improve their confidence in their trust decisions by utilising the targets reputation; they will adapt the level of the risk they take based on the level of confidence (they can muster) in their trust belief. When an agent uses reliance trust the agent only bases their trust on their own experiences, they no longer compute trust from all their sources of information, they have now formed a relationship with the target though a series of personal interactions and rather than considering why they have a trust relationship, they use its existence as a short-cut in their decision making.

2.4 Simulation of Agent societies

Axelrod [9] and Edmonds [46] both identify key reasons for the use and application of multi-agent based simulation (MABS). Three application domains are common to their analyses — those of science, entertainment and performance. Axelrod [9] describes performance motivated MABs as those involving agents carrying out specific tasks in a manner analogous to humans and employing theories in human perception, decision making and social interaction in order to support task performance.

It should be noted in particular that we do not use simulation as a means to explain or predict human behaviour, nor are we adopting the sociological approach evident in other studies (see [35, 36, 118] for discussion of simulation with a sociological emphasis). We are instead experimenting with socio-cognitive theories which, while being informed by analogous concepts in the social sciences, do not necessarily provide reciprocal insights into human societies and reasoning when interpreted in a simulation context.

The design and construction of the test-bed is such that our four key underlying components can be rapidly developed namely: the agent architecture, the socio-cognitive models (used for internal formalisation of cognitive states, reasoning and decision-making), the social interactions of the agents (in the form of protocols) and the physical characteristics of their environment (the network architecture and physical objects).

This bottom-up approach to testbed design reflects a *methodologically individualistic* stance on the study of agent societies. Central to this modelling stance is the belief that social phenomena derive solely from the activities of individual agents (and thus need not be modelled separately). This approach has been criticised — for example, by Sawyer (in [35]), for not explicitly modelling the large-scale social phenomena associated with such scenarios. In the experiments reported here, however, the primary concern is the performance of individual agents and system, rather than the evolution of the society. Our aim is to create and experiment with artificial systems whose performance criteria benefit from the use of social theories. The distinction is further illustrated through reference to the synthetic method underlying artificial intelligence (AI) research ([114]). The main steps, depicted in Figure 2.1, involve generalising some observed phenomena to produce a theory, from which an artificial system can be constructed. The resulting system performance serves to support or refute the theory.

Figure 2.2 is an adaption of Figure 2.1 and provides further detail on the concepts motivating our experimental approach. In this diagram, a clear distinction is made between the informing social sciences [1, 32, 49, 64] which serve to motivate the computer science activities. The transition from theory to artificial system is no longer direct and now includes the generation of a computational formalism as an intermediate step. In addition, the results of observed performance are not used to justify or refute the informing theory,

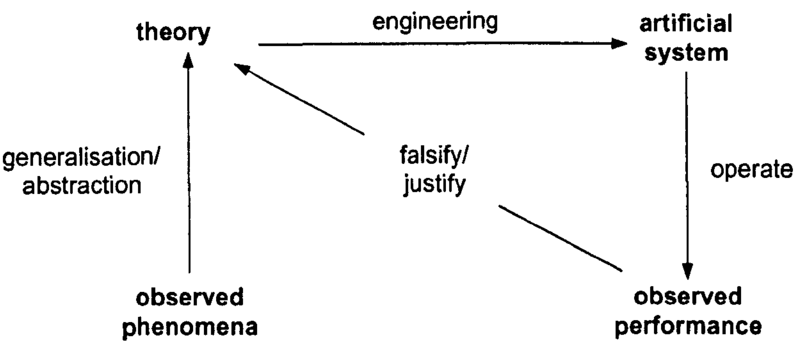


Figure 2.1: The Synthetic Scientific Method ([114])

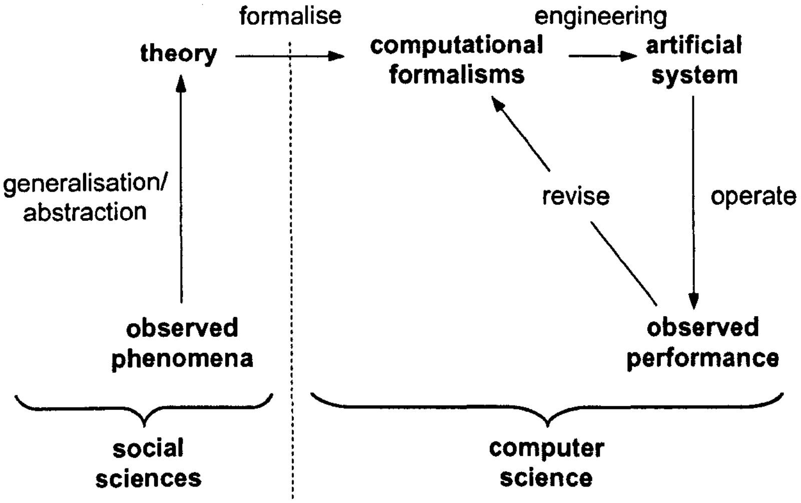


Figure 2.2: An Adaptation of the Synthetic Method

but are instead used to adjust the formalism behind the artificial system.

By simulating systems composed of such agents and observing the outcome, we aim to tailor formalisms to achieve the desired global performance and hence demonstrate an agent design that is suitable for the distributed application. It is our intent to evaluate the performance of the applied models based on the efficiency, fairness and dynamics of the resulting communities. By giving particular attention to the suitability of the fundamental assumptions of the scenarios, for instance the economic models employed, we ensure that the results of our future simulation work will accurately portray the behaviour of an actual distributed multi-agent system.

The nature of the measurements used to determine efficiency, fairness and dynamics is highly application dependant. Concepts from social science such as fairness are necessarily vague, while the reader may have their own belief about the term, historically attempts to define a universal measure of fairness (justice) have failed. For instance compare Bentham's utilitarian approach [17] which states you should seek the highest benefit for the largest number of people (in essence - lets maximise the utility in the system as a whole, without special concern for the individual, the ends justifying the means) to Kant's categorical imperatives, which state we must always respect the individuals dignity, requiring us to look at the distribution of the utility and consider the morals of the individual actions which lead to that utility. From this Kantian perspective the analysis of our case study in chapter 4 looks at how utility is distributed across the population, comparing results for agents which have majority vs. minority needs. While chapter 5 shows both the total utility in the system (Benthamism) as well as showing how reputation reduces the gap between the best performing and worst performing agents (Kantian).

2.4.1 Related Work

Rapid prototyping and animation of agent societies in a logical form has been effectively used in order to demonstrate and verify properties of agent societies. In [120] Vasconcelos et al present an approach to rapid prototyping multi-agent systems through the definition of a global interaction protocol. The global protocol defines the types and order of interaction between the components and is used to automatically generate a set of agents which are then simulated to check for desirable properties in the protocol. CaseLP [90] is a logic-based prototyping environment for specifying and verifying complex distributed applications. It also provides tools for specifying certain network properties when developing prototypes of distributed systems e.g. reliability or latency of connections.

Muli-agent Based Simulation (MABS)[38] is a micro-level approach to simulation of complex systems. Whereby the behaviour of system components or individuals are modelled as agents. A number of MABS tools exist [61] including Swarm[1][91], Repast [2] and MASON [89] and are widely used for Agent Based Social Simulation (ABSS)[34, 35, 39].

ABSS uses MABS techniques to model human interactions within a multi-agent system, generally with relatively simple behaviours (on an individual agent basis) that when sim-

[1] www.swarm.org

[2] http://repast.sourceforge.net/

ulated their interactions lead to complex global behaviour. These results are subsequently used to understand and elaborate social theories. Our social agent experiments [93, 94] cross-fertilise with the theories and formalisations of the ABSS field and the wider social sciences. However our interest lies in using this knowledge to solve problems related to open agent societies which diverges from their use to further understand human society. Given our focus on agent systems, their environments and the encapsulation of agents in their environment as network nodes, we require support for heterogeneous agent architectures, low-level representation of agent communication languages and communication networks. While Repast can be configured to support various agent architectures it lacks the necessary flexibility with respect to agent communication.

Multi-agent system development tools (for an evaluation see [106]) such as AgentBuilder[3] and JADE support analysis, design, development and deployment of multi-agent applications. More specifically, JADE[4] [14–16] is a robust middleware for developing FIPA[5] compliant agent applications. The JADE agent framework provides the developer with an API for message syntax and parsing and a set of standard interaction protocols thus simplifying the process of developing interoperable agents. There also exists a deployment tool [24] which supports the configuration and deployment of JADE agent based applications.

2.5 Chapter Summary

This chapter has identified the importance of social relations, such as are required for collaborative filtering and e-commerce, in engineering agent societies and/or socio-technical systems. For example multi-agent systems systems in which the interaction between autonomous computational components are conditioned by some kind of social relations, or systems which require that those components exhibit or understand some kind of social relation because they are embedded in some human-centred legal, social or organisational context.

[3] www.agentbuilder.com

[4] http://jade.tilab.com

[5] www.fipa.org

SIMULATION PLATFORM *PreSage* 3

3.1 Introduction

PreSage is a Java based simulation platform for rapid prototyping of Agent Societies. It enables designers to investigate the effect of agent architecture, algorithm design, parametrisation, network properties and the physical environment on individual agent behaviour and long-term collective global performance.

We view PreSage prototyping as a step before frameworks like JADE or AgentBuilder; providing a platform for investigating system wide performance, emergent behaviour, optimising interaction protocols and algorithms. PreSage is being developed in order to support these investigations as well as the development of our case studies and the applications we will present in section 3.6, as such the platform is required to support:

- animation of computationally-intensive algorithmic intelligence, and simulation of 'large' populations of agents;
- simulation of multiple classes of networks;
- simulation of the 'real' physical environment in which the MAS is embedded;
- simulation of systems with the characteristics described above, and
- systematic experimentation, visualisation, versatility, customisation and reconfiguration.

The rest of this chapter is structured as follows, section 3.2 provides a set of non-functional requirements and a brief overview of how a user develops and runs a test-bed using the platform. The platform architecture including the underlying simulation model and core modules are discussed in section 3.3. In section 3.4 we describe in detail the agent model and agent communication language. Following this section 3.6 summarises the research which has been carried out using the platform.

3.2 An Overview

To satisfy the functional requirement of developing a rapid prototyping and animation environment for agent societies we have paid particular attention to developing a highly customisable and extensible simulation architecture. However, in order to support the designers goals of observing social behaviour, long term global performance and adaptation we also specifically identify a set of non-functional requirements, namely:

- abstraction: the system allows the designer to tailor the degree of abstraction in their models. In particular, the primary objects of study, the agents and the network, can be as simple or as complex as necessary. For example, the agents can range from reactive stubs to fully-fledged BDI agents;
- flexibility: the platform should be simple to configure for numerous input parameters. This supports systematic experimentation as the platform can be configured to run with the independent variables set over a range of values, and the measures of interest (dependent variables) collected for each run;
- extensibility: the platform is provided with a pre-programmed set of libraries, but the designer may extend the functionality using scriptable methods and component plug-ins;
- interaction: particular emphasis is given to simplifying the front-end to 'program' an experiment, to visualise the animation as it is running, data logging, and access to external applications, such as Gnuplot[1] for graphical representation of data;
- scalability: the architecture of the system has been designed to support both single-processor and distributed animation, allowing (dependant upon agent complexity) simulation to feature societies comprising many hundreds of agents.

[1] "Gnuplot is a portable command-line driven graphing utility" http://www.gnuplot.info/

In developing a prototype the a user can create their agent participant types through optional use of the supplied abstract class; to ensure compatibility with the simulation calls and provide core functionality like message handling etc. They can then choose from one of the pre-defined network and physical environment modules or extend the basic Network and PhysicalWorld classes to suit their purpose. Finally they may add functionality to the platform in the form of scriptable methods and plugins.

A basic input-output overview of our simulation platform is illustrated by Fig. 3.1. The experimenter configures each simulation run via XML[2] input-files; these files serve four main purposes, parametrising the general simulation variables, configuring the simulated agents (participants), scripting events and initialising the required plug-ins.

Once the platform has initialised as specified it enters the simulation thread and loops for the number of iterations specified. During this time the user can view the progress of the simulation via visualiser plug-ins, record data using data archiver plug-ins, execute methods and launch extra plug-ins during runtime.

At the end of the simulation, the platform can be scripted to organise and archive results and input-files. It can also call external applications for example Gnuplot to create publication ready graphs.

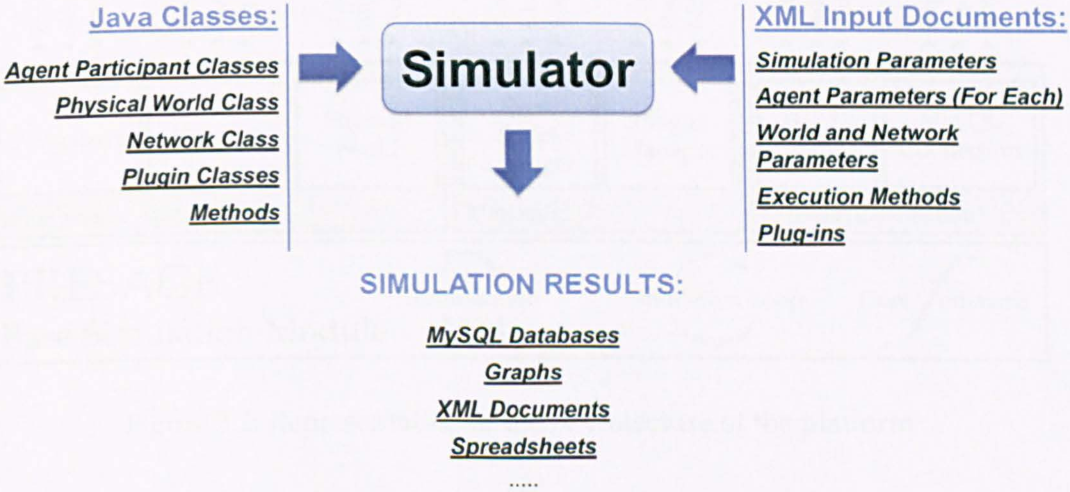


Figure 3.1: A generic Input-Output Overview of the Simulation Platform, the experimenter extends the provided Java classes and parameterises the configuration via XML files, The Simulation will then provide results in a number of formats.

[2] http://en.wikipedia.org/wiki/XML

3.3 Platform Architecture

The PreSage architecture is illustrated as a software stack (Fig. 3.2) depicting the base simulation module, the interfaces and abstract classes, simulation managers, and the platforms connectivity to external processes. Above this we have given some examples of how the user could utilise the classes and modules e.g. an auction scenario operating over a unstructured P2P network without a physical world. The user can select some of the existing examples to use, our Manet Class (Mobile Adhoc Network) has been utilised in conjunction with Brownian Motion (agents move like bouncing particles) a number of times (see the list our applications built by others in section 3.6 and the visualisation in 3.5(b)). In this section we address each of the modules in more detail.

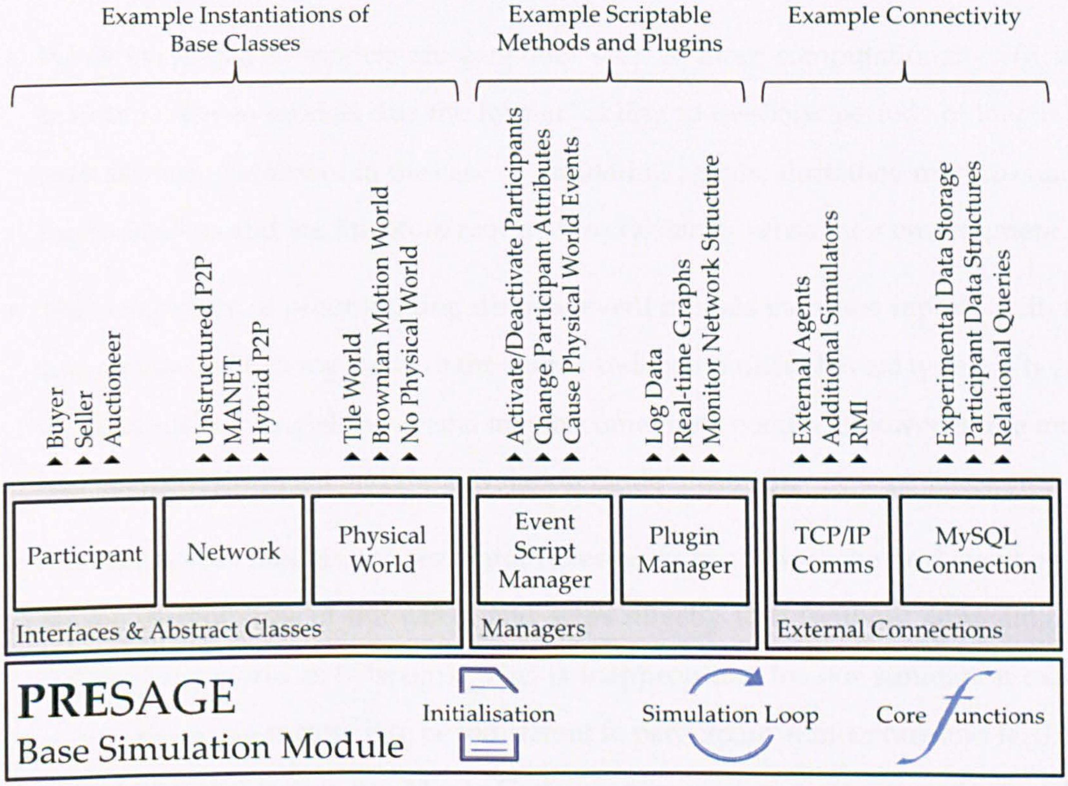


Figure 3.2: Representation of the Architecture of the platform

3.3.1 The Base Simulation Module

The role of the base simulation module is to perform parameter initialisation, manage simulation execution, and provide generic functions to higher level modules and classes.

PreSage takes a multi-agent discrete time-driven approach. In this simulation execution model each loop of the simulation control thread equates to a simulation time slice. For every time slice the state of the network and physical world is updated, scripted events execute, plugins perform their duties and the agent participants are given a turn to perform physical and communicative actions. By handling the agent process execution as a centralised time-driven model we ensure pseudo-concurrent execution of agent actions thus affording the advantages of Multi-agent based simulation (MABS) outlined in [38], and providing the user and agent a centralised notion of time. Concurrency is enforced by queuing actions until the end of each time slice.

We have developed a time-driven as opposed to an event-driven model of execution because:

- While event-driven models are generally seen as more computationally efficient than time-driven models due the former' ability to overlook periods of inactivity, such efficiency is absent in the case of simulating agents, since they react to changing conditions and are therefore required to constantly sense their environment.
- The complexity of programming discrete-event models increases rapidly with the complexity and heterogeneity of the agents and the number of event types. Whereas in a time-driven model the agents may become more complex, however, the interface between the agent and the simulation model does not.
- In event-driven models, the simulator determines in advance the next event based on the current state of the world and steps directly to it (without animating the states of the world in-between). This is inappropriate for our simulation execution model as we require it to be indifferent to participant architecture and facilitate probabilistic behaviour (for Monte Carlo experiments), pro-activity and adaptability.

3.3.2 Managers

This section introduces the three simulation managers which afford control over the simulation execution, plugins and the execution of extraneous events. The simplest of these is the Control Panel. This primarily lets the user run/pause and step through the simulation. In addition to providing progress information and allowing the user to prematurely end a simulation whist still executing post processing, archiving and tidying up of the databases and connections.

The Event Script Manager (ESM) uses the Java reflection API to facilitate runtime execution of Java methods. This allows the user to script the execution of a method at a certain time point with specific variables independent of the platforms compile time behaviour. This script initially takes the form of an input file, but events can be added through the GUI during runtime. Methods can also be scripted for execution before or after the simulation run such that the user can use them for initialisation or post processing. Given the generic nature of scripting method execution there are a vast array of possible uses, these include, triggering events in the simulated physical world, adapting the network topology, altering parameters and timing each agent's entrance or exit from the simulation.

The Plugin Manager (PM) allows the user to launch plugin modules from input files or a GUI during runtime; the key difference between plugins and methods being that plugins persist between simulation cycles meaning that they are repeatedly executed, have memory between simulation cycles and can include a user interface. As a result they form the basis of the many possible data archiving and visualisation tools. The PM, like the ESM auto-detects available plugins and allows the user to launch and remove them during runtime. A plugin can be created by simply using the `plugin` interface. The power of the plugin architecture is illustrated by two key plugins, the DataArchiver and the Visualiser.

DataArchiver: One key feature of any simulation platform is the ability to log experimental data. A basic DataArchiver plugin class is provided in the platform API that can create results logs in the form of spreadsheets. The specific nature of the data and its layout in the output-file is defined by the user as it is scenario dependant. This is relatively easy process of instantiating the DataArchiver's abstract method `getDataRow()` to return a row of data in the form of an array. In each simulation cycle the plugin will then get the required data and archive it in a comma separated file.

Visualisation Plugins: We provide a small group of plugins specifically designed to enable the user to create real-time visualisations of experimental data. At this time we have created three basic forms: line graphs, radial plots and network visualisation. While it is our intention to extend this library further in the future, the user can, of course, create their own as needed.

3.3.3 External Connections

The platform supports many types of external connection. In this section, we review three, TCP/IP connections, MySQL, and access to other external applications.

TCP/IP Communication consists of a client/server pair for communicating with external processes such as situated agents, remote servers and networking the platform to additional simulators.

The MySQL connection is managed by the platform for providing short-cut methods to perform queries and updates. This enables users to store large volumes of simulation data including event logs for post-processing. The participants can also use SQL to: store beliefs, form temporary data structures from more than one table, perform mathematical functions on large datasets or quickly and efficiently search and organise a large amount of information.

External Process Invocation is handled by a number of convenience methods allowing the execution of system commands and external applications from within the platform. These can either be called by user defined code in the network, physical world, participant, or plugin classes; or by a scripted event. This is particularly useful for launching agent processes outside the simulation, calling on Gnuplot or a spreadsheet application to post-process simulation results.

3.3.4 Environmental Interfaces and Abstract Classes

Agent systems operate in a number of physical and network environments from fully connected static networks without the need to model a physical world to vehicular adhoc networks (VANETS). The individual properties of these environments pose unique challenges to the agent system developer, therefore it is essential that agent simulation platforms support the custom specification of these environments. In order to achieve this the PreSage platform contains two abstract classes namely the Network Simulation Module and a Physical World Simulation Module.

Network Simulation Module: The network module's core function is to facilitate the exchange of messages between connected peers and to simulate dynamic connectivity between the participants. Network modules are simple to create by extending the basic abstract class to determine the required behaviour. The following network types have been created: static fully/partially connected, unstructured P2P, hybrid P2P and mobile adhoc networks.

Physical World Simulation Module: The platform supports the inclusion of a simulated spatial environment for the agents. Like the simulated network, the physical world is an interface class which allows for custom specification by the experimenter. The basic interface supports the addition and removal of participants from the world and facilitates the sensing and effecting of their environment. It is up to the user to define the valid actions and their effect on the state of the world, in addition to any rules of the environment not determined by agent behaviour.

An example of using the physical world and network interfaces is an instance of a wireless mobile ad-hoc network (MANET) simulation. In our configuration the agents can move in a two dimensional environment and can sense the relative position of nearby peers. The world itself bounces agents when they interact with its boundaries causing the agents to move in a Brownian motion. The spatial data of the world is used by the network module to calculate the physical connections of the network based on relative distances between peers and their wireless transmission ranges. The physical network topology can then be used to infer the logical connectivity of the participants. A real-time visualisation of the physical world and the resultant network is provided by a plugin shown in Fig. 3.3.

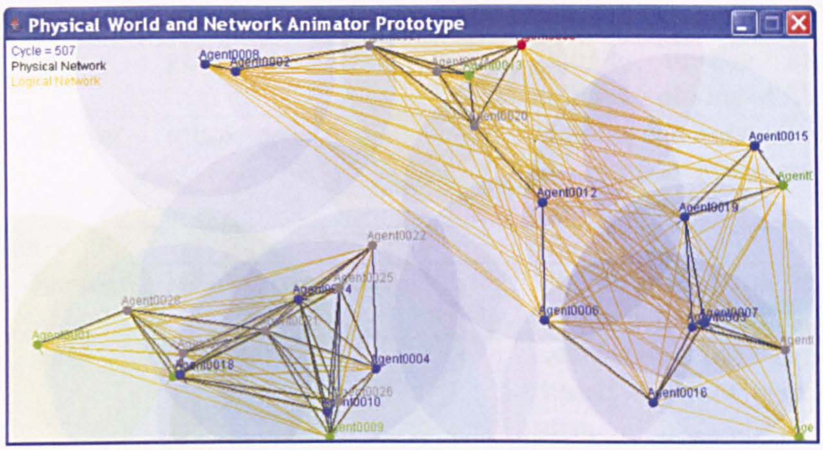


Figure 3.3: Plugin creates a realtime animation of the changing positions of the agents in the simulated physical world and the effect this has on the topology of the physical and logical networks.

3.4 Agents, Participants and Communication

The principal component of the platform is its collection of agents, whose interaction with one another and their environment is our primary interest. In theory it would be ideal for the platform not to constrain the design of the agents in any way. However in order to interact with the base simulation model and ensure the interoperability of participants a degree of homogeneity is required. Table 3.1 lists these prerequisites. Externally the agent must have a globally unique identifier (GUID), defined roles and communicate via a common agent communication interface (as defined in the following section). However, internally the requirements simply facilitate the interaction with the simulation platform, for instance activation of the agent, drawing a representation of the agent onto a graphics context and calling the agent to take its turn via a public method `execute()`. Within these constraints the user is free to develop their own agent architecture be it reactive, deliberative, BDI or otherwise. As such the platform is neutral with respect to the agents' internal architectures.

3.4.1 The Participant Class

It is expected that the majority of users of the platform will be primarily interested in the interaction between agents and the evolution of behaviour within a simulated agent soci-

Variables			
public	String	gUID	globally unique identifier: defined from input file
public	Queue	inbox	to allow the network module to enqueue messages to the agent
Methods			
public	boolean	isRole(String role);	returns true if role is one of the participants roles.
public	void	execute();	called by the simulation thread upon a participants turn.
public	void	onActivation();	called by the platform when the agent becomes active in the simulation.
public	void	onDeActivation();	called by the platform when the agent is removed from the simulation.

Table 3.1: Required methods and variables for a simulation participant, if the Agent is Java based, otherwise a wrapper class can be used to translate the Java interface (this works very well with Prolog).

ety. As such we have developed a root agent class `Participant` from which researchers can derive heterogeneous agents for participation in their simulations. Figure 3.4 shows how one might derive the necessary classes for an online auction scenario in a Virtual Organization and instantiate an heterogeneous population from them. Notice that the class hierarchy allows us to define more or less sophisticated agent strategies: from the simple buyer, socio-cognitive buyer, and onto machine learning or game theoretic buyers. The participant class handles as much of the agent's internal operation as possible (without sacrificing scenario flexibility). Its architecture is a combination of deliberative and probabilistic models, this has proved sufficiently complex for our experiments in emergent behaviour.

To create the individual participants the derived classes must then be launched and parametrised via an input-file. Each row of the file launches and specifies an individual agent's parameters. The core inputs the user must specify for each agent include, among other things, the Java class that includes the agents reasoning and communication protocols which extend our generic participant class, the agents globally unique identifier, the

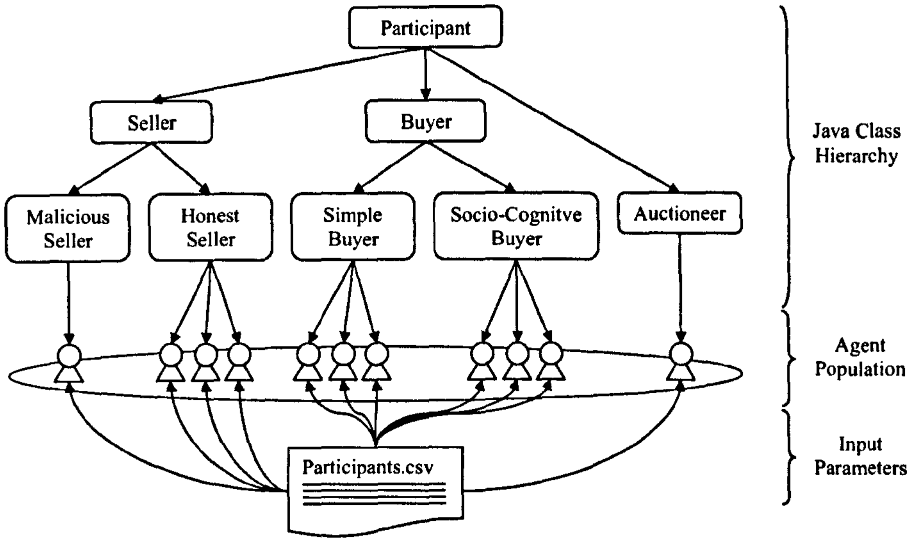


Figure 3.4: Using Java OOP, Participant class and input files to define a heterogeneous agent population

initial roles to be assigned to the agent. In addition the user can provide scenario specific parameters for example in our trust and e-commerce scenario the participants input file also defines what trust model each agent will adopt, its economic constraints/preferences and its character type e.g. its inclination towards and strategy for illegal, unethical and antisocial behaviour.

3.4.2 Agent Communication

The simulation platform aims to put minimal restriction on the internal characteristics of the participating agents. However, in order for the agents to communicate effectively some *a priori* knowledge as to the mechanisms and semantics of communication are required. Pitt and Mamdani [99] argue for the use of a protocol based semantics in the external specification of agent interactions specifically between agents with behavioural and architectural heterogeneity, therefore within the participant class, we provide the necessary mechanisms for handling protocol based communication between the agents. In fact all the mechanisms from message sending and parsing to maintaining the state of current interactions is built in; effectively reducing the users work load to defining the protocol and the reasoning of the agents at each stage of that protocol.

In this section we discuss the defined agent communication interface which permits and facilitates the exchange of information between peers. The interface consists of a higher and lower level component pair: the agent communication language (ACL) and the mechanism for transmitting messages. Message transmission is achieved by calling the

```
sendMessage(Message, InetAddress)
```

method of the Network module. The Network module will either send the message via TCP/IP sockets or enqueue the message to the recipient's inbox queue; depending on whether the recipient is internal or external to the platform. We define our ACL in terms of three components: The message syntax, the mechanisms maintaining the state of a communicative context (a conversation) and the external semantics of the protocols. The following three subsections discuss the way that messages, conversations and protocols are represented, in order to give the user an understanding of how to implement a protocol and associated agent behaviour within our framework.

Message Syntax

In order for agents to parse and interpret information exchanged between them there must be an agreed upon message syntax. In our ACL a message takes the form of a seven-place term:

$$\text{message}\,(R, S, C, P, CKr, CKs, Content)\,;$$

Where the terms R and S denote the intended recipient and the sender respectively; these are instantiated with the agents $GUID$ values. Element C defines the type of communicative act (i.e query or purchase) being performed. P determines the protocol (i.e. CNP or Hello) under which the communicative act is being issued. CKs and CKr are the conversation keys ($ConvKey$) of the sender and recipient respectively; these are used by the agents to recognise the ongoing context in which a message should be interpreted (Pitt and Mamdani [99]). When an agent initiates a conversation they create a conversation object and a instantiate it with a locally unique conversation key. This key is then sent with all subsequent messages. When an agent receives a message without an instantiated CKr it signals that this is the first message of a new conversation; the recipient will then create a new conversation and instantiate its key before processing the message. The format of the message *Content* is determined by the message performative C and the protocol P being followed. An instantiated example is given below, where $Agent0022$ is sending an *introduction* message as part of a *hello* protocol to $Agent0056$. The recipients ConvKey (CKr) is uninstantiated (.) as this message is not part of an ongoing conversation. The senders Convkey (CKs) is instantiated (4.0) such that $Agent0056$'s response can be put into context. The contents of $Agent0022$'s message is simply their name, the roles they believe they perform (consumer), and the address that can be used to communicate with them $127.0.0.1 : 9436$.

$$\begin{aligned}\text{message}\quad & (\text{Agent0056}, \text{Agent0022}, \text{introduction}, \text{hello}, \\ & (.), (4.0), (\text{Agent0022}, \langle \text{consumer}, 127.0.0.1 : 9436\rangle));\end{aligned}$$

Conversations

As an agent executes an interaction protocol with a peer it must maintain local information about the context of that interaction. The agents achieve this by creating a *conversation* object for every multi-stage interaction initiated. A conversation object has the following structure:

$$\text{conversation}\,(CKm, CKt, tID, P, S, To, Beliefs)\,;$$

CKm and CKt are the agent's ConvKey and its peer's respectively. These ConvKeys are used to link incoming messages to an ongoing conversation and to instantiate the ConvKey fields of any replies. The fields tID and P identify whom the conversation is with and which protocol they are following. The state of the conversation S identifies at which point of the protocol (and therefore which section of the agents reasoning) the next message or timeout should refer to. To is the time at which the conversation is internally called, this can happen for a number of reasons: it could be used to end a period of open bidding in an auction protocol or simply to call a conversation to resend a message or tidy up if a peer has failed to respond. Finally the *Beliefs* field is a set of temporary beliefs which the agent wishes to directly associate with a conversation, for instance the current highest bid in an auction.

It is necessary that the participant are able to carry out multiple conversations at any given time; the set of active conversation objects are stored in the conversations KB. Periodically the agent checks to see if any of the conversations have timed out or have completed. If the state of a conversation is completed then the conversation is removed from the KB. However if the conversation has timed out: the code associated with the protocol is passed the conversation. When we refer to the code associated with a conversation, we are referring to the user defined method that defines the agents behaviour at each stage of the protocol as describe in the next section.

Protocols and user defined semantics

The Participant class uses the Java reflection API in order to provide a user extensible protocol library. To add a protocol to the agent the user simply creates two methods:

```
protected void protocol_name(Message msg)
```

```
protected void protocol_name(Message msg, ConvKey
                        convkey)
```

The first method is called on receipt of any message claiming to conform to the protocol. This method performs a number of checks before calling the second method; for instance if the message is part of an ongoing conversation and if that conversation actually exists; or if the message is intended to start a new conversation in which case it will create a new conversation object. The second method is called in three situations: agent receiving a message (via the first method), a conversation timeout in which case the message is null and finally as a result of a child conversation returning. It is in the second method that the user codes the relevant agent behaviours for each stage of the protocol.

This handling of messages and conversations is added to the Participant class for the convenience of users whom do not require a specific agent architecture. With more advanced applications users can override built in conversation and messaging functions allowing messages to instead be passed over to code written in languages supported by the Java Native Interface including among others Prolog, C++ and Smalltalk.

3.5 A Simple Example of Using PreSage

In order to use the PreSage platform you need to import the Java packages to a suitable IDE such as eclipse, if you want to develop components that utilise relational databases you will also need MySQL installed and the MySQL connector visible on your path. The following is the a brief description of the process of creating a simple simulation experiment.

1. Define your scenario in terms of the behaviour of the environment the interactions of the agents with each other and their environment. For this incredibility simple example: all agents can move in a 2D space and send messages to each other which fail to be received with probability $P(sendfailed)$. The messages the agents exchange will contain their current location.

2. Use this description to determine the Actions that the agents can take and the Inputs they will receive from their sensors, and create Java classes for each of these that implement the provided Action and Input interfaces. So in this case we would:

   (a) Create a class called Move implementing the Action interface. This will have as an additional parameter a vector describing the agents move attempt.

   (b) We would also imagine that our agents can sense their location so we create a class called Position implementing the interface Input. This would contain an X,Y coordinate of the agents current location.

   (c) The Message Class is provided by the PreSage platform, and is both an Action and an Input. Lets assume our agents simply wish to inform the others of their current location. We would extend the Message class to create a class called PositionInformMsg. Which would have variables for the x,y coordinates of the sending agent and a defined method for the receiving peer to access them.

3. The Environment component is the arbiter of action and input, and the maintainer of the objective state of the world. To create our ExampleWorld we need to extend AbstractEnvironment in the following way:

   (a) **Create the 'World State':** The state of the world is kept in a object called an Environment Data Model. This data model is written in such a way so as to be serializable and deserialisable into XML format. It therefore acts as the configuration file for the environment and a simple way for storing the objective representation of the world at $time = t$ for later analysis. We need to create one for our simple example. We do this by extending the abstract class AEnvDataModel to the class ExampleEnviroinmentDataModel and including a parameter for $P(sendfailed)$ as well as a mapping between the agent's identities and their current positions.

   (b) **Adapt the behaviour of ExampleWorld by coding the abstract methods in AbstractEnvironment:** These methods update the state of the world based on factors outside of the influence of the Agents, e.g. the weather.

   (c) **Handling the allowable actions in your environment class:** We've defined two Actions in our world Move and Message, how do we code our Example world to deal with them? We create a MoveHandler and MessageHandler class which implements the ActionHandler interface and we add them to our ExampleWorld. Inside MessageHandler will be a method called handle(Action action) in here we will add a piece of code that drops a message with probability $P(sendfailed)$ and returns null, otherwise we will enqueue the Message onto the recipients input queue. MoveHandler is similar in that we might limit the movement of the agents in some way.

4. Create the Agent Participants: The experimenter can create their agent participant types through conforming to the Participant interface; to ensure compatibility with the simulation calls. Since the platform is neutral with respect to the internals of the agent this can be achieved in a number of ways. It is however recommended that the experimenter uses a data model approach like that described for the simulation environment.

5. Create Event methods: If you wanted to see what would happen if the simulation parameter $P(sendfailed)$ doubled at $time = t$, you would create a DoubleFailuresEvent class by implementing the Event interface. Where in the Execute method you would double the variable $P(sendfailed)$ in the ExampleWorld's data model. You then add this to the EventScriptManager to execute at $time = t$.

6. Creating Plugins: These have access to all the information in the Environment Data Model and if the agent participants use the data model approach the information (beliefs) of the agents. These can then be used to:

   (a) Log Data to an XML file

   (b) Show a Real Time Map

   (c) Create and save results graphs

7. Running a Simulation

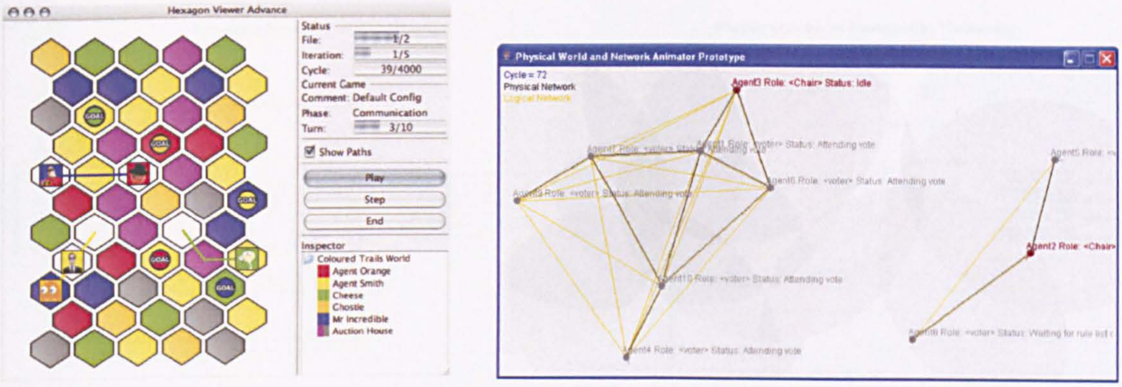
   (a) **Creating the Input Files:** Rather than write large XML files by hand, the easier way is to create all the agents, the environment, the plugins and scripts using their respective Java Class Constructors. Then convert them to XML using the tools provided (see XMLWrite.java).

   (b) **Running your simulation:** Choose between a command-line tool (faster) e.g. "java -Xmx 1024m Presage.java example.xml" or a GUI (good for demonstrations) for example the provided ControlCentre GUI or write your own which interacts with the Simulation class by way of listening for events.

   (c) **Creating Multiple Simulations to cover a Parameter Space:** XMLWrite.java can be looped to create multiple sets of configuration files, incrementing a parameter each time it can also create a command-line script to execute them in turn.

3.6 Demonstrator Applications

PreSage is a platform for developing testbeds, it is itself neutral to the specific scenario, facilitating the use of the platform in a variety of agent based experimental tasks. This



Status
File: 1/2
Iteration: 1/5
Cycle: 39/4000
Current Game
Comment: Default Config
Phase: Communication
Turn: 3/10

☑ Show Paths

Cycle = 72

Agent3 Role: <Chair> Status: Idle

Agent1 Role: <voter> Status: Attending vote

Agent11 Role: <voter> Status: Attending vote

Agent6 Role: <voter> Status: Attending vote

Agent9 Role: <voter> Status: Attending vote

Agent10 Role: <voter> Status: Attending vote

Agent4 Role: <voter> Status: Attending vote

Agent5 Role: <

Agent2 Role: <Chair>

Agent8 Role: <voter> Status: Waiting for rule list

(a)

(b)

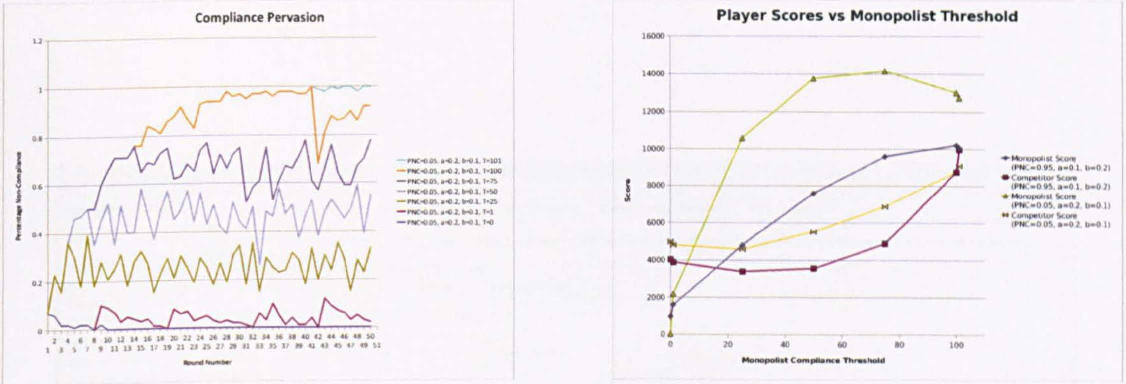
Figure 3.5: Example visualisation Plugins for a) Colored Trails and IPR tile world b) Paxos algorithm voting in MANET's

section outlines the use of the platform to develop in eight independent testbeds for experimenting with agent societies.

Trust and Economics in Decentralised Trading: The first application of PreSage was in a simulation study into the use of socio-cognitive theories of trust and reputation for regulating agent behaviour in an agent mediated marketplace which was intrinsically unmoderated and dynamic [93, 94]. This work demonstrates the platforms ability to support complex agent interaction between heterogeneous agents taking place in the context of a simulated economic environment. See Chapter 5

Coloured Trails and IPR: In demonstration of PreSage's physical world environment we have developed an agent society version of the Harvard Coloured Trails [58] scenario, using its tile world representation of goals, paths and tasks to support research into Intellectual Property (IP) concepts. More specifically we intend to use this test environment to investigate the provision of legal services such as Regulatory Compliance, Conflict Prevention, Alternative Dispute Resolution and Law Making to agent societies. See Fig. 3.5a).

Copyright Games: Studies the pervasion of compliance to norms in a population (by norm, here it is meant as a social rule to conform to a pattern of behaviour. In our copyright games, there are two types of players: a Monopolist (content producer, rights holder), and Players (content consumers). The game is played iterated one versus many, i.e. in each round, the Monopolist plays each of the players. The Monopolist strategies are de-



(a)

(b)

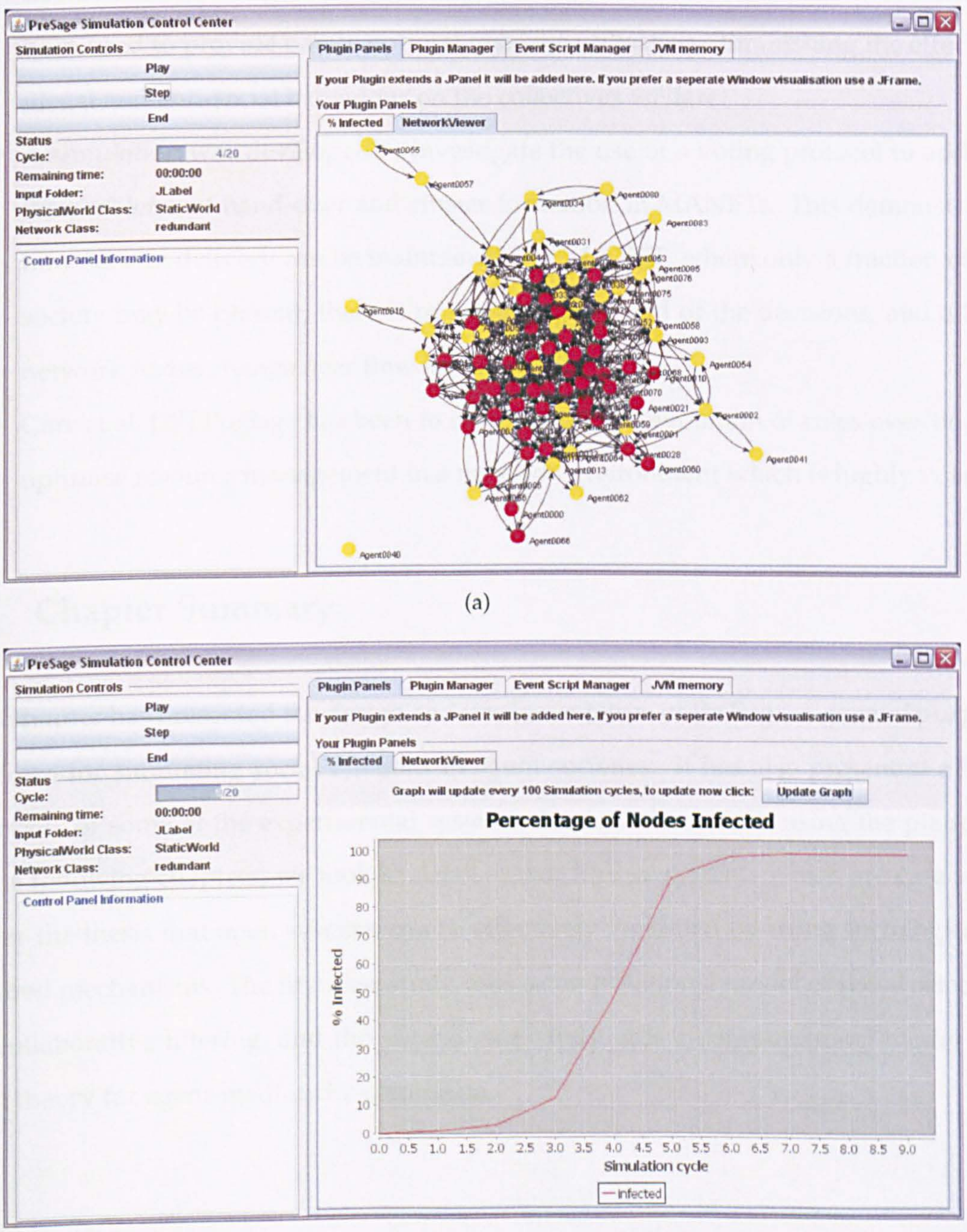
Figure 3.6: Results from simulating compliance pervasion in copyright games

fensive (litigate) or passive; each Player can either be compliant or non-compliant. The results demonstrated that irrespective of population, it was the Monopolist who dictates level of compliance (by litigating when the populations' non-compliance exceeded its threshold), but it was the Players' tendencies which determine utility (cost) of the monopolist's strategy (see Fig. 3.6). They results reinforced some recent findings by the UK Demos think-tank on user attitudes to (il)legal downloading and internet law [43]. Details can be found in [95].

Collaborative information filtering on unstructured P2P networks: We are using PreSage to simulate an agent society whereby peers self-organise the P2P overlay network based on their peer models such that the network itself becomes a collaborative information filter. This has been used to investigate the effect of different agent preference distributions and peer selection models on the structure of the emergent network and population welfare distributions. See Chapter 4

Node Infection: A simple testbed to look at the spread on node infection across a network. Mainly used as a teaching and development example for the new interface design, may be useful for future work on Memes, norm adoption, information dissemination. See Fig.3.7

Resource management: In certain types of network, for example in virtual organization and ad hoc networks, it is a common problem to define a 'fair' collective decision-making mechanism for the allocation of a common resources and to safeguard against either the



PreSage Simulation Control Center

Simulation Controls

Play

Step

End

Status

Cycle: 4/20

Remaining time: 00:00:00

Input Folder: JLabel

PhysicalWorld Class: StaticWorld

Network Class: redundant

Control Panel Information

If your Plugin extends a JPanel it will be added here. If you prefer a seperate Window visualisation use a JFrame.

Your Plugin Panels

(a)

Graph will update every 100 Simulation cycles, to update now click: Update Graph

(b)

Figure 3.7: Example testbed, using standard PreSage GUI a) Random Network failing due to node infection b) Real-time graph showing infection over time

over exploitation (e.g. The Tragedy of the Commons [67]) or under utilisation of resources. PreSage has been used as the basis of three investigations in this domain:

- Local inter-agent negotiation, combined with sanction and trust mechanisms, have been used to provide pareto optimal resource utilisation, diminishing the effects of illegal and anti-social behaviour on the collectives welfare.
- A simulation was developed to investigate the use of a voting protocol to address the problems of hand-over and cluster formation in MANETs. This demonstrated how service delivery can be maintained in a MANET, where only a fraction of the society may be present, there is no centralised record of the decisions, and all the network nodes change over time (see Fig. 3.5b).
- Carr et al. [27] PreSage has been to investigate the adaptation of rules over time to optimise resource management in a network environment which is highly volatile.

3.7 Chapter Summary

This chapter has presented the design and implementation of PreSage, a general purpose platform for simulating social relations in agent societies. It has also presented a brief overview of some of the experimental systems that have been built using the platform. In the following chapters, we look in detail at two further systems which are case studies for the thesis that open systems can be effectively regulated by using sociologically-inspired mechanisms. The first case study uses a computational model of social selection for collaborative filtering, and the second case study uses a computational model of a trust theory for agent-mediated e-commerce.

Self-Organising Social Recommendation

4

4.1 Introduction

In chapter 3 our simulation platform was introduced, specifically we presented an experimental architecture, API and tool set for investigating the effect of agent behaviour and network architecture on the performance of multi-agent systems. The simulation platform is intended as a test-bed for examining socio-cognitive behaviours such as peer modelling, relationship formation and collective belief. This chapter covers the first of two case studies intended to demonstrate the the effectiveness of our platform and examine socio-cognitive behaviours.

4.2 Overview: Information Dissemination

Open multi-agent systems can be formed from heterogeneous peers with disparate goals, requirements and architectures. These peers typically operate in a decentralized fashion and so local knowledge is often partial, inaccurate and subjective. It is therefore necessary to provide some mechanism to enable agents to find 'similar' peers with whom to interact, for commerce, coalition formation, content recommendation, and so on. In this work, we investigate a recommender system based on each agent using a cognitive model of 'the other'. Agents can then subjectively model the preferences/requirements or behaviour of their peers, and utilise this knowledge to form local sub-networks for directing and personalising information dissemination. We show how this self-organization can create a network topology which functions as a collaborative filter, and how different peer selection strategies (and their parametrization) affects the topology of the network and the overall performance of the system.

Example application domains in which this approach could be utilised would include: information dissemination in sensor networks, organisation of trust networks, recommendation, discovery and choice of P2P content or services; and the distribution and selection of plans in robot/agent swarms. The common challenges of these scenarios and thus the features we aim to capture in our abstract game are:

- Highly Distributed: agents cannot reasonably obtain a global view; by which we mean that data replication techniques would prove too costly.
- Information Overload: The scale of the network, the number of options, and the volume of opinions means each individual node would not be capable of processing the global data set in real time, or even a fraction of it.
- Mutable: Knowledge is only viable for a short period due to changing external factors, this period being shorter than an agents own usage cycle.
- Uncertain and Partially specified: Inability to a-priori determine the likely outcome of a choice based solely on logical analysis.

The remainder of this chapter is structured as follows Section 4.3 outlines an abstract plan selection and recommendation scenario on which we base our prototype system and simulations. Sections 4.4 and 4.5 describe in detail our agent models and the interaction between them. In Section 4.7 we describe a set of experiments designed as a proof of concept and the presentation and analysis of the results.

4.3 The Abstract Agent Swarm Scenario

From the perspective of the abstract robot/agent swarm scenario. Assume we have a population of agents with heterogeneous goals and architectures. These agents will execute plans in order to achieve their goals as best they can, we will refer to these agents as operators. The effectiveness of any given plan will be dependant upon a number of factors including an agent's "goal satisfaction metric", its architecture and the local environment be it physical or virtual; we will refer to these factors as an agent's requirements.

A human planner can analyse an agent or a group of agents' requirements before creating a custom plan and sending it to those specific robots/agents, these custom plans are the most effective in fulfilling the agents' goals. However, the planner can only create plans for a small number of agents in any given time frame and certainly doesn't have the resources to analyse each agents' situation and abilities; and subsequently disseminate plans to all the agents individually. Thus, in order for agents to achieve their goals they must exhibit a degree of autonomy in self selecting their plans.

Through this scenario and the experiments defined in Section 4.7 we aim to address the following questions:

- Can self organisation be used to create a network topology which functions as a collaborative filter?
- How do different peer selection strategies effect the topology of the resulting network?
- How do peer selection algorithms and their parametrisation effect the performance of the system with regards the individual agent's decision utilities?

4.3.1 Plans

Our simulated operators will be unable to execute and rate actual plans and nor will the simulated planners be able to create them. Therefore the agent simulation will have to create and operate on mock content. Our design of this mock content follows the following specification: it consists of two parts; the plan instructions (INST) and a Plan Information Descriptor (PID).

A real world PID would likely be a complex XML document containing a significant amount of information most of which is unnecessary for our simulation purposes. Our mock PID is structured as follows: $PID(name, host, port, gUId)$ where name identifies the planner, host and port are the TCP/IP and port combination with which the planner can be contacted and finally the gUID is a globally unique identifier for the plan.

The plan instruction is simply modelled as a floating point number in the range $[0 - 2\pi]$

the reasons for this are made clear in the sections on shared preference modelling and plan creation (4.5.2 and 4.4.1). Thus each plan is the concatenation of the PID with the instruction, the PID itself can be copied and distributed, an example of this is recommendations where the PID is shared with the agents opinion of the plan the PID refers to. When the PID is shared in our recommendation system users are able to not only collate the recommendations referring to the same content by matching the gUID's but also identify the planner and where the plan can be obtained from.

4.4 Planning Agents

For our simulation the planning and execution behaviours belong to distinct agent types[1]. The planning agents are significantly simpler in design, their role is simply to create the mock plans and in a targeted manner seed them into the network. The aim of this being to encourage the operator agents to self select their own plans and thus to reuse the created plans as many times as possible. In effect the planner aims to maximise the autonomy of the agents operating in its planning niche. However the planning agents themselves are a-priori unaware as to which actors will find their plans of use, so they need a simple form of learning to identify those agents first.

4.4.1 Plan Creation

For simplicities sake our planners can create a new plan every $t_{creation}$ simulation cycles ($t_{creation} = 30$ in the following experiments). Creating the mock plans during runtime allows us to continue creating new items for as long as we wish to run the simulation. As mentioned earlier the actual plan is represented as a number in the range $0 - 2\pi$. Each planner has a niche within this range defined by an upper and lower bound. To "create" the new plan the planner agent selects a random number from a uniform distribution within their content creation niche.

[1] Of course another scenario could be that the execution agents themselves form the plans on the fly and execute them before sharing with the network.

4.4.2 Plan Seeding

Operator agents will discover plans though the recommendation system, so the planners need to illicit some recommendations preferably good ones to spread the word about their new content. In our system the planners achieve this by seeding a number operators with the new plan. The planners need to direct their plans to illicit positive recommendations, this is especially important as the first few agents opinions will determine is the others will self select. They do this by requesting feedback from the operators, based on this information they seed subsequent plans to the specific agents who in the past provided the most positive feedback. The protocol for this is shown in Fig.4.1.

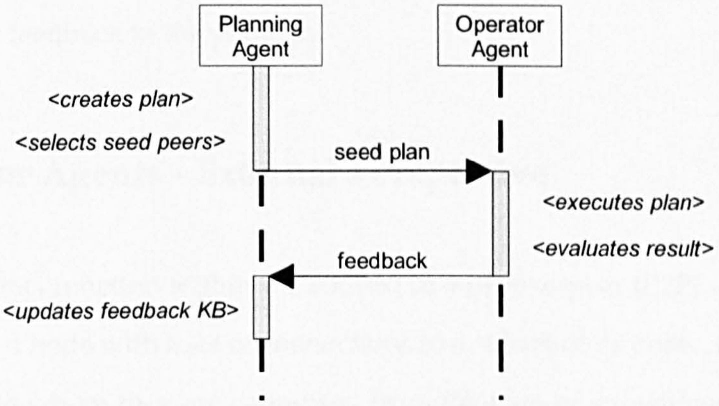


Figure 4.1: Seeding Protocol in AUML

The number of plans a designer seeds the system with is decided by the individual planner themselves. It is updated using a simple derivative-follower algorithm; when the planner creates a new plan it increments the number it will seed. It does this until the number of plans the operator agents self-select in that time period drops below the those in the previous round it then reverses the direction of its increments. The effect of this is to ascend the gradient to a local maxima in operator agent autonomy.

In summary:

- Set of Planning agents $PA = \{pa_1, pa_1, ...pa_n\}$;
- Set of Operator Agents $OA = \{oa_1, oa_1, ...oa_n\}$;
- each Planning agent has a planning niche defined by upper c_u^i and lower c_l^i limits;

where $0 \leq c_l^i \leq c_u^i \leq 2\pi$

- each Planning agent has a set of plans $P_i = \{p_1, p_1, ...p_n\}$
- each plan is the combination of PID(planner_name,host,port,gUID) and INST.
- where gUID is a globally unique identifier, and INST is the plans instruction defined as a random number in the range $[c_l^i, c_u^i]$
- each Planning agent creates a new plan P_{n+1} every $t_{creation}^i$ simulation cycles.
- it seeds this plan to a set of operators $B \subset OA$.
- The seed selection function $B = f(OA)$ chooses the operators which have provided the highest feedback to the planner.

4.5 Operator Agents - External Perspective

The operator agents function within the context of a peer-to-peer (P2P) overlay network each agent being a node with a set of connections to a subset of its peers. Operator agents select the peers to whom they are connected from their set of known contacts. This section describes how this network is initially formed, the interactions which subsequently occur between the operator agents in organising this network and discusses the implication such organisation has for an agents beliefs such as its awareness of available plans, known peers, or the subjective desirability of known plans.

4.5.1 Initial Network Formation

Before our agents can begin to communicate and organise the network topology, a network must already exist. This bootstrapping problem is handled in our experiments in two ways. Firstly when an agent is introduced into the simulation it will be given one contact peer at random from the set of peers already introduced. Ensuring that all peers in the resulting network have a path to all the others, it also means there are nodes which are more central (have more connections) these tend to be the first nodes introduced. Immediately following connecting to its initial random contact, the operator agent will

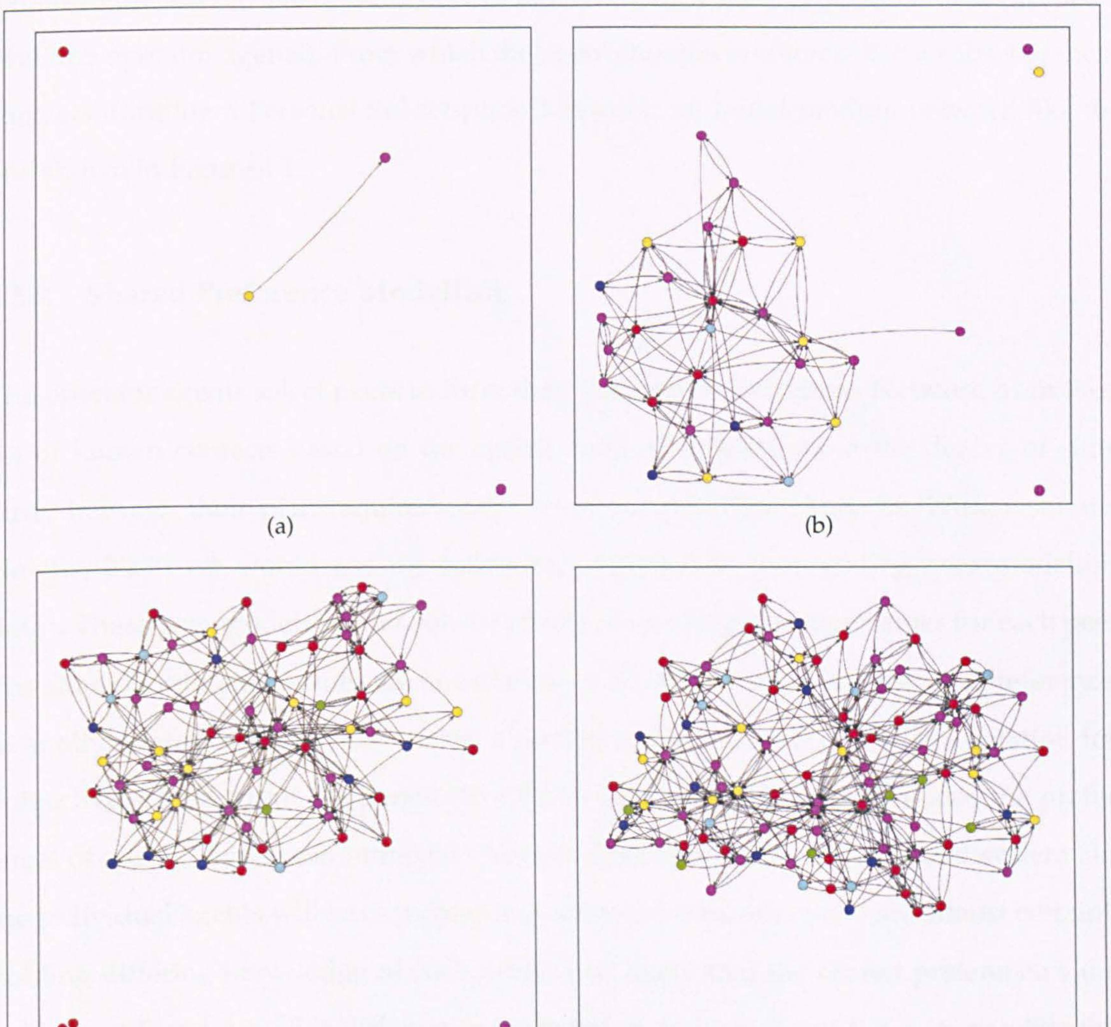


Figure 4.2: The initial random network at 4 stages of being grown.

execute the Hello-walker protocol, sending a message containing the new agents contact details, which traverses the network in a depth first search. Each receiving node randomly routes the message on to one of its known contacts until it is returned to the sender or its time to live is decremented to zero. The contact details are used by the receiving peers to add the sender to its list of known contacts and to send their contact details to the sender. This results in each agent having a set on known contacts (a subset of all the operator agents). From which the agent chooses to connect to a a subset of these contacts, forming a Personal Subscription Network, an initial random network like the one shown in Figure 4.2.

4.5.2 Shared Preference Modelling

The operator agents select peers to form their Personal Subscription Network from their set of known contacts based on the agents subjective belief about the degree of similarity between their plan requirements. From our previous efforts in (Witkowski and Neville, 2003) we would use the following approach to representing peer modelling data[2]. These peer models would consist of sets of shared preference values for each peer. The shared preference values are an estimate of the similarity of one entity's preferences to another entity's preferences within a certain context. $\mathbf{S_A SP_{A,B,\theta}}$ is the notation for: Subjectively from agent A's perspective the degree to which agent B shares the preferences of agent A given the context θ. Note that because this is a distributed system and the individual agents will have incomplete, sometimes incorrect and are almost certainly holding differing knowledge of each other; it is likely that the shared preference value held about B and A by B is different to the belief of A about A and B (i.e. its possible that $\mathbf{S_B SP_{B,A,\theta}} \neq \mathbf{S_A SP_{A,B,\theta}} \neq \mathbf{S_C SP_{A,B,\theta}}$). Also $\mathbf{S_A SP_{A,B,\theta}}$ and $\mathbf{S_A SP_{B,A,\theta}}$ do not refer to the same belief, in fact $\mathbf{S_A SP_{B,A,\theta}}$ would refer to A's belief as to what B believes about it [A].

The context parameter θ can be used to provide a finer grained distinction between agents than simply using their identities, such as the agent's emotional state, current goal and

[2] We will keep the original acronym $\mathbf{S_A SP_{A,B,\theta}}$ and simply state for the purposes of this work preference and requirement are used interchangeably

location. The use of θ may be necessary if the shared preference between two peers is very high in some contexts but low in others. In the following work we have omitted θ for clarity; and the system is formulated as if there is a single context.

4.5.3 Recommendation via the Personal Subscription Network

The Personal Subscription Network of Agent A (PSN_A) is the set of agents to which agent A is subscribed (connected), and whose recommendations Agent A is willing to pass on further to its subscribers. In our implementation the members of the Personal Subscription Network (PSN_A) are a subset of top peers selected by agent A's $\mathbf{S_A SP_{A,B,\theta}}$ beliefs. This group is constantly updated as the agent learns more about the effectiveness of certain plans, meets new peers and receives more information about existing peers.

The agent autonomously subscribes to peers in order to receive their recommendations and selected recommendations from third parties, this allows the agent to discover plans ranked by similar peers and rank the desirability of those plans based on the opinions of its PSN and those connected to its PSN.

When the agent executes a plan it forms an opinion of the effectiveness of the plan and uses this information to compare its opinion with those of its peers. It subsequently reorganises its connections and subscriptions based on the similarity of those peers' opinions.

Before passing on third party recommendations the client considers the source and its degree of shared preference, if it is not sufficient or the third party recommendation disagrees with their user's opinion the recommendation is not propagated. This stops recommendations from penetrating certain areas of the overlay network whilst directing them to others. The emergent collaborative filtering effect provides users with tailored content recommendations.

Unlike centralised recommender systems SID is based on three complimentary principles; network self organisation, social networks and collaborative filtering. These work in unison to control and direct the flow of information (in this case recommendations) between the agents and across the evolving network. The effect is to reduce the number of recommendations handled by each node whilst increasing the relevance of the opinions to those nodes which receive them. Thus the network itself pre-filters recommendations reducing the load on the collaborative filtering algorithms inside the individual agent nodes.

Social networks are used in many domains to represent the relationships between the actors in a population. The nature of the relationship varies depending on the nature of the analysis required. In our system a relationship between nodes is one of information exchange; this exchange of information occurs via the subscription protocol Fig. 4.3. Which is effectively a registration of interest in a peers opinions on the part of the subscribing agent. Following subscription the recipient becomes part of the subscribers personal subscription network (PSN). The recipient will send the initiator both their recommendations and forward those from peers in its personal subscription network until they receive an unsubscribe message. Relationships are formed and broken on the basis of the agent's Shared preference estimates thus the PSN is constantly updated as the agent meets new peers. After an agent recalculates its Shared preference estimates it will re-evaluate the set of peers to which it subscribes. It is with these peers the agent will temporarily maintain subscription relationships with. Therefore each agent has its own personal subscription network defined by connections to its estimated closest matching peers. The contents of recommendation messages sent in this protocol are a 3-tuple consisting of what the recommendation is referring, the identity of the recommendations source, and the opinion itself:

$$Rec(DID, SourceId, Opinion)$$

The subscription relationship is unilateral in that an agent may subscribe to another without the subscription being reciprocated. A major disadvantage of this is that once the agents have organised to select the best peers for their PSN it does not allow new peers to become important recommenders. This is because unknown peers are given a default $\mathbf{S_ASP_{A,B,\theta}} = 0.5$ which is likely to be less than peers already selected into an agents PSN. Thus they are not selected to provide recommendations and their recommendations are not forwarded on possibly ignoring an important source of information. To overcome this and to prevent agents getting caught in local maxima, agents are able to

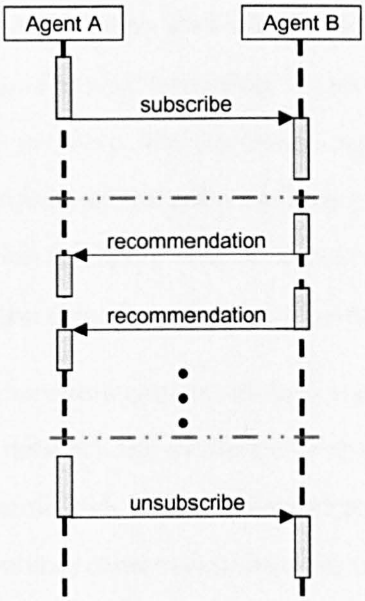


Figure 4.3: Subscription Messages

pro-actively inform a peer of their opinions. While these recommendations will not be forwarded on, they allow a peer to tell others what they think and thus force them to maintain an $\mathbf{S_{A}SP_{A,B,\theta}}$ for them. A peer will pro-actively forward to the agents which they subscribe to, in effect it makes the subscription protocol bilateral even when one side has not subscribed to the other.

These simple rules and protocols give rise to a number of system wide properties namely, the localised and targeted discovery of content and peers, organisation of the network based on preference and the subjective and localised perception of reputation.

Organising the global subscription network: Given the dynamic nature of the subscription relationship as an agent collects recommendations it will self-organise its connections. Eventually it will subscribe to those peers in the system with which they have the highest degree of shared preferences. If we were to map this organised global network we would find a lattice structure where agents holding opposing preferences are positioned on opposite sides of the structure, and those with intermediate preferences would link them over a number of hops. This spatial arrangement creates a preference gradient across the network where any localised sub-net are a cluster of peers sharing similar preferences.

Localised perception of content reputation: The subjective reputation of an item of content is collectively informed by an agent's peers and dependant upon the set of recommendations available to the evaluator. Thus by restricting the recommendations received based on the agents position in the network, the perceived reputation of an item of content will depend upon the evaluators current and previous position within the global subscription network. This position is ideally within an organised cluster and therefore the agent's subjective opinion of the object's reputation is informed by like minded peers.

Content discovery: The agent discovers content though word of mouth, as recommendations are spread through the network the existence of that item of content and where it can be obtained is also communicated. When the agent receives a recommendation they update their list of known content (content database) to include the item. Subsequently every time the agent receives a new recommendation it updates the rating given to the content.

From a scalability perspective an agent may be unaware of specific items of content and therefore not store or process information about them. This occurs because at some stage in the propagation of recommendations across the global network peer agents evaluate the content as not desirable enough to consume. As such they do not form an opinion and consequently do not spread their recommendation of that content. Agents further across the global network may never learn of the existence of the content and thus never need to maintain and process a reputation for it.

Peer discovery: There are two mechanisms through which the agents discover their peers namely introduction protocols and the sharing of recommendations. When peers first connect to the system they are assumed to be a-priori aware of one peer and will enact the hello protocol to form an initial connection. This is quickly followed by sending a hello-walker message 3.4.2 in order to obtain a few more initial contacts (in the following simulations we set the hello-walker protocol to obtain five additional contacts). Barring this preliminary bootstrapping the agents discover peers by receiving forwarded recommendations, these act as a referral to the source of the recommendation. The result of this is to reduce the set of peers an agent must model to a fraction of the total peers in the system while simultaneously improving the average shared preference of its contact set.

4.5.4 Internal Agent Framework

This section takes a closer look at the operator agents cognitive framework Fig 4.4. The framework defines the interactions between the different reasoning tasks undertaken from the perspective of an agent requiring a new plan. However it is also possible for the framework to be initiated from an alternative point, i.e. upon receipt of a new recommendation or a planner agent seeding the operator a plan. The stages are numbered for clarity.

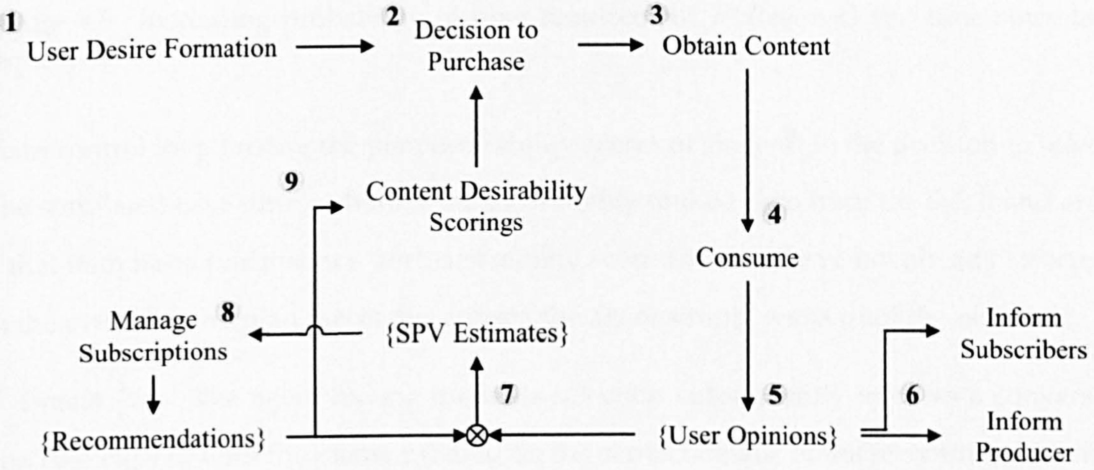


Figure 4.4: Control and feedback framework of the Operator agent

1 *Plan Requirement:* In a real agent swarm there would be large number of factors that would determine an operator agent's requirement for new plans. Since we taking a abstraction of the scenario, we have opted for a Monte Carlo approach, where as the time since the operator last choose a new plan (dt) increases so does the probability that the agent will make a choice in this time cycle ($P(Request)$). The equation in Fig 4.5 determines $P(Request)$ from dt. This way the exact time and order in which the agents select plans is randomised but constrained such that the agent must at least try to select a desirable plan when $P(Request) = 1$. Note if it considers that none of the plans it is aware of have a positive desirability score it will give up and wait.

2 *Plan Selection:* When our simulated operator agent decides its time to obtain some new a new plan they generate a short list of the plans it has discovered. This list of plans is scored and sorted by the agents belief about the desirability of each item. This closes the

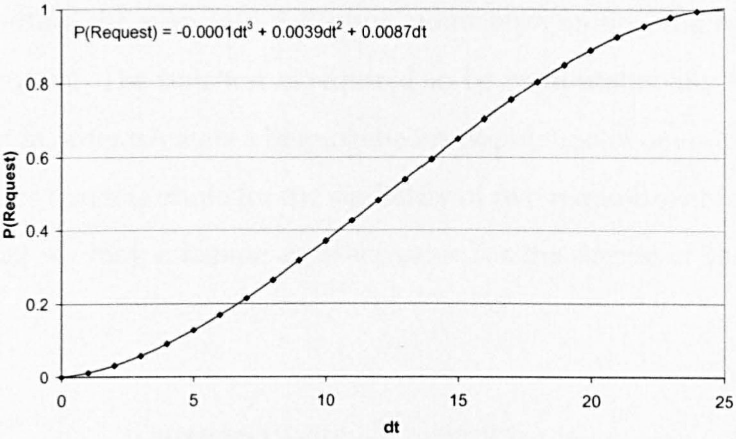


Figure 4.5: Increasing probability of plan requirement $P(Request)$ vs. time since last request dt

main control loop linking the plan desirability scores of stage **9** to the decision to select. The simulated user simply chooses the most highly ranked item from the list, if and only if that item has a positive non-zero desirability score and they have not already selected. In the event that no plan meets this criteria the agent simply waits until the next time.

3 *Obtain Plan:* The agent having made its selection subsequently initiates a conversation (see Fig.4.6) with the planner (based on the contact details in the recommendations). Where the plan is transferred to the operator agent. The remainder of the protocol namely the feedback to the planner is addressed in sections **5** and **6** of the walk through.

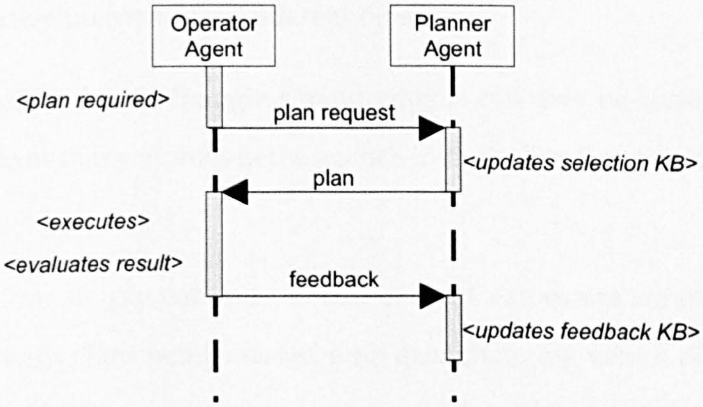


Figure 4.6: Plan Transfer Protocol in AUML

4 *Plan Execution and Outcome Utility:* At this stage in the framework we are looking to define the Operator requirements Model this is a simulation model of the outcome of an operator executing a plan. Essentially it is a function to translate the floating point number representing the plan into a floating point representing the operators' utility upon plan execution. The function is required to be computationally simple and easily parametrised in order to create a heterogeneous population of operator requirements. It is also necessary that a formula for the similarity of two requirement functions be well defined such that we may calculate an exact value for the degree of shared preference between peers.

$$\textbf{Outcome Utility} = \mathbf{5} \times \textbf{sin}\,(\theta + \varphi) + \mathbf{c} \tag{4.1}$$

With these factors in mind operator utility is defined by equation (4.1) with a default range of [+5 to -5]. Altering the amplitude of the function sets how extreme a peers views are. Plan (θ) is defined as by a variable in the range $0 - 2\pi$ radians as discussed in section 4.4.1. The phase variable φ determines how the plan values translate to utility and which peers have similar requirements (the lower the phase difference the higher the degree of shared requirements). As illustrated in Fig.4.7. Constant c determines the peers bias towards succeeding with any plan and in conjunction with the amplitude can be used to define the consumers range of outcomes.

- Different planners create plans in different niches which means they appeal to certain phase adjustments and so different operators.
- Preference shifts due to changing requirements can also be simulated by simply adjusting one or two variables in the agent's initial utility function as the simulation continues.
- Because we can simply continue to add content values the simulation can go on indefinably with plans being created with ever changing values of θ. Thus there is no need to predefine the plan outcome set - for specific predefined plans at initialisation, which would otherwise limit the simulation to a finite game.

By using equation (4.1) as our operator requirements and utility function we can derive the actual Shared Preference value between two operators A and B using equation (4.2). Unlike the subjective evaluation of $A's$ Shared Preference value ($S_ASP_{A,B}$) which is based

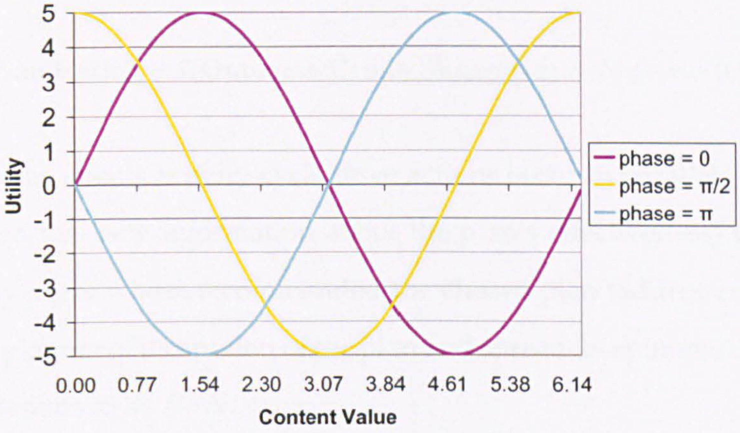


Figure 4.7: Content Utility vs. Content Value for phase values 0, $\pi/2$ and π

on its possibly incomplete set of $B's$ recommendations, equation (4.2) calculates the actual degree of similarity between the requirements functions whereby $SP_{A,B} = SP_{B,A}$ but not necessarily equal to $S_ASP_{A,B}$ or $S_BSP_{B,A}$. Equation (4.2) is important as it enables us to measure the agents' ability to seek out their most closely matched peers and the accuracy with which they evaluate their subjective beliefs.

$$\mathbf{SP_{A,B}} = \frac{10\left(2\pi\right) - \int_0^{2\pi} \left|5\sin\left(\theta+\varphi_A\right) - 5\sin\left(\theta+\varphi_B\right)\right| d\theta}{10\left(2\pi\right)}$$

$$= \frac{2\pi - \left| \begin{array}{c} -\cos\left(\frac{3\pi-\varphi_B+\varphi_A}{2}\right) + \cos\left(\frac{\pi-\varphi_B+\varphi_A}{2}\right) \\ -\left(-\cos\left(\frac{3\pi+\varphi_B-\varphi_A}{2}\right) + \cos\left(\frac{\pi+\varphi_B-\varphi_A}{2}\right)\right) \end{array} \right|}{2\pi} \tag{4.2}$$

5 *Returning Outcomes:* Whilst Equation 4.1 gives us the exact value of the outcome utility gained by our simulated operator, it would be considered unfair to pass this directly to its agent or their peers in the form of recommendations. As the percision of Equation 4.1 is implausible for a real world agent. This would subsequently provide a significant advantage at the Shared Preference modelling stage. We therefore obfuscate the data by increasing its granularity to a simple integer rating in the range (-5 to +5) using equation 4.3. The operators outcome rating is then tagged onto the plan's entry in its database.

$$\textbf{Plan Rating} = \textbf{f(Outcome Utility)} Where f(x) = floor(x + 0.5) \tag{4.3}$$

At this point in the agents activity cycle three actions occur in parallel. Specifically, the agent will (given this new information about the plan's effectiveness) reassess its SPV estimate for any peers whom recommended the chosen plan (addressed at stage 7), it will inform the planner of its opinion of the plan and spread its opinions into the network via recommendations to its PSN.

6 *Recommendation* At this stage in the framework the agent will communicate its opinion with peers on the network.

7 *Calculating Shared Preference Estimates:* As discussed in section 4.5.2 $\mathbf{S_ASP_{A,B}}$ represents Agent A's estimate of the overlap between its requirements and User B's preferences. The function of this stage in the framework is to attempt to evaluate the current $\mathbf{S_ASP_{A,B}}$ based on the Agent A's available knowledge of its outcome ratings and those of agent B. The Agent achieves this by comparing the n most recent recommendations received from a peer B to which agent A is aware of its corresponding ratings. This stage of the framework is initiated when the agent receives a new recommendation from a peer or executes a plan. Equation 4.4 takes the sum of the difference of opinion between the peers and normalises it to the range [0-1] where $\mathbf{S_ASP_{A,B}} = 1$ is a perfect preference match. U_B is a set containing the n most recent recommendations made by B that Agent A is aware of and for which Agent A has a corresponding opinion. This ensures that $\mathbf{S_ASP_{A,B}}$ is adaptive to a peer's changing requirements over time. The constant 10 is determined by the ratings scope [-5,+5].

$$\mathbf{S_ASP_{A,B}} = 1 - \sum_{did \in U_B} \frac{|Opinion_{A,did} - Recommendation_{B,did}|}{(n \times 10)} \tag{4.4}$$

8 *Manage Subscriptions - Organise Social Network:* Taking as input its contact database and its updated Shared Preference Estimates the agent sorts its peers and selects those peers with which it has the highest Shared Preference Estimates. This subset of peers becomes the agent's new personal subscription network (PSN) once the necessary subscribe and unsubscribe messages are sent. For more details on this process refer back to Section

4.5.3. Reorganising the PSN has a knock-on effect on the set of recommendations the agent receives, ideally improving their relevance to the agent which is useful for our next stage where the agent recalculates its content desirability scores.

The method by which the agent selects its PSN is one of the experimental parameters during our simulations, where we compare two selection algorithms.The first is based on a minimum $\mathbf{S_ASP_{A,B}}$ required to be in agent A's PSN (Threshold approach) and the second selects the X highest $\mathbf{S_ASP_{A,B}}$ rated peers (Top X approach).

9 *Plan Desirability Scoring:* A plans desirability score ($\mathbf{Des_{PID}}$) is an agent's collectively informed opinion of the expected outcome utility which executing a plan will generate. To update $\mathbf{Des_{PID}}$ the agent selects from its database the recommendations which refer to PID. This creates the set of recommendations R_{PID} in Equation 4.5. Where n is the cardinality of R_{PID}. Rec_{PID} is an individual opinion of the plan in the range [-5 to +5].

$$\mathbf{Des_{PID}} = \frac{\sum\limits_{Rec_{PID} \in R_{PID}} Rec_{PID}}{n+1} \tag{4.5}$$

An alternative approach to calculating $\mathbf{Des_{PID}}$ would have been to take a mean of the recommendations in R_{PID}. However this would give a single recommendation equal credence as having many recommendations for the content. Allowing content with a single +5 vote to be more desirable than another with nine +5 and a single +4 recommendation. By using $n+1$ as the divisor in Equation 4.5 we correct the desirability value towards zero for low n and towards the average recommendation in R_{DID} as n increases. A further alternative to equation 4.5 would be to use a variant of Bayesian Rating, which we intend to investigate as part of our future work.

This approach differs again from our earlier work in [124] where a cut-off pre-filter was applied using the agents top x peers, and the credibility weighting applied to each recommendation was based on the agents' Shared Preference Estimates. Here we do none of this; in effect once a recommendation is received it will affect the desirability score. This of course is less discriminating and if this was used in a centralised system, simply summing a set of recommendations for each piece of content would not work.

However in this case once the agent has self-organised within a suitable cluster of peers; recommendations that are detrimental to the peer are filtered by the network. So in effect equation 4.5 simply gives us the most credible estimate it can given the agent's set of received recommendations, while the information exchange protocols and network organisation aim to keep unsuitable opinions out of the agents' database in the first place. This helps satisfy our objective for reducing the storage and computation at each node. The agents will simply be unaware of information which is of lesser or detrimental value to it. Reducing the number of options and opinions to compute.

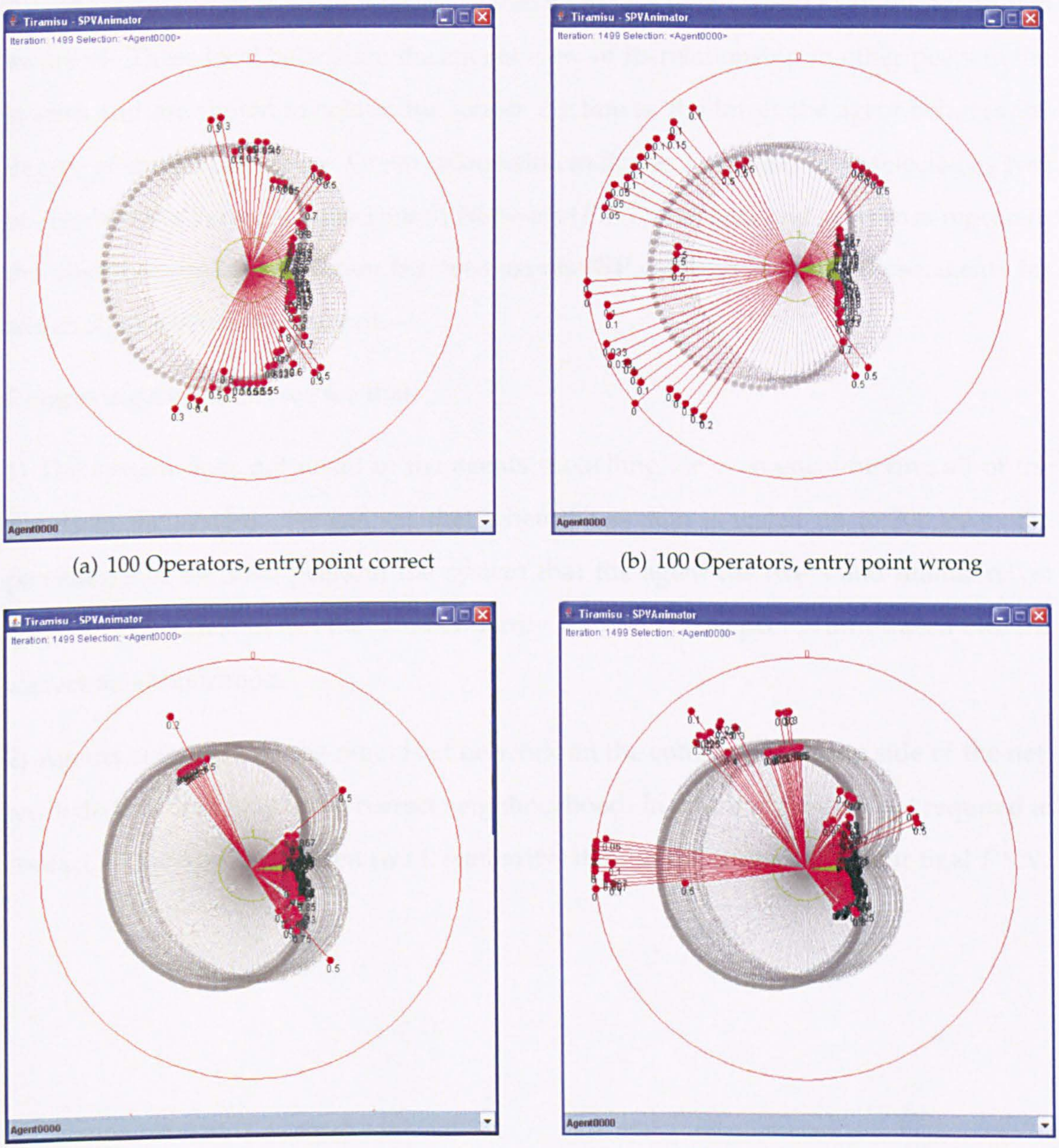
Thus each operator agent maintains an up to date set of beliefs about the desirability of the available plans ready for the next time one is required [2].

4.6 Initial Experiment - New Agent Introduction

The following four simulation runs were performed with a view to investigate the behaviour of agents when added to the system, after the initial agents had organised the network topology. In order to verify that the new agent would be able to discover the best peers, and do so efficiently.

The experimental parameters for these experiments:

- Two system sizes of 100 and 300 Operator agents.
- All the runs use the Top X peer selection algorithm where X=10.
- A single operator agent (Agent0000) is introduced to the simulation at $t = 1000$
- Agent0000's initial contact is not random but specified as a parameter. Where the contact maximises or minimises the objective measure of similarity between Agent0000 and its first contact.
- The result of this is to add the agent into the organised network, either in the perfect or the worst location for its requirements.


Tiramisu - SPVAnimator
Iteration: 1499 Selection: <Agent0000>
Agent0000
Tiramisu - SPVAnimator
Iteration: 1499 Selection: <Agent0000>
Agent0000
Tiramisu - SPVAnimator
Iteration: 1499 Selection: <Agent0000>
Agent0000
Tiramisu - SPVAnimator
Iteration: 1499 Selection: <Agent0000>
Agent0000

(a) 100 Operators, entry point correct

(b) 100 Operators, entry point wrong

(c) 300 Operators, entry point correct

(d) 300 Operators, entry point wrong

Figure 4.8: These radials diagrams show (grey) all the agents in the simulation and their objective shared preference in relation to a specific agent (Agent0000). The Agent's known peers and its subjective opinion of them (in colour) and the agent's selected Personal subscription network (in green)

4.6.1 Results

The radial diagrams Figure 4.8 present the subjective evaluation of an agent (in this case Agent0000) about its shared preference value $\mathbf{S_A SP_{A,B}}$ with respect to all the agents it is aware of. These local beliefs are the agents view of its relationship to other peers in the system and are shown in colour, the longer the line is, the lower the agent believes the degree of shared preference. Green colouration indicates a peer has been selected as part of Agent0000's Personal Subscription Network (PSN). Background grey lines represent the objective similarity relation between agents ($\mathbf{SP_{A,B}}$) and includes those agents for which Agent0000 is unaware of.

Comparing the radials we see that:

1) The system does not result in the agents modelling, or even encountering all of the agents in the system. We can see that when the system is scaled up to 300 peers the percentage of the total peers in the system that the agent discovers and maintains an entry for decreases. In fact the number hardly increases if the peer is introduced into the correct neighbourhood.

2) Agents entering into the organised network on the completely wrong side of the network do find their way to the correct neighbourhood. In addition they are not required to contact all the agents between (w.r.t. similarity) their initial contact and their final PSN.

4.7 Main Experiment

The main set of experiments in this chapter will use our platform to examine the effect of various Operator Agent parameters e.g. peer selection strategy and preference distribution on the resulting network and the performance of the agents. Our scenario is highly reconfigurable, however some of the variables will be kept the same between simulation runs, namely the number of Planning Agents (20), the niche for each planning agent (equal divisions of the range $[0 - 2\pi]$), the number of Operator Agents (100). Despite this the variables specified in this experiment still represent a sizeable state space, in order to explore the possible configurations we will present the results from 22 simulation runs; comprised of 11 distinct peer selection function configurations run against 2 different preference distributions.

In summary our configuration Parameters:

- Peer selection functions:
  - Top X where $X = [10, 20, 40, 60, 80, 100]$
  - Threshold where the $Threshold = [0.9, 0.8, 0.7, 0.6, 0.3]$
- Preference distributions - i.e. how φ_i from Equation 4.1 (which determines the agents' requirements) is distributed across the population):
  - Uniform - φ is uniformly distributed from $[0 - 2\pi]$
  - Majority/Minority where φ is designated by an normal distribution, resulting in the majority of peers having similar requirements

Our experiments will record the following dependent variables:

- Network Structure
  - Frequency of node degree.
  - Distribution of node degree.
- Decision making in the form of mean absolute error (MAE).
- Agent Welfare
  - Overall utility of the individual agents.
  - Utility distribution.

4.7.1 Simulation Results

Network Topologies

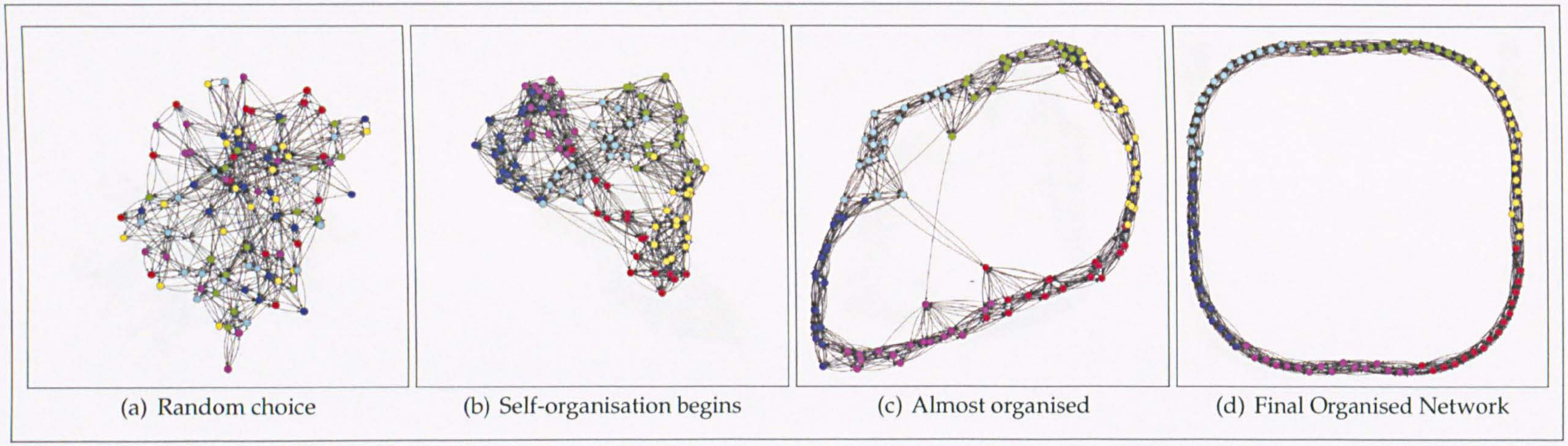
Figures 4.9, 4.10, 4.11 and 4.12 illustrate the affect that altering the experimental parameters has on the resulting recommendation network topologies. For four of the simulation configurations they show the progression from the initial randomly formed network to the final network at the end of the simulation.

When the agents requirements are uniformly distributed the resulting topologies take the form of lattice rings (Figures 4.9, 4.10). This arises from using $\varphi = [0, 2\pi]$, such that the first agent ($\varphi_{Agent0} = 0$) has an equal affinity to the second agent ($\varphi_{Agent1} = 2\pi/N$) as it does last agent in the range ($\varphi_{AgentN} = 2\pi - 2\pi/N$). Figures 4.13b and 4.14b show that $TopX = 10$ results, unsurprisingly, possess an average node degree of 10 whereas $Threshold = 0.9$ has a higher degree of approximately 18. Hence in our topology illustrations the $Threshold = 0.9$ configuration results in a tighter more connected topology than the $TopX = 10$ configuration, however the shape remains the same; dictated by the uniform preference distribution.

Looking to the topologies that arise from the combination of Threshold selection and a Majority/Minority preference distribution (Figure 4.12) we see a ring structure with a cluster on one side. The cluster being comprised of the agents whose φ parameters are closer together (the Majority biased). As the filter opens more connections are made to and from the minority peers, the ring protrusion gets progressively assimilated into the cluster until as we near a $Threshold = 0$ we are left with a fully connected network. This can be seen quantitatively in Figure 4.16a which shows the number of connections coming into each agent node (the node degree distribution) and Figure 4.16b which shows the frequency of occurrence of a node degrees. The shape of the node degree distribution clearly shows the distinction between the highly connected majority peers (the edges of the graph) and the lower connectivity of the minority peers (centre of the graph), in Figure 4.16b we see two spikes in frequency also indicating the two groups. In effect the Threshold approach reflects the actual similarities between the agents in the topology it forms.

This is not the case when we analyse the topologies resulting from combining TopX selection and a Majority/Minority preference distribution Figure 4.11 we see that $TopX = 10$ results in a broken ring (or line) caused by the presence of a discontinuity in the Majority/Minority distribution. Unlike a uniform distribution the first and last agent specified are dissimilar enough such that when the peer selection mechanism is very tight e.g. $TopX = 10$ the peers at either end of the requirements spectrum do not connect and the ring breaks. The ring reforms for higher X values, at $TopX = 20$ it appears almost identical to ring structure seen with the uniform distribution at $TopX = 20$, and at $TopX = 100$ it resembles an amorphous lump with every peer fully connected to rest. So in these cases ($TopX = [20, 100]$) the TopX selection method overrides the preference distribution's ability to dictate the general topology thus the topology metrics in Figures 4.13a,b and 4.15a,b are very similar.

Where $TopX = [40, 60, 80]$ the distribution of node subscriptions becomes less uniform (as distributed across the population see Figure 4.15a) showing peaks and troughs of connectivity across the population. This is the result of two factors, firstly TopX selection forces each agent to create X connections regardless of the utility of the X^{th} connection, secondly connections have a direction when viewed relative to an agent. So when each agent must select 40 peers to connect to, they don't necessarily connect to 20 agents to the left (lower φ) and 20 to the right (higher φ). The direction of an agents connections is biased towards the direction in which the differences between agents is shortening, so firstly away from the most isolated peer and towards the mainstream; and to a lesser extent away from the discontinuity between $Agent0000$ and $Agent0099$.



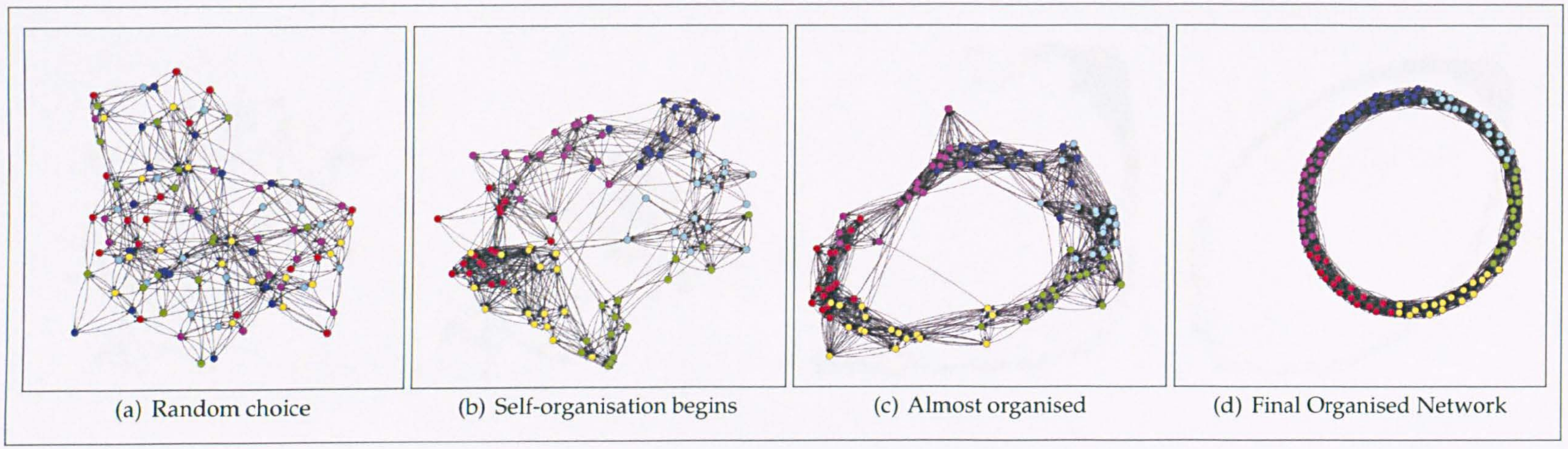
(a) Random choice

(b) Self-organisation begins

(c) Almost organised

(d) Final Organised Network

Figure 4.9: Global Agent Subscription network using a Top X algorithm (X=10) and a Uniform Heterogeneity distribution



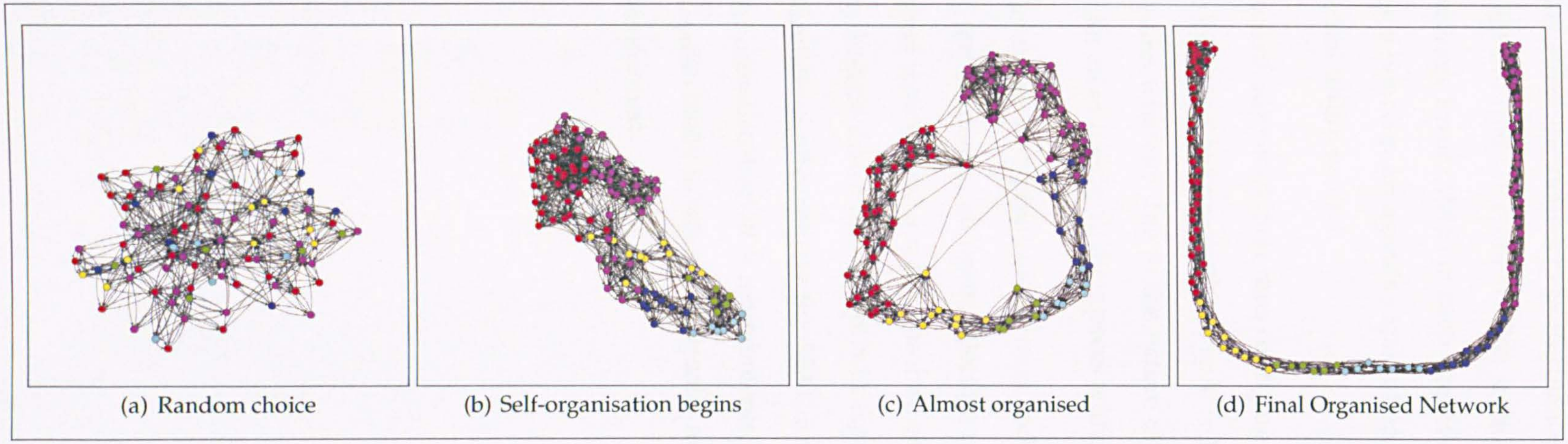
(a) Random choice

(b) Self-organisation begins

(c) Almost organised

(d) Final Organised Network

Figure 4.10: Global Agent Subscription network using a Threshold algorithm (T = 0.9) and a Uniform Heterogeneity distribution



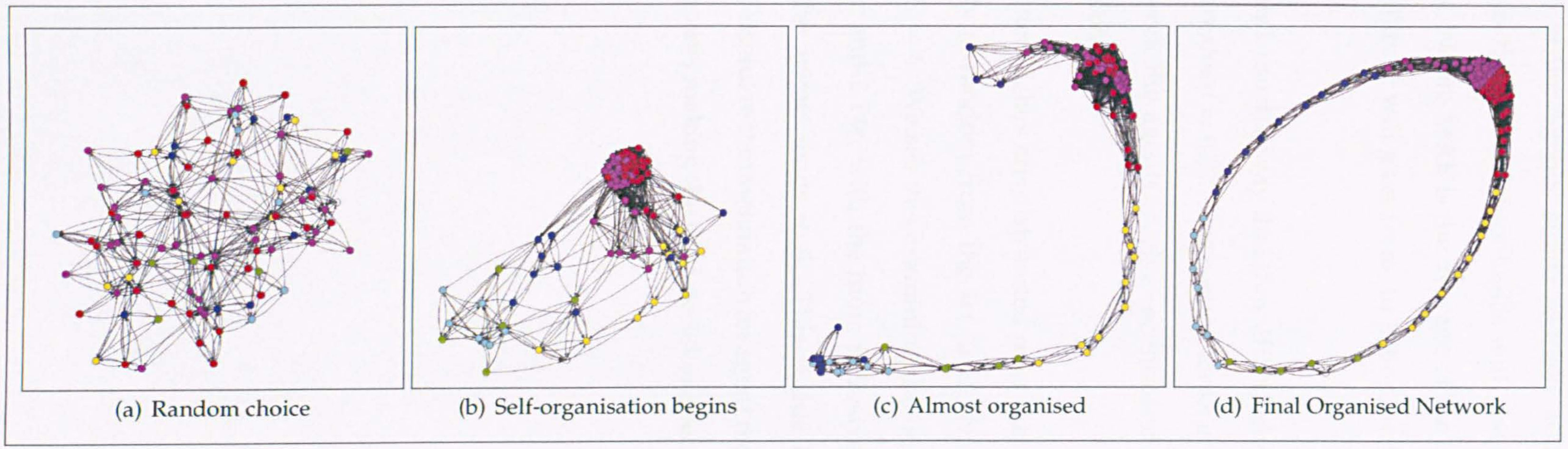
(a) Random choice

(b) Self-organisation begins

(c) Almost organised

(d) Final Organised Network

Figure 4.11: Global Agent Subscription network using a Top X algorithm (X=10) and a Majority/Minority Heterogeneity distribution



(a) Random choice

(b) Self-organisation begins

(c) Almost organised

(d) Final Organised Network

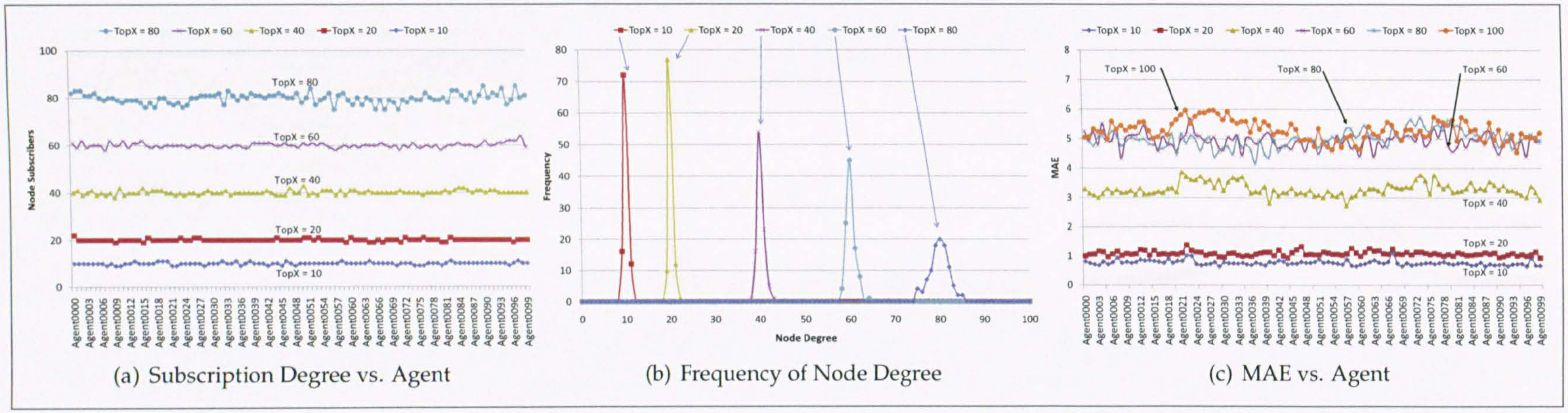
Figure 4.12: Global Agent Subscription network using a Threshold algorithm (T=0.9) and a Majority/Minority Heterogeneity distribution

Mean Average Error - MAE

Having seen the topologies which our experimental parameters have formed, how does this affect the networks performance in terms of the Agents' ability to make accurate, well informed decisions? This section contrasts the network topologies with the Mean Average Error (MAE) of each agents decisions. Where MAE is the average of the difference between an agent's expectation of the utility it will gain from an action, with the actual utility gained.

Two of our simulation runs configure the agents in such a way that they effectively form fully connected topologies ($TopX = 100$ or $Threshold = 0.3$). This section looks at these results whereby due to the nature of the network the agents receive recommendations from all or nearly all their peers without selection.

Given the way plans are formed and the planner niches are distributed over the range of requirements, an agent selecting completely at random from the set of all available plans would have a mean absolute error $MAE = 5$. We see this examining the uniform preference distribution results in Figures 4.13c and 4.14c, with the more inclusive peer selection configurations the MAE result for the agents tends to 5. This is due to the uniform distribution of requirements, for every useful recommendation an agent receives it is also likely to receive an equally misleading one, making the final decision effectively uninformed.

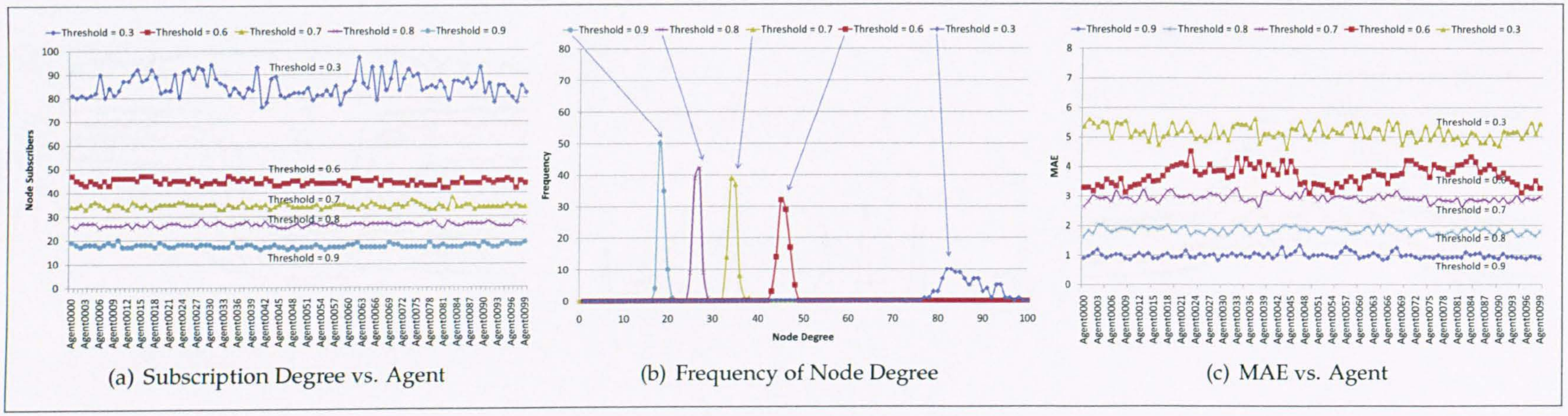


(a) Subscription Degree vs. Agent

(b) Frequency of Node Degree

(c) MAE vs. Agent

Figure 4.13: Using a TopX algorithm and a Uniform Heterogeneity distribution

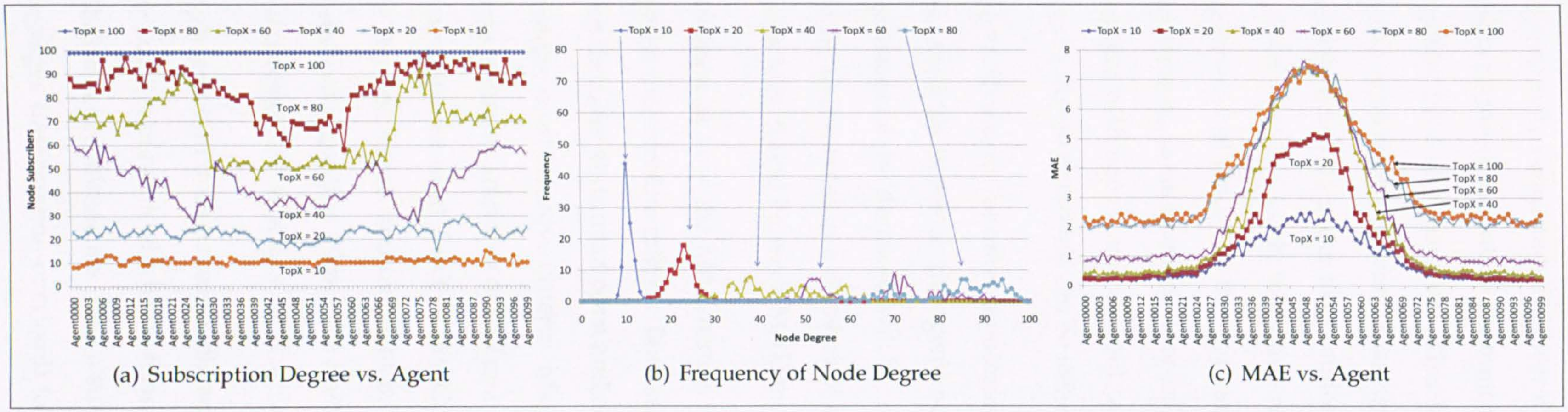


(a) Subscription Degree vs. Agent

(b) Frequency of Node Degree

(c) MAE vs. Agent

Figure 4.14: Using a Threshold algorithm and a Uniform Heterogeneity distribution

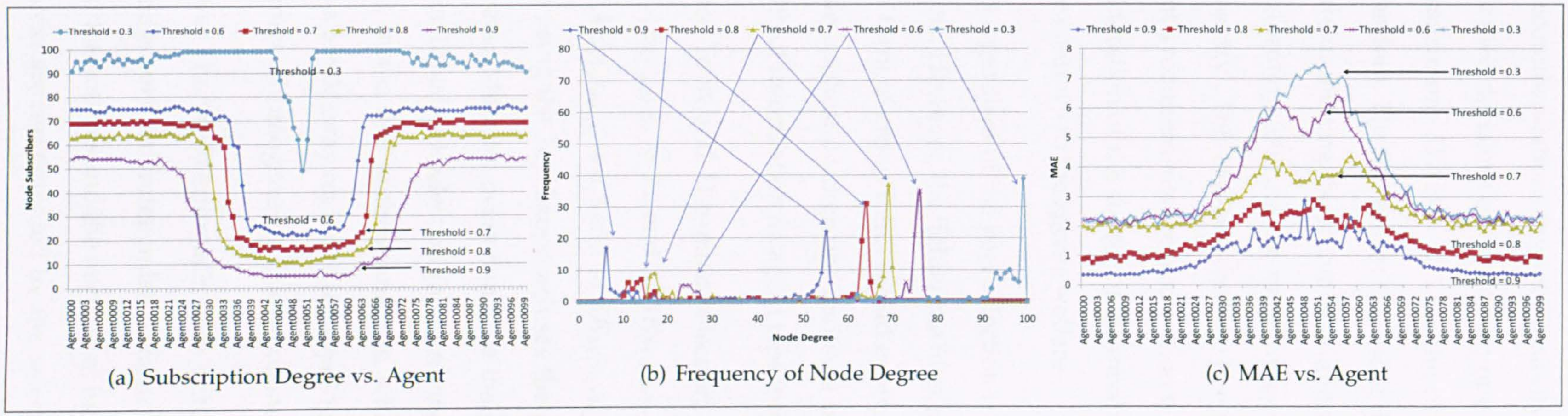


(a) Subscription Degree vs. Agent

(b) Frequency of Node Degree

(c) MAE vs. Agent

Figure 4.15: Using a TopX algorithm and a Majority/Minority Heterogeneity distribution



(a) Subscription Degree vs. Agent

(b) Frequency of Node Degree

(c) MAE vs. Agent

Figure 4.16: Using a Threshold algorithm and a Majority/Minority Heterogeneity distribution

This is not the case for our Majority/Minority distribution experiments Figures 4.15c and 4.16c. Here the unequal distribution of peer preferences, means that as the peer selection tends to fully connected, the sheer number of recommendations from the majority belief agents causes the agents in the minority to make their decisions based on the populations majority held requirements. This leads to a maximum $MAE \approx 7.5$ for the minority agents, a performance worse than selecting at random. The flip side of this being that the majority peers do better than random even without filtering the recommendations with a $MAE \approx 2.3$. Clearly without a network performing filtering the minority peers are negatively influenced by the 'Tyranny of the Majority'. This also sets the base line for our experiments, demonstrating that unless an agent's requirements are in common with the majority held needs, and without careful consideration of the source of information, an agent's decision making can be influenced to be contrary to the agents welfare.

The MAE results for other topologies show that regardless of the peer selection rule, the finer the filter the better the agents make decisions. However, the tighter the filter the less experimental the decisions will be, in essence a form of group think would evolve. This can be detrimental in a number of our example application domains and so a balance needs to be found between MAE and the breadth of information an agent taps into.

Lets look at how the MAE results transition for TopX and Threshold selection, as we tighten the selection criteria. Both start with $MAE_{Majority} \approx 2.3$ and $MAE_{Minority} \approx 7.5$ with the filter at its most open and end with $MAE_{Majority} \approx 0.3$ and $MAE_{Minority} \approx 2.0$ with the most selective criteria. Majority peers using the TopX approach see their MAE score suddenly improve once $TopX = 60$ however Minority peers don't see their MAE score improve until $TopX = 20$. Conversely, in Threshold selection the minority peers are the first to see improvement in their MAE, this is due to the manner in which the Threshold algorithm allows the agents to build a break away region of the topology with low connectivity whereas TopX attempts to enforce a homogeneous level of connection.

TopX selection does have one big advantage over Threshold approach, with Thresholds the number of peers which will form connections is not fixed independently from the distribution of requirements. As a result the network resources and the number of incoming messages (information overload) for each process are not bounded by the selection algorithm, but are instead dependant upon the number of peers with similarity measures above the threshold. Under certain circumstances a small change in Threshold selected could mean a huge change in network utilisation. This is highlighted by comparing the number of connections the majority peers make to achieve the same MAE score ≈ 0.3; using $TopX = 10$ results in 10 connections whereas with $Threshold = 0.9$ the cluster in the Threshold topology creates around 50 per agent.

In summary Top X doesn't transition well if the agents' requirements are not widely held, in this case the agent would benefit from choosing peers based directly on similarity as opposed to aiming for a set number of relationships. However the Threshold approach doesn't scale (from a network load perspective) if you hold majority requirements.

4.8 Chapter Summary

This chapter has presented our first case study, demonstrating PreSage's ability to model large scale distributed agent systems, which feature complex adaptive network topologies with software agents encapsulated into the network nodes. We have designed and built socio-cognitive agents who adapt their communicative interactions based on social beliefs of related similarity, and shown how this in turn creates the system-wide social network structure. This structure having a direct influence on the welfare of the agents and forming a feed-back loop resulting in self-organisation. We also demonstrate PreSage's ability to animate numerous populations of agents with differing initial configurations, allowing the user to explore the micro $\rightarrow$ macro effects of subtle changes to agent reasoning. Figures 4.17 and 4.18 show how we have extended the provided classes of presage to create our simulation agents, the network and data logging plugins, and how the XML configuation and subsequent running of the simulation has created the results in this chapter.

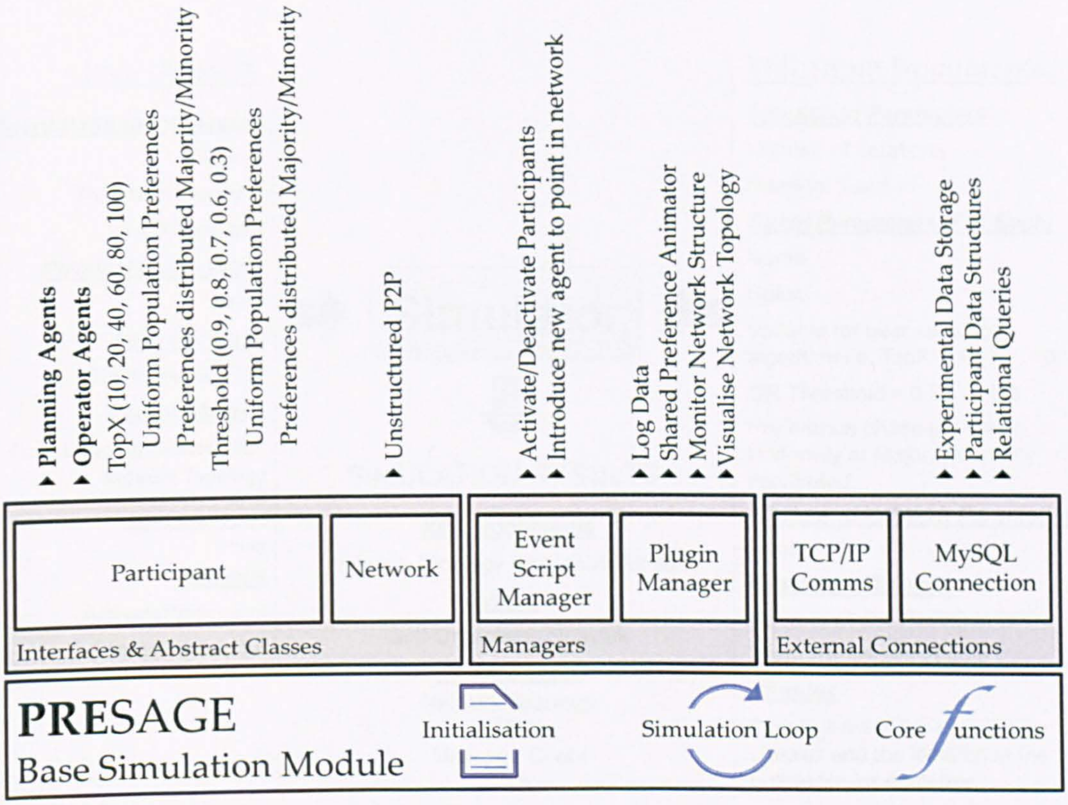

▸ Planning Agents
▸ Operator Agents
TopX (10, 20, 40, 60, 80, 100)
Uniform Population Preferences
Preferences distributed Majority/Minority
Threshold (0.9, 0.8, 0.7, 0.6, 0.3)
Uniform Population Preferences
Preferences distributed Majority/Minority
▸ Unstructured P2P
▸ Activate/Deactivate Participants
▸ Introduce new agent to point in network
▸ Log Data
▸ Shared Preference Animator
▸ Monitor Network Structure
▸Visualise Network Topology
▸ Experimental Data Storage
▸ Participant Data Structures
▸ Relational Queries
Participant
Network
Interfaces & Abstract Classes
Event Script Manager
Plugin Manager
Managers
TCP/IP Comms
MySQL Connection
External Connections
PRESAGE
Base Simulation Module
Initialisation
Simulation Loop
Core functions

Figure 4.17: 'Toast Rack' instantiation of the platform architecture, for Simulating Social Recommendation Agents

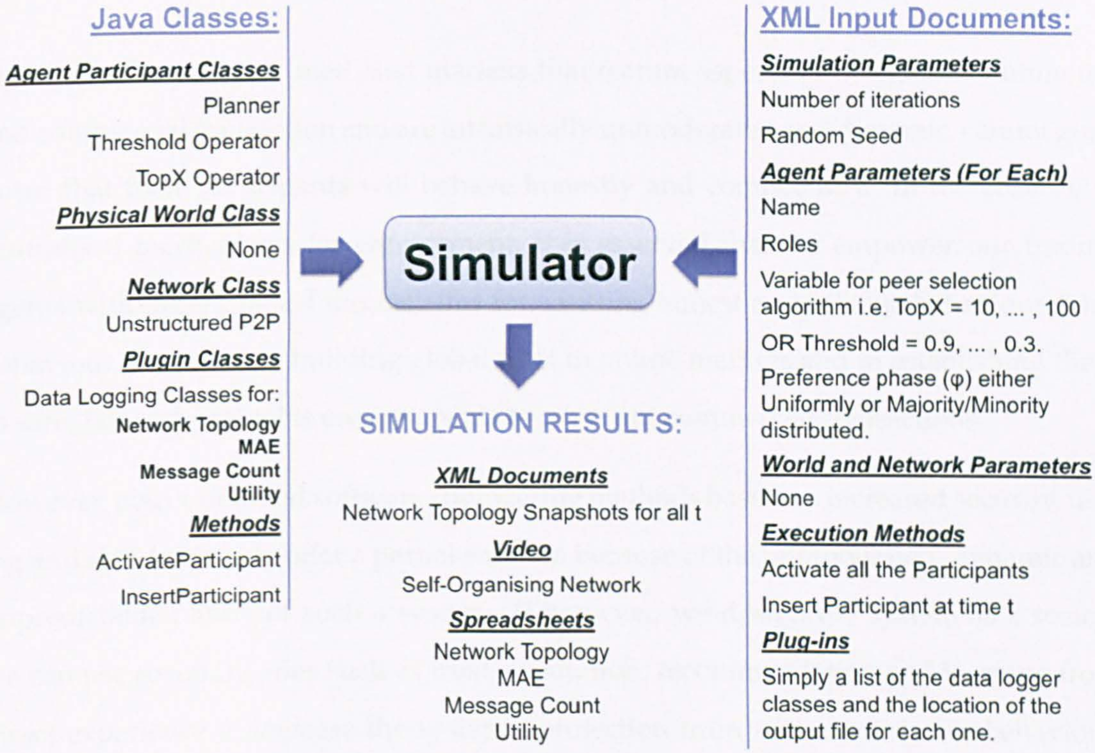

Java Classes:
Agent Participant Classes
Planner
Threshold Operator
TopX Operator
Physical World Class
None
Network Class
Unstructured P2P
Plugin Classes
Data Logging Classes for:
Network Topology
MAE
Message Count
Utility
Methods
ActivateParticipant
InsertParticipant
Simulator
SIMULATION RESULTS:
XML Documents
Network Topology Snapshots for all t
Video
Self-Organising Network
Spreadsheets
Network Topology
MAE
Message Count
Utility
XML Input Documents:
Simulation Parameters
Number of iterations
Random Seed
Agent Parameters (For Each)
Name
Roles
Variable for peer selection algorithm i.e. TopX = 10, ... , 100
OR Threshold = 0.9, ... , 0.3.
Preference phase (φ) either Uniformly or Majority/Minority distributed.
World and Network Parameters
None
Execution Methods
Activate all the Participants
Insert Participant at time t
Plug-ins
Simply a list of the data logger classes and the location of the output file for each one.

Figure 4.18: Input/Output chart for Simulating Social Recommendation Agents

THE COMPUTATIONAL SOCIO-COGNITIVE AND ECONOMIC FRAMEWORK (CS_CEF) 5

5.1 Introduction

Open distributed agent mediated markets that feature aspects of delegation, autonomy and commercial transaction and are intrinsically unmoderated and dynamic, cannot guarantee that their participants will behave honestly and competently. In the absence of centralised mechanisms for enforcement, it is essential that we empower our trading agents with decentralised mechanisms for assuring honest and reliable behaviour. This behaviour is integral to building global trust in online markets and in establishing them as safe, fair and profitable environments to carry out commercial transactions.

However, object-oriented software engineering methods based on increased security, testing and standards only offer a partial solution because of the unmoderated, dynamic and unpredictable nature of such a system. If however, we design the system as a society we can use social theories such as trust, reputation, recommendation and learning from direct experience to increase the system's protection from such undesirable behaviour. In support of this, Conte [34] argues that reputation plays a crucial role in decentralised mechanisms for the enforcement of social order.

In this chapter we advance this argument by developing a Computational Socio-cognitive and Economic Framework (CS_CEF) where the actions of agents in market-level interactions are influenced by their relationships on a social level. Therefore the agents' social interaction acts as a means to provide accountability to market level actions and thus discourages malicious behaviour and isolates incompetent agents. We argue the framework would also increase consumer information regarding potential sellers and therefore the efficiency of the market.

The key features of the CS_CEF are based on the inter-disciplinary study of two social sciences, namely sociology and economics. The framework formalises social theories of trust, reputation, recommendation and learning from direct experience and integrates these socio-cognitive formalisms with the agent's economic reasoning. This produces an agent whose behaviour in commercial transactions is influenced by its social interactions, whilst being motivated and constrained by its economic considerations.

The distinctive features of our computational framework are:

- Both economic and social factors are utilised in the agents decision to trust.
- The framework represents recommendation as a generic task, as a result evaluating trust in recommenders and recommending recommenders requires no special formalisms or protocols.
- Our functions for determining the certainty measure associated with a belief are based on the age, source and quantity of information used to form the belief.
- The formation of experiences though the agent's actions in the commercial arena, provides positive feedback to the socio-cognitive elements of the framework.
- The framework's numerical formalisms are amenable to immediate computational implementation.
- Agents transition from blind trust through risk based trust to reliance trust based on increasing confidence in a trust relationship.

Therefore our framework provides a comprehensive solution to issues ranging from the evolution of trust beliefs from individual experiences and recommendations to the use of those beliefs in market place level decisions. Which is compatible with our aim of creating an artificial system to test which formalisms and parameters will provide desirable system performance in diverse real world applications.

5.2 Socio-cognitive Modelling

Our investigation of socially motivated behaviour is based on formal models of anthropomorphic socio-cognitive relations. Accordingly, our socio-cognitive model defines three component beliefs of each agent about its peers in the society: trust, direct experience and reputation. This section defines the socio-cognitive model and the software developed to facilitate experimentation with agent societies whose members are implemented with a (parametrised) version of the model.

Our computational representation of trust is based on the formal model of Castelfranchi and Falcone [29, 49]. The essential conceptualisation is as follows: the degree to which Agent A trusts Agent B about task τ in (state of the world) is a subjective probability; this is the basis of agent A's decision to rely upon B for τ. Our method incorporates this stance, defining trust as the resultant belief of one agent about another, borne out of direct experience of that other party and/or from the testimonies of peers (i.e. reputation). We define direct experience as the belief one agent has about the trustworthiness of another, based on first-hand interactions. Having 'trusted' another agent to perform task τ and assessed the outcome, an agent will update its direct experience beliefs concerning the delegated agent accordingly. The outcome of delegating a task may be either successful or unsuccessful: this categorisation is the basis of an experience update rule [123] that calculates the revised level of trustworthiness to associate with the agent in question. Our motivation in investigating reputation mechanisms by simulation is provided by Conte [32], who outlines the need for 'decentralised mechanisms of enforcement of social order' noting that 'reputation plays a crucial role in distributed social control'. In an agent society, reputation is the collectively informed opinion held by a group of agents about the performance of a peer agent within a specific social context. By consulting its peers, an agent can discover the individual reputation of an agent. However, the received testimonies may be affected by existing relationships and attitudes (for example, agent C may be willing to divulge reputation information to agent A but not to agent B). Thus, reputation is also a subjective concept which we define as a belief held/derived by one agent. The Subjective Reputation Evaluation Function (SREF) formulates subjectively that is, from the perspective of an agent A the reputation of an agent B, based on information from the peers of A and B. SREF may take several forms; examples are a weighted sum or a fuzzy relation. In our current work, we use the former approach; a summing function in which n assertions received from peers regarding agent B are weighted by the agent's trust (confidence) in each peer to make accurate recommendations. This approach incorporates the credibility of each assertion's source (the credibility of a belief being dependent upon the credibility of its sources, evidences and supports [29]).

For now, we briefly address a proposed method of combining direct experience with reputation to formulate the mental state of trust. Each belief has associated with it a degree of confidence signifying an agent's trust in the belief's accuracy. This confidence measure is essential to what trust will be based on, be it one or indeed both of the beliefs, experience or reputation. An agent with strong confidence in its experience beliefs and little confidence in the accuracy of its reputation beliefs should rationally choose to calculate its degree of trust (DoT) primarily from its experiences. An agent recently introduced to the system should have little confidence in its own (in)experience and should thus base its trust solely on reputation. The number of direct experiences is therefore significant in evaluating confidence in a belief about direct experience. It is similarly the case for reputation, where the number of agreeing testimonies is a decisive factor. Experience and reputation thus become influences of trust, the weighting assigned to them dependent upon an agent's confidence in their respective accuracies.

Open distributed agent mediated markets that feature aspects of autonomy and commercial transaction and are intrinsically unmoderated and dynamic, cannot guarantee that their participants will behave honestly, ethically and competently. Therefore it is essential that we empower our trading agents with decentralised mechanisms for enforcing honest and reliable behaviour. This enforcement s integral to building global trust in online markets and establish them as safe, fair and profitable environments to carry out commercial transactions. To achieve this we propose a Computational Socio-cognitive and Economic Framework (CS_CEF) as a mechanism for regulation. Moreover this framework functions without the need for centralised law makers and enforcers therefore making it appropriate for P2P based e-commerce.

The key features of the CS_CEF are based on the inter-disciplinary study of two social sciences, namely sociology and economics. The framework formalises social theories of trust, reputation, recommendation and learning from direct experience and integrates these socio-cognitive formalisms with the agent's economic reasoning. This produces an agent whose behaviour in commercial transactions is influenced by its social interactions, whilst being motivated and constrained by its economic considerations. The framework thus provides a comprehensive solution to a number of issues ranging from the evolution of a trust belief from individual experiences and recommendations to the use of those beliefs in market place level decisions.

If the agent framework is going to be adopted it is necessary that it not only protects against malice and incompetence, but also allows the agent's performance to compliment the motivation of its owner. Hence a producer agent must be capable of the the long-term maximisation of its owner's profit and consumer agents must maximise their owner's utility given their budgetary restrictions. Essentially this means that when improving the agents defenses we must not make them economically sub-optimal, as this would inhibit the market's ability to allocate resources efficiently .

In order to verify that the application of our framework in open distributed agent mediated markets results in an improvement in system performance, we use multi-agent based simulation (MABS) [35] to investigate how CS_CEF agents perform in a simulated environment. To achieve this we have developed a simulation platform that provides tools to facilitate socio-cognitive and economic specification of agents, control of market economic factors, data logging, results analysis and visualisation.

In this chapter, we present a series of experiments which follow our methodology, the adapted synthetic method Fig. 2.2. Specifically the formalisation of social theories, the engineering of a simulation and the revision of the system based on the results of animating the agent society. In a sense, we show how we use PreSage to follow a form of iterative social design.

We present a specification for a artificial retail market scenario. Then we detail our computational formalisms of social theories, namely the economic model for producer/seller agents, the socio-cognitive framework and the economic rationale of the consumer agents. We go on to present the results of simulating the system with an increasingly refined socio-cognitive model, in order to demonstrate the value added by each feature of the model. Ultimately we show that social order in competitive market systems can be created and supported by the integration of social behaviour into the trading agent architecture.

5.3 The Social Market Scenario

To address issues pertaining to agent e-commerce using simulation methods, we first need to specify a suitable market based scenario. There are many possible types of market that could be used, however we have chosen to focus on software agent mediated e-commerce within a manufacturing retail market place, as in [124]. In these markets agents buy and sell information goods or services such as multi-media products, content hosting or information retrieval.

The market model comprises two groups of agents, one group represents the producers of a service or product the other its consumers. Our trading agents execute a variant of the contract-net protocol (CNP — [113]). Whereby producers sell information to consumers. In summary, consumers having formed a desire for the goods or services will make a call for proposals(bids) to those producers it is aware of. The producers then submit their bids and await either acceptance or rejection. Having selected the product or service they require, the consumers communicate their order to the producer and on receipt of payment the producer should supply the product to the consumer.

Effectively the role of the producer agents in our proposed simulation environment will be to the test the ability of the consumer agents' socio-cognitive framework to protect against malicious or incompetent behaviour. Hence the producer agents will be implemented with both an economic model and a character type, some of these characters will aim to defraud the consumer agents. Given that the market mechanism proposed, inherently protects the producer agents from risk they are not simulated as socio-cognitive. We intend to address this simplifying assumption in future work, as the producers could benefit greatly from knowing their own reputation and those of their competitors. The consumer agent model presented is both socio-cognitive and economic, socio-cognitive in the sense that it forms a social network of peers with which it communicates its opinions as well as receiving and reasoning about the opinions of others. Its behaviour is economic in that consumer agents aim to maximise their owner's utility.

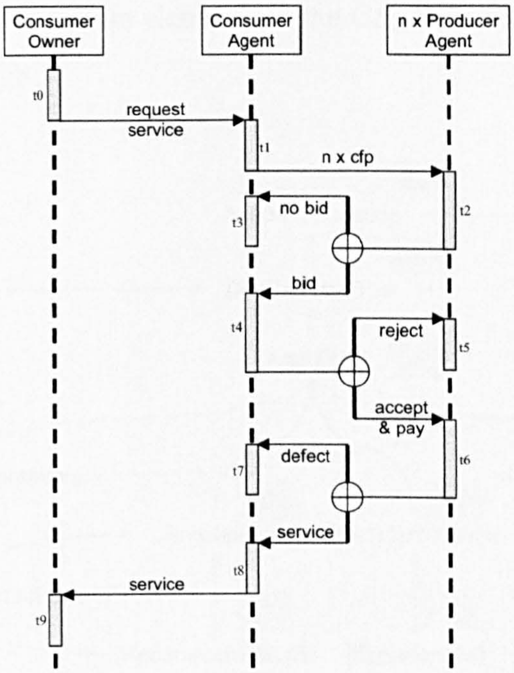


Figure 5.1: Contract Net Protocol for the Social Market Scenario in AUML

The agents of the social market scenario communicate using an interaction protocol (5.1). For clarity reasons, we have not included timestamps on this diagram and we have merged the roles of the many producers involved into one participant.

This scenario bears some resemblance to a one-sided iterated prisoners dilemma and therefore could have been tackled with a game theoretic approach. However due to the increased complexity as a result of numerous participants, heterogeneous participants and the addition of learning in terms of both the consumers and producers, a bottom up agent based simulation represents a more tractable approach.

5.4 The Consumer Agent

The consumer model combines economic and socio-cognitive decision making in order to create an agent whose behaviour on a market level is both enabled and directed by its social relationships, attitudes and interactions, whilst remaining motivated and constrained by its economic considerations. In this section we present a walk though demonstrating how these social and economic components are integrated in response to a request from the consumer agents owner for a service, this is summarised by Fig. 5.2 which portrays the connections between the main elements of the CS_CEF and is numbered in correlation with the walkthrough.

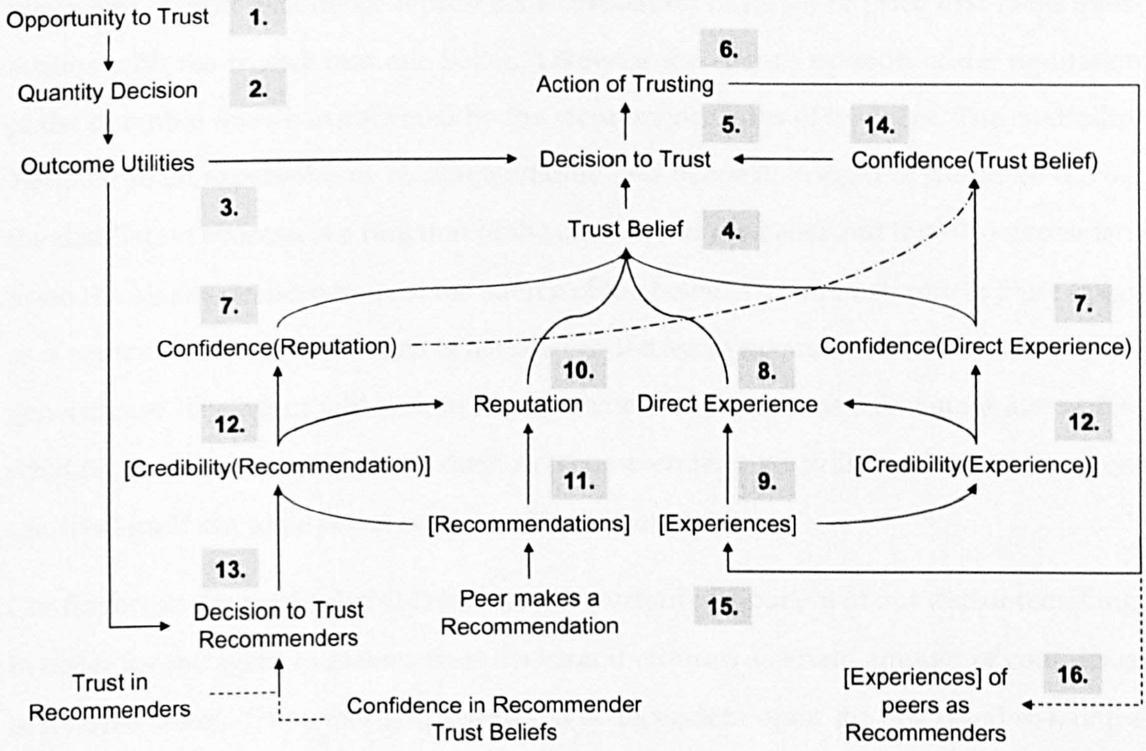


Figure 5.2: Economic and Socio-cognitive elements of the CS_CEF

In our agent-mediated market place a (simulated) human consumer makes a request to its agent to obtain some goods or services. In response the consumer agent communicates with any producers it is aware of, in order to find the current prices charged by them. Upon receiving bids from the producer agents, the agent is faced with an opportunity to trust (1) one or none of them to supply the goods or services. Taking each bid in turn, the consumer agent decides the quantity it wishes to order from the producer based upon the price, its economic model and budget constraints (2). Given this quantity the agent will then estimate the utility that would be gained by its owner consuming those goods or services, or lost in the event that the producer fails to supply those products to the consumer. These outcome utilities (3) are the economic influence to the agents decision to trust, they represent the possible pay-off from relying on the producer.

The other variable in the decision to trust is the agent's trust belief, this being the agent's subjective evaluation of the probability of a successful outcome of trusting the peer. The agent's trust belief, is computed from the combination of the agent's belief about its direct experiences and the reputation of the potential trustee. The relative influence of these beliefs on the trust belief is determined by the agent's confidence in their respective accuracies. Direct experience represents a distillation of its set of prior first hand interactions with the trustee into one belief. Likewise the agent's opinion of the reputation of the potential trustee is informed by the recommendations of its peers. The credibility assigned to an experience or recommendation and hence its weight of influence during the distillation process, is a function of the currency of the belief and it is also dependent upon the agents decision to trust the source of the belief. This opportunity to trust a peer as a source of recommendations is handled in the same manner as defined here for the generic case. The agent will look to its experiences of the peer as a recommender and at what its peers recommend about them as recommenders. We will assume that the agent can trust itself not to lie about or distort it's own experiences.

Confidence in the trust belief (14) forms an important component of our decision making. In order for the agent to make a trust decision it requires a certain amount of confidence in its trust belief. This confidence threshold is dependent upon the perceived risk of the interaction. If the confidence in the trust belief is less than that required the agent may reduce the order quantity thus reducing the perceived risk, the agent then recalculates the trust belief given these new economic variables.

Else, if the agent is sufficiently confident it can use its trust belief and the outcome utilities to calculate the expected utility of that purchase. The agent compares this to the expected utilities it calculated for the other competing producers. The consumer decides to trust the producer with the highest positive expected utility, if none of the opportunities have a positive utility then the consumer takes no action.

In the cases where the consumer agent decides to trust one of the potential trustees, the will take the action of trusting. The resultant experience of the trustee is added to the agents set of prior experiences (15). Experiences are also formed about those agents that have made recommendations referring to the trustee and subsequently the agents that recommended them and so on (16). When the agent forms each experience belief, it can then decide if it would like to make a recommendation of the target agent. Our agents decide to make a recommendation when they are sufficiently confident in their own direct experience belief. The confidence required to make a recommendation is dependent upon the nature of that recommendation, for instance an agent can make a negative recommendation with less confidence than for a positive one.

5.5 Economic Rationale

5.5.1 Consumer Economic Rationale

Our economic model of the consumer agent focuses on estimating the utility gained by the consumer from consuming the goods and services it purchased. In addition the consumer agent estimates the utility lost in the event that a producer fails to supply those products to the consumer. These utility measures are employed in the integration of socio-cognitive and economic influences as part of the consumer agent's decision to trust a producer.

Each of our consumer agents have a designated budget to spend on the generic resource in each time period. The agent's budget is one of our simulation parameters. By changing the budgets of the consumer agents we control the monetary value of the markets demand for the resource. Our producer agents set the per unit price of their goods and services (as addressed in section 5.5.2).

The key factors which need to be taken into account when calculating the value of a purchase to the consumer are:

- A unit of resource is of highest value when the amount of resource consumed is equal to or near zero.
- The more of a resource that is consumed the less an additional unit is valued (diminishing marginal utility).

- More of a resource is always better than less.

The value assigned to an extra unit of resource given the number of units currently consumed (in this time period) is given by the derivative (5.1). This addresses the key factors outlined above, the constants in the derivative are used to tailor the valuations to a specific consumer's profile. The constant ξ is the value of an extra unit resource when the quantity already consumed is zero and γ determines the rate at which the value of an extra unit decreases as the amount consumed increases. Fig 5.3 shows formula (5.1) for $\xi = 500$ with diminishing utility at a rate of $\gamma = 0.1, 0.2$ and 0.3. Given the price per unit resource, the consumer can maximise its utility by consuming S1 units where $S = S1$ solves the marginal utility function (5.1) equal to the price per unit. If it consumes more than $S1$ the utility gained by consuming the additional units will be less than what it has paid for them where as consuming less than $S1$ units would be sub-optimal. Sometimes the consumer will be unable to maximise its utility as it cannot afford $S1$ units ($S1 > Budget/Price$), in which case it will do best to purchase as much as it can afford ($S1 = Budget/Price$). To calculate the utility of a successful purchase we integrate (5.1) from zero to the number of units in the proposed purchase (consumption $S1$), we then subtract the cost of the purchase giving us (5.2). In the case of an unsuccessful outcome the achieved consumption $S1$ is zero and so (5.2) reduces to (5.3), the utility lost is the money paid. The significance of these results and especially how they inform the socio cognitive model are explained in the following section.

$$\frac{\mathbf{dU(Success)}}{\mathbf{dS}} = \frac{\xi}{e^{\gamma S}} \tag{5.1}$$

$$\begin{aligned} \mathbf{U(Success)} &= \int_0^{S1} \frac{\xi}{e^{\gamma S}} dS - (Price \times S1) \\ &= \frac{\xi}{-\gamma}\left(e^{-\gamma S1} - 1\right) - (Price \times S1) \end{aligned} \tag{5.2}$$

$$\mathbf{U(Failure)} = -\,(Price \times S1) \tag{5.3}$$

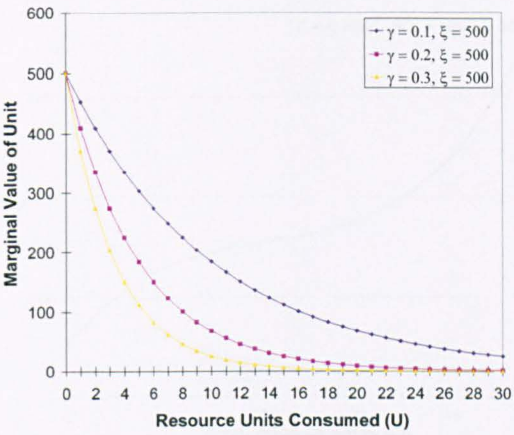


Figure 5.3: $\frac{\mathrm{dU}}{\mathrm{dS}} = \frac{\xi}{e^{\gamma S}}$ for $\xi = 500$, and $\gamma = 0.1, 0.2$ and 0.3.

5.5.2 Producer Economic Rationale

In this section we define both our economic model of a producer and the determinants of the producer agent's behaviour. A producer agent's goal is the maximisation of their owner's profit. We define profit as the difference between the business agents' revenue and the total cost of producing its product, $Profit = TotalRevenue - TotalCost$. It is thus necessary that the agents have a model of their total costs.

The total cost (TC) of producing a good is the sum of the total variable cost (TVC) and total fixed cost (TFC) of production. Dynamic behaviour of the total variable cost is the result of increasing returns to scale as the quantity produced increases, followed by diminishing returns to scale. The agents' cost function is best represented as a cubic polynomial, the coefficients of which are experimental parameters. A possible producer cost function for use in the simulation is shown in Fig. 5.4.

Total revenue (TR) is the product of the quantity of goods demanded and at what price, this said the quantity of good demanded is itself dependent on the price of the product. In our simulation environment the producer agents are responsible for setting the price at which they sell their wares for that time period. The consumers then decide whether or not to purchase the product at that price and how much to order. Producers are expected to supply the quantity demanded by the consumer. Kephart et al use this *dynamic posted pricing* mechanism in [83]. For the producer to maximise its profit they must optimally set the price of their goods and services. This method represents only one possible pric-

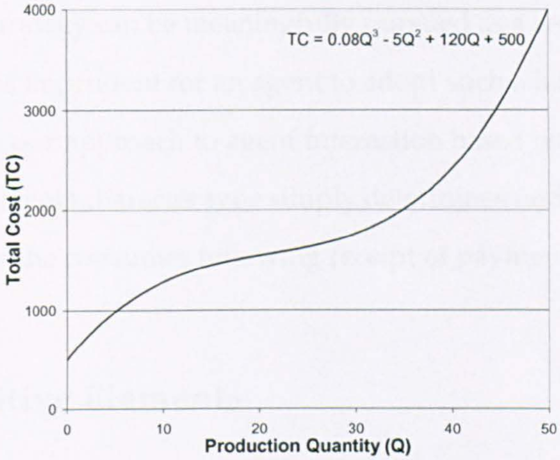


Figure 5.4: Example producer cost curve

ing mechanism, examples of others would include the many different types of auction. Auctions require the opposite set of decisions, as the producer decides the quantity of product and the consumers set the price they are willing to pay.

To set their price the agents employ the derivative-follower algorithm from Greenwald and Kephart [65]. In each time period the derivative follower increments its price. It does this until its profit in that time period drops below the profit in the previous round it then reverses the direction of its price increments. The effect of this is to ascend the gradient to a local maxima in the profit. In addition the agent needs to decide upon an initial price at which to start its search for the maximum. In the case where there is already a market for competing products the agent aims to undercut the competition in its first time period. However when there are no competing producers we assume that the owner of the producer agent is capable of providing a suitable first price.

So far we have described the characteristics of a totally reliable, honest and cooperative producer agent acting within a error free environment. What is missing is the malice or incompetence of some agents and the unreliability inherent in multi-agent system (MAS) environments and real-world applications. To model this environment, each producer has an associated competence level and character type. The competence variable is defined as the probability the producer agent will succeed in its task given that it attempts to do so. If the agent tries to supply the consumer but fails due to incompetence then it still incurs the cost of that action. This fallibility factor is absent from the simulations of

[8], where a *tit for tat* strategy can be meaningfully pursued as a result. In the presence of fallibility, however, it is imprudent for an agent to adopt such a harsh line of judgement. This further motivates our approach to agent interaction based on social models including trust aspects. The agents character type simply determines under what circumstances they attempt to supply the consumer following receipt of payment.

5.6 Socio-cognitive Elements

5.6.1 Consumer Socio-cognitive Model

Our socio-cognitive modelling is based on social theories of trust, reputation, recommendation and learning from direct experience. There are many approaches and conceptualisations of trust proposed in current computing and social sciences research. Our computational representation of trust is based on the semi-formal models appearing in the work of Castelfranchi and Falcone [28–30, 49]. We have chosen this trust framework due to its strong connections with other research in the ALFEBIITE project and because of its clear potential for implementation. They define trust not only as a truster's evaluation of a trustee (its trust belief), but also as the decision to and the action of trusting. This section outlines our conceptualisation of an agent's trust belief and a mechanism by which this belief and the agent's utility evaluations influence its decision to trust. We go on to introduce our method for subjectively evaluating the trustworthiness of a peer from its direct experiences and peer testimony.

5.6.2 Formal Model of Trust and Trustworthiness

The Trust Belief

Our computational representation of an agent's trust belief is based on the formal model of Castelfranchi and Falcone [29, 49]. The essential conceptualisation is as follows: the degree to which Agent A trusts Agent B about task τ in (state of the world) Ω is a subjective probability $\mathbf{DoT}_{A,B,\tau,\Omega}$. This is the basis of agent A's decision to rely upon B for

τ. Our method incorporates this stance, and defines trust as the resultant belief of one agent about another, born out of direct experiences of that other party and/or from the testimonies of peers (i.e. reputation). Although the formal model outlined above identifies the trustworthiness of an agent and an agent's beliefs as being dependent upon the state of the world, we omit the Ω parameter in subsequent descriptions for reasons of simplicity. It should also be noted that the socio-cognitive framework is generic and can be widely applied to many interpretations of the task τ. In the case of our e-commerce scenario this task is considered to be the provision of a service or information good. In the specific case of peer recommendations $\tau = srec$ indicates that the task τ is for the peer to act as a reliable "source of recommendations". For purposes of readability throughout the rest of the paper we assign the following agent identities and roles. Agent A is our consumer agent, agent B is a peer consumer and agent C is a producer agent.

The Decision to Trust

In defining the mental state of trust Gambetta [59] refers to the decision to trust, as an evaluation 'that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him.'. He goes on to note that this assessment is based on both the degree of trust and the perceived risk. Indicating that as the risk associated with an interaction increases, the degree of trust needed to trust (decide to rely upon) also increases. We have implemented a decision to trust function (5.4) which is guided by this theory. The function takes the form of predicting the expected utility of the action of trusting. With bipolar outcomes of trustee success or failure, the calculation is straight forward. The agent estimates the utility of the successful scenario ($\mathbf{U(Success)}_{A,C,\tau}$) where its trading peer Agent C cooperates and succeeds at task τ and conversely its losses ($\mathbf{U(Failure)}_{A,C,\tau}$) in the event its trading partner fails. The trust belief ($\mathbf{DoT}_{A,C,\tau}$) of the agent is the probability of the successful scenario occurring and its distrust ($1 - \mathbf{DoT}_{A,C,\tau}$) the probability of failure. Knowing the pay-off of each outcome in advance and having an estimate of their probabilities of occurrence, the agent can calculate the expected utility of trusting its peer. If the expected utility is positive then the agent estimates that it can benefit from trusting its peer and so should make the decision to trust. The expected outcome method captures the intuition that as the cost of failure increases the degree of trust needed to decide to trust increases and vice versa. Markets may provide a choice of a number of trading partners, in this case the agent takes the action of trusting the partner with the highest positive expected utility.

$$\begin{aligned}&\textbf{Expected Utility} = \\ &\quad \textbf{DoT}_{A,C,\tau} \times \textbf{U}\,(\textbf{Success})_{A,C,\tau} + (1 - \textbf{DoT}_{A,C,\tau}) \times \textbf{U}\,(\textbf{Failure})_{A,C,\tau}\end{aligned} \tag{5.4}$$

5.6.3 Experiences and Recommendations

The agent having delegated a task τ to Agent C, evaluates the outcome of trusting Agent C about τ at time t. This outcome evaluation ($\textbf{Experience}_{C,\tau,t}$) is heavily application dependent, for instance when forming an experience of a peer as a recommender the evaluation takes the form of a continuous variable ($\textbf{Experience}_{C,\tau,t}$ valued between -1 and 1) which represents a degree of accuracy or similarity measure. In other cases such as contractual scenarios the outcome can be characterised by a discrete bipolar evaluation i.e. Success or Failure to meet the agreed to contractual obligations ($\textbf{Experience}_{C,\tau,t}$ = -1 or 1).

Recommendations are the testimonies by which the agents share their experiences with their peers. They are integral in informing the reputation of agents and therefore in applying pressure for agents to act honestly and ethically. A recommendation by agent B regarding its experience of an agent C about a task τ which is received at time t is represented by $\textbf{Recommendation}_{B,C,\tau,t}$. Only the most current recommendation from each peer is maintained by the agent, e.g.:

$$\textbf{Recommendation}_{B,C,\tau,t_2} \; \textit{replaces} \; \textbf{Recommendation}_{B,C,\tau,t_1} \; \textit{for} \; t_2 > t_1. \tag{5.5}$$

Recommendations take the form of a continuous variable in the range [0-1]. It should also be noted that chained recommendations are not implicitly catered for by this representation. Chained recommendations occur where agent B informs agent A that agent D

recommends agent C to do a task τ. Instead the third party (agent B) can first introduce the recommendation's source (agent D) and secondly agent B can recommend agent D as a source of recommendations. Now agent A can query agent D regarding agent C and using agent B's recommendation about agent D decide whether to trust agent D's recommendation.

5.6.4 Credibility of a Belief

The credibility attached to a belief is defined as its quality and power to elicit belief. We view this credibility as being a function of the agents trust for the beliefs source and how long ago the assertion was made (its currency). It must be proportional to the agents trust in the source of the belief (which may be itself or a peer) and inversely proportional to the age of the belief $\mathbf{\Delta t}$. Functions (5.10) and (5.11) are examples of suitable functions for deriving the credibility of $\mathbf{Recommendation}_{B,C,\tau,t}$ and $\mathbf{Experience}_{C,\tau,t}$ respectively.

The first term of each function determines the rate by which a belief is discredited with age, $\mathrm{e}^{-\alpha\mathbf{\Delta t}} = 1$, for $\mathbf{\Delta t} = 0$ i.e. the belief is current and its credibility is judged purely by the agents trust in its source. As $\mathbf{\Delta t} \Rightarrow \infty$ the term is asymptotic to the x-axis. The constant α governs the rate of decay of credibility with age. The trust belief, the second term in functions (5.10) and (5.11) is the degree of trust Agent A has in Agent B as a "source of recommendations". For function (5.11) we assume that the agent implicitly trusts itself as a source of outcome evaluations (source of experiences abbreviated to sexp) and so $\mathbf{DoT}_{A,A,sexp} = 1$ [1]. Fig. 5.5 plots the credibility assignment of a belief against the age of that belief, for different values of α and assuming $\mathbf{DoT}_{A,B,sbeliefs} = 1$.

Earlier we argued that the agent must decide to trust a peer given both its trust belief in that peer and the perceived risk of trusting them. This is also the case when deciding to trust a peer as a source of beliefs. We argue that the risk of trusting a peer's recommendation is a function of the risk of trusting the recommendations target about the task τ to which the recommendation refers. Rearranging function (5.4) gives us a formula for $\mathbf{ReqDoT}_{A,C,\tau}$ (function (5.6)) this represents the minimum required trust belief needed

[1] This is not to say that for the general case $\mathbf{DoT}_{A,A,\tau} = 1$ or that $\mathbf{DoT}_{A,A,\tau} \geq \mathbf{DoT}_{A,B,\tau}$ in fact a number of hypothetical scenarios can be envisioned where $\mathbf{DoT}_{A,A,\tau} < \mathbf{DoT}_{A,B,\tau} \leq 1$

for Agent A to decide to trust Agent C about τ given the outcome evaluations. If Agent B's $\mathbf{Recommendation}_{B,C,\tau,t}$ is greater than or equal to $\mathbf{ReqDoT}_{A,C,\tau}$ then Agent B is effectively recommending that Agent C be trusted about τ. In this case the outcome of trusting Agent B's recommendation is to trust Agent C about τ, therefore the outcome evaluations (and hence the required degree of trust) of trusting Agent B as a source of recommendations is equal to those for trusting Agent C about τ. Conversely, if Agent B's $\mathbf{Recommendation}_{B,C,\tau,t}$ is less than $\mathbf{ReqDoT}_{A,C,\tau}$ then its recommendation is to not trust Agent C about τ. In this case the outcome of trusting Agent B's recommendation is to decide not to trust Agent C about τ, if trusting Agent B's recommendation is a success then Agent A has saved the cost $\mathbf{U}(\mathbf{Failure})_{A,C,\tau}$ by Agent C, and if Agent B was wrong then Agent A has lost the successful outcome $\mathbf{U}(\mathbf{Success})_{A,C,\tau}$ of trusting Agent C about τ. These two cases are summarised by the equations (5.7) and (5.8) which provide outcome utilities for trusting Agent B as a recommender these are used by equation (5.9) to calculate the minimum required trust belief needed to decide to trust Agent B's recommendation about Agent C. There exists a set of equations of the same structure as (5.7), (5.8) and (5.9) to calculate $\mathbf{ReqDoT}_{A,A,sexp}$.

For both equation (5.10) and (5.11) when $\mathbf{DoT}_{A,B,sbeliefs} < \mathbf{ReqDoT}_{A,B,sbeliefs}$ then the credibility of the evidence and correspondingly its influence on the decision to trust Agent C about τ is set to zero. These conditions act as the agent's decision to trust a peer or itself as a source of beliefs.

$$\mathbf{ReqDoT}_{A,C,\tau} = \frac{\mathbf{U}\left(\mathbf{Failure}\right)_{A,C,\tau}}{\mathbf{U}\left(\mathbf{Failure}\right)_{A,C,\tau} - \mathbf{U}\left(\mathbf{Success}\right)_{A,C,\tau}} \tag{5.6}$$

$$\mathbf{U}\left(\mathbf{Success}\right)_{A,B,srec} = \begin{cases} \mathbf{U}\left(\mathbf{Success}\right)_{A,C,\tau}, & \text{if } \mathbf{Recommendation}_{B,C,\tau,t} \geq \mathbf{ReqDoT}_{A,C,\tau} \\ -\mathbf{U}\left(\mathbf{Failure}\right)_{A,C,\tau}, & \text{if } \mathbf{Recommendation}_{B,C,\tau,t} < \mathbf{ReqDoT}_{A,C,\tau} \end{cases} \tag{5.7}$$

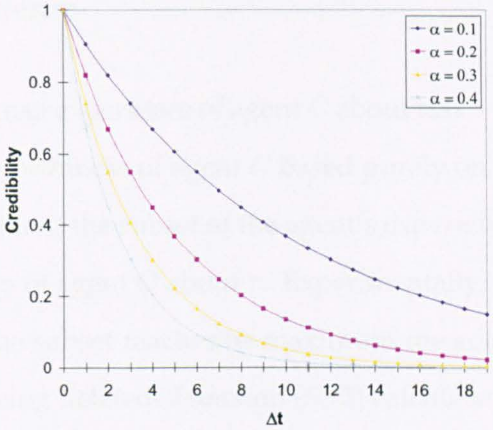


Figure 5.5: Belief Credibility vs $\mathbf{\Delta t}$, for $\mathbf{DoT}_{A,B,sbeliefs} = 1$ and $\alpha = 0.1, 0.2, 0.3$ and 0.4.

$$\mathbf{U}\left(\mathbf{Failure}\right)_{A,B,srec} = \begin{cases} \mathbf{U}\left(\mathbf{Failure}\right)_{A,C,\tau}, & \textbf{if } \mathbf{Recommendation}_{B,C,\tau,t} \geq \mathbf{ReqDoT}_{A,C,\tau} \\ -\mathbf{U}\left(\mathbf{Success}\right)_{A,C,\tau}, & \textbf{if } \mathbf{Recommendation}_{B,C,\tau,t} < \mathbf{ReqDoT}_{A,C,\tau} \end{cases} \tag{5.8}$$

$$\mathbf{ReqDoT}_{A,B,srec} = \frac{\mathbf{U}\left(\mathbf{Failure}\right)_{A,B,srec}}{\mathbf{U}\left(\mathbf{Failure}\right)_{A,B,srec} - \mathbf{U}\left(\mathbf{Success}\right)_{A,B,srec}} \tag{5.9}$$

$$\mathbf{Credibility}(\mathbf{Recommendation}_{B,C,\tau,t}) = \begin{cases} \mathbf{e}^{-\alpha\mathbf{\Delta t}} \times \mathbf{DoT}_{A,B,srec}, & \textbf{if } \mathbf{DoT}_{A,B,srec} \geq \mathbf{ReqDoT}_{A,B,srec} \\ 0, & \textbf{if } \mathbf{DoT}_{A,B,srec} < \mathbf{ReqDoT}_{A,B,srec} \end{cases} \tag{5.10}$$

$$\mathbf{Credibility}(\mathbf{Experience}_{C,\tau,t}) = \begin{cases} \mathbf{e}^{-\alpha\mathbf{\Delta t}} \times \mathbf{DoT}_{A,A,sexp}, & \textbf{if } \mathbf{DoT}_{A,A,sexp} \geq \mathbf{ReqDoT}_{A,A,sexp} \\ 0, & \textbf{if } \mathbf{DoT}_{A,A,sexp} < \mathbf{ReqDoT}_{A,A,sexp} \end{cases} \tag{5.11}$$

5.6.5 Direct Experience

We define agent A's direct experience of agent C about task τ ($\mathbf{Exp}_{C,\tau}$), as the belief agent A has about the trustworthiness of agent C based purely on its first-hand interactions of agent C. By this we refer to the subset of the agent's experiences consisting specifically of the agent's experiences of agent C about τ. Experimentally, we impose a maximum size on the subset, when the subset reaches its maximum the addition of further experiences results in the oldest being deleted. Function (5.12) calculates the direct experience belief by summing the agents most recent experiences each of which weighted by their assigned credibilities, $E_{C,\tau}$ denotes the set of times at which the agent had experiences of agent C about τ, the constants in the formula bound the result to within [0-1].

$$\mathbf{Exp}_{C,\tau} = \frac{0.5 \times \sum_{t \in E_{C,\tau}} \mathbf{Experience}_{C,\tau,t} \times \mathbf{Credibility}(\mathbf{Experience}_{C,\tau,t})}{\sum_{t \in E_{C,\tau}} \mathbf{Credibility}(\mathbf{Experience}_{C,\tau,t})} + 0.5 \qquad (5.12)$$

5.6.6 Reputation

In our agent system, reputation is defined as the collectively informed opinion held by an agent about the performance of a peer agent within a specific context. Agents form a belief about an agent's reputation from the recommendations of their peers. However, the received testimonies may be affected by existing relationships and attitudes. Thus, reputation is also a subjective concept which we define as a belief held/derived by one agent.

Equation (5.13) formulates subjectively, from the perspective of agent A the reputation $\mathbf{Rep}_{C,\tau}$ of an agent C about a task τ. The reputation is the weighted sum of the recommendations of its peers, weighted by the credibility measure assigned to each of those recommendations (equation (5.10)). The set $R_{C,\tau}$ contains the identifiers of agents b whom made recommendations of C about τ.

$$\mathbf{Rep}_{C,\tau} = \frac{\sum_{b \in R_{C,\tau}} \mathbf{Recommendation}_{b,C,\tau,t} \times \mathbf{Credibility}(\mathbf{Recommendation}_{b,C,\tau,t})}{\sum_{b \in R_{C,\tau}} \mathbf{Credibility}(\mathbf{Recommendation}_{b,C,\tau,t})} \qquad (5.13)$$

5.6.7 Confidence and the Opportunity to Trust

Our framework utilises the agents confidence in the trust belief in three ways. Firstly, as a mechanism to force the build up of evidence in a trust belief before allowing large risks to be taken. The perceived risk increases with the size of the order, therefore uncertain agents make smaller purchases and so take less risks. Secondly, the agent evaluates the reputation of the target only when it is not confident enough based upon its own experiences. And finally the agent ceases evaluating recommendations and their sources when it has accrued sufficient confidence, this allows for scalability of the framework with increasing numbers of recommendations. This avoids the possibility that as the number of recommendations about a target increases not only will the time taken by a agent to evaluate the reputation of that target increase but it also risks allowing the influence of the reputation belief to overpower an agent's an own experiences. This is a key aspect of the model allowing the agent to transition from risky reputation trust to reliance trust.

The mechanism also allows the consumer agents to change the nature of its opportunity to trust by altering the quantity they intend to purchase. This results in a change in the calculated situational variables (required trust, required confidence and the outcome utilities). Meaning that each bid by a producer agent represents a number of possible opportunities for the consumer. If the consumer fails to make a decision to trust a producer, not as a result of distrust but because of a lack of confidence, the agent will reduce the risk by lowering the quantity selected, thus reducing the $reqConf$ and $reqDoT$ making both these conditions more easily satisfied. Our agents search the quantity space in order to find the quantity which maximises utility whilst satisfying the confidence criteria.

5.6.8 Combining Experience and Testimony to Formulate Trust

Earlier we defined the trust belief $\mathbf{DoT}_{A,C,\tau}$ as being a subjective evaluation based on the agent's experiences and reputation. In this section we describe the final stage in determining the agent's degree of trust by addressing the combination of the agent's direct experience and reputation beliefs. Intuitively an agent with strong confidence in its direct experience belief and little confidence in the accuracy of its reputation beliefs should rationally choose to calculate its trust belief primarily from its direct experiences. Conversely an agent with little experience should base its trust on reputation.

To achieve this we associate a degree of confidence with both our direct experience and reputation beliefs. Three factors are important in determining this confidence measure. They are the trust in the sources of the evidence, the currency of the evidence and the amount of evidence used to form the belief. The first two of these factors are addressed when determining the credibility of the individual evidences themselves. We therefore define the confidence in the beliefs $\mathbf{Exp}_{C,\tau}$ and $\mathbf{Rep}_{C,\tau}$ as the sum of the supporting beliefs respective credibility measures (equations (5.14) and (5.15)). The weighted combination of the two beliefs takes the form of function (5.17), where the result is scaled to between [0-1] by the denominator. We see that the relative magnitude of the beliefs influence on the trust belief is determined by an agent's confidence in their respective accuracies.

$$\mathbf{Confidence(Exp}_{C,\tau}\mathbf{)} = \sum_{t \in E_{C,\tau}} \mathbf{Credibility(Experience}_{C,\tau,t}\mathbf{)} \tag{5.14}$$

$$\mathbf{Confidence(Rep}_{C,\tau}\mathbf{)} = \sum_{b \in R_{C,\tau}} \mathbf{Credibility(Recommendation}_{i,C,\tau,t}\mathbf{)} \tag{5.15}$$

$$\mathbf{DoT}_{A,C,\tau} = \frac{\mathbf{Confidence(Exp_{C\tau})} \times \mathbf{Exp_{C\tau}} + \mathbf{Confidence(Rep_{C\tau})} \times \mathbf{Rep_{C\tau}}}{\mathbf{Confidence(Exp_{C\tau})} + \mathbf{Confidence(Rep_{C\tau})}} \tag{5.16}$$

$$\mathbf{Confidence(DoT}_{A,C,\tau}\mathbf{)} = \mathbf{Confidence(Exp_{C\tau})} + \mathbf{Confidence(Rep_{C\tau})} \tag{5.17}$$

5.7 Control Experiments – Economic Simulation

This section briefly explores the dynamics of the underlying economic trading market scenario. The purpose of this is to verify that from a trading perspective the simulation demonstrates the expected behaviour required to support the conclusions drawn with respect to the effect of socio-cognitive reasoning on the behaviour of the system. As investigated in the following section 5.8. More specifically these initial experiments are to check that the interaction between our model of producer price setting (the profit derivative follower) and consumer economic choices (based on the marginal utility function) can create the desired macro-economic effects.

5.7.1 Monopoly vs. Competition

The first experiment demonstrates the interaction of the producer price setting and consumer choice, under monopoly and competitive conditions. In both cases we simulate 50 consumer agents with homogeneous utility functions. Under monopoly a single producer agent acts as the sole trader and for competition 2 producers with the same cost structure compete to sell a commodity. Note the producers are both completely trustworthy and competent.

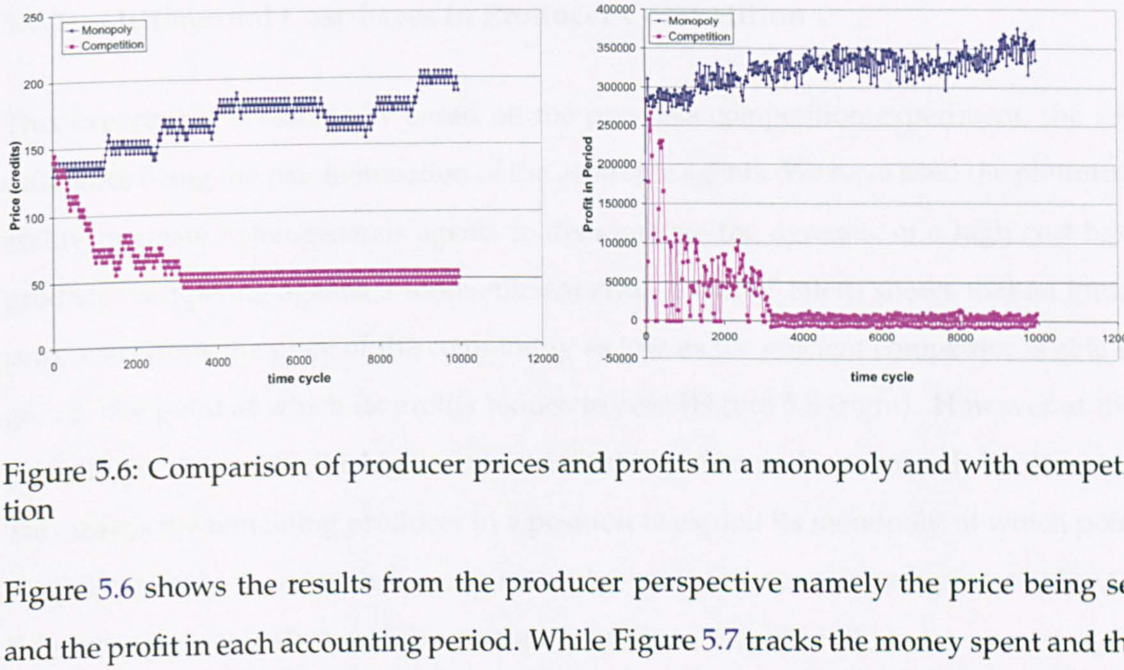


Figure 5.6: Comparison of producer prices and profits in a monopoly and with competition

Figure 5.6 shows the results from the producer perspective namely the price being set and the profit in each accounting period. While Figure 5.7 tracks the money spent and the utility gained by the consumers. As is expected under a monopoly the producer is free to set its prices high and take for itself the majority of the economic surplus, the consumers spend their maximum budget. By contrast the price war that ensues from the competition scenario forces prices down as low as possible within the limit of the producers' cost function. The consumers spend significantly less but capture all the economic surplus, resulting in significant utility gains and reducing the producers' profit to effectively zero.

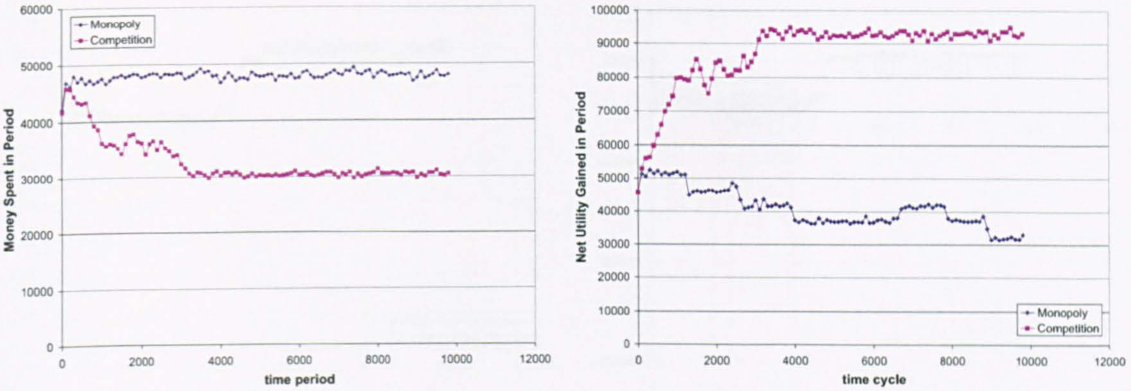


Figure 5.7: Comparison of consumer spending and utility gained in both a monopoly and with competition

These results are in keeping with what we would expect to see in a commodity market functioning under perfect competition.

5.7.2 Differential Cost-bases in Producer Competition

This experiment is essentially based on the previous competition experiment, the key difference being the parametrisation of the producer agents. We have used the platforms ability to create heterogeneous agents to demonstrate the dynamic of a high cost base producer competing against a more efficient rival. Figure 5.8(left) shows that an initial price war drops the price of the commodity as low as the efficient competitor is able to go, i.e. the point at which its profits reduce to zero (Figure 5.8 (right). However at this price the producer with the high cost base is making a loss and is eventually bankrupted. This leaves the remaining producer in a position to exploit its monopoly, at which point the price quickly rises and the economic surplus previously creating a high net utility for the consumer is transferred to the monopoly producer (Figure 5.9).

5.8 Economic and Socio-cognitive Simulation

The experimental results presented in the following section focus on 4 hazards in the agent mediated market place. These hazards are market entry, incompetent producers, malicious producers and dishonest recommenders. For each hazard we simulate a num-

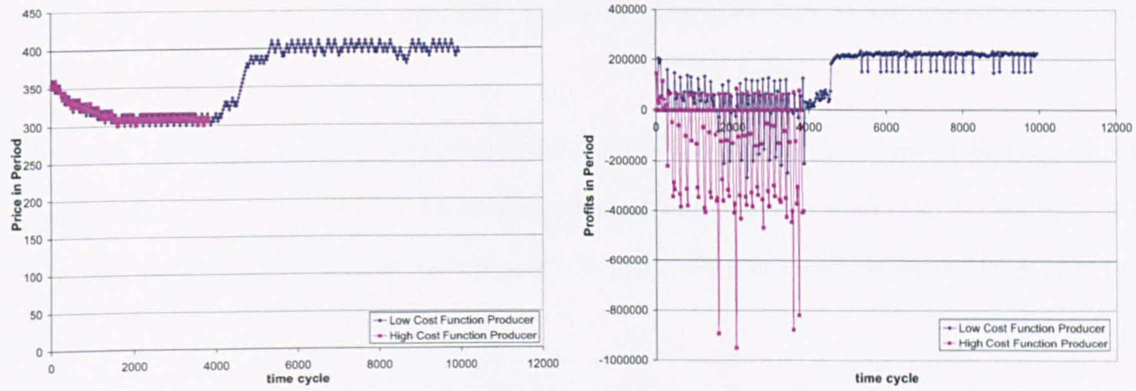


Figure 5.8: Pricing (left) and Profit (right) over time during competition between producers with differing cost-bases.

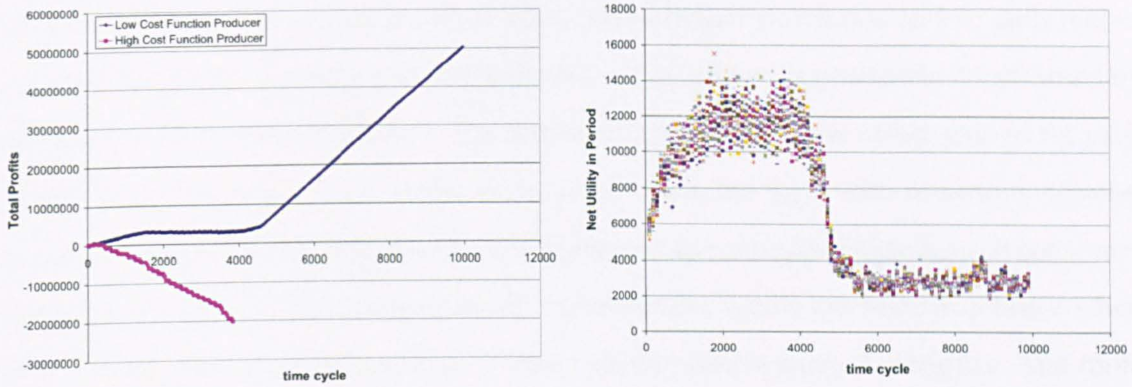


Figure 5.9: Cumalative producer profit (left) and Consumer utility (right) gained each accounting period, during competition between producers with differing cost-bases.

ber of agent configurations, i.e. purely economic agents, those able to learn from direct experience and those with and without recommendations. We do this to demonstrate the performance of our agents in a number of scenarios with differing degrees of socio-cognition.

5.8.1 Consumer Market Entry

Consumer market entry is a dangerous time for the new consumer, because it will have no prior knowledge as to the trustworthiness of the other participants. If there are many malicious agents in the market identifying them all will be a very costly exercise. This is likely to act as a significant barrier to entry and so discourage consumers from entering the market. The following simulations demonstrate the effect of recommendations on the rate at which our CS_CEF enabled agents are able to bootstrap and the accuracy with which they do so.

We ran 4 simulations, all of which featured a simple market system consisting of ten consumers. These consumers are introduced into the system every 50 cycles (starting with cAgent0 at t = 0). The producers' price is static throughout to ensure results are not affected by price changes.

Fig. 5.10 shows the results from the first two simulations, when only a single infallible altruistic producer agent exists. The consumers net utility increases with each interaction because as it gains more experiences it becomes more confident in the trustworthiness of the producer, this confidence allows it to make bigger purchases and so gain higher utilities. The order quantity and hence the net utility gained is ultimately constrained by the agent's economic parameters. The left hand graph shows the utility gained for each agents first 10 interactions without recommendations, the right with recommendations. It can be seen that without communication between the consumer agents, each consumer must build its confidence independently. However the agents can bootstrap faster when they utilises the recommendations of those joining before them 5.10(right). The more recommendations for both the producer and for the recommenders of the producer there are in the system, the faster they can bootstrap.

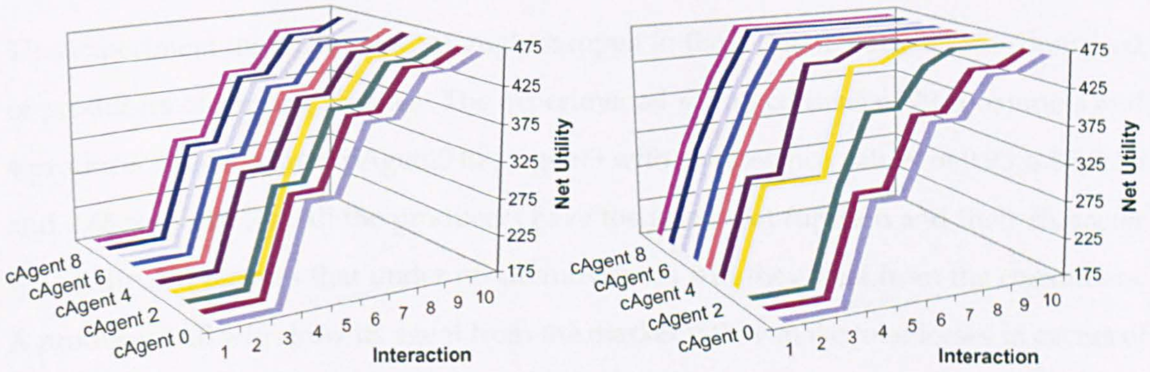


Figure 5.10: Consumer Market Entry: Single Altruist Producer, Graphs show the Utility gained from successive interactions. Left graph shows utility gained by each agent in each interaction without recommendations, right with recommendations

In the final two simulation runs the single infallible altruistic producer agent is joined by four malicious agents with no intention of providing services in return for payment. Our malicious agents have no costs and so can always afford to price lower than the Altruist.

As can be seen in Fig. 5.11 these low prices attract new agents whom know no better than to trust the malicious agents. Without recommendations each new consumer must work its way through all the malicious producers to find the altruist and then it must slowly build its confidence in them. With the addition of the recommendation system only the initial consumer is the victim of the malicious behaviour, subsequent entrants may then bootstrap as in Fig. 5.10(right).

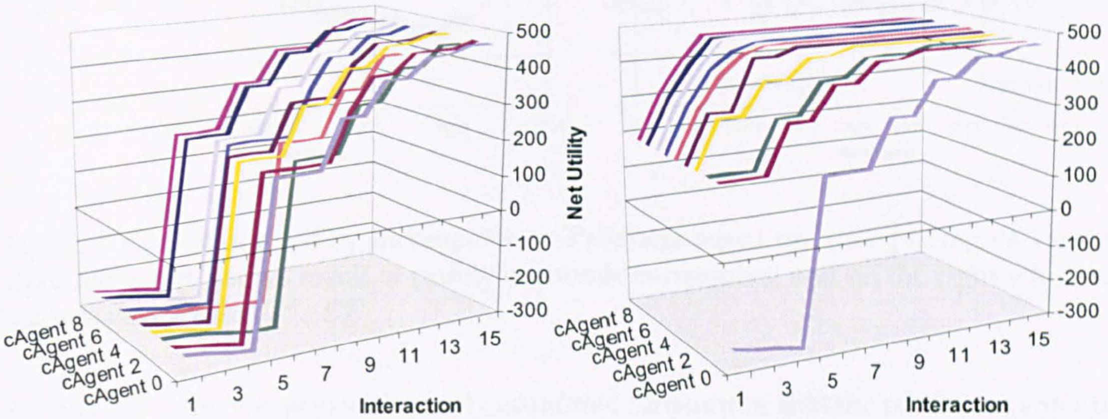


Figure 5.11: Consumer Market Entry: Malicious producers present, Graphs show the utility gained from successive interactions. Left graph shows utility gained by each agent in each interaction without recommendations, right with recommendations

5.8.2 Surrounded by Incompetence

This experiment investigates what might happen in the e-Retail marketplace comprised of producers of varying abilities. The experimental set-up consists of 20 consumers and 4 producer agents named pAgent0 to pAgent3 with competence values of 0.95, 0.85, 0.75 and 0.65 respectively. All the producers have the same cost function and their character type (Altruist) dictates that under no circumstances will they steal from the consumers. A producer will withdraw its agent from the market if they make total losses in excess of 200000 credits.

We ran this set-up with two configurations:

1. The consumers are not socio-cognitive, they base their decisions on the economic components of the framework.

2. The consumers use the full CS_CEF.

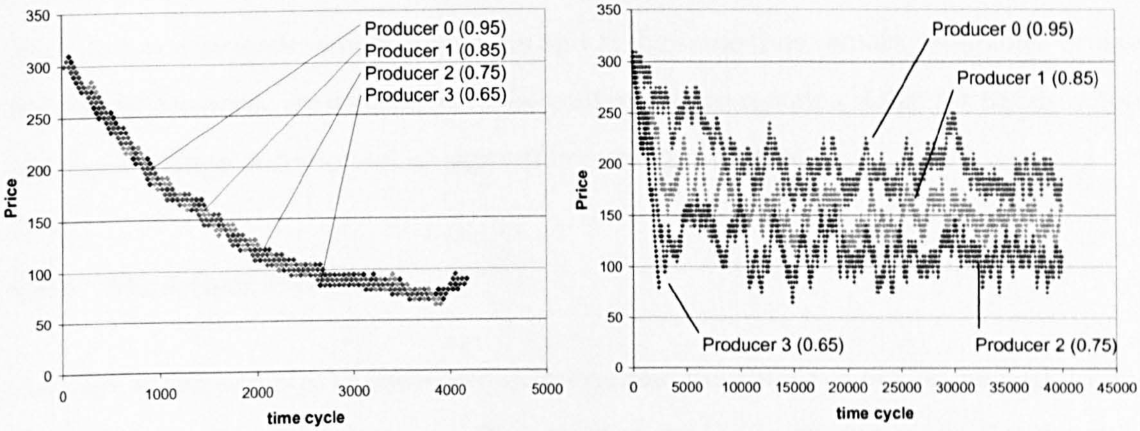


Figure 5.12: Surrounded by Incompetence: Producer agent resource pricing vs Time Period, left graph shows result of purely economic consumers, and on the right when consumers employ the CS_CEF

We can see in the scenario with purely economic consumers, that the producers enter into a price war this turns out to be mutually destructive (Fig. 5.12(left)). Within 4200 cycles all the producers competent and incompetent alike are eliminated from the market place, each having made significant losses (Fig. 5.13(left)).

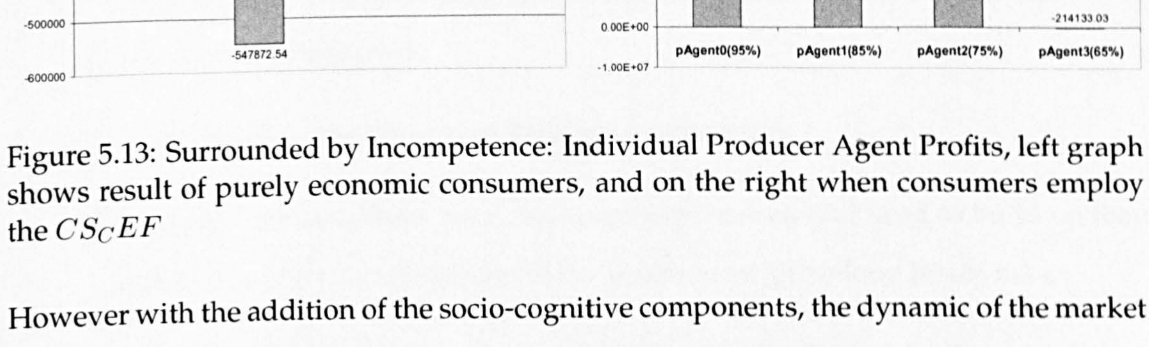


Figure 5.13: Surrounded by Incompetence: Individual Producer Agent Profits, left graph shows result of purely economic consumers, and on the right when consumers employ the CS_CEF

However with the addition of the socio-cognitive components, the dynamic of the market changes considerably specifically in terms of its fairness and stability. With the socio-cognitive consumers present, the producers find equilibrium prices (Fig. 5.12(right)). These prices reflecting their relative competency, allowing the more competent producers to price higher than the less competent. As a result the total profits gained by the producers also reflects this ordering (Fig. 5.13(right)). With the socio-cognitive agents only one producer is eliminated from the market, who due to its incompetence was unable to set its price low enough to attract customers and at the same time remain profitable. Unlike the purely economic market, the socio-cognitive system remains stable for the duration of our simulation and showed no signs of becoming unstable.

5.8.3 Malicious Producers

The aim of this series of experiments is to examine the ability of our framework to defend against malicious behaviour. Our outright malicious producers are simulated as never rendering services ordered, because of this and unlike the altruistic producers they have zero costs. The malicious agents alter their identity every 200 cycles in an attempt to periodically shed their negative reputations, this reflects the ease with which an agent can change its identity in many distributed systems. The simulations feature 20 consumers and 6 producers, of which 4 producers are malicious and the remaining two altruists have competency values of 0.95 and 0.90. As a preliminary experiment we investigate the effects of adding increasing layers of socio-cognition, on the total profits and net utility gained by our producers and consumers over the course of the simulation.

We ran this scenario with 5 different consumer agent configurations:

1. The agents are not socio-cognitive, they base their decisions on the economic components of the framework.
2. The agents may only learn from their own experiences.
3. The agents may only learn from their own experiences, and need to build up their confidence in the trustworthiness of the producer before taking larger risks.
4. The agents utilise all the features of the CS_CEF, i.e. The agents form their trust belief from their own experiences and the recommendations of its peers, and they must build up their confidence in the trustworthiness of the producer.

5. We removed the malicious producers, the agents utilise all the features of the CS_CEF. This simulation is a best case control experiment.

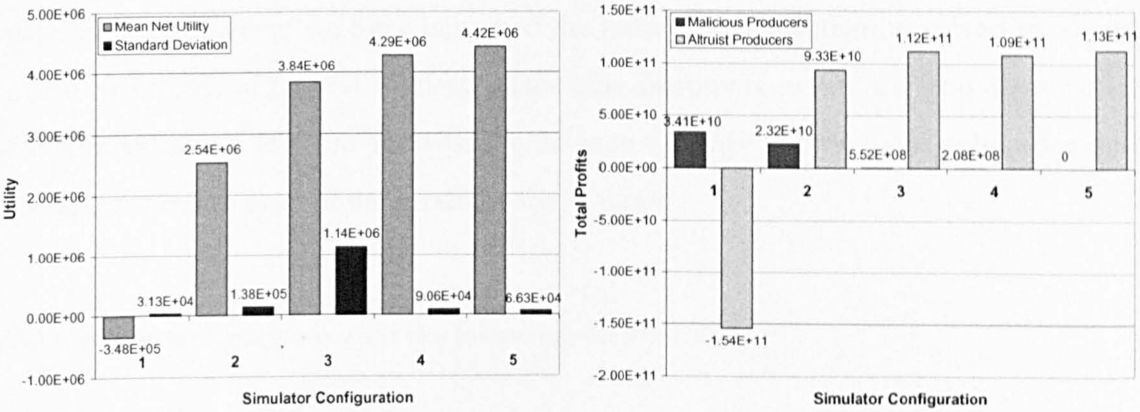


Figure 5.14: Malicious Producers: Left compares mean utility gained by, and the standard deviation between individual consumers in each configuration, right compares the total profits accrued by malicious and altruistic producers for each configuration

The key trend apparent from Fig. 5.14 is that with every increase in the consumers level of socio-cognitive reasoning the closer the results of the simulation match those of our positive control (simulation 5). Our first simulation run is our worst case example and results in losses of utility and profits for our altruistic consumers and producers, whilst allowing the malicious producers to turn a profit. This makes it evident that where there is the possibility that malicious agents may have access to the market place, solely economic trading agents are highly undesirable. The second run demonstrates how enabling the consumers to form experiences and reason based on them, reverses this trend by facilitating positive altruistic producer profits and mean consumer utility meanwhile reducing malicious producer profits. Simulation 3, continues this trend but also has the highest standard deviation between the results of the individual consumers. Even though on average consumers will do well in this market the gap between the most fortunate and least fortunate is the largest, thus the society formed is far from egalitarian. From the consumer point of view, using agents with all the features of the CS_CEF enabled (simulation 4) allows the market to best approximate an environment where there are no malicious agents (simulation 5).

5.8.4 Malicious Recommenders

The participating producer agents are specified the same as in the malicious producers experiments, however we have increased the number of consumers involved to 30 and made two thirds of them dishonest. Malicious consumers in our scenario distort their recommendations, they are motivated to do so in the hope that they can reduce demand for, and hence the price of the producer they favour.

We ran this configuration with the following configurations:

1. The agents assume all recommenders are trustworthy.
2. The agents use the CS_CEF to its full extent.

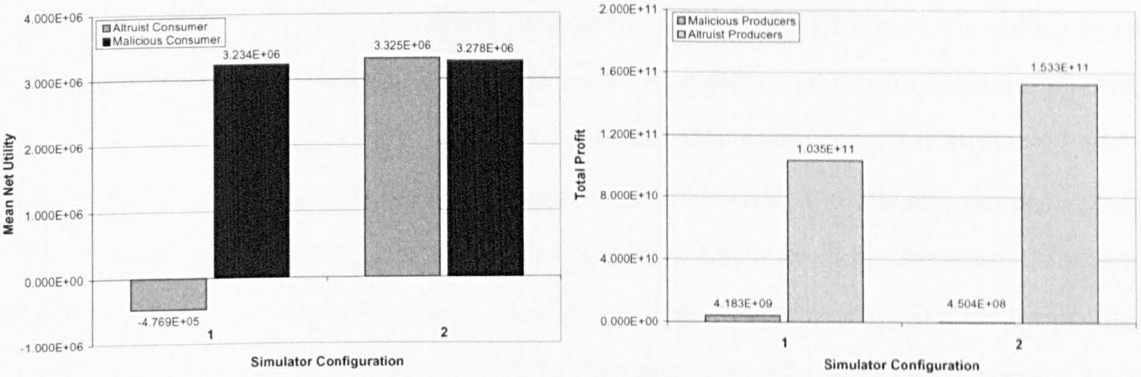


Figure 5.15: Malicious Recommenders: Left compares mean utility gained by malicious and altruistic consumers for both configurations, right compares the total profits accrued by malicious and altruistic producers for both configurations

Our first configuration relies upon significantly greater numbers of honest recommenders than malicious ones in order to ensure sufficient accuracy of the reputation belief to avoid distorting the agents decision to trust. Our framework however does not make this assumption, as we model the trustworthiness of recommenders, make the decision whether or not to trust recommenders and allow the most trusted recommenders to have a greater influence on our agent's reputation belief. As can be seen from Fig. 5.15 this minimises the damage which malicious recommenders may precipitate in our multi agent system. For instance from the supply side of the market, the removal of the assumption results in a reduction of malicious producer profits by approximately 89%, accompanied by an increase of 48% for the altruist producers. From the consumer perspective we witness an absolute reversal of fortune for our altruistic consumers.

5.9 Chapter Summary and Conclusions

This chapter has introduced our computational socio-cognitive and economic decision framework (CS_CEF). In order to support the development and testing of the CS_CEF we have devised an experimental scenario based on a unmoderated, open, distributed, agent mediated marketplace. As opposed to distilling an abstract game we have grounded this scenario using economic theory, this allows the agents in our examples to reason about their delegations and commercial transactions in terms of their actual economic utility. It also means that our simulations can capture properties absent in abstract games, such as price stratification, monopoly and competition effects. In addition to the ability to see how an agent's future opportunities and their associated outcome utilities are affected by their past performance/behaviour. In the future this grounding of utilities will allow us to add to the system: alternative actions, new interaction models and decision models without changing the underlying simulation scenario/models. An example would be to add producer initiated auctions (alongside consumer initiated calls for proposals), this would allow for the transfer of risk to the producer and enable consumer agents to interact with producers when trust is low or uncertain.

Our simulation results have shown that social order in competitive multi-agent systems can be created and supported by the introduction of a socio-cognitive layer in support of the agent's economic reasoning. We have demonstrated that the framework benefits law abiding and competent members of the agent-mediated market place and hence the human society in which they are embedded. To achieve this we have developed an agent simulation environment using the PreSage platform, providing tools facilitating the specification of agents, control of the specified market economic factors, data logging, results analysis, demonstration and visualisation. We have presented simulation results that show that our framework also protects against the actions of dishonest consumers who spread inaccurate recommendations in an attempt to reduce demand for reliable producers services and hence drive down prices. Our simulation package has provided experimental verification for our framework in a number of scenarios, by simulating heterogeneous groups of producer and consumer agents.

Figures 5.16 and 5.17 show how we have extended the provided classes of presage to create our simulation agents, the network and data logging plugins, and how the XML configuation and subsequent running of the simulation has created the results in this chapter. In summary we developed three basic agent classes the Consumers, the Producers and a Name Server and Recommendation Database. These were subsequently further extended through the addition of increasingly complex algorithms for decision making.

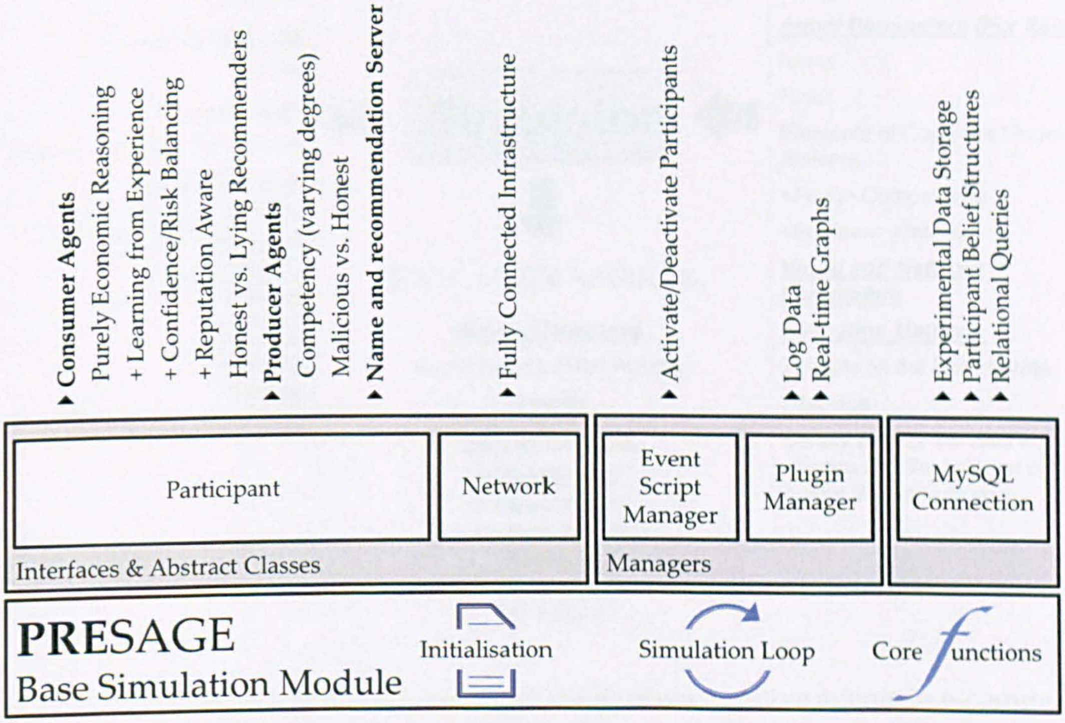

Consumer Agents
Purely Economic Reasoning
+ Learning from Experience
+ Confidence/Risk Balancing
+ Reputation Aware
Honest vs. Lying Recommenders
Producer Agents
Competency (varying degrees)
Malicious vs. Honest
Name and recommendation Server
Fully Connected Infrastructure
Activate/Deactivate Participants
Log Data
Real-time Graphs
Experimental Data Storage
Participant Belief Structures
Relational Queries
Participant
Network
Interfaces & Abstract Classes
Event Script Manager
Plugin Manager
Managers
MySQL Connection
PRESAGE
Base Simulation Module
Initialisation
Simulation Loop
Core functions

Figure 5.16: 'Toast Rack' instantiation of the platform architecture, for simulating socio-cognitive agents in e-commerce

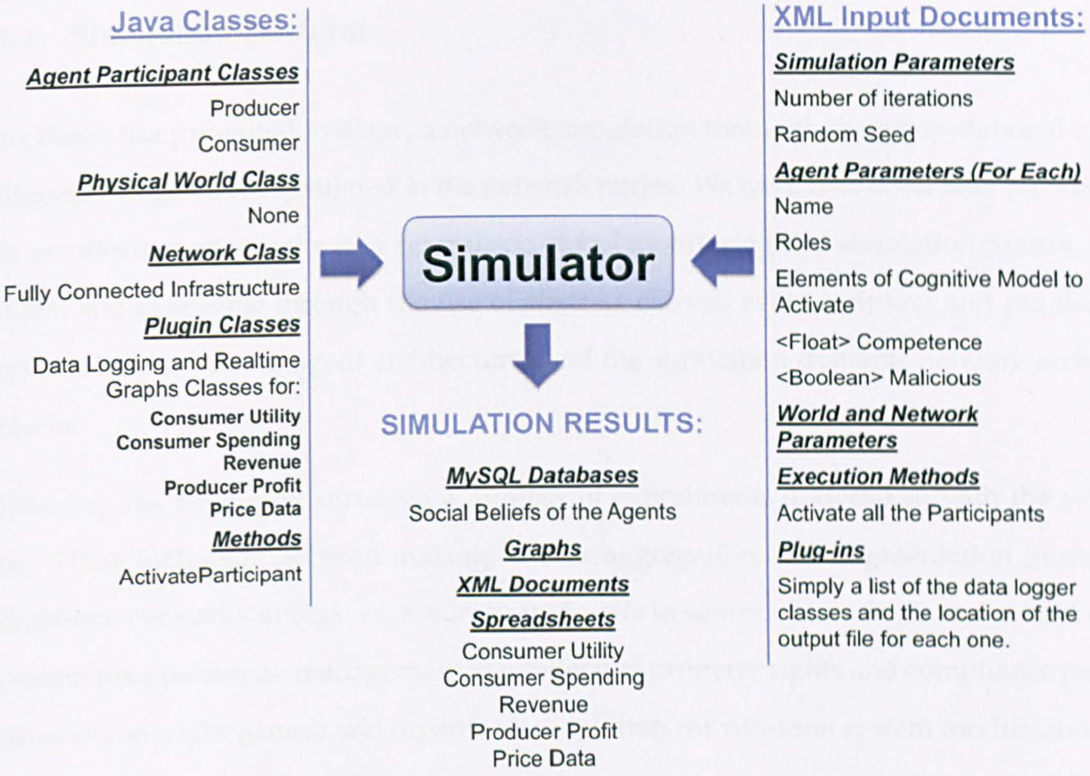


Figure 5.17: Input/Output chart, for simulating socio-cognitive agents in e-commerce

CONCLUSION 6

6.1 Summary

6.1.1 Simulation Platform

This thesis has presented PreSage, a network simulation tool with the computational intelligence of agents encapsulated in the network nodes. We have described how our system architecture affords the user centralised global monitoring and simulation control, is flexible and extensible through the use of abstract classes, event-scripting and plugins; supports heterogeneous agent architectures and the simulation dynamic network architectures.

Following this we briefly surveyed a number of experiments undertaken with the system. These included: decision-making, cluster aggregation and fragmentation in mobile ad-hoc networks; energy vs. accuracy trade-offs in sensor networks; management of common pool resources; management of intellectual property rights and compliance pervasion in copyright games; and organized adaptation for run-time system modification. The range of applications and their varying characteristics demonstrate that PreSage is a versatile and reconfigurable tool for simulation and animation of intelligent agents and multi-agent systems.

6.1.2 Socio-cognitive Agents - Case Studies

The second dimension of this thesis focuses on engineering distributed systems whereby the interactions between components are conditioned by social relations and socio-cognitive factors (e.g. trust reputation, forgiveness, kinship and similarity). We have developed two examples of how MAS are improved by the addition of social concepts. These demonstrate that empowering agents with social relations and socio-cognition can have positive regulatory and performance outcomes for large open and distributed groups of agents, and more specifically that both information dissemination and security can be supported by the emergent social order.

Socially adapted information dissemination

In Chapter 4 we presented the first of our two case studies where we investigated how an agent can model its peers and use this knowledge in an information dissemination task. The resulting cognitive model utilised concepts from recommender systems (i.e. collaborative filtering) and self organisation. The aim of the case study was two fold, firstly we demonstrated PreSage's ability to model and visualise dynamic networks (in this case a P2P model) and secondly to investigate the effect that peer selection approaches have on the resultant network structures, the welfare of the individual agents, and the macro effect on the welfare profile of the agent society.

The results demonstrated:

- Selecting Peers based on a minimum similarity threshold is better for niche peers than aiming for a set number of relationships.
- Threshold selection can be unpredictable w.r.t scalability.
- Social Relationships between agents can be effectively used to pre-filter information, reducing agent workload and network costs.

Trust

Finally in chapter 5 we described an experimental evaluation of a (computational) socio-cognitive and economic framework (CS_CEF) for agent decision making in distributed e-commerce markets. The framework formalises social theories of trust, reputation, recommendation and learning from direct experience and integrates these socio-cognitive formalisms with the agent's economic reasoning. Together the socio-cognitive and economic elements are designed to enable agents to cope with malicious or incompetent behaviour in an e-commerce environment. The results of our simulations show that the integration of social behaviour into the trading agent architecture can not only act as an effective mechanism for regulation of distributed agent-mediated market places, but also improve the agents ability to maximise its owners utility.

In summary, our initial experiments have shown that the employment of the CS_CEF in hostile trading environments:

- significantly reduces the profitability of malicious producers.
- minimises the damage which malicious recommenders cause.
- ensures the most reliable and law abiding sellers are the most profitable.
- reduces the variation in performance of individual participants.
- results in the overall performance of the market approximates that of a safe environment.

6.2 Limitations

This section outlines the limitations of our three main sections.

6.2.1 PreSage

Scalability While multi-threading and cluster based architectures for PreSage are currently being investigated the version of PreSage presented in this thesis is restricted to a single thread of execution on a single machine. While a number of separate simulations can be run concurrently across a cluster, support for utilising distributed computation to support very large scale simulations (1000's of nodes) is not supported. This is of particular interest as part of our future work involves simulating systems of systems, such as intelligent power supply grids, which will require tens of thousands of nodes. However it is not a simple task to parallelise the presage code, due to the level of interaction between nodes and centralised functions such as the physical world and dataloggers.

Usability PreSage's target audience is comprised mainly of computer scientists investigating agent systems, given such a group are generally capable of finding their way around programming API's our requirements have not focused on elements such as user interfaces, drag and drop development and configuration; as such these are less developed in comparison with other simulation packages. The platform does provide an API for interfacing custom GUI's to the main simulation package, along with an example interface, further GUI elements are not planned. In general accessibility should be improved by providing training resources when we publish the platform, though the API's will become more complicated when we add support for distributed computing in order to address the scalability issue.

Resource constrained agents PreSage's turn based approach to simulating the agents provides mechanisms to ensure the agents actions are randomly interleaved to ensure protection against first mover advantage, agents taking multiple actions in a time frame etc. However one aspect of fairness has not been addressed, namely the equality of computational resources provided to agents in any given simulation time period. There is no benefit incurred for agents whom calculate their actions quicker than those whom make use of slower but more exhaustive decision making. This limitation makes it difficult to simulate certain classes of experiment for example using anytime algorithms or investigating the benefit of socio-cognitive frameworks as short-cuts in time/resource constrained environments (discussed more later). A number of testbed/experiment level workarounds have been discussed however no platform level solution which could apply generically has been found. Recently a Java virtual machine aimed at constraining resources consumed by mobile agents within their host environment was announced, and may be ideal for solving this limitation.

6.2.2 Socially adapted information dissemination

The first limitation of the work in Chapter 4 is our use of formulas to define the requirements distribution across the peers and the creation of the resources for the agents. This approach was initially intended as a stop gap, allowing the development of the agents and initial testing until user trials could provide live data as part of the Tiramisu Project; this stage of the project remains incomplete. However, one advantage of using the parametrised approach is that we are able to simulate for longer (we don't run out of data) and simulate trajectories through the state space that may not be present in collected data.

The results indicate that regardless of the agents requirements the fewer peers an agent selects to receive information from the more accurate the recommendations are and the better the agents decision making. This is an oversimplification, while an agent with more connections will receive less tailored information, it will receive it quicker. In dynamic environments this trade off is important. This brings us to our second item of future work; resources (plans) that become less effective the more they are used. Agents which have larger numbers of connections will receive information quicker should have an advantage against isolated agents, acting as a negative feedback against tighter peer selection thresholds. Combine this with agents that adapt their peer selection thresholds based on utility or error, and we can investigate the affect of the rate of resource depletion (environmental dynamism) on the balance between higher accuracy (less connections) and lower latency (more connections).

6.2.3 Trust - CS_CEF

Generic Risk CS_CEF currently determines the risk of an interaction by calculating the loss of economic capital that can arise from a given trust decision. This risk is then utilised to determine an agent's minimum decision making confidence and act as a worst case scenario for calculating the expected outcome. A reasonable approach for a framework aimed at integrating trust and economic models. However our future plans for trust model research are not limited to marketplace scenarios and so a generic model of risk/reward perception is required.

Required Confidence As with an agents risk perception our algorithm for required confidence is based on simple assumptions (more risk implies greater required confidence) which can be easily calculated from the economic context of a trust decision. Once again this is reasonable in the context of a marketplace scenario, but is less useful in other scenarios such as trusting a peer with sensitive information where the final outcomes are not easily measured in monetary terms. Furthermore the heuristic for determining required confidence only uses the economic conditions of the current decision, it may be useful to consider the wider context for the agent. For example the agent may become more risk adverse as its bank balance depletes, or where it has suffered a series of deceptions.

Tasks, Context and Logic Our representation of trust $\mathbf{DoT}_{A,B,\tau,\Omega}$ included two parameters for defining the task to be delegated (τ) and the wider context in which the delegation will occur (Ω). For our experiements τ is implemented as a label defining the task as either to provide the purchased service or accurate recommendations; these tasks are definined by the agent designer. Ω was omited with the intention to introduce it in future scenarios. This limits our framework to reasoning about predefined tasks and scenarios and ignores the context of the environment. We will need to adapt our representations of task and context defining them logically in an appropriate action language. This would enable agents to first reason if it is possible for a peer to fulfill a given task in a given context, before employing the CS_CEF to determine if it can be trusted. It would also enable agents to distinguish between failures caused by agents and those that can be attributed to the environment (more on this in future work).

6.3 Thesis Contributions

The contribution of work detailed in this thesis can be categorised into three groups, namely the Method, the Platform and the Experiments. From a methodological perspective, in chapter 2 we show how taking the step from natural language theories to the description of formal computational models delineates the relationship between the Social and Computer Scientist. We go on to adapt the synthetic method [114] to describe how computer scientists can use the (often and necessarily) loosely defined theories of the Humanities and Social Sciences domains, in engineering agent technologies which benefit from formal models of social phenomena.

It is this methodology, the *adapted synthetic method*, which we have developed the PreSage platform to support. The animation of artificial systems inspired by, and governed by social phenomena as opposed to the simulation of human social systems. As such PreSage fills a gap in the available tools, between agent based social simulation tools such as Mason, Netlogo, Swarm and agent development middleware such as Jade or AgentBuilder. It is differentiated by:

- supporting conventional rules e.g. $p \dashrightarrow q$ in context c.
- supports full agent architectures, allowing for experimentation with computationally intensive algorithms.
- its discrete time approach allows experimenters to order interactions and to integrate with event calculus models.
- support for heterogeneous agent architectures.
- dynamic models of the environment including communication network properties.

The outcomes of our case studies themselves represent a contribution to the state of the art and in doing so support our argument for the novelty and contribution of the PreSage system. The first of these investigates self-organising recommendation where we take a holistic approach to developing our platform drawing on concepts from a broad range of domains. While the first contribution is the demonstration of PreSage's ability to investigate systems that cover a wide parameter space, and feature a rich representation of social networks. We also contribute to the communities understanding of algorithm selection for systems looking to use social networks in service discovery, peer selection and recommendation. When investigating centralised collaborative filters Herlocker et al [71] conclude that selecting a users best n peers (as opposed to all peers above a threshold) to solicit recommendations provides the best way to form recommendations. While our work in [124] supports this view from a simulation perspective. The simulations in chapter 4 show that when such systems are distributed and self-organised, the interplay between the neighbourhood selection and the underlying agent preferences cause the system to evolve very different network topologies. Subsequently, the performance of the system is highly dependant upon the initial conditions as well as the algorithm selected.

Finally our agent-mediated e-commerce scenario builds on market economics e.g. price setting, diminishing marginal utility, economies of scale and competition vs. monopoly effects in a methodologically individualistic manner i.e. as the decisions and behaviours of the population of software agents. The market test-bed itself is therefore a useful outcome for researchers investigating web-services, negotiation, auctions etc, which are situated in the context of a human economy. We also formalise a trust framework that integrates with these economic factors, to create an economically optimal but normatively governed system. We developed a scalable heuristic for agents to reason and integrate third party testimony into their trust beliefs which protects individual agents against group think. The same heuristic demonstrates the transition of decision making based on blind trust, to reputation based trust to reliance trust. Ultimately, this final case study demonstrates our ability to simulate systems comprised of nodes encapsulating complex algorithmic intelligence in the context of a complex dynamic environment.

6.4 Further Work

In addition to addressing the limitations outlined in section 6.2 we also hope to continue to develop the platform, expand the work on socio-cognitive models of trust and define future scenarios and applications where our approach can be of value.

6.4.1 PreSage

Diversity of included modules The platform currently provides a number of 'off the shelf' modules from which users can customise and build experimental testbeds. Future work will include greater variety of predefined agent architectures and network types; increased realism and world interactivity; and a greater number of plugins for visualisation and interfacing with external tools. Most of these should be addressed as more experiments are performed and the resulting code is rolled back into the platform.

Agents which use simulation as a cognitive process Motivating the need for simulation of agent systems is the desire to provide tools for developers and researchers that explain the long term results of design time decisions. Analogous to this, agents capable of simulating the consequences of runtime decisions could garner an advantage in complex interactive scenarios. An example of this would be conflict prevention in IPR law; which we investigated as part of the EU ALIS project. Cognitive agents could build models of their peers' behaviour and use simulation to predict outcomes and avoid conflict. Effectively transferring design time tools into services for agent decision making.

Games and Simulations involving both humans and Agents Another application of this is in developing systems whereby agents act as a cognitive prosthesis for their human owners. Our earlier case studies have described agents whom make decisions on the behalf of their owner mainly from the perspective of acting in the users absence, or acting to reduce the amount of work the user must perform. However there remains another important motivation for the use of socio-cognitive agents; instead of basing our models of trust on theories of how people reason about trust we can design our agents reasoning in such a way as to improve the decision making of the user by addressing weaknesses in the user's cognitive process. Creating an agent that augments the users trust behaviour in the form of a cognitive prosthesis.

6.4.2 Socio-cognitive Trust

Trust as a cognitive short-cut We argue that trust is not a rational calculation on the part of the trustor. Trust is not a calculated reliability score due to imperfect knowledge, time constraints or dynamic environments which change faster than a person or agent can compute a rational decision. The time bounded nature of decisions and excessive cognitive workload (in analysing all possible avalible information) is why people make irrational and quick decisions, errors which occur due to these short-cuts are referred to as cognitive bias and have been studied by psychologists. In order for real-time applications to scale socio-cognitive agents will likely employ cognitive short-cuts as opposed to rational analysis.

Most trust models (ours included) are heuristic decision making algorithms and can therefore be viewed as a form of cognitive short-cut. Indeed our CS_CEF framework already incorporates the concept of limiting cognitive load; where the model will stop searching for and evaluating evidence when the level of confidence required is reached. The result of this is that agents exert more cognitive effort the more risky they perceive the decision. This however assumes that agents have the required time to come to a decision, or that failing to come to a decision in time is more desirable that taking a chance. In this sense most trust scenarios underplay the cost of indecisiveness.

This motivates the case for further research into trust models based on Anytime Algorithms. These are a class of algorithm which can be interrupted mid-computation and still provide the best possible answer given the time constraints. For instance an anytime trust model could continue to search for supporting evidence (recommendations) and calculate outcomes for a set of possible delegations until such a time as it is forced to delegate the task or risk failure. Agents could execute anytime algorithms as a background activity, such that when faced with a time constrained decision they are capable of selecting the 'best' known option instantly. The use of anytime algorithms would enable us to investigate trust as a cognitive short-cut in simulations where the agents computational resources and time available could be constrained.

Causes of Behaviour A key aspect of social cognition is how should agents infer the causes behind any given behaviour or event in a social system, and in particular for a socio-cognitive theory of trust, how do we decide if an action taken by a peer is intentional. There are limits to performing causal attribution in P2P environments; namely a lack of external observation of a peer's interaction with others (surveillance), and also no observation of the environment (the network, or a nodes local physical surroundings). Heider [70] laid the groundwork for investigations into this issue which are categorised as Attribution theories. Stating that people attribute the cause of a persons behaviour or outcome in one of two ways. As either being inherent of some characteristic of the peer (i.e. their intelligence or trustworthiness) which he referred to as dispositional attribution. Or as a result of the environment and circumstance thus forming a situational attribution.

Clearly, in dynamic environments an agent's ability to correctly attribute an outcome of a dependence relationship be it positive or negative to either dispositional or situational factors is of great importance. It would enable an agent to avoid attributing blame to an agent when the outcome is outside of their control and also avoid increasing trust where the agents behaviour had little to do with obtaining a successful outcome. In addition to developing a trust model which can learn the consequence of the environment on predictions of reliance and also form a key factor in any theory of forgiveness which may be used to repair trust and protect social capital.

Other Of course trust in agent societies is a large field with many avenues for further investigation, many of which we feel can be studied using both PreSage and our agent mediated e-commerce testbed.

- **Economic Models:** It is also our aim to investigate other price setting mechanisms in our simulation experiments, we are specifically interested in socio-cognitive price setters.
- **Social Justice:** Currently the malicious recommenders and producers are not punished for their misbehaviour, merely isolated. Further investigation could be made into strategies for punishment and reparation, such as reciprocal behaviour, banishment and sanctions.
- **Mechanism Design:** Introduce the ability to negotiate the selection of the trading protocol. This would allow agents to assign risk to either party, this offers a way for less trusted peers to prove their capability (if not their willingness) to perform. In addition to solving certain bootstrapping issues.
- **Risk inclination:** Agents can be endowed with 'mind sets' in the form of a contextual state machines that define their inclination towards risk taking behaviour.
- **Selective Recommendation:** As a extension of the work on information networks in Chapter 4 agents may choose to direct their recommendations to peers either based on contextual information or simply on the similarity of the agents stated

beliefs. In this sense agents could as in Chapter 4 remove the need to communicate and process recommendations which provide them little useful knowledge.

6.5 Final Remarks

The scope of the case studies in this thesis has focused on the role of social relations between computational agents. We have demonstrated that these interpersonal relationships have a direct effect on the performance and efficiency of societies by enabling agents to manage risk and uncertainty, discover products, services, additional relationships and opportunities.

6.5.1 Socially Adapted Recommendation

Our case study on Socially Adapted Recommendation provided a challenge which tested and refined the platforms provision for creating large numbers of configuration files, the archiving of results data and the scripting of a number of experiments to execute consecutively. It has also demonstrated that experiments using PreSage can identify the transition from micro causation to properties at the macro level. However we also take a view that it is not simply the internal model of an agent and its reasoning which effect its trajectory but also the macro self organisation of the society and the mechanisms in place for social interaction. I.e. the current state of the marco social structure directly effects the actions agents are either capable or willing to enact. It is these actions which determine the social structure of the future (as shown in the network structures of Chapter 4). Thus PreSage has enabled us to animate both the micro ⟶ marco and macro ⟶ micro causal link; this is an integral facet of theories of social interaction.

6.5.2 Trust

Integrating socio-cognition into agents operating in open distributed multi-agent systems can provide protection against malicious and incompetent actors. We do not view our work as an alternative to traditional security methods based on authentication, encryption and certification. In fact we suggest that the combination of these two approaches leads to synergies between trust and security, benefiting law abiding and reliable members of the agent-mediated market place and their human owners.

We have demonstrated that building a socio-technical layer of computation, complementary to (and exploiting functionality of) a lower level of security, is technically feasible, and indeed offers a more tractable solution to certain security problems in e-commerce [21, 115] and digital rights management [19] than, say, ever more sophisticated encryption algorithms or encoding technology. For example, advocates of the Extensible rights Mark-up Language (XrML) position XrML as the foundation of trusted system development [127]: a higher-level approach is to use trust as the foundation of trusted system development.

6.5.3 PreSage

PreSage was originally developed as part of the EU ALFEBIITE project which inspired the socio-cognitive experiments on trust. However it was further developed on the EU TIRAMISU project and EU ALIS project where it was respectively applied to social recommendation systems and animating legal processes. Through its application to a number of projects it has demonstrated itself to be a highly reconfigurable programming environment. We have described the system architecture, and in addition to the two case studies in this thesis, Chapter 3 described eight independent testbeds built with it (counting Coloured Trials and Copyright Games as separate applications) Namely:

- Open Agent Mediated Marketplace
  - Trust, Reputation and Economics
- Social Networks and Recommendation
  - Distributed Recommendation, Network Self-Organisation
- Adaptation of Voting Rules
  - Paper: Hugo Carr and Jeremy Pitt OAMAS'08
- Voting in MANETs
- Coloured Trials (based on Harvard AIRG's Colored Trails)

  - Undergraduate Group Project
  - RC, ADR, LM in IPR law (EU ALIS Project)
  - CL & GT & LR in Virtual Enterprises
- Copyright Games
- Manet Opinion Formation approaches and their effect on Node Power Consumption
- Viral infection of networked machines

In reference to these applications we evaluate PreSage in terms of the requirements that we originally specified, and find that:

- all the applications allow animation of computationally-intensive algorithmic intelligence, using PreSage's discrete time execution model so that all agents complete their processing;
- multiple classes of networks have been simulated across the range of applications;
- the MANET and Coloured Trials applications demonstrate simulation of different physical environments;
- the applications collectively address issues of openness (MANET), fault-recovery (dispute resolution in Coloured Trials), volatility (CPR) and decentralisation (all);
- PreSage supports systematic experimentation, multiple types of visualisation, extensive re-use, customisation for different networks and environments, and reconfiguration (i.e. PreSage-MS).

The largest population of agents that has been simulated has been in the range of a few hundred separate processes. A major design and engineering challenge is scaling up PreSage to handle 'large' numbers of agents for micro-population simulation, and nested systems of rules for evidence-based policy making. This requires a new approach to the software and hardware configurations, and is the major focus of ongoing research and development. The applications reported here demonstrate that in its current stage of development PreSage is an effective network simulation tool when there is a requirement to encapsulate the computational intelligence of agents in the network nodes. As such, we plan to release PreSage open source with concomitant user support, for others to use as a highly re-configurable programming environment for simulation and animation of intelligent agents and multi-agent systems.

BIBLIOGRAPHY

[1] J. S. Adams. Inequity in social exchange. *Advances in experimental social psychology,* 2(267-299), 1965. 39

[2] J. Alspector, A. Kolcz, and N. Karunanithi. Comparing feature-based and clique-based user models for movie selection. In *Proceedings of the Third Conference on Digital Libraries,* pages 11–18, 1998. 29

[3] Josep Ll. Arcos, Marc Esteva, Pablo Noriega, Juan A. Rodríguez-Aguilar, and Carles Sierra. Engineering open environments with electronic institutions. *Eng. Appl. Artif. Intell.,* 18(2):191–204, 2005. 19

[4] Kenneth J. Arrow. *The Limits of Organization.* W. Norton & Company, 1974. 35

[5] A. Artikis and J. Pitt. A formal model of open agent societies. In *Autonomous Agents 2001 Conference,* pages 192–193. ACM Press, 2001. 19

[6] A. Artikis, J. Pitt, and M. Sergot. Animated specifications of computational societies. In *Proc. of the First International Conference on Autonomous Agents and Multi-Agent Systems,* pages 1053–1062. ACM Press, 2002. 16

[7] R. Axelrod. The evolution of cooperation, 1984. 37

[8] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984. 110

[9] Robert Axelrod. Advancing the art of simulation in the social sciences. In Rosaria Conte, Rainer Hegselmann, and Pietro Terna, editors, *Simulating Social Phenomena,* volume 455 of *Lecture Notes in Economics and Mathematical Systems,* pages 21–40. Springer-Verlag, 1997. 38

[10] Norman T. J. Bailey. *The Mathematical Theory of Infectious Diseases and Its Applications.* Griffin, 1975. 32

[11] A. Bandura. Social cognitive theory: an agentic perspective. *Annual review of psychology*, 52:1–26, 2001. 22

[12] K. Suzanne Barber and Joonoo Kim. Soft security: Isolating unreliable agents from society. In Rino Falcone, K. Suzanne Barber, Larry Korba, and Munindar P. Singh, editors, *Trust, Reputation, and Security*, LNAI 2631, pages 224–233. Springer, 2003. 37

[13] Nicholas J. Belkin and Bruce B. Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, December 1992. 29

[14] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade - a fipa-compliant agent framework. In *Proceedings of PAAM-1999*, pages 97–108, April 1999. 17, 42

[15] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with jade. In *ATAL*, pages 89–103, 2000.

[16] Fabio Bellifemine and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Softw. Pract. Exper.*, 31(2):103–128, 2001. 42

[17] J. Bentham, J.H. Burns, and H.L.A. Hart. *An introduction to the principles of morals and legislation*. Jeremy Bentham Works. Clarendon Press, 1996. 41

[18] Henri Bergson. *The two sources of morality and religion*. Macmillan, London :, 1935. 18

[19] J. Bing. Managing copyright in a digital environment. In I. Butterworth, editor, *The Impact of Electronic Publishing on the Academic Community*, pages 52–62. Portland Press, 1998. 143

[20] Andrew D. Birrell, Roy Levin, Michael D. Schroeder, and Roger M. Needham. Grapevine: an exercise in distributed computing. *Commun. ACM*, 25(4):260–274, 1982. 32

[21] M. Blaze, J. Feigenbaum, John Ioannidis, and Angelo D. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming:*

Security Issues for Mobile and Distributed Objects, number 1603 in Lecture Notes in Computer Science, pages 185–210. Springer, 1999. 143

[22] Sissela. Bok. *Lying : moral choice in public and private life / by Sissela Bok.* Pantheon Books, New York :, 1st ed. edition, 1978. 35

[23] Jeffrey M. Bradshaw, Niranjan Suri, nas Alberto J. Ca Robert Davis, Kenneth Ford, Robert Hoffman, Renia Jeffers, and Thomas Reichherzer. Terraforming cyberspace. *Computer*, 34(7):48–56, 2001. 18

[24] Lars Braubach, Alexander Pokahr, Dirk Bade, Karl-Heinz Krempels, and Winfried Lamersdorf. Deployment of distributed multi-agent systems. In *ESAW*, pages 261–276, 2004. 42

[25] Robin Burke. Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems*, page 2000. Marcel Dekker, 2000. 29

[26] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002. 29

[27] H. Carr and J. Pitt. Adaptation of voting rules in agent societies. In *Proceedings AAMAS Workshop on Organised Adaptation in Multi-Agent Systems (OAMAS)*, pages 36–53, 2008. 63

[28] C. Castelfranchi and R. Falcone. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In Y. Demazeau, editor, *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 72–79. IEEE Computer Society, 1998. 110

[29] C. Castelfranchi and R. Falcone. Social trust: A cognitive approach. In C. Castelfranchi and Y.-H. Tan, editors, *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer Academic Press, 2000. 20, 35, 99, 100, 110

[30] C. Castelfranchi and Tan. Introduction. In *Deception, Fraud and Trust in Virtual Societies*. Kluwer Academic Press, 2000. 110

[31] James S. Coleman. Social capital in the creation of human capital. *The American Journal of Sociology*, 94:S95–S120, 1988. 20

[32] R. Conte. A cognitive memetic analysis of reputation. Technical report, Project Deliverable, EU ALFEBIITE Project (IST-1999-10298), 2002. 39, 99

[33] R. Conte and C. Dellarocas, editors. *Social Order in Multi-Agent Systems.* Kluwer: Dordrecht, 2002. 18

[34] Rosaria Conte. Agent-based modeling for understanding social intelligence. *Proceedings of the National Academy of Sciences of the United States of America*, 99(10):7189–7190, may 2002. 41, 97

[35] Rosaria Conte, Bruce Edmonds, Scott Moss, and Keith Sawyer. Sociology and social theory in agent based social simulation: A symposium. *Computational and Mathematical Organization Theory*, 7(3):183–205, 1999. 39, 41, 101

[36] Rosaria Conte, Nigel. Gilbert, and Jaime Simao Sichman. Mas and social simulation: A suitable commitment. In J. S. Sichman, R. Conte, and N. Gilbert, editors, *Proceedings of the First International Workshop on Multi-Agent Based Simulation*, volume 1534, pages 1–9. Springer Verlag, 1998. 39

[37] Partha Dasgupta. Trust as a commodity. In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, chapter 4, pages 49–72. Basil Blackwell, New York, NY, 1988. 35

[38] Paul Davidsson. Multi agent based simulation: Beyond social simulation. In *MABS*, pages 97–107, 2000. 41, 47

[39] Paul Davidsson. Agent based social simulation: A computer science view. *J. Artificial Societies and Social Simulation*, 5(1), 2002. 41

[40] Tom De Wolf and Tom Holvoet. Emergence versus self-organization: Different concepts but promising when combined. In S. Bruecknet, editor, *ESOA 2004*. Springer-Verlag Berlin Heidelberg, 2005. 33

[41] S. DeLoach, W. Oyenan, and E. Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16(1):13–56, 2008. 16

[42] Alan Demers, Dan Greene, Carl Houser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. *SIGOPS Oper. Syst. Rev.*, 22(1):8–32, 1988. 32

[43] Demos. Digital music survey. Technical report, Demos, 2009. 61

[44] Giovanna Di, Marzo Serugendo, Marie pierre Gleizes, and Anthony Karageorgos. Self-organisation and emergence in mas: An overview. *Informatica*, 30:45–54, 2006. 33

[45] Emile Durkheim. *On morality and society; selected writings. Edited and with an introd. by Robert N. Bellah.* University of Chicago Press Chicago, 1973. 20

[46] Bruce Edmonds. The use of models - making MABS more informative. In *Proceedings of the Second International Workshop on Multi-Agent-based Simulation (MABS-2000)*, number 1979 in Lecture Notes in Artificial Intelligence, pages 15–32. Springer-Verlag, 2001. 38

[47] Leo Egghe and Loet Leydesdorff. The relation between Pearson's correlation coefficient r and Salton's cosine measure. *Journal of the American Society for Information Science and Technology*, 60(5):1027–1036, 2009. 30

[48] P. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie. From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, May 2004. 32

[49] Rino Falcone, Munindar P. Singh, and Yao-Hua Tan, editors. *The socio-cognitive dynamics of trust: Does trust create trust?*, volume 2246 of *Lecture Notes in Computer Science*. Springer, 2001. 20, 39, 99, 110

[50] Gutknecht O. Michel F. Ferber J. From agents to organizations : an organizational view of multiagent systems. In *Agent-Oriented Software Engineering (AOSE) IV*, volume LNCS2935, pages 214–230. Springer-Verlag, 2004. 19

[51] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. Kqml as an agent communication language. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, pages 456–463, New York, NY, USA, 1994. ACM. 19

[52] J. Fink and A. Kobsa. A review and analysis of commercial user modeling servers for personalization on the world wide web. *User Modeling and User-Adapted Interaction*, 10:209–249, 2000. 30

[53] FIPA. FIPA'97 specification part 2: Agent communication language. FIPA (Foundation for Intelligent Physical Agents), http://www.fipa.org, 1997. 19

[54] Leonard N. Foner. Yenta: A multi-agent, referral-based matchmaking system. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97), 301–307*. ACM Press, 1997. 31

[55] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer-Verlag, 1996. 21

[56] Noah E. Friedkin. Information flow through strong and weak ties in intraorganizational social networks. *Social Networks*, 3(4):273–285, 1982. 31

[57] Jonathan Gabbai, Hujun Yin, W A Wright, and Nigel Allinson. Self-organization, emergence and multi-agent systems. In S. Wang, editor, *IEEE International Conference on Neural Networks and Brain*. Springer Verlag, 2005. 33

[58] Y. Gal, B. Grosz, S. Kraus, A. Pfeffer, and S. Shieber. Agent decision-making in open-mixed networks. *Artificial Intelligence*, 2010. 60

[59] Diego Gambetta. Can we trust trust? In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, chapter 13, pages 213–237. Basil Blackwell, New York, NY, 1988. 35, 36, 111

[60] L. Gasser. Mace: High-level distributed objects in a flexible testbed for distributed ai research. In *Proceedings of the 1988 ACM SIGPLAN workshop on Object-based concurrent programming*, pages 108–110, New York, NY, USA, 1988. ACM. 21

[61] Nigel Gilbert and Steven Bankes. Platforms and methods for agent-based modeling. *Proc. of the National Academy of Sciences of the United States of America*, 99(10):7197–7198, 2002. 41

[62] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992. 30

[63] David Good. Individuals, interpersonal relations, and trust. In Diego Gambetta, editor, *Trust: Making and Breaking Cooperative Relations*, chapter 4, pages 49–72. Basil Blackwell, New York, NY, 1988. 35

[64] M. S. Granovetter. The Strength of Weak Ties. *The American Journal of Sociology*, 78(6):1360–1380, 1973. 31, 39

[65] Amy R. Greenwald and Jeffrey O. Kephart. Shopbots and pricebots. In *Agent Mediated Electronic Commerce (IJCAI Workshop)*, pages 1–23, 1999. 109

[66] Garrett Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, December 1968. 35

[67] Garrett Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, 1968. 63

[68] F.A. Hayek and B. Caldwell. *The road to serfdom: text and documents*. The Collected Works of F. A. Hayek. University of Chicago Press, 2007. 23

[69] Barbara Hayes-Roth. An architecture for adaptive intelligent systems. *Artif. Intell.*, 72(1-2):329–365, 1995. 22

[70] F. Heider. Social perception and phenomenal causality. *Psychological Rev.*, 51(6):358–374, 1944. 140

[71] Jon L. Herlocker, Joseph A. Konstan, A. Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR-99)*, pages 230–237, 1999. 29, 137

[72] Peter Huber. *Robust Statistics*. Wiley, New York, 1974. 30

[73] Jomi Fred Hübner, Jaime Sim ao Sichman, and Olivier Boissier. Moise+: towards a structural, functional, and deontic model for mas organization. In *AAMAS '02: Pro-*

ceedings of the first international joint conference on Autonomous agents and multiagent systems, pages 501–502, New York, NY, USA, 2002. ACM. 19

[74] David Hume, L. A. Selby-Bigge, and P. H. Nidditch. *A treatise of human nature / by David Hume ; edited with an analytical index by L. A. Selby-Bigge.* Clarendon Press ; Oxford University Press, Oxford : New York :, 2d ed. / with text rev. and variant readings by p. h. nidditch. edition, 1978. 35

[75] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993. 35

[76] N. R. Jennings. Agent-oriented software engineering. In *12th Int. Conf. on Industrial and Engineering Applications of AI*, pages 4–10, 1999. (Invited paper) Also appearing in Proc. 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-99), Valencia, Spain, 1-7 (Invited paper). 21

[77] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001. 35

[78] N. R. Jennings and M. Wooldridge. Applications of intelligent agents. In *Agent technology: foundations, applications, and markets*, pages 3–28. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998. 21

[79] C. M. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. In F. J. Garijo and M. Boman, editors, *Multi-Agent System Engineering, Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1647 of *Lecture Notes in AI*, pages 221–232. Springer Verlag, 1999. 37

[80] Lloyd Kamara, Alexander Artikis, Brendan Neville, and Jeremy Pitt. Simulating computational societies. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents World III, Third International Workshop, ESAW 2002, Madrid, Spain, September 16-17, 2002, Revised Papers*, volume 2577 of *Lecture Notes in Computer Science*, pages 53–67. Springer, 2003. 19

[81] Henry Kautz, Bart Selman, and Mehul Shah. Referralweb: Combining social networks and collaborative filtering. *Communications of the ACM*, 40:63–65, 1997. 31

[82] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003. 16

[83] Jeffrey O. Kephart, James E. Hanson, and Amy R. Greenwald. Dynamic pricing by software agents. *Computer Networks (Amsterdam, Netherlands: 1999)*, 32(6):731–752, 2000. 108

[84] P Kollock. The emergence of exchange structures: An experimental study of uncertainty, commitment and trust. *Am J Soc*, 100(2):313–345, 1994. 37

[85] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997. 30

[86] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986. 19

[87] Ken Lang. Newsweeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995. 29

[88] Niklas Luhmann and Niklas. Macht. English Luhmann. *Trust ; and, Power : two works by Niklas Luhmann / [translated from the German by Howard Davis, John Raffan and Kathryn Rooney] ; with an introduction by Gianfranco Poggi.* Chichester [etc.] : Wiley, 1980. Translation of: 'Vertrauen', 2., erw. Aufl. Stuttgart : Ferdinand Enke, 1973 ; and of 'Macht', Stutt. 35

[89] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005. 17, 41

[90] M. Martelli, V. Mascardi, and F. Zini. A logic programming framework for componentbased software prototyping, 1999. 17, 41

[91] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system, a toolkit for building multi-agent simulations, 1996. 17, 41

[92] C. Müller-Schloer and B. Sick. Controlled Emergence and Self-Organization. In Würtz, R. P., editor, *Organic Computing, Understanding Complex Systems*, pages 81–+. 2008. 33

[93] Brendan Neville and Jeremy Pitt. A computational framework for social agents in agent mediated e-commerce. In A. Omicini, P. Petta, and J. Pitt, editors, *Engineering Societies in the Agents World IV*. Springer-Verlag, 2004. 42, 60

[94] Brendan Neville and Jeremy Pitt. A simulation study of social agents in agent mediated e-commerce. In *Proceedings of the Seventh International Workshop on Trust in Agent Societies*, 2004. 42, 60

[95] Brendan Neville, Daniel Ramirez, Lloyd Kamara, and Jeremy Pitt. Computational logic and compliance pervasion. Technical report, Project Deliverable 1.5, EU ALIS Project (IST FP6), 2009. 61

[96] Hyacinth S. Nwana and Martlesham Heath. Software agents: An overview, 1996. 21

[97] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5 - 6):393–408, December 1999. 29

[98] J. Pitt and A. Mamdani. Designing agent communication languages for multi-agent systems. In F. Garijo and M. Boman, editors, *In Multi-Agent System Engineering: Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, volume 1647 of *Lecture Notes in Artificial Intelligence*, pages 102–114. Springer-Verlag, 1999. 20

[99] J. Pitt and A. Mamdani. A protocol-based semantics for an agent communication language. In *Proceedings 16th International Joint Conference on Artificial Intelligence IJCAI'99*, pages 485–491. Morgan-Kaufmann, 1999. 53, 54

[100] Jeremy Pitt, Abe Mamdani, and Patricia Charlton. The open agent society and its enemies: a position statement and research programme. *Telematics and Informatics*, 18(1):67–87, 2001. LocalNets: Environments for community-based interactive systems. 18

[101] Karl Raimund Popper. *The open society and its enemies*. Routledge, London :, 1945. 18

[102] Conte R. and Castelfranchi C. *Cognitive and social action*. Londra: London University College of London Press, 1995. 21

[103] P. Resnick. Beyond bowling together: Sociotechnical capital. In John Carroll, editor, *HCI in the New Millenium (ACM Press)*. 2001. 20

[104] P. Resnick. Impersonal sociotechnical capital, icts, and collective action among strangers. In William Dutton, Brian Kahin, Ramon O'Callaghan, and Andrew Wyckoff, editors, *Transforming Enterprise*. 2004. 20

[105] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM. 30

[106] Pierre-Michel Ricordel and Yves Demazeau. From analysis to deployment: A multi-agent platform survey. In *ESAW '00: Proceedings of the First International Workshop on Engineering Societies in the Agent World*, pages 93–105, London, UK, 2000. Springer-Verlag. 42

[107] S. J. Russell and Norvig. *Artificial Intelligence: A Modern Approach (Second Edition)*. Prentice Hall, 2003. 22

[108] J. Sabater and C. Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of the 4th Workshop on Deception, Fraud and Trust in Agent Societies*, 2001. 37

[109] S. Sen, A. Biswas, and S. Debnath. Believing others: Pros and cons, 2000. 37

[110] Sandip Sen. Reciprocity: A foundational principle for promoting cooperative behavior among self-interested agents. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press, 1995. 37

[111] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. 30

[112] C. Sierra, J. Rodríguez-Aguilar, P. Noriega, M. Esteva, and J. Arcos. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional*, V(4):33–39, 2004. 16

[113] R. Smith and R. Davis. Distributed problem solving: The contract-net approach. In *Proceedings of the 2nd Conference of Canadian Society for CSI*, 1978. 102

[114] Luc Steels. Building agents out of autonomous behaviour systems. In *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*, pages 83–121. Lawrence Erlbaum Associates, 1995. 11, 39, 40, 136

[115] V. Swarup and J. T. Fabrέga. Trust: Benefits, models and mechanisms. In *In Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in Lecture Notes in Computer Science, pages 3–18. Springer, 1999. 143

[116] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 1 edition, May 2005. 30

[117] Jeffrey Travers, Stanley Milgram, Jeffrey Travers, and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969. 31

[118] Klaus G. Troizsch. Social science simulation — origins, prospects, purposes. In Rosaria Conte, Rainer Hegselmann, and Pietro Terna, editors, *Simulating Social Phenomena*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 41–54. Springer-Verlag, 1997. 39

[119] Roy M Turner and Roy M. Turner. The tragedy of the commons and distributed ai

systems. In *in Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 379–390, 1993. 35

[120] W. Vasconcelos, D. Robertson, C. Sierra, M. Esteva, J. Sabater, and M. Wooldridge. Rapid prototyping of large multi-agent systems through logic programming. *Annals of Mathematics and Artificial Intelligence*, 41(2-4):135–169, 2004. 17, 41

[121] Max Weber. Basic concepts of sociology, 1962. 21

[122] M. Witkowski, A. Artikis, and J. Pitt. Experiments in building experiential trust in a society of objective-trust based agents. In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber Societies*, LNAI 2246, pages 110–132. Springer, 2001. 37

[123] M. Witkowski, A. Artikis, and J. Pitt. Experiments in building experiential trust in a society of objective-trust based agents. In R. Falcone, M. Singh, and Y.-H. Tan, editors, *Trust in Cyber Societies: Integrating the Human and Artificial Perspectives*, number 2246 in Lecture Notes in Artificial Intelligence, pages 111–133. Springer-Verlag, 2001. 99

[124] Mark Witkowski, Brendan Neville, and Jeremy Pitt. Agent mediated retailing in the connected local community. *Interacting with Computers*, 15(1):5–32, 2003. 81, 102, 137

[125] M. J. Wooldridge and N. R. Jennings. Agent theories, architectures and languages: A survey. In M. J. Woolridge and N. R. Jennings, editors, *94 Workshop on Agent Theories Architectures and Languages*, pages 1–32, 1994. 21

[126] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995. 21, 22

[127] XrML. Xrml: Extensible rights mark-up language, 2002. 143

[128] Yao and P. J. Darwen. How important is your reputation in a multi-agent environment. In *IEEE Conference on Systems, Man, and Cybernetics*. IEEE Press, 1999. 37

[129] Bin Yu and Munindar P. Singh. Towards a probabilistic model of distributed reputation management. In *The Fourth Workshop on Deception, Fraud, and Trust in Agent Societies*, pages 125–137, 2001. 37