Edinburgh Napier UNIVERSITY

Applying Semantic Technologies to Multi-Agent Models in the Context of Business Simulations

Thomas Farrenkopf

A thesis submitted in partial fulfilment of the requirements of Edinburgh Napier University, for the award of Doctor of Philosophy

January, 2017

Abstract

Agent-based simulations are an effective simulation technique that can flexibly be applied to real-world business problems. By integrating such simulations into business games, they become a widely accepted educational instrument in the context of business training. Not only can they be used to train standard behaviour in training scenarios but they can also be used for open experimentation to discover structure in complex contexts (e.g. complex adaptive systems) and to verify behaviours that have been predicted on the basis of theoretical considerations.

Traditional modelling techniques are built on mathematical models consisting of differential or difference equations (e.g. the well-known system dynamics approach). However, individual behaviour is not visible in these equations. This problem is addressed by using software agents to simulate individuals and to model their actions in response to external stimuli.

To be effective, business training tools have to provide sufficiently realistic models of real-world aspects. Ideally, system effects on a macroscopic level are caused by behaviour of system components on a more microscopic level. For instance, in modelling market mechanisms market participants can explicitly be modelled as agents with individual behaviour and personal goals. Agents can communicate and act on the basis of what they know and which communication acts they perform. The evolution of the market then depends on the actions of the participants directly and not on abstract mathematical expressions.

Generally, agent-based modelling is a challenging task, when modelling knowledge and behaviour. With the rise of the so-called semantic web ontologies have become popular, allowing the representation of knowledge using standardised formal languages which can be made available to agents acting in a simulation. However, the combination of agent-based systems with ontologies has not yet been researched sufficiently, because both concepts (web ontology languages and agent oriented programming languages) have been developed independently and the link has not yet been built adequately.

Using ontologies as a knowledge base allows access to powerful standardised inference engines that offer leverage for the decision process of the agent. Agents can then determine their actions in accordance with this knowledge. To model agents using ontologies creates a new perspective for multi-agent simulation scenarios as programming details are reduced and a separation of modelling aspects from coding details is promising as business simulation scenarios can be set up with a reduced development effort.

This thesis focuses on how ontologies can be integrated utilising the agent framework Jadex. A basic architecture with layered ontologies and its integration into the belief-desire-intention (BDI) agent model is presented. The abstract level of the approach guarantees applicability to different simulation scenarios which can be modelled by creating appropriate ontologies. Examples are based upon the simulation of market mechanisms within the context of different industries. The approach is implemented in the integrated simulation environment AGADE which incorporates agent-based and semantic technologies. Simulations for different scenarios that model typical market scenarios are presented.

Keywords: business simulation, ontology, BDI agents, business game.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Neil Urquhart and Michael Guckert for the continuous support of my PhD study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my PhD study.

Besides my advisors, I would like to thank the rest of my thesis committee: David Benyon, Jyoti Bhardwaj and Simon Wells for their insightful comments and encouragement, but also for the hard questions which invented me to widen my research from various perspectives.

I thank my friends for the stimulating discussions in the following institution Technische Hochschule Mittelhessen, Germany.

Last but not least, I would like to thank my family for their support in general and especially a heartfelt thanks goes out to my girlfriend for all your love, support and patience when I spent my free time and weekends away from her and was only thinking about strange OWLs :-)

Contents

List of Acronyms **x**

List of Algorithms **xiii**

List of Figures **xiv**

List of Tables **xvii**

1 Introduction **1**

1.1 Motivation 3

1.2 Problem Statement 4

1.3 Objectives 5

1.4 Research Questions 7

1.5 Summary of Contribution 8

1.6 Outline of the Thesis 9

2 Background **11**

2.1 Multi-Agent Systems 11

2.2 Ontologies 13

2.3 Web Ontology Language 16

2.3.1 Introduction 16

2.3.2 Web Ontology Language (OWL)-Sublanguages 16

2.3.3 Basic Elements of OWL DL 18

2.3.4 Modelling Standards for Applied Ontologies 21

2.3.5 Open-World Assumption 22

2.4 System Dynamics 24

2.5 Summary of Chapter 2 27

3 Literature Review 28

3.1 Business Games 28

3.2 Approaches for Multi-Agent Based Simulation 31

3.2.1 Computational Economics 31

3.2.2 Micro-Economic Models 33

3.2.3 Monte Carlo Simulations 35

3.3 Review of Relevant Work Involving Multi-Agent Systems and Semantic Technologies 36

3.3.1 Ontologies in Multi-Agent Systems 36

3.3.2 Ontology Alignment 37

3.3.3 Semantic Technologies Applied to Agent-Models . . . 38

3.3.4 Overview of Available Frameworks 43

3.3.4.1 JADE 46

3.3.4.1.1 Architecture 46

3.3.4.1.2 Modelling Behaviour 47

3.3.4.1.3 Ontology Support 48

3.3.4.2 MadKit 50

3.3.4.2.1 Architecture 51

3.3.4.2.2 Modelling Behaviour 52

3.3.4.2.3 Ontology Support 53

3.3.4.3 FIPA-OS Toolkit 53

3.3.4.3.1 Architecture 53

3.3.4.3.2 Modelling Behaviour 55

3.3.4.3.3 Ontology Support 56

3.4 Conclusion Drawn from Literature Review 56

4 Social Structures and Marketing Theory 58

4.1 Social Structures 58

4.1.1 Small-World Scale-Free Networks 59

4.1.2 Preferential Attachment 61

4.2 Stimulus-Response Model 65

4.3 Opinion Leadership and Buying Influence 69

4.4 Personal Preferences 71

4.5 Summary of Chapter 4 73

5 Agents and Ontologies 74

5.1 Fundamentals 74

5.1.1 Agent Design Methodology 74

5.1.2 Rules 76

5.2 Incorporating Ontologies into BDI-Agents 79

5.3 Layered-Ontology Model 82

5.4 Communication and Learning Capability 85

5.5 Summary of Chapter 5 88

6 Agent Software Framework 90

6.1 Ontology-based Business Simulations Framework 90

6.2 Architecture 91

6.2.1 Distributed Environment 92

6.2.2 Abstract OWL Agent 95

6.2.3 Ontology-based Plan Selection 97

6.2.4 Complex Calculations 101

6.2.5 Ontology Query Languages 103
6.2.5.1 SPARQL 104
6.2.5.2 SPARQL-DL 105
6.2.5.3 DL Query 106
6.2.5.4 OWL API 107
6.2.5.5 Benchmarking 107
6.2.6 Round-based Management 109
6.3 AGADE Overview 110
6.4 Summary of Chapter 6 115

7 Results and Validation 117
7.1 Mobile Phone Market Scenario 117
7.1.1 Opinion Leadership 118
7.1.1.1 Summary 118
7.1.1.2 Mobile Phone Market Ontology 118
7.1.1.3 Parameters and Results 123
7.1.2 Personal Preferences 126
7.1.2.1 Summary 126
7.1.2.2 Mobile Phone Market Ontology 126
7.1.2.3 Parameters and Results 130
7.2 Fuel Market Scenario 136
7.2.1 Summary 136
7.2.2 Fuel Market Ontology 140
7.2.3 Parameters and Results 145
7.3 Business Game Experiments 148
7.3.1 Summary 149

7.3.2 Results and Discussion 152
7.4 Summary of Chapter 7 154

8 Conclusions and Future Research 155
8.1 Conclusion on Ontology Based Business Simulations 155
8.1.1 Research Question 1 156
8.1.2 Research Question 2 157
8.1.3 Research Question 3 158
8.2 Future Work on Ontology Based Business Simulations 160

Appendix A Published Work 161

Appendix B Description Logic Symbols 163

Appendix C Ontology Modelling Notation 164

Appendix D Certificate 168

Appendix E RMI vs. Web Services 169

Appendix F Calculation of Happiness Value 172

Bibliography 173

List of Acronyms

ACE	Agent-Based Computational Economics
ACC	Agent Communication Channel
ACL	Agent Communication Language
ADL	Abstract Domain Layer
AGADE	Agile Agent Development Environment
AID	Agent Identifier
AMS	Agent Management System
API	Application Programming Interface
ARCOL	ARTIMIS Communication Language
BDI	Beliefs-Desires-Intentions
CE	Computational Economics
CORBA	Common Object Request Broker Architecture
CT	Container Table
CWA	Closed World Assumption
DAML	DARPA Agent Markup Language
DF	Directory Facilitator
DL	Description Logic
FIPA	Foundation for Intelligent Physical Agents
FOL	First-Order Logic
GADT	Global Agent Descriptor Table
GPL	GNU General Public License
GUI	Graphical User Interface
IDL	Individual Domain Layer
IRI	Internationalised Resource Identifier

IIOP	Internet Inter-ORB Protocol
LADT	Local Agent Descriptor Table
LGPL	GNU Lesser General Public License
IDE	Integrated Development Environment
JADE	Java Agent Development Framework
KIF	Knowledge Information Format
KR	Knowledge Representation
KQML	Knowledge Query and Manipulation Language
LNAI	Lecture Notes in Artificial Intelligence
LNCS	Lecture Notes in Computer Science
MAS	Multi-Agent System
MATES	Multi-agent System Technologies
MIT	Massachusetts Institute of Technology
MTS	Message Transport Service
ODM	Ontology Definition Metamodel
OIL	Ontology Inference Layer
OWA	Open World Assumption
OWL	Web Ontology Language
PAAMS	Practical Applications of Agents and Multi-Agent Systems
RDF	Resource Description Framework
RMI	Remote Method Invocation
RuleML	Rule Markup Language
SDL	Specific Domain Layer
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
S-R	Stimulus-Response

SWRL	Semantic Web Rule Language
TILAB	Telecom Italia Lab
UML	Unified Modelling Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

List of Algorithms

1 Preferential attachment. Build Adjacency Matrix Function . 64

2 Preferential attachment. Pick Function 65

3 S-R-Agent 66

4 Simple BDI agent reasoner 80

5 Add information to knowledge base 87

6 Run next agent actions 101

7 Opinion leadership plan. 123

8 Update happiness value 172

List of Figures

1 The agent-environment interaction 11
2 OWL ontology example 20
3 OWL/relational structural analogies 23
4 Simple human population feedback loop 25
5 Equation view of a Vensim model 26
6 Result of running feedback loop simulation 26
7 Example of emergent phenomenon 33
8 Ontology-based communication model 42
9 JADE Architecture 46
10 Content reference model 49
11 Jade support for content languages and ontologies 50
12 Aalaadin core model 51
13 MadKit architecture 52
14 FIPA-OS architecture. 54
15 FIPA-OS task model 56
16 Sociogram 59
17 Random versus scale-free graphs 60
18 Power-law graph 61
19 Preferential attachment 62
20 Linear Decision-Making process 66
21 Agent decision-making process 68
22 A demonstrative conversation between Steve and Bill 68
23 Calculation of of points for comparison 72

24 A high-level abstraction of a Beliefs-Desires-Intentions (BDI) agent 75
25 OWL-BDI-Mapping 81
26 Small example of OWL / BDI agent model 82
27 Three-layered ontology 83
28 Three-layered ontology architecture example 84
29 Inconsistency ontology example 88
30 Abstract view on the AGADE architecture 91
31 AGADE nodes model 94
32 AbstractOWLAgent Java class 96
33 Interaction of layered ontology and agent classes 97
34 Ontology-controlled plan selection 99
35 An example ontology used for the query variants 103
36 Four phases simulation 110
37 AGADE start new simulation step I 111
38 AGADE start new simulation dialog step II 112
39 Adjacency matrix setup 113
40 AGADE simulation GUI 113
41 Current ontology state of an participating agent 115
42 Sub-concepts of the mobile phone market SDL 119
43 Rudimentary mobile phone market IDL 122
44 Phone distribution after 100 rounds 124
45 The rate of adoption of Tetracycline by doctors 125
46 Calculation of scores of phone attributes 128
47 Brand distribution chart after 100 rounds 135
48 Sub-concepts of the biofuel market SDL. 141

49 Consumer IDL Biofuel 142
50 Seller IDL Biofuel 142
51 Biofuel adoption curve 147
52 Unit sales volume market share 148
53 AGADE user input dialog 150
54 Results of the questionnaire 153
55 IBM Award of Scientific Excellence 168

List of Tables

1 FIPA-ACL Message Parameters 41
2 Comparing Matrix of Agent-Frameworks 45
3 Benchmark test results 108
4 Properties relevant for the comparison 134
5 Brand distribution in survey compared with brand distribution in AGADE after 100 rounds 135
6 Survey questions 152
7 Basic logic symbols 163
8 List of modelling notations 164

1 Introduction

Traditionally, business simulation is used for a wide range of business purposes such as business training, education, analysis of a particular business situation and decision making [1]. Classically, the models used have been equation-based and modelled with cause-and-effect relationships which more or less ignore the role of individuals and their behaviour. Nevertheless, crowd behaviour is formed by the individual behaviour of the members of the crowd and therefore depends on effects that can be ascribed to the crowd itself and interactions between its members. Software simulations of these phenomena are common in the field of sociology and related disciplines, i.e. simulations are essential for understanding complex systems. Markets can be treated as complex systems. The focus of a complex system model is on representing the interactions among the elements, which in an economic model involves agents (individuals) and institutions (firms, regulatory agencies, etc.).

A paradigm not yet extensively applied to business simulations is the agent-based approach ([2] and [3]), even though this technique provides a natural description of an economic environment where simulated human beings are modelled as agents, interacting with some of their peers as well as with their environment. Agent-based systems are perfectly suited to represent individuals rather than describing effects through equations and thresholds.

However, realistic behavioural patterns can only be achieved by appropriately representing knowledge and intelligence. Agents must be able to act according to what they know and must be able to learn. With the rise of the semantic web [4], ontologies became popular and we now have standardised

formal languages to represent knowledge which can thus be made available to agents acting in a simulation.

In computer science, an ontology is a formal representation of knowledge by means of a set of concepts within a domain and the relationships between those concepts [5]. Using ontology languages such as OWL[1], knowledge can be represented in a formal standardised way. The history of OWL dates back to the 1990s: A number of research efforts were set up to explore how the idea of knowledge representation as it was used in the area of artificial intelligence could be used on the Web to make machines understand content. One particular project in this field named DARPA Agent Markup Language (DAML) was started in late 1990s, with the goal of creating machine-readable representation for the Web. The main outcome of the project was an agent markup language based on Resource Description Framework (RDF) – which is a general-purpose language for representing information on the web (see [6]) – named DAML [7]. OWL started as a research-based revision of DAML and Ontology Inference Layer (OIL) and has become a formal W3C recommendation in 2004 (see [8]).

Ontologies for agents can be used to represent knowledge such as knowledge about an agent's external environment such as knowledge about products, knowledge about personal preferences, knowledge about relations, etc. In multi-agent systems, ontologies are gradually becoming more popular for describing objects involved in the agents' interactions. Nevertheless, the

[1]The natural initialism for Web Ontology Language would be WOL instead of OWL. The story dates back to December 2001, the days when the OWL group was working on OWL. Tim Finin, in an e-mail dated back on 27 December 2001, suggested the name OWL based on these considerations: OWL has just one obvious pronunciation that is also easy on the ear; it yields good logos, it suggests wisdom, and it can be used to honour the *One World Language* project, an artificial intelligence project at MIT in the mid-1970s (see `http://lists.w3.org/Archives/Public/www-webont-wg/2001Dec/0169.html`)

combination of agent-based systems with ontologies for running business simulations has not yet been researched sufficiently, because both concepts (web ontology languages and agent oriented programming languages) have been developed independently and the link has not yet been built.

This research is being undertaken with the aim of building stronger connections between semantic technologies and multi-agent systems. Agents and their knowledge will be modelled by means of ontologies. Models that incorporate individual behaviour into business simulations are considered. The generic approach will allow the simulation of different scenarios. The representation of individuals in simulations with socio-cultural and personal patterns can then lead to more realistic market simulations in business games.

1.1 Motivation

Business games are an important tool when teaching students the consequences of decision making within a business context. In a business game environment the player typically suggests a plan of activity, through making a set of decisions e.g. producing a certain quantity of products, with a given set of features, a pricing strategy or a marketing budget. Such decisions are governed by constraints within the game scenario such as budget or production capacities. The business game then evaluates the input of the player and presents feedback detailing the effect of the decision. Such feedback will typically be an estimate of the quantity of products sold, the market share and the resulting profit/loss. The players continue to interact "turn about" with this model over a number of iterations. Typically, the model attempts to estimate likely sales figures for the product portfolios offered

by the companies under simulation. Classically, business games are created with traditional modelling techniques like the well-known system dynamics approach. In a system dynamics approach the system dynamics modeller does not program the behaviour of individual agents. Instead, he has to model how populations of agents behave as a whole. System Dynamics models are highly sensitive to their settings and small changes can have significant effects on the behaviour of the system. The modeller has to calibrate parameter settings with time costly experiments to adjust the model. He cannot make use of knowledge about individuals directly as this is not part of the controls of the system. In contrast, appropriate agent-based models are based on descriptions of individuals and their behaviour. Therefore, agent based models and especially multi agent models are a more natural tool to model complex systems which depend on individuals rather than approaches in which the effect of the system is described – in a sense indirectly – with equations and thresholds. Using agents in business simulations, there is a necessity to represent the knowledge base of an agent in an appropriate way. Ontologies allow access to powerful standardised inference engines that offer leverage for the decision process of the agent. This thesis will concentrate on building a stronger connection between semantic technologies and multi-agent systems for business simulations.

1.2 Problem Statement

Simulation of real-world scenarios in business games is an important application of complex adaptive social systems. According to Holland, complex adaptive systems can be defined as "... systems that have a large numbers of components, often called agents, that interact and adapt or learn." [9]

These complex adaptive systems will gain more and more importance for business simulations to describe and explore complex structures e.g. organisations or markets and therefore the science of complexity will become one of the core disciplines of the future. While many aspects of complex systems might be put into sets of difference or differential equations (see Forrester in his System Dynamics approach [10]) or other methods that emphasise quantitative aspects (see [11]). Agents provide a natural instrument to model complex dynamic systems as they allow direct modelling of components and interactions. A restructuration of the representational infrastructure of complex social systems has occurred as Wilensky has put forth [12] and this will in consequence have an impact on the quality of business simulations.

1.3 Objectives

The effectiveness of a business game as an educational tool depends entirely on its feasibility in real-world scenarios and on the feedback provided by the game which should both appear as realistic as possible (see [13] and [14]). The more realistic the model the more valuable the educational experience. In specific simulation scenarios, the business game involves specifying, producing and pricing a consumer product which is then released into a marketplace for sale. Products that are desirable will outsell those that are perceived as less attractive. It is proposed that a realistic simulation of consumer marketplaces can be achieved by the use of agents to model consumer behaviour and communication.

According to classical definitions, agent-based models consist of agents that are autonomous computational individuals interacting with each other and their environment. They are active as well as reactive: external stimuli,

internal state and available information are used to determine actions an agent performs to reach given goals. That is why agents are particularly well suited to represent social actors in complex environments which we use as a basis for business game scenarios.

A conceptual framework that guides agent activities is the BDI (Beliefs-Desires-Intentions) model. The BDI model is a well-established paradigm for the development of agents [15]. A BDI agent has knowledge about its world (beliefs) and pursues goals (desires) while following given strategies (intentions). Agents must be able to act according to their knowledge of their internal and external world which must therefore be expressed in an appropriate format. Standardised formal languages to represent knowledge are available now and can be used to model the knowledge of an agent using ontologies. They can be used to equip agents acting in a simulation with personalised knowledge and learning capability. The idea of supporting communities of agents with ontologies which describe the objects involved in the interactions of the agents was originally formulated by Gruber [16] and subsequently been discussed by various other authors e.g. [17]. Currently, frameworks mostly just perform syntactic matchings to detect analogies between data exchanged during the communication process of a simulation [17, pp. 1–35]. The research presented here aims at building stronger connections between semantic technologies and multi-agent systems for business simulations. A strong and well-designed integration of semantic technologies and multi-agent systems can have a significant effect on the quality of multi-agent simulations.

The overarching objectives of the research are:

- creating simulations with semantically advanced technology in the context of business games.
- mapping BDI agents flexibly to corresponding OWL ontologies.
- creating simulations that can support running scenarios based on real individual behavioural patterns.

1.4 Research Questions

Again, extending an agent programming environment with the reasoning power of ontologies can have a significant impact on the ways agent oriented programming and business simulations work. The practical integration of such technologies for developing business simulations is still a challenge. In fact, extending agent programming languages with the ability of a transparent use of ontologies with a focus on the perspective of real market based simulations is not yet described in the literature. The aim to create ontology based business simulations with various scenarios, where plans are carefully written to use ontological descriptions which have effects on the agents' beliefs, leads to the following main research questions which will be answered in this thesis:

- To what extent can software agent methodologies be utilised in the context of a business game to enhance realistic gaming behaviour?
- How can ontologies handle and influence the activities of BDI agents?
- How can ontologies be adapted to different simulation scenarios?

1.5 Summary of Contribution

The contributions contained within this thesis may be summarised as:

A community of agents for the simulated market place. By using a complex structure with a large number of agents making individual decisions based on local criteria and the influence of other local agents the responses should be sufficiently complex to model market-place behaviour, but should not encourage the students to learn the game rules to achieve goals, but to concentrate on making realistic business decisions.

Supplying agents with ontologies in modelling simulation scenarios. A major benefit of supplying agents with ontologies in modelling simulation scenarios is a shift in perspective that separates modelling aspects from coding details and focuses on the model rather than on programming details. This increases efficiency and leads to a clearer separation of concerns.

A layered-ontology model. It allows agents to share knowledge and create a basic common understanding of their environment while enabling reuse of fundamental concepts. Ontological commitment is ensured without which communication between the agents would not be possible. Overall individual behaviour and decision processes can be modelled and used in simulations for different scenarios. This creates a new perspective for multi-agent simulation scenarios by modelling agent knowledge with the help of semantic technologies.

A framework that incorporates semantic technologies and network analysis algorithms. The framework called AGADE (Agile Agent Development Environment) shows a practical integration of a multi-agent based system and semantic technologies for a realistic modelling of individuals participat-

ing in a dynamic market environment, where plans are carefully written to use ontological descriptions.

1.6 Outline of the Thesis

The remainder of the thesis is set out as follows:

Chapter 3: Literature Review.

This chapter gives a review of published work that is relevant to the research contained within this thesis.

Chapter 4: Social Structures and Marketing Theory.

A brief overview of marketing theories is given and behavioural patterns relevant for the thesis are discussed. Selected marketing mechanisms are explained and the scenarios (e.g. opinion leadership) are introduced.

Chapter 5: Agents and Ontologies.

Theoretical foundations of multi-agent systems and ontologies are summarised. An integration pattern for ontologies and agent frameworks following the BDI paradigm is presented. A blueprint is then created of a layered ontology architecture that makes the concept universally applicable.

Chapter 6: Agent Software Framework.

The agent-based framework called Agile Agent Development Environment (AGADE) is introduced in this chapter in which the concept of agents with ontology support is implemented. It is a highly configurable tool that can run BDI agent simulations where agents communicate with each other, have knowledge of their environment and act in structured social environments.

Chapter 7: Results and Validation.

This chapter describes the process of applying the underlying AGADE mechanisms to simulation scenarios. Results obtained using AGADE are compared with simulation results published earlier.

Chapter 8: Conclusions and Future Research.

This chapter draws final conclusions from the thesis and proposes topics to be discussed in future work.

2 Background

In this chapter, the basic formalisms and technologies underlying the work in this thesis are introduced. Apart from giving background information, it will also touch on the standardised OWL language in the area of ontologies. Expert readers can skip (parts of) this section and continue directly to Chapter 3.

2.1 Multi-Agent Systems

Before discussing *multi-agent systems (MAS)*, the concept of an agent has to be defined. An agent is an autonomous software entity which observes an environment, reacts to impulses (internal or external) and acts independently within a certain setting ([18]).

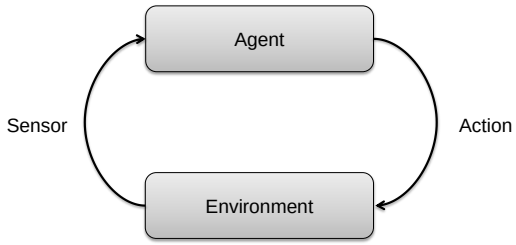


Figure 1: The agent-environment interaction.

Fig. 1 shows an abstract representation of an agent in relation to its environment [19] which follows the definition of Russell and Norvig [20]. An agent receives information about its environment through sensors. The information is then processed and may cause the agent to perform an action in its environment. Agents can focus their activities on achieving given goals. Literature also uses the term *intelligent agent* or *software agent*. Both terms are interchangeable. An intelligent agent can use existing

knowledge and can acquire new knowledge (i.e. learn) while pursuing goals. Intelligent agents are used in various research areas such as artificial intelligence, information retrieval, database and knowledge-base systems, and distributed computing. Because researchers use agents with different goals and therefore have different requirements, there is no single generally accepted definition of an agent. The most commonly used and cited definition follows Wooldridge et al.: "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 delegated objectives." [18]

Generally speaking, agents are autonomous parts of a software application that show independent reactive and proactive behaviour. The following definitions describe the basic characteristics of agents adapted from Wooldridge et al.:

Autonomy

Agents act independently without direct control through external interventions. They control their own actions.

Social ability

Agents interact with other agents through some kind of *language* (such as KQML and FIPA ACL). They also communicate with users to answer questions and to report on their own work progress.

Reactivity

Agents can perceive their environment and respond to their perceptions.

Proactivity

Agents can take the initiative to operations and can act proactively to achieve their objectives. They are able to perform goal-directed actions.

Multi-agent systems are systems which typically contain numerous interacting agents within an environment. Multi-agent systems allow modelling complex situations in which agents mutually can have common, conflicting or even contradictory goals. They exhibit unexpected, emergent behaviour as a result of the complexity of agent behaviours and their interactions. Agents may act in complex situations, e.g. as participants in markets. For this purpose, agents may decide to cooperate for mutual benefit or to egoistically pursue and achieve their own goals.

2.2 Ontologies

Knowledge representation (KR) and reasoning aim to have machine-interpretable representations of the world and to allow the drawing of conclusions, similar to humans [21]. According to Davis et al. [22]:

> "A KR is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e., by reasoning about the world rather than taking action in it. [...] It is a set of ontological commitments, i.e., an answer to the question: In what terms should I think about the world? [...]
>
> A KR is a Medium of Human Expression: knowledge representations are also the means by which we express things about the world, the medium of expression and communication in which we tell the machine (and perhaps one another) about the world. This role for representations is inevitable so long as we need to tell the machine (or other people) about the world, and so long as we do so by creating and communicating representations. [This] role for knowledge representations is thus as a medium of expression and communication for use by us."

Currently, knowledge representation appears in different forms, the most prevalent of which are based on semantic networks, rules and logic. The use

of ontologies serves two purposes in the approach: according to Gruber's classic postulate [23] it defines a common vocabulary for communication and beyond that it allows an abstraction of certain elements of the simulation. For example, simulating market mechanisms with typical actors is structurally equivalent for different markets. They differ in detail only e.g. in the goods being traded and specific behavioural patterns.

Referring to the statement that "agents must be able to act according to their knowledge of their internal and external world which must therefore be expressed in an appropriate format" the characteristics of ontologies are summarised here: ontologies are used to declare facts (i.e. structural knowledge) and to define how conclusions can be drawn (i.e. inference knowledge) by using available reasoning instruments. Formally, an ontology $\mathcal{O}$ is a triple $(\mathcal{C}, \mathcal{R}, \mathcal{I})$ where $\mathcal{C}$ is a set of concepts, $\mathcal{R}$ a set of relations, and $\mathcal{I}$ a set of individuals. Concepts formally denote subsets of individuals. From a perspective of formal logic such subsets of individuals are the extension of concepts itself while concepts define the intentional representation of the corresponding sets of individuals. An individual that belongs to a concept is called an instance of that concept. The elements of $\mathcal{R}$ are relations (also called roles or object properties, as they manifest the linking on the level of the individuals) having subsets of $\mathcal{C}$ as domain and range. The extension of a role is then a set of pairs (c, d) with $c, d \in \mathcal{I}$. Additionally, individuals can have data properties through which they get linked to primitive data e.g. strings or numbers. Typically, ontologies are formulated by means of description logics with differing levels of expressiveness [24]. Usually, description logics are proper subsets of first order logic where expressiveness has been traded for decidability. Inference knowledge is implicitly given by

the underlying mechanisms of the available reasoning instruments (see [25] and [26]). The following shows some typical constructors using description logic. Let the ontology consists of concepts *Human*, *Male*, *Female* and *Doctor*. Assume that the concept of "A man that is married to a female and has at least two children, all of whom are doctors" has to be defined. This concept can be described with the following concept description (see appendix B for basic description logic symbols):

$$Human \sqcap \neg Female \sqcap \exists married.\exists Female \sqcap (\geqslant 2 hasChild) \sqcap \forall hasChild.Doctor$$

This description uses the intersection ($\sqcap$) constructor, the negation ($\neg$) which is interpreted as set complement, as well as the existential restriction constructor ($\exists R.C$), the value restriction constructor ($\forall R.C$), and the number restriction constructor ($\geqslant nR$). Let an individual, Alice, belongs to $\exists married.\exists Female$ if there exists an individual that is married to Alice (i.e., is related to Alice with the *married* property) and is a male person (i.e., belongs to the concept *Male*). Similarly, Alice belongs to ($\geqslant 2 hasChild$) if she has at least two children, and she belongs to ($\forall hasChild.Doctor$) if all her children (i.e., all individuals related to Alice using the *hasChild* property) are doctors.

As OWL is a well accepted standardised general knowledge representation language for formulating *ontologies*, OWL was the language of choice (see the following section).

2.3 Web Ontology Language

2.3.1 Introduction

OWL has been standardised by the World Wide Web Consortium (W3C) as an ontology language finding increasing acceptance in various areas since the 2000s [27]. In 2009, the W3C OWL Working Group published an extension and revision of the first standardisation. The previous version has been referred to as OWL 1[2]. Languages for modelling knowledge face two principal problems. The first is the problem of complexity, i.e. evaluating language expressions is potentially exponential in time and space. The second problem is that of undecidability, i.e. first order logic expressions can be formulated that can neither be evaluated to be true nor to be false. Obviously, language expressiveness and efficiency are concurrent goals. Particular emphasis in the development of OWL was laid on striking a balance between expression strength and efficient and scalable inference algorithms. There are currently three sub-languages of OWL available for modelling: OWL *Full*, OWL *Description Logic (DL)* and OWL *Lite*. OWL Lite is a sub-language of OWL DL, which in turn is a sub-language of OWL Full: OWL Full $\supseteq$ OWL DL $\supseteq$ OWL Lite.

2.3.2 OWL-Sublanguages

In the following the most important characteristics of each species of OWL are listed [27]:

OWL Full

[2]Whenever it is necessary to differentiate between the version number of OWL in the following text, it will be specified.

- equals full first-order logic FOL
- is very expressive
- is undecidable
- has only limited support by current software tools

OWL DL

- is a sub-language of OWL Full
- is decidable
- is almost completely supported by current software tools
- NExpTime[3]

OWL Lite

- is a sub-language of OWL DL and thus also of OWL Full
- is decidable
- is least expressive
- ExpTime[4] complexity in the worst case scenario

After becoming a W3C Recommendation, OWL is becoming increasingly widely accepted and used [29]. Particular worthy of mention is that OWL DL is decidable by its design. This means that the question whether or not a statement can be inferred from a set of facts within an ontology can be answered by an algorithm whose termination is guaranteed. In practise, there are only few constructs that cannot be modelled with the use of OWL DL and those constructs are normally only necessary for a rather limited purpose. For example, a resource which is not allowed in DL is a resource

[3]NExpTime or NEXP and EXPTIME or EXP are complexity classes of decision problems that can be accepted by a nondeterministic Turing machine in a limited time complexity in the worst case scenario [28, pp. 101-104]
[4]See previous footnote

which is simultaneously a class and an individual [30]. These and similar restrictions are accepted to ensure the decidability. OWL Lite, however, has very limited expressive power and therefore only bears minor importance i.e. OWL Lite has a limited notion of cardinality for properties, to be explicit, the only cardinalities allowed are 0 or 1, etc. (see [30]). The OWL 2 document does not list OWL Lite anymore and states that all OWL Lite ontologies are OWL 2 ontologies. In addition, OWL 2 defines three sub-languages based on OWL DL as profiles that allow the expressive power to be tailored in order to balance the efficiency of reasoning - *OWL 2 EL*, *OWL 2 QL* and *OWL 2 RL*. This reduction in expression level is accepted in order to achieve optimum calculation times for reasoning in specific applications.

2.3.3 Basic Elements of OWL DL

The basic building blocks of an OWL DL ontology are [31]:

Namespaces

Each ontology that contains knowledge about objects has its own namespace for identifying the unique ontology. The namespace is identified by an Uniform Resource Identifier (URI).

Classes

Classes are interpreted as *sets* that contain individuals with certain properties. For example, class *Person* defines the characteristics of every object in a given universe of discourse that is considered to be an element of the set that corresponds to the concept *Person*. The set is the extension of the class, while the class is the intention of the set. Classes may be organised into a superclass-subclass hierarchy, which leads to a named taxonomy. Subclasses specialise (are subsumed by)

their superclass. Classes represent an important part of the metadata as they are the basic building blocks of an ontology.

Object Properties

Object properties are relations defined by the set of classes (and possibly by the set of individuals) in ontologies. They are defined by classes and link individuals. For example, the property *hasSibling* defined by the class *Person* may link the individuals *Smith* and *Jones*. Object properties cannot be represented by means of the taxonomy alone. Like attributes, these object properties are also inherited along with the taxonomy. Object properties can have an inverse relationship, i.e. the inverse of *hasOwner* is *isOwnedBy*.

Datatype Properties

Datatype properties describe relationships between individuals and data values. OWL uses the facilities of the standardised *XML Schema Datatypes* [32]. For example, a datatype property for a person may be *first name* or *age* which links to date of the type *xsd:string* or *xsd:integer* respectively. Further, it is possible to assign ranges for datatypes, i.e. age with range int$[>= 20, < 99]$.

Instances or Individuals

Instances are concrete expressions in which attributes have concrete values. They are often referred to as facts or individuals. In combination with the ontology they represent the knowledge base.

Rules

Rules extend an ontology with special relations that occur only under clearly definable circumstances. An often-used example is the ancestral relationship: If a father of a child has a brother, then the brother is the

uncle of the child. By using rules it is possible to create such relations between concepts that are unique for specifying circumstances, such as the ancestral relationship which exists only in a part of the existing instances.

Reasoner

Reasoners are able to automatically draw conclusions based on the OWL expressions of the ontology by inferring logical consequences from a set of explicitly asserted facts or axioms. Reasoners typically provides automated support for reasoning tasks such as classification and consistency checks [33].

Fig. 2 shows a simplified example of the basic building blocks described above, where the element with a yellow circle is a class, and those with dark diamonds are instances.

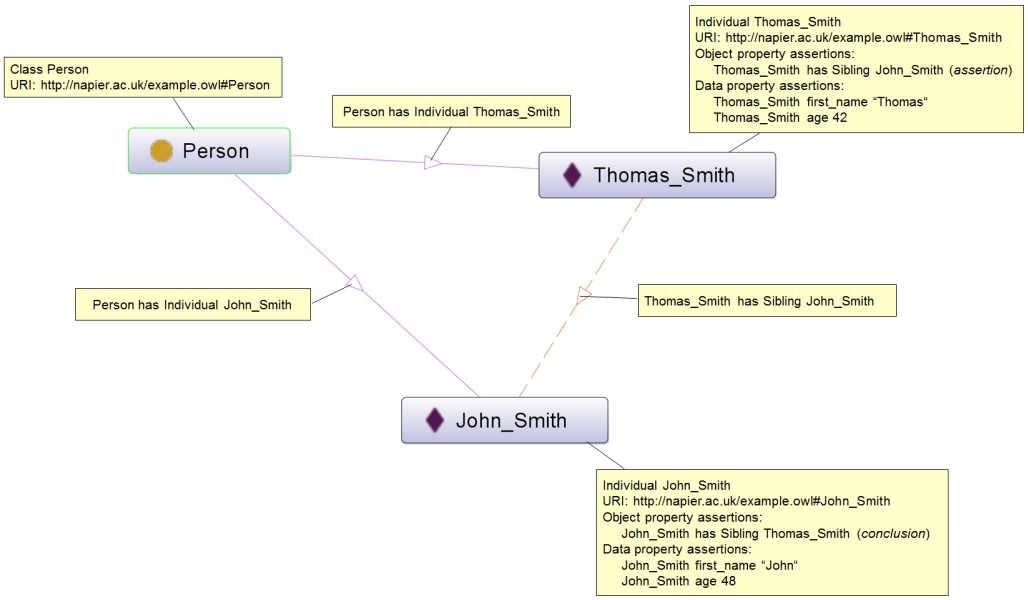


Figure 2: OWL ontology example.

2.3.4 Modelling Standards for Applied Ontologies

The Unified Modelling Language (UML) is a standardised general-purpose tool for software engineering which aims to be a standard modelling language which can model concurrent and distributed systems [34]. Cranefield et al. published various papers about the combination of UML, agent messages and ontologies. They discussed the use of UML for representing instance information within agent messages and presented technologies to support knowledge representation [35]. In further work, they proposed ontologies for interaction protocols where ontologies used for describing the input and output data are processed during the protocols execution, together with the actions and decisions that agents must perform. Cranefield et al. follow the aim that an ontology would then be able to interpret any interaction protocol which is defined in an ontology [36]. As an example, Cranefield et al. prove the suitability of UML to model agent-based systems in a travel booking scenario [37].

The paper *"Is it an ontology or an abstract syntax?"* shows a system of interlinked ontologies to describe the concepts underlying FIPA agent communication [38]. It presents a meta-modelling approach for modelling both the object-oriented domain ontologies and the abstract models of agent communication and content languages. For example, the concepts of an ontology were referred to message types and internal agent operations were modelled as a combination of classes and objects which define the performing operations.

Collier et al. present the paper *"Agent Factory Development Methodology"* demonstrating with a brief example how these modelling languages

can be used to support developers assigned with the task of building an agent-oriented application [39]. The idea of modelling ontology domains during the implementation phase is considered, but the idea has not been discussed in depth. *Object Management Group* published a standard ontology definition meta model (ODM) for mapping OWL and UML in 2009 [40]. High expectations were raised and the community is waiting for implementation approaches that work in practise such as the ATL Use Case - ODM implementation (see [41]). In the meantime, whenever an ontology will be visualised in this thesis, the modelling notation from Appendix C will be used.

2.3.5 Open-World Assumption

The open-world assumption (OWA) is an essential aspect which is applied in formal system of logic like OWL or Datalog (see [42, p. 16] and [43]). It is the assumption that the truth value of a statement may be true irrespective of whether or not it is known to be true. The following section gives a more detailed explanation and better understanding of OWA.

The structure of OWL statements about individuals and their relationship with other individuals or values (i.e. object properties and datatype properties) shows analogies to structures in relational databases. Considering a simple database schema with a one-column table for each OWL class, containing all individual names that are individuals of this class, and a two-column table for each property, containing pairs of individuals (i.e. object property) or values associated with individuals (datatype property) known to be related by the property.

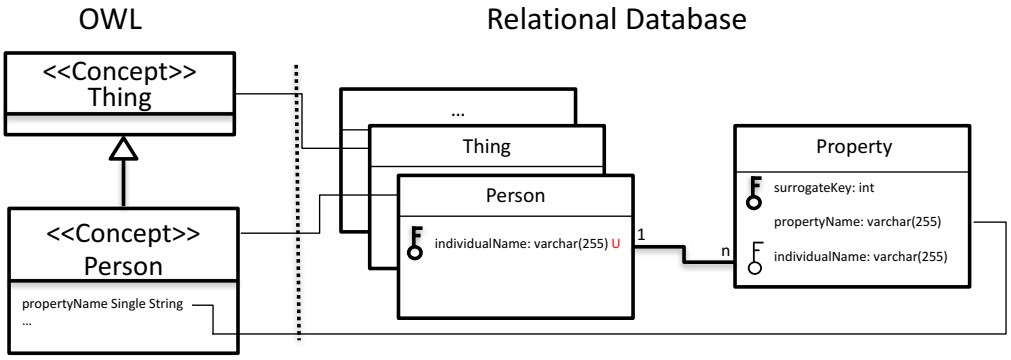


Figure 3: OWL/relational structural analogies.

Note that the occurrence of individuals in a table is based on either explicit assertions or implicit OWL reasoning results. OWL assumes that the information in these tables is incomplete by default – *open-world assumption*. The closed-world assumption which is commonly used in standard databases, not found is *false*. The open-world assumption would interpret non-occurrence of information as *unknown*, since the the open-world assumption assumes that if something is not known, just because nobody has formulated that an assertion is *not* true [44, pp. 233-238].

For instance, considering a data source about a customer c who owns a product p_1, then the query about the fact whether customer c owns product p_2 would be answered using the closed-world assumption with *false*. However, using the open-world assumption, the answer is *unknown*, as there could be another data source containing this information. If customer c is not in possession of product p_2, this has to be expressed explicitly in the ontology to receive *false*.

This assumption has a significant impacts on how information is modelled and interpreted. Typically, information systems operate with a closed-world assumption and assume that information is complete and known (everything

stated by the database, either explicitly or implicitly, is *true*; everything else is false). However, in the context of the semantic web it is useful to consider information as incomplete, because the absence of information on the world wide web is based on the assumption that the information has not been made explicit.

2.4 System Dynamics

Crowd behaviour is formed by individuals and depends on effects that can be ascribed to the crowd itself and the interactions between its members. Software simulations of these phenomena are fairly common in the field of sociology and related disciplines. A popular example is Forrester's world model presented to the Club of Rome in the 1970s, which tried to make predictions about future population levels, increasing pollution levels and rates of consumption of natural resources [45].

The system dynamics approach is generally used for models such as that described above. It is a modelling technique that can express cause-and-effect relationships between variables. Each variable needs to have an equation which is used to determine the value of the variable at any future point in time (i.e. time units). This in turn also depends on the values of other variables and describes the behaviour of the components of such a model. Such models are able to handle direct causal links such as how growth in population leads to increased depletion of resources and feedback loops: population growth depends on the food supply – but food supply in turn also depends on the level of the population [46]. Fig. 4 shows an example of a feedback loop modelled in *Vensim*[5].

[5]Vensim is a software tool from Ventana Systems, Inc. (See `http://www.vensim.com`).

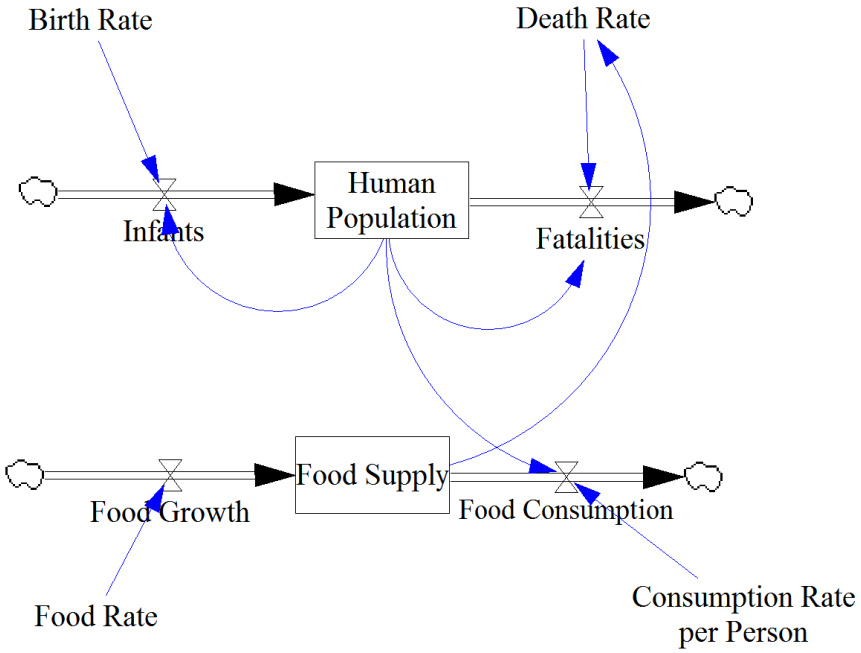


Figure 4: Simple human population feedback loop.

The model in Fig. 4 uses factors for representing natural birth rate, natural death, amount of food supply which all have an effect on the number of the human population. It is possible to set an equation for every variable as shown in Fig. 5. For example, there is a random birth rate between 0.05 and 0.2 and a variable death rate of 0.06 and 0.12 which depends on the amount of food supply. The food rate is randomised between 0.05 and 0.4 while the food consumption depends on the size of the human population. By running this simulation on the basis of the assumed data, it is suggested that the human population increases steadily at the beginning and drops slightly at the end, because in this model it is assumed that the growth of the food supply is limited. At the end the consumption rate is higher than the food growth and the human population drops slightly (see Fig. 6). Simulations with significantly more factors were used to describe various phenomena e.g. the human population growth.

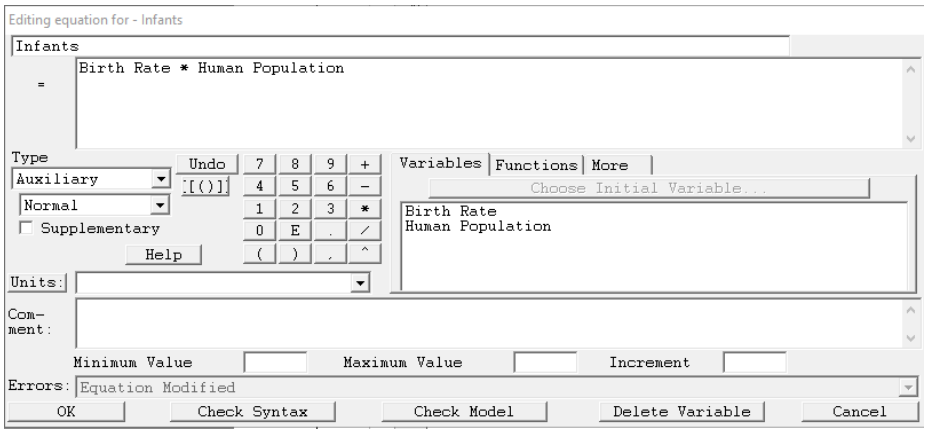


Figure 5: Equation view of a Vensim model.

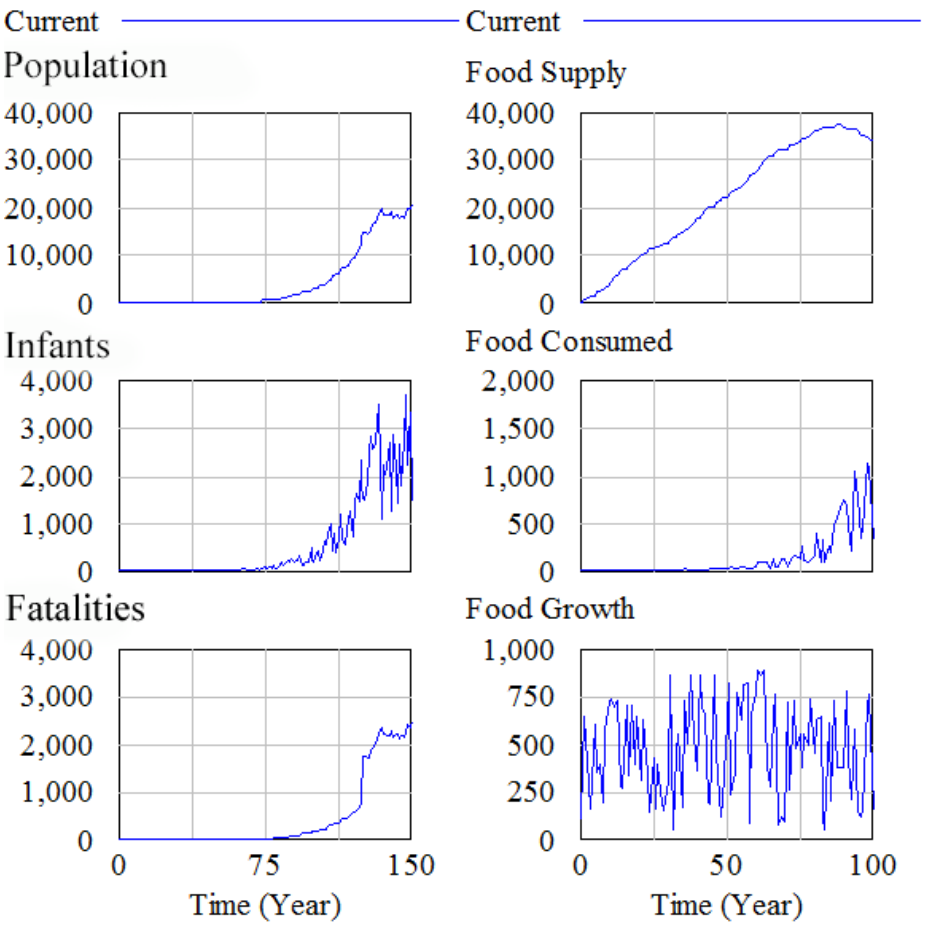


Figure 6: Result of running feedback loop simulation.

2.5 Summary of Chapter 2

This chapter has given a brief overview of the background relevant to this thesis. After introducing multi-agent systems in Section 2.1, knowledge representation has been considered. Section 2.3 introduced the OWL ontology language standardised by W3C followed by the current situation of available ontology modelling notation standards. Section 2.4 introduced the OWA, which is an essential aspect in OWL. The last section in this chapter described the system dynamics approach including a typical example of a simple human population model.

3 Literature Review

This chapter provides a description of literature relevant for the research area. The first part of this review focuses on business games, the second part focuses on multi-agent simulation models and methods and the third part, relevant work involving multi-agent systems and basic technologies is reviewed.

3.1 Business Games

Business games (also called business simulation games) originated from military war games which appeared in the late 1950s, spawned by the experience of many ex-military managers who had learned war game simulations in World War II [47]. Nowadays, business games can be described as virtual business worlds with the ability to allow users to interact and obtain feedback about possible outcomes of these interactions. The focus usually lies on economic process management using a simplification of the business environment [48]. Business software simulations often refer to simulation games that are used as an "educational tool" for teaching business. Participants make strategic decisions by analysing company and market data and then set different variables in the simulation to values determined as a result of decision process. Participants manage a simulated company and tackle life-like business challenges while competing in teams against one another. The simulation then processes the decisions of the participants and determines how successful these strategies would be in the simulated economic system. Results are usually calculated immediately. The feedback and time dimensions added by using simulations allow participants to evaluate

their accomplishments and commercial effectiveness after each round of the simulation [49]. The three business games listed are representative products of this area.

SimVenture

SimVenture was developed by Paul and Peter Harrington of Venture Simulations in 2006. It is a business simulation game for apprentices who are learning the realities of running a business startup. SimVenture allows participants to setup and run their own virtual company, allowing the user to learn about business and about being an entrepreneur in an active, authentic and engaging way. The participant takes on the role of business owner, managing time and money to develop the business and resolve any issues that may arise over the course of a game. The participant receives feedback from the software in relation to any decision made. Details of the underlying technology and algorithms have not been published.

Simultrain

Simultrain is a project management simulator used in project management training programs. A group of 4 people plays the role of a project manager and manages a 3-month project in 6-8 hours. It allows participants to acquire core project management competencies as well as teamwork and leadership skills. The first version was released in 1996. It is based on Sauter's PhD thesis *La modélisation des facteurs humains dans la gestion de projets* [50]. The participants have to complete the project within the deadline and budget with various scenarios: IT service development project, production project, marketing project and organising a sport event. It allows the multidimensional

optimisation along selected criterion of the shortest time or the lowest cost. Progress index may be modelled directly through the adopted earned values or as a quality parameter in the simulation. Again, details of the underlying technology and algorithms have not been published.

TopSim

TopSim is an umbrella brand which offers business games for different areas i.e. banking, biotech, business development, change management, destinations management, eCommerce, facility management, global management, insurance, logistics, marketing, portfolio management, social management, etc. Originally developed by Unicon, the software is under control of TATA Interactive Systems since 2006. A representative example of a business game is the *e-Commerce* scenario: the focus of this business game is the simulation of converting a conventional company into an online company. The main goal is to improve cost effectiveness, which can be achieved by formulating and reaching goals such as customer loyalty, customer satisfaction and others determined by the participants. Market analysis, product mix, pricing, the right ratio of online and traditional marketing, the support of IT (call centers and logistics capabilities) are critical success factors in this scenario. During the simulation, the products on sale and certain business parameters can be modified by the participants. In conjunction with rapid market growth, the seasonal fluctuations create a strong momentum in the game. Participants have to make assumptions about the market and act according to their hypothesis. This business game is based on the system dynamics approach (see Section 2.4). TopSim eCommerce

currently implements 78 of those cause-and effects relations and each relation can be edited by the seminar leader e.g. the parameter of attractiveness of a website which has an effect on market share [51, pp. 17-20].

The University of Applied Sciences Friedberg has used TopSim games since 2008. Despite the successful use in the classroom and the positive feedback from students, there are rigid game restrictions that cause a gap between the implemented model and reality. Lecturers as well as students note that it does not give sufficient feedback to draw conclusions on cause and effect. This statement can certainly be made for the most of the business games which are based principally on global cause and effect description. Again, such models are highly sensitive according to their setting. Agent-based models are a more natural tool to represent individuals in a market rather than describing effects through equations and thresholds.

3.2 Approaches for Multi-Agent Based Simulation

3.2.1 Computational Economics

Computational Economics (CE) is a branch of economic research which is mainly interested in system dynamics and the computational study of economic processes by running simulations. A popular approach for implementing heterogeneous agent models is also called Agent-Based Computational Economics (ACE). In the paper *Heterogeneous Agent Models in Economics and Finance* [52], Hommes explains reasons for the increase in popularity of ACE. Certain economic models have traditionally relied on the assumption that humans are rational and will attempt to maximise their utility for

both monetary and non-monetary gains (i.e. *Homo Oeconomicus*). Modern behavioural economists, however, have demonstrated that human beings are not as rational as has long been assumed. Based on Hommes laboratory experiments, he supports the statement that individuals often do not behave rationally.

Smith and Kahneman are pioneers in the discipline of experimental economics [53]. They received the Nobel prize in economic sciences for their work in 2002 [54]. Economic experiments check psychological principles of individual actions in economically relevant decision-making situations. Smith conducted a classroom experiment in 1956 where half of the participants act as buyers and the other half act as sellers [55]. Buyers profited from exchange and had a secret demand schedule while sellers had a secret supply schedule. The two-sided auction began: buyers announced bids, sellers announced asking prices. Contracts were made on acceptance. The market converged to price and volume represented by theoretical intersection of supply and demand curves. Market simulations are now often designed in a very abstract way and models derived from *decision* and *game theory*.

Multi-agent environments have proven to be successful tools for simulations that has seen a number of applications in the last few years, including applications to real-world business problems [56]. There are a lot of approaches to pure agent-based simulations. Bonabeau shows an example of an emergent phenomenon where simple individual rules lead to coherent group behaviour [57]. Particularly worth emphasising is the ability to run the simulation involving humans as well as by agents.

There is a group of 10 - 40 people with the following rules:

- Each member has to randomly select two individuals and define them

as person A and person B.

- The selected person A is considered as a protector and person B is considered as the opponent.
- Every member tries to keep person A between them and person B.

Then the simulation starts where each member has to ensure that they always keep A between them and B (A is their protector from B). After a short period of time each member has to keep itself in between A and B (members are now the protector). The system will end up in everyone clustering in a tight knot (see Fig. 7).

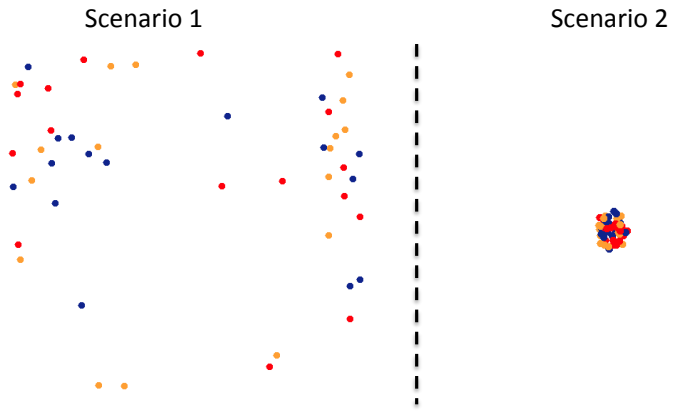


Figure 7: Example of emergent phenomenon.

Again, this short agent-based example shows group behaviour of individuals and additionally how small changes in rules can have a dramatic impact.

3.2.2 Micro-Economic Models

Micro-economic is a branch of economics that studies the behaviour of individuals and firms in making decisions regarding the allocation of limited resources [58, p. 19]. Steiglitz describes one of the simplest models wherein agents produce, consume and trade in a gold-food economy e.g. agents

will bid higher if the inventory of food is low and will sell food at a lower price if the inventory is high [59]. By running this simulation with 1,000 independent agents and some speculator agents who are able to store gold and food depending on the exchange rate of food to gold, the price stabilised dramatically. This model utilises multi-agent systems and shows the impact of rules on the agents' behaviour.

In addition to pure agent-based models, there is an approach for building simulations which is called microsimulation. The original version of microsimulation was set out in [60]. It is a modelling technique that "operates at the level of individual units such as persons, households, vehicles or firms" [61]. This modelling technique uses historical records or representative national surveys e.g. of households, and includes data with unifying identifiers, e.g. a list of persons with age, sex, annual income and employment status. A set of rules allows changes in state or behaviour like changes in taxes, marrying or salary increases to be simulated. These rules are in general deterministic or stochastic. There are a lot of hybrid approaches in which agent-based models are based on microsimulation models in the area of economics and financial activity which might combine the best features of both approaches (see [62], [63], [64], [65] and [66]). Agents are well suited for representing individuals and the simulation runs with realistic historical data. However, there are disadvantages:

- High modelling complexity of agents and their appropriate mapping with the historical data. For example, the change of a state of an agent in relation to time has to be represented, e.g. the possibility that an individual earns more or less money next year which in turn can be reinvested in the market.

- Microsimulation does not allow any interaction between agents. Each agent is individually represented by its properties and isolated in the world.

Weighing up the advantages and disadvantages and examining the scientific assessment, an agent-based microsimulation allows a quick development of individual simulations in which the effects of changing rules can be visualised but the representation of individual behaviour of humans, especially the communication among themselves, will be ignored.

3.2.3 Monte Carlo Simulations

Monte Carlo methods are also applied in agent-based simulations, typically for assigning valuations to agents or for simulating noise. In 1993, Gode and Sunder published an article in which they have conducted various market experiments. They run experiments in which human traders are replaced with so called *zero-intelligence* traders. Zero-intelligence trader is a simple algorithm trader in a market that submits random bids and offers in a range bounded by their reservation prices (they can avoid transactions which incur losses) [67]. Cliff and Bruten analysed models that use zero-intelligence trader and showed that they do not correspond to real-world behaviour. Two more articles from [68] and [69] underline, that more than zero-intelligence traders are needed for realistic simulations - especially in double auction market environments - in which buyers can determine the price and the highest bid typically wins. However, Das et al. have shown that zero-intelligence programs consistently out-perform human traders in human-against-robot experimental economics marketplaces in real-time [70]. In 2001, Cliff published an article based on the result of this analysis in

which he presents a parameter set for more realistic behaviour of software agent traders, but the models are highly sensitive in the parameter setting [71].

This demonstrates the breadth of approaches of agent-based simulation models with attempts to build realistic economical simulations. For running simulations it is important that agents are represented in a way that is as similar as possible to the behaviour of human beings. The next section deals with these approaches.

3.3 Review of Relevant Work Involving Multi-Agent Systems and Semantic Technologies

3.3.1 Ontologies in Multi-Agent Systems

It is not a new idea to use ontologies to share terms among agents. Malucelli and colleagues introduced the *ForEV* platform which is implemented using a multi-agent system in which partners negotiate about common standards of which an ontology is part [72]. Stuckenschmidt and Timm argue that ontologies play a key role in multi-agent communication and mention that there is a need for ontologies for interacting agents to understand the content of messages [73]. Ontologies have also been applied to help in *solving the heterogeneity problem in e-commerce negotiations* [74]. Furthermore, sharing terms are also important in robotic applications e.g. the RoboCup robot soccer domain to share knowledge and also to represent real-world objects in software applications [75].

3.3.2 Ontology Alignment

Ontology alignment aims at finding meaningful correspondences between two ontologies represented as a collection of entities [76]. Ontologies can be used to support the integration of heterogeneous information tasks. Singh and colleagues present semantic modelling techniques that enable the integration of stable information resources and pioneered the use of agents to provide interoperation among autonomous systems [77]. The paper *"Coordinating Heterogeneous Information Services based on Approximate Ontology Translation"* shows approximation mechanisms for ontology translations for achieving semantic interoperability [78]. In this prototype, agents can coordinate regional information services provided by the GeoLink system [79].

Soh's paper deals with the collaborative understanding of distributed ontologies in multi-agent frameworks [80]. Each agent manages an information database in a distributed information retrieval simulation. For facilitating collaborative understanding, each agent maintains its own ontology and a translation table. Agents determine whether some translation is worth learning, which neighbours to communicate with, how to handle and distribute queries, and how to plan for agent activities. The translation table is to map between each concept the agent knows and its neighbours.

Deen shows an ontology integration for a multi-agent environment [81]. Each agent is autonomous and has knowledge of its own schema. Additionally, each agent holds an ontology of its acquaintances i.e. other agents with similar interests. The integration has to be carried out whenever a new acquaintance is added or, when the local ontology of an acquaintance changes.

This approach implies that agents communicate only with a few agents and ignore others.

Among the approaches mentioned before the concept of ontology based communication is the most important of this thesis. Since one of the main objectives of this thesis is the creation of simulations with semantically advanced technology, ontology concepts will be used for communication task. Using a standard modelling language to share terms among agents, enables these agents to communicate appropriately.

3.3.3 Semantic Technologies Applied to Agent-Models

Up to now, multi-agent systems have been set up on rather similar paradigms regarding the methods for expressing data, behaviour and knowledge. Agents only have a limited knowledge of their external world. There have been some rudimentary efforts to give agents access to structured representations of what they know. For example Knowledge Query and Manipulation Language (KQML), which is a language and a protocol for communication among software agents and knowledge-based systems that allows a structured way for knowledge exchange [82]. As a result of this a logical language Knowledge Information Format (KIF) was built for the representation of knowledge [83]. It is a computer-oriented language for the interchange of knowledge among different programs. It allows agents to express properties of humans or objects like Peter is 1.87 meters tall or Neil is the supervisor of Thomas. But KQML was proposed without a defined semantic. This criticism led researchers to define the new language *Agent Communication Language* standardised by the Foundation for Intelligent Physical Agents (FIPA), or (FIPA-ACL).

FIPA was founded in order to improve the interoperability between agent systems. As a result, FIPA-ACL is now a widely adopted standard language for agent communications [84]. FIPA-ACL is in turn based on the ARTIMIS Communication Language (ARCOL) [85]. ARCOL became the basis for the FIPA-ACL rather than KQML because ARCOL defines formal semantics, whereas KQML does not. It should be mentioned that there are no organised efforts to develop KQML any further or even to maintain a list of resources about it. The KQML effort was mainly subsumed by FIPA's activities and their concentration is on FIPA-ACL [86].[6] In order to model ontology-based communication FIPA published the following specifications:

- The *FIPA-ACL Message Structure Specification*
- The *FIPA Ontology Service Specification*
- The *FIPA-SL Content Language Specification*
- The *FIPA Communicative Act Library Specification*

These specifications define speech acts in agent systems with the help of a formal model. There are more specifications than listed here, but the status for other content languages is still experimental. A closer look at the ontology service and SL content language specification which enable agents to manage explicit, declaratively represented ontologies shows that SL is in general undecidable. There are two profiles, SL1 and SL2 which are subsets of SL. They have restricted statements and have to be decidable by definition. Schiemann mentioned in his work that there are disadvantages in

[6] All three communication languages are based on a philosophical theory called *speech act theory*. This theory is originally due to the philosopher of language, Austin in 1962 [87], and extended by Searle (1969, 1983) and Searle and Vanderveken (1985) [88], [89] and [90]. In such an approach, interactions among agents take place at least at two levels: one is corresponding to the informational content of the message and the other is corresponding to the intention of the communicated message.

the specification [91]. First, building reasoners for message contents is a very complex task. The way in which FIPA-SL is applied determines whether one of the profiles is satisfied or not. Second, the speech act *request-when* must have another speech act as its content. The *request-when* allows an agent to inform another agent that a certain action should be performed as soon as a given precondition, expressed as a proposition, becomes true. The separation of the semantics of speech act and content in FIPA-SL is difficult (see [91]). Khalique and colleagues made an analysis of FIPA-SL and OWL based on the factors of knowledge representation and expression power. They came to the conclusion that both languages have advantages as well as disadvantages. But OWL has become a W3C recommendation. OWL is constantly being developed and becomes really more applicable. ACL messages contain one or a set of more parameters, but every message has to set the *performative* parameter which specifies the type of communicative act. The types of communicative acts used most often are *request*, *inform*, *reuse* and *not-understood*. All types are listed in table 1.

Table 1: FIPA-ACL Message Parameters.

Parameter	Description
performative	Type of communicative acts
sender	Sender of the message
receiver	Receiver of the message
reply-to	Specifies the reply address
content	Content of message
language	Language in which the content is written
encoding	Encoding of the Content
ontology	Specifies the ontology
protocol	Specifies the interaction protocol
conversation-id	Unique Identifier, which identifies an ongoing communication
reply-with	Control of conversation
in-reply-to	Reference to a previous action
reply-by	Time or date value as deadline for a response

To realise parallel communication between agents, each message has a unique identifier: the *conversion-id* parameter. Thus agents are able to associate messages with a specific task. The language parameter is the representation of an ACL message for several types of encoding like FIPA-SL. Within the FIPA-ACL specifications it is also possible to set an ontology parameter as language parameter. The ontology parameter is used in conjunction with the language parameter to support the interpretation of the content expression by the receiving agent. Fig. 8 shows a typical

ontology-based communication model: agent A sends a message to agent B. The content of the message can e.g. be a question about any part of a mobile phone. Both agents, A and B, use the same ontology. Thus, the receiving agent B knows how to interpret the incoming ACL-based message.

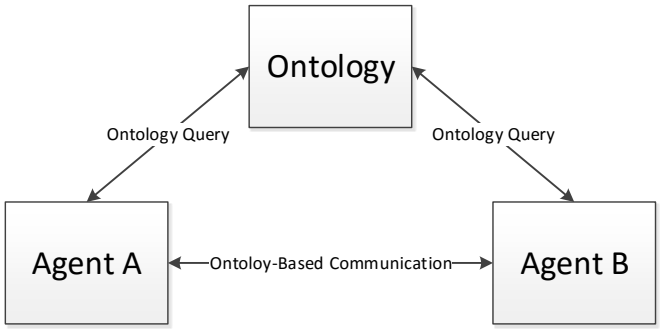


Figure 8: Ontology-based communication model.

Moreira et al. published a paper which set the general basis for the idea of combining agent-oriented programming with the use of reasoning techniques of ontologies [92], but additional research activities for a practical implementation for running business simulation scenarios such as ontology based communication (see Section 5.4) and rules (see Section 5.1.2) for agent behaviour have to be carried out. Schiemann et al. use contents that are exclusively formulated in OWL DL for FIPA ACL compliant messages [91]. In practise, it is more realistic that agents not only use the same ontology but that they also have an individual ontology. The main challenge of giving agents individual ontologies is that ontologies can differ in various ways. The difference can exist between the concepts, properties and the facts (i.e. individuals or instances of concepts). In a further approach, Schiemann tackles the problem of merging OWL DL ontologies used in multi-agent systems to allow communication between agents that use semantic content for dynamic knowledge bases [93]. He considered a semi-automatic method

which uses the principle of minimal change of belief revision theory [94, pp. 129-147].

3.3.4 Overview of Available Frameworks

There are over 20 different multi-agent systems from one-man open source projects to commercial platforms with IDE support (see [95] and [96]). Among them it is possible to identify two groups. While the first group focuses on the implementation of communication platforms and standards like FIPA [97], e.g. JADE [98] or FIPA-OS [99], the second group provides support for the implementation of agent internal structures for methodological behaviourism in which each agent is defined in terms of its goals, knowledge and social capability, e.g. AgentBuilder [100], JACK [101] or JAM [102]. These systems are based on different agent languages because there is no general consent about which language concept is suitable for the description of those structures. For example, AgentSpeak [103] or the extended version of it called Jason [104] are well known agent-oriented programming languages which are based on the Beliefs-Desires-Intentions-Concept (BDI-Concept). The origins of the BDI model lie in the theory of human practical reasoning developed by the philosopher Michael Bratman, which focuses particularly on the role of intentions in practical reasoning [15]. The theory of human practical reasoning asserts that agents are influenced by their beliefs, desires and intentions in their reasoning about which action to perform. BDI agents are defined as systems that are situated in a changing environment, receive continuous perceptual input and take actions to affect their environment, all based on their internal mental state [103]. In 2007, Silva et al. shows an approach called Argonaut [105] which shows a combination of Jason with

internal actions to handle OWL ontologies but has disadvantages (see [106]).

Most agent platforms, both FIPA and non-FIPA, offer flexible integration possibilities at the agent level, but the platforms themselves are mostly considered as a *black box*. The agent community agrees that the flexible and dynamic attaching of agents from different systems to the agent platform requires a common language [107]. Ontologies are suitable to be used as a common language as they can represent an agreement on common terminologies. W3C's OWL has gained wide acceptance in the agent community and it has already been used in many agent frameworks like JADE. Those frameworks implement ontologies in general for describing elements that agents can use within the content of messages. The ontology is used to define a vocabulary and the relationships between the elements in such a vocabulary.

Table 2 shows the results of a study comparing various agent frameworks based on [108, p. 282]. Firstly, the ontology support of agent frameworks was evaluated. Secondly, criteria such as the free use of, support, available literature and a quick initial training in getting the framework runnable was evaluated. The three top-rated agent-based frameworks that made it into the closer selection will be analysed with detailed descriptions in the following three sections.

Table 2: Comparing Matrix of Agent-Frameworks based on [108, p. 282]. A 1-5 scale was used for the values: (1) excellent, (2) very good, (3) good, (4) fair, (5) poor.

Criteria / Agent Platform	JADE	FIPA-OS	Zeus-Tool	MadKit	agentTool	JACK	Jason
Developers	Telecom Italia Lab	FIPA-OS Community Project	BT Communications	MadKit Team	Kansas State University / Multiagent & Cooperative Robotic Lab	Agent Oriented Software Pty. Ltd.	Hübner and Bordini
Latest Version	JADE 4.5.0 (June 2017)	FIPA-OS 2.2.0 (March 2003)	Zeus 2.0 (January 2006)	MadKit 5.1.1 (March 2017)	agentTool III 1.1.4 (June 2012)	JACK 5.6 (July 2015)	Jason 2.1 (December 2016)
Open Source	Yes	Yes	Yes	Yes	Yes	No	Yes
FIPA compliant	Yes	Yes	No	Since v5.0	Yes	Yes (Extension)	No
Further Development	Yes	Unknown	Unknown	Yes	Yes	Yes	Yes
Availability	Download	Download	Download	Download	Download	Download (60 Day Trial)	Download
Ontology Support	Yes	Indirectly	Yes	Indirectly	Planned	No	Indirectly
Installation	3	3	2	1	2	1	2
Comprehensibility	2	2	2	2-3	3-4	1-2	2-3
Range of Applications	1	1	2-3	1	2	2-3	3
Reliability	2	2	2-3	1	3-4	1	2
Support/Forum	1	3	4	2	2-3	2	3
Literature	1	3	3	4	2	2	2
Average	1.7	2.33	2.17	1.25	2.58	1.7	2.41

3.3.4.1 JADE

JADE was developed by Telecom Italia Lab. JADE is an abbreviation for *Java Agent Development Framework*. It is implemented in Java and based on the FIPA specifications for agents. It is distributed under the GNU Lesser General Public License (LGPL), which guarantees end-users the freedom to use, study, share (copy) and modify the software for individual needs.

3.3.4.1.1 Architecture

Fig. 9 shows the architecture of the JADE agent platform. It consists of different containers which may be connected or distributed over a network. Each agent runs in an individual Java thread and lives in a container. A container manages all services which are needed for running an agent.

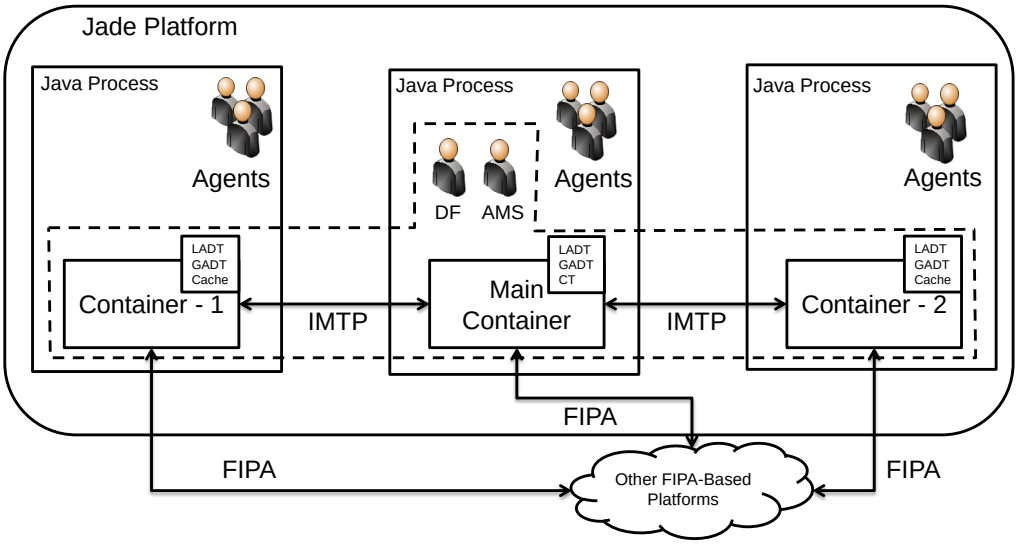


Figure 9: JADE architecture based on [98].

The main container plays a particular role in JADE. It is the starting point of the agent platform and is loaded initially. All other containers must be linked to this main container. The main container manages the *container*

table (CT) which contains the registrations for all object references and transport addresses of linked containers. In addition, the main container also manages the *global agent descriptor table* (GADT). It contains the information of running agents, their location and status. Furthermore, the main container hosts two agents: the *Agent Management System* (AMS) and the *Directory Facilitator* (DF). Both are agents which are connected and instantiated automatically by JADE. The AMS controls and manages the access to the agent platform. It also provides a called *white page service* which controls life cycles of agents: a new agent will be registered at the AMS and will receive an *Agent Identifier* (AID) which identifies an individual agent. The *Directory Facilitator* (DF) provides the called *yellow page service* in which agents can register their own services or search for services of other agents. To handle performance issues in the agent platform, all other containers contain two tables: the *Local Agent Descriptor Table* (LADT) and *Global Agent Descriptor Table* (GADT). If a container has to locate the receiver (agent) of a message, the LADT will be used. If the search fails, the main container has the task of locating and caching the reference of the agent. Due to the dynamics of the system, agents can migrate, terminate or instantiate during a running simulation. This could lead to incorrect entries in the LADT and an internal error message will be thrown. If this happens, the container has to update its cache entries from the GADT of the main container.

3.3.4.1.2 Modelling Behaviour

The tasks an agents can perform are implemented as *behaviours* in JADE. A behaviour describes the reaction of an agent on its environment. All classes

are sub-classes of the abstract class `Jade.core.behaviour.Behaviour`. Every modelled agent has to inherit from the class `Jade.core.Agent` which supports two methods of controlling behaviour:

- addBehaviour(Behaviour b) and
- removeBehaviour(Behaviour b).

They facilitate the adding of new or removing no longer required behaviour. It is possible to add or remove behaviour during runtime. A scheduler, internal to the base class `Agent` and hidden to the programmer, automatically manages the scheduling of all defined behaviours until the method `action` is called. JADE differentiates between two behaviour classes: the `SimpleBehaviour` and the `CompositeBehaviour`. The `SimpleBehaviour` is an abstract class which models behaviour that is made by a single, monolithic task and cannot be interrupted. The `CompositeBehaviour` is a superclass for behaviour composed of many parts. This class holds a number of children behaviours inside. When a `CompositeBehaviour` receives its execution quantum from the agent scheduler, it executes one of its children according to some policy.

3.3.4.1.3 Ontology Support

Jade supports the use of ontologies for defining a knowledge domain. This enables the creation and communication of complex data types in agent languages. Fig. 10 shows a class diagram of the content reference model.

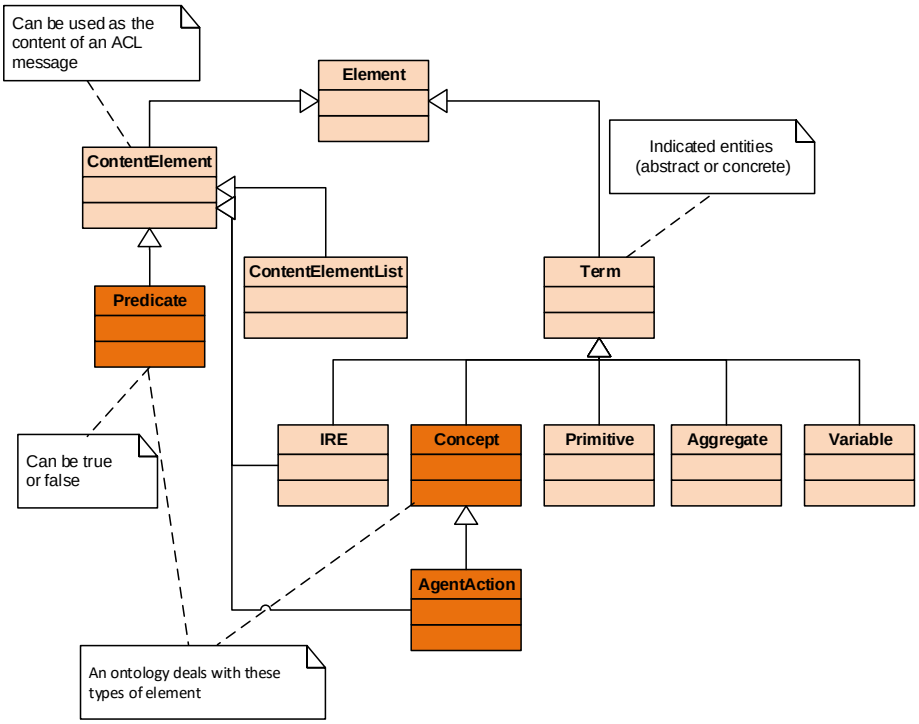


Figure 10: The content reference model (based on [109]).

The important types in this model are *Predicate*, *Concept* and *AgentAction* since these are the types a JADE ontology deals with. They are marked in orange and are defined as follows [109]:

- *Predicate* is an expression which describes the status of the modelled world and usually evaluates to true or false.
- *Concept* is an expression that indicates entities and that agents talk and reason about.
- *AgentAction* is an expression that indicates something that can be executed by some agent.

The other classes appearing in Fig. 10 are provided to support JADE ontologies. They are generally used to combine user defined content elements in order to put them into an ACL message. Ontologies are modelled by

the Java class `Jade.content.onto.Ontology`. This class can be used for modelling concepts, predicates and agent actions in message content. An individual Java class instance is needed for each of the modelled elements. Objects can be generated by the content manager which prepares (encodes) the message for the use as message content of an ACL message. The content manager can decode the message content on the receiver side into the original object structure including validation without any further programming efforts. Fig. 11 shows the process of code and decode by the `Content-Manager`.

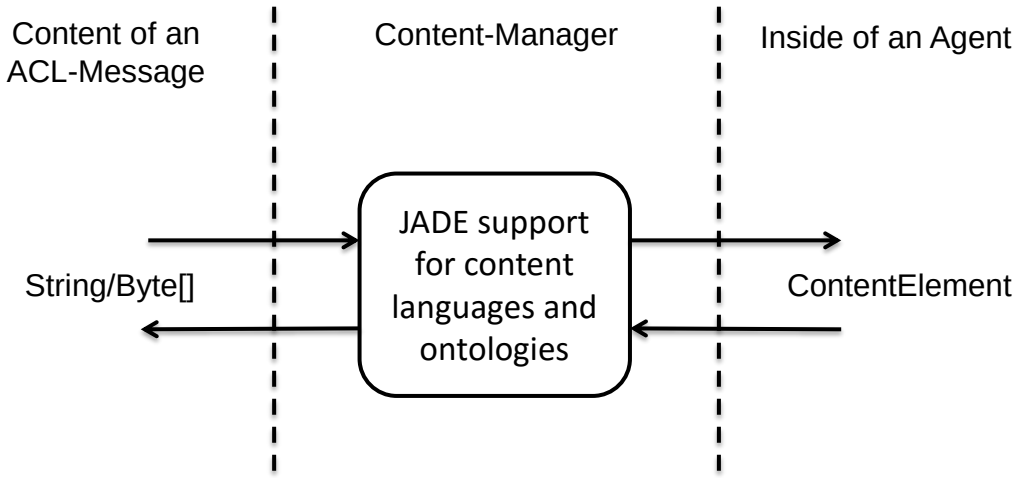


Figure 11: Jade support for content languages and ontologies (based on [98]).

JADE allows the development of additional modules implementing features that extend Jade with functionality. Braubach and others developed an add-on called Jadex which follows the BDI architecture. Thus allowing the modelling of BDI in a widely used agent plattform.

3.3.4.2 MadKit

MadKit is a Java-based platform which is built upon an organisational model. It was developed by Gutknecht and Ferber and the help of a MadKit team

at the Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier [110]. The main classes of MadKit are distributed under the LGPL.

3.3.4.2.1 Architecture

MadKit is based on the *Aalaadin* meta model. The Aalaadin model is not a specific agent methodology, but a meta model for describing organisations of agents which are based on three core concepts: *agent*, *group* and *role*:

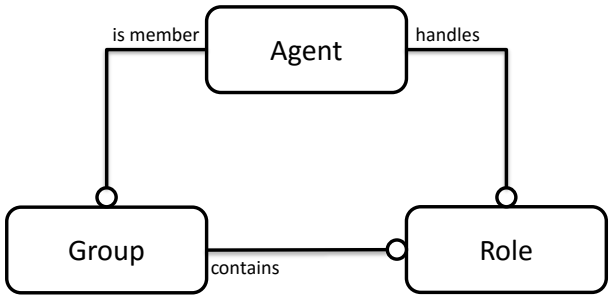


Figure 12: Aalaadin core model based on [111, p. 3].

The following provides a short description of Fig. 12 [111]: The *Group* is defined as an atomic set of agent aggregation. A group represents any usual multi-agent system. The *Role* is an abstract representation of an agent function, service or identification within a group. Each agent can handle several roles, and each being local to a group. The model does not place any constraints or any formalism on agents' internal architecture.

The organisational structure is the focus of MadKit. The architecture and the structure of any individual agent have to be defined by the developer. Fig. 13 shows the essential components of MadKit [112]:

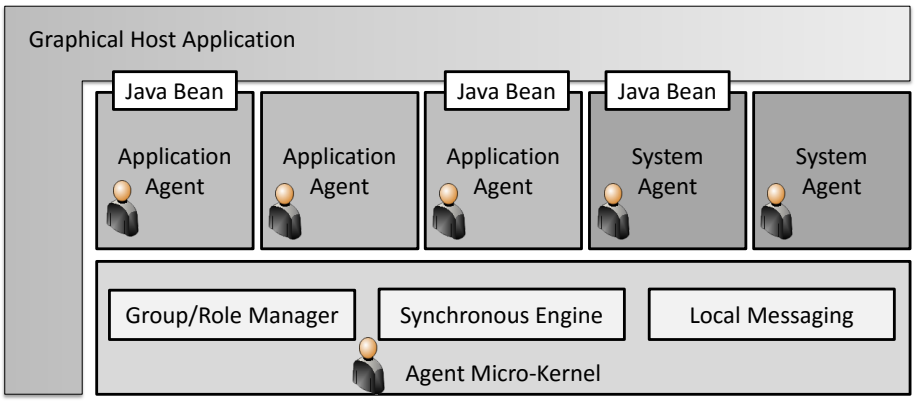


Figure 13: MadKit architecture (based on [112]).

The Agent platform consists of a micro-kernel and a set of system agents which handle certain system services. The micro-kernel is a small (less than 40 Kb) agent kernel and can be understood as a program – similar to micro-kernel operating systems – which provides an infrastructure for running system services. Further, it handles the management of local groups and roles, controls the life cycle of all agents and handles the communication between local agents. The micro-kernel is also represented as an agent – the *KernelAgent*.

In contrast to other agent frameworks, MadKit uses agents to achieve things like distributed message passing, migration control, dynamic security, and other aspects of system management. The developers are pursuing the goal of a very high level of customisation, as these agents can be replaced without great difficulty.

3.3.4.2.2 Modelling Behaviour

MadKit is a Java library for designing and simulating multi-agent systems which provides a set of tools i.e. agents which can be used to help the development of multi-agent applications. Such agents provide methods for

building communities, groups and roles. The group and role system is always available to an agent and provides both action and information calls. The agent developer is completely free to define the agent behaviour, but the organisational model will always be present.

3.3.4.2.3 Ontology Support

MadKit does not provide a mapping model of an ontology similar to JADE which is described in Section 3.3.4.1. Since the release of MadKit 5 in 2012 the API specifies an option for using the FIPA protocol as a communication language which can specify an ontology as a parameter. There is no further support for ontologies and the use of the ontology parameter is up to the developer.

3.3.4.3 FIPA-OS Toolkit

The FIPA-OS is a component-based toolkit and was originally developed as an experimental agent framework by *Nortel Networks Ltd.* [113]. Since 2000 the company *emorphia Ltd.* has led the development of the FIPA-OS toolkit. It was founded by members of the agent group at Nortel Networks for the purpose of commercialising agent technology, network services, mobile technology and building solutions. The toolkit is distributed under the *Nortel Networks FIPA-OS Public Licence* which is similar to GNU General Public License (GPL) [113].

3.3.4.3.1 Architecture

Fig. 14 shows the architecture of the FIPA-OS toolkit.

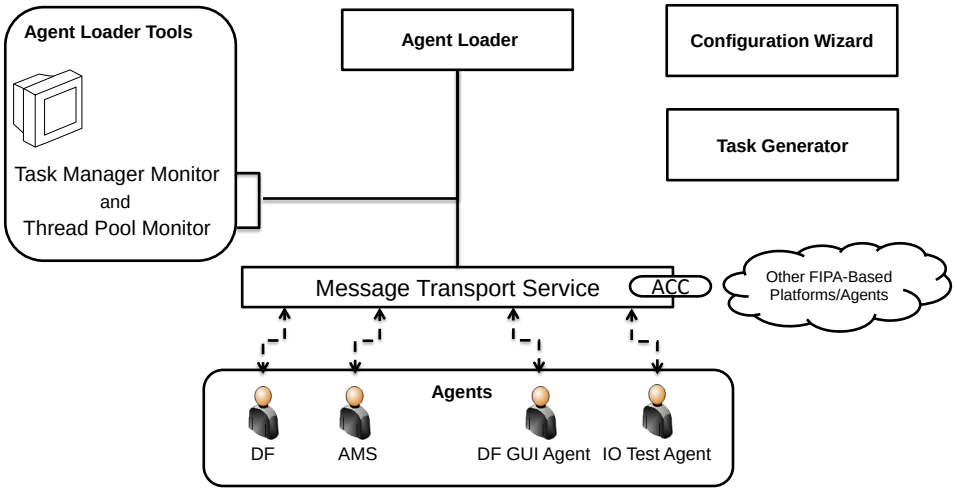


Figure 14: FIPA-OS architecture based on [113].

An essential component of the FIPA-OS agent platform is the *Agent Loader* which starts and stops the agents. For monitoring threads or tasks of running agents the *Agent Loader Tools* can be used. The FIPA-OS model consists of the DF and the AMS agents similar to JADE in Section 3.3.4.1. The DF provides the yellow page service in which agents can register their own services or agents can search for services from other agents. The AMS provides the white page service which controls life cycles of agents. Both agents are started automatically during the initialisation of the platform by the *Agent Loader*. A *DF Cross Registration GUI* agent which is loaded automatically is needed to link several DF agents. It provides a `register with remote DF` and `add remote DF` function. The *IO Test* agent is a simple test agent to send messages to other agents and prints out the responses. The Message Transport Service (MTS) ensures the communication between agents and supports the message transport protocols Remote Method Invocation (RMI) and Internet Inter-ORB Protocol (IIOP). MTS is part of the Agent Communication Channel (ACC) which enables

the communication to other compliant agent platforms. The *Configuration Wizard* can be used to set all running parameters for FIPA-OS easily. The *Task Generator* facilitates the integration of communication protocols in agent classes by generating Java classes depending on the protocol definition.

3.3.4.3.2 Modelling Behaviour

FIPA-OS provides a lot of Java library classes for developing a multi-agent system. A detailed description of the more than 450 existing Java classes can be found in the developers guide of FIPA-OS [114]. The tasks performed by agents are implemented and handled by the class `TaskManager`. It provides the ability to split the functionality of an agent into smaller, disjoint units of work known as tasks. Generally every modelled agent has to extend the class `FIPAOSAgent`, and a number of `Task` implementations that contains the basic functionality of an agent. The `FIPAOSAgent` class enables direct access to the `TaskManager` by the `_tm` variable. The class `TaskManager` references all active `Task`s which can be understood as a set of specific events to be processed. The different types of events are handled by the `TaskManager` i.e. *initialisation of a task*, *termination of a sub-task* and *termination of not correctly processed sub-task*. The order of upcoming events is handled by the `TaskManagerListener` which will be informed by the `TaskManager`. The default processing of upcoming events is in the order in which the events occur. Fig. 15 shows the logical interaction between a number of parent and child tasks. The method `startTask()` is invoked after the method `newTask()` has been invoked on a task and `doneX()` is automatically invoked on the parent after a child task invokes its `done()` method.

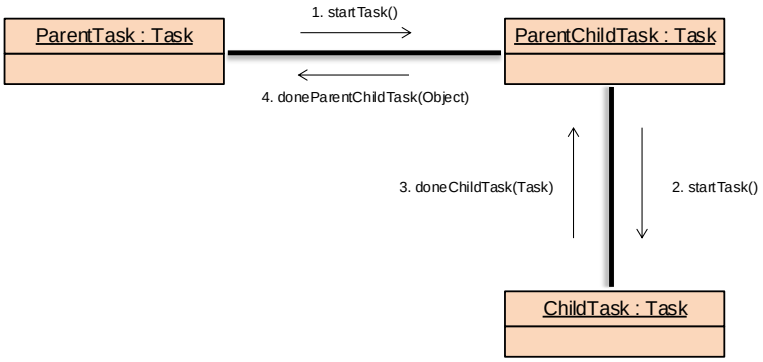


Figure 15: FIPA-OS task model (based on [113, p. 17]).

3.3.4.3.3 Ontology Support

FIPA-OS does not provide a mapping model of an ontology similar to JADE which is described in Section 3.3.4.1. The API of FIPA-OS only specifies an ontology package in which an individual ontology class can be placed if a class belongs to an ontology component. FIPA-OS supports the majority of the FIPA experimental specifications and is being continuously improved as a managed open source community project. This means that the listed protocol specifications in Section 3.3.3 can be used in FIPA-OS.

3.4 Conclusion Drawn from Literature Review

The literature review shows the breadth of approaches of agent-based simulation models, which allow flexible and dynamic simulations. Despite the success of business games, rigid game limitations cause a relevant gap between the implemented model and reality, e.g. micro simulations do not allow any interaction between individuals. When running simulations, it is important that individuals are represented in a way as close to reality as possible. Regarding business simulations, social interactions such as commu-

nication are a major requirement for representing consumers appropriately in a market. The modelling of interaction can create new explanation possibilities for the phenomena of crowd and individual behaviour. Multi-agent environments have proved to be successful tools for simulations [56]. These models in which agents represent individuals can extend business games with more realistic behavioural patterns but those patterns can only be achieved by appropriately representing knowledge and intelligence. Agents must be able to act according to what they know and must be able to learn. The standardised agent communication language FIPA already allows for the integration of ontologies, but currently agent frameworks have not yet made full use of this option. The available FIPA Ontology Service Specification has disadvantages, which are described in Section 3.3.3. FIPA defines a knowledge manipulation based on content languages i.e. FIPA-SL, which is powerful but lacks any interconnection with commercial tools and standards. The disadvantages can be reduced by using description logics [91]. Description logics are the core of ontology languages, such as OWL [115]. These can be used to give agents access to a structured representation of what they know. The challenge is to minimise the described lack in integrating ontologies and agent frameworks as described in the previous sections.

4 Social Structures and Marketing Theory

This chapter provides a brief overview of social structures, marketing theories and behavioural patterns relevant to the simulations used for the experiments. Well understood marketing mechanisms are explained and the scenarios (i.e. opinion leadership and personal preferences) are introduced.

4.1 Social Structures

Social structures are the basic concept of the proper understanding of society. For a long time, many efforts have been made to define *social structure* but still there is no unanimity of opinion on its definition. According to Radcliffe-Brown social structure is "an arrangement of parts or components related to one another some sort of a larger unity" [116].

In other words, individuals can be considered as components and individuals occupy in some kind of a position in the world. The relationships among these positions make up the social structures.

Social structures can be visualised using mathematical graph theory: A graph $G = (V, E)$ consists of a set of nodes V (vertices) and links E (edges) between pairs of nodes. Edges can either be directed or undirected. Graphs can be applied in real world problems: in the context of social networks, nodes typically represent individuals and edges relations between individuals. This representation of interpersonal relationships is also called a sociogram. Depending on the type of relation modelled the edges will be directed or undirected. The number of edges leaving a node is called *out-degree* and the number of edges entering is called *in-degree*.

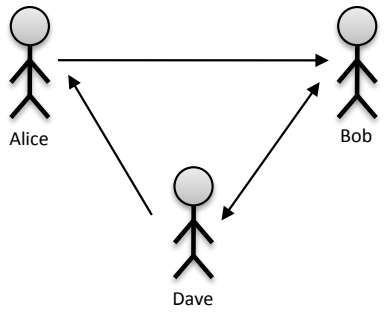


Figure 16: Sociogram.

Fig. 16 shows a rudimentary sociogram containing three individuals that are linked together: Alice knows Bob and Alice is known by Dave (directed link). Bob and Dave know each other (undirected link). Note, links do not necessarily represent a *knows* relationship. They may also represent various factors such as involvement, appeal, co-operation, etc. The in-degree of Alice and Dave is both one and of Bob is two. The out-degree is one for all individuals.

4.1.1 Small-World Scale-Free Networks

A small-world network is a type of a graph in which, although most nodes are not neighbours of one another, most nodes can be reached from every other node by a small number of links. Analysis of real world communities showed that social structures are not as random as long assumed. According to Watts et al., small-world networks are a class of networks that are "highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs" [117] (see Fig. 17).

Because of the characteristic that nodes in the network can reach each other through short paths these networks are called small-world networks. This fact was suspected by several authors and probably first stated dat-

ing back as far as 1967 by Milgram [118]. Later on with more elaborate techniques (computer power) the fact could be demonstrated by analysis of data describing networks (see by Barabási [119] and [120]). Recent results show that virtual social networks (like Facebook) do not really follow that typical pattern [121]. However, concentrating on real-world communities the considerations are still valid.

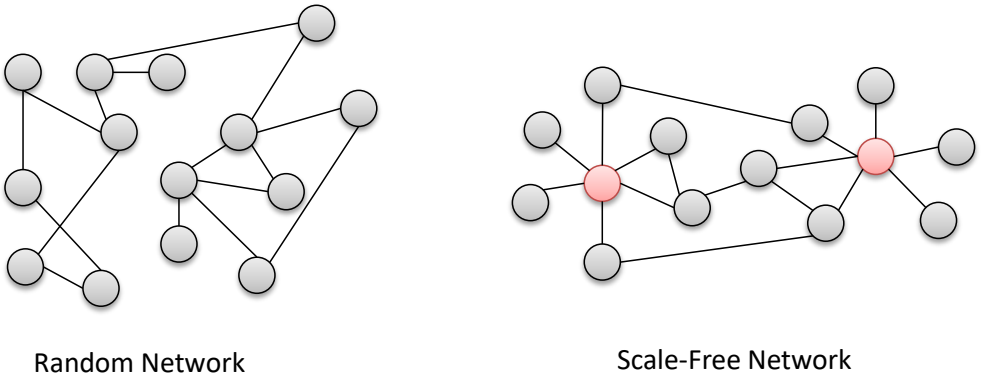


Figure 17: Random versus scale-free graphs.

Mapping communities to graphs (members to vertices, relations to edges) often leads to a graph that consists of few highly connected nodes and a majority of nodes with only a small number of neighbours i.e. scale-free network. The distribution of the node degrees (number of neighbours of a node) follows a power law distribution. Barabási and collaborators coined the term *scale-free network*, to describe the class of networks that exhibit a power-law degree distribution. *Scale-free* means that there is no intrinsic scale in these networks. The most commonly used definition of a scale-free network is "A network is called scale-free network if its probability degree distribution $P(k)$ is asymptotically for large k (degree) a simple power law of the form: $\lim_{k\to\infty} P(k) \propto k^{-\gamma}$ where by the definition of power law, $\gamma > 0$." [122, p. 18]

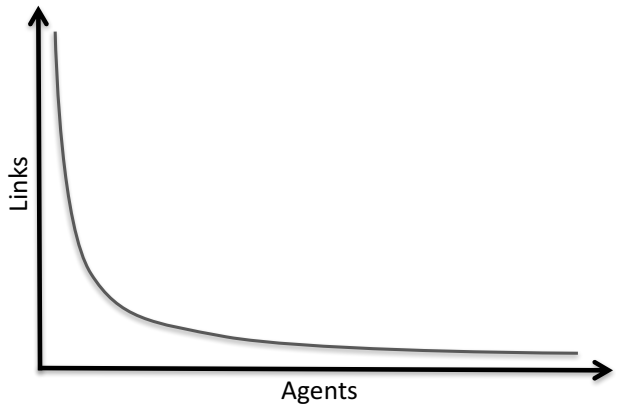


Figure 18: Power-law graph.

Fig. 18 shows an example power-law graph, being used to demonstrate ranking of popularity. To the right is the long tail, and to the left are the few that dominate.

4.1.2 Preferential Attachment

Observing that communities rather evolve over time than being created in a single act, Barabási's preferential attachment algorithm is an appropriate means for creating small-world networks.

The following describes the preferential process: Nodes are added successively one at a time. When a new node is added to the network it creates m edges (m is a parameter which is constant for all nodes). The edges are not placed at random but preferentially, i.e., with a probability p_i, this new node is linked to an existing node i with in-degree k_i, according to a probability proportional to its (in-)degree (degree for an undirected graph or in-degree for a directed graph) relative to the (in-)degree of the other nodes (eq. 1).

$$p_i = \frac{k_i}{\sum_j k_j} \tag{1}$$

Fig. 19 illustrates the preferential attachment process. Starting from three connected nodes ($t = 1$), in each timestep a new node (shown as an empty circle) with $m = 2$ is added to the network. In timestep $t = 2$ each node has therefore an in-degree of 2 and the total number of edges is 3, then each node has the same probability $p_i = 0.\overline{66}$ for node k_i ($p_{1,2,3} = \frac{2}{3}$). In step $t = 3$, k_1 and k_3 as well as k_2 and k_4 has the same in-degree. The total number of edges is 5. Therefore the probability $p_{1,3} = \frac{3}{5} = 0.6$ and $p_{2,4} = \frac{2}{5} = 0.4$ and so on. In other words, it is more likely to see the new nodes prefer to attach to the more connected nodes – *preferential attachment*.

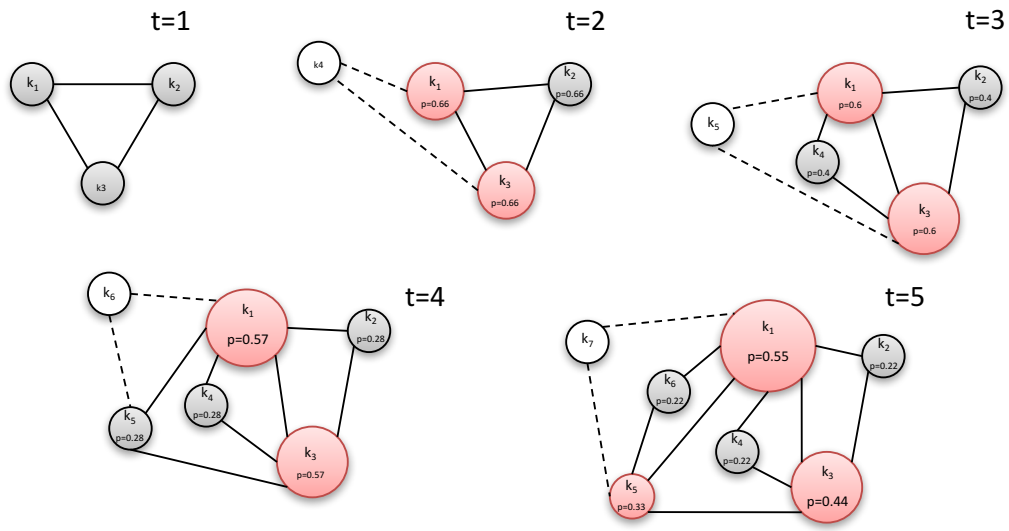


Figure 19: Preferential attachment.

A slightly modified version of the algorithm is used that creates *directed* arcs whose tails are chosen randomly with a probability that depends on the number of nodes already linked to that node. This models the fact that a newcomer will more likely get connected with someone popular (directed link). The hubs are the result of *the rich get richer* phenomenon [120].

Adjacency matrices are likely to be used to represent a graph. The preferential attachment is used to directly operate the adjacency matrix

with entries of 1 and 0. The entry 1 in all i, j indicated that an edge with head i and tail j exists. The adjacency matrix for iteration 5 in Fig. 19 ($(i = 5)$) will be as follows:

$$adjacencyMatrix = \begin{array}{c} \begin{matrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & k_7 \end{matrix} \\ \left(\begin{array}{ccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \right) \begin{array}{l} k_1 \; i_1 \\ k_2 \; i_2 \\ k_3 \; i_3 \\ k_4 \; i_4 \\ k_5 \; i_5 \\ k_6 \; i_6 \\ k_7 \; i_7 \end{array} \\ \begin{matrix} j_1 & j_2 & j_3 & j_4 & j_5 & j_6 & j_7 \end{matrix} \end{array}$$

The algorithm presented in pseudocode shows the implementation of preferential attachment as it is used in the scenarios.

Algorithm 1 Preferential attachment. Build Adjacency Matrix Function.

```
Ensure: gamma ≥ 1
  function createAdjacencyMatrix(nodes, gamma)
    admat: two-dimensional array          ▷ adjacency matrix
    for i = 0, ..., nodes − 1 do
      pvect: array                        ▷ probability vector for node i
      csum: array                         ▷ sum of the columns (i.e. in-
                                            degree of node i)
      tsum = 0                            ▷ auxiliary variable for the total
                                            sum of existing links in the
                                            social network
      for j = 0, j < i, j + 1 do
        for k = 0, k < i, k + 1 do
          csum[i]+ = admat[j][k]          ▷ sum of entries with value 1 for
                                            column j in admat
        end for
        tsum+ = csum[j]                   ▷ add it to the total sum of ex-
                                            isting links
      end for
      for j = 0, j < i, j + 1 do
        if i > gamma then                 ▷ special case for a new social
                                            network (i.e. current number
                                            of nodes < number of edges)
          pvect[j] = csum[j]/tsum         ▷ probability of node j
        else
          pvect[j] = 1                    ▷ when current number of nodes
                                            is less then number of edges,
                                            each node will have the same
                                            probability
        end if
      end for
      pick(admat, i, pvect, gamma)        ▷ build links to existing nodes
    end for
  end function
```

Algorithm 2 Preferential attachment. Pick Function.

```
function PICK(admat, i, probs, gamma)
    count = 0                          ▷ break if all links have been built per node
    tries = 0                          ▷ auxiliary variable for the probability array
    while count < gamma & count < i do
        if randomValue() < probs[tries%i]   ▷ throw the dice and if the value is below the probability then create a directed link between two nodes
            & admat[i][tries%i] == 0 then
            admat[i][tries%i] = 1      ▷ create directed link
            if randomValue() < δ then  ▷ δ represents the probability of creating a symmetric link
                admat[tries%i][i] = 1  ▷ create 2nd directed link
            end if
            count + 1                  ▷ increase the link counter
        end if
        tries + 1                      ▷ increase the number of tries for building links counter
    end while
end function
```

4.2 Stimulus-Response Model

One of the commonly accepted consumer buying behaviour models is the so called stimulus response model S-R [123, p. 24]. Here stimulus and response can be interpreted as cause and effect of consumer behaviour. Mapping S-R to the agent paradigm means that a desire is created in an agent through its sensors or internal state and that this agent has plans how to satisfy it. The template of the behaviour of such an agent can be described as follows:

The agent searches a rule whose condition matches the corresponding situation and performs the appropriate action.

The response part of S-R can be further elaborated by modelling the

Algorithm 3 S-R-Agent.

```
function S-R-Agent(percept)
    input interpret-input(percept)
    rule rule-match(input)
    action description-action(rule)
    return action
end function
```

decision-making process. According to [124, pp. 1–20] decision-making includes the following five steps:

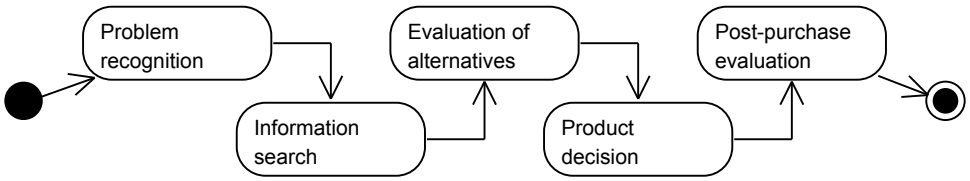


Figure 20: Linear Decision-Making process. In practice it may contain loops and any step may be linked to previous process steps.

Problem Recognition.

The first phase *problem recognition* in decision-making process meaning if there is no problem, there is no action. This situation arises when a consumer is happy with his current product. However, not all the problems end up as buying behaviour depending on the level of importance he attributes to the need. The recognition of a problem can be caused by internal stimuli or external stimuli:

Internal stimuli is a need for change. It may occur when a consumer is not satisfied with his current situation any more. This situation is modelled by a *happiness value*. This value is used as an indicator of how likely an agent will start an action, e.g. if a consumer is not

satisfied it will start a buying action. This is handled by individual thresholds for happiness i.e. if the happiness value is lower than the given threshold the agent becomes active following its plans trying to make amends by starting actions. The happiness value deteriorates continuously over time to model internal stimuli.

External stimuli is a stimuli that may arise from the environment e.g. advertising.

Information Search and Evaluation of Alternatives.

The phases *information search* and *evaluation of alternatives* (i.e. evaluate the most suitable to their needs and choice (i.e. *product decision* the one he think it is best for him) are described in detail in the next sections.

Post-Purchase.

In the last activity *post-purchase* the consumer will evaluate the adequacy of his decision.

The decision process can then be implemented as follows:

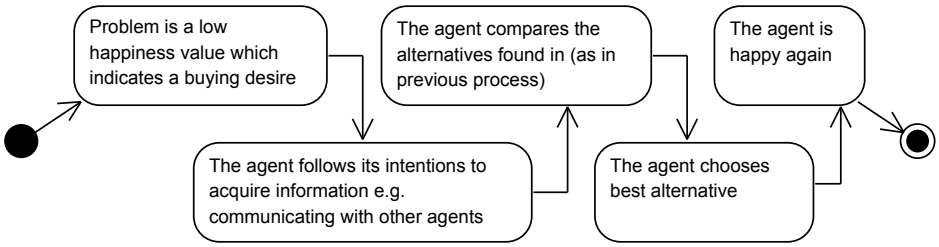


Figure 21: Agent decision-making process. This is a linear process stage representation. In practice it may contain loops and any step may be linked to previous process steps.

The agent decision-making process represents the most basic behavioural pattern of consumers. The following scenario in Fig. 22 demonstrates what this mechanism might look like in practice.

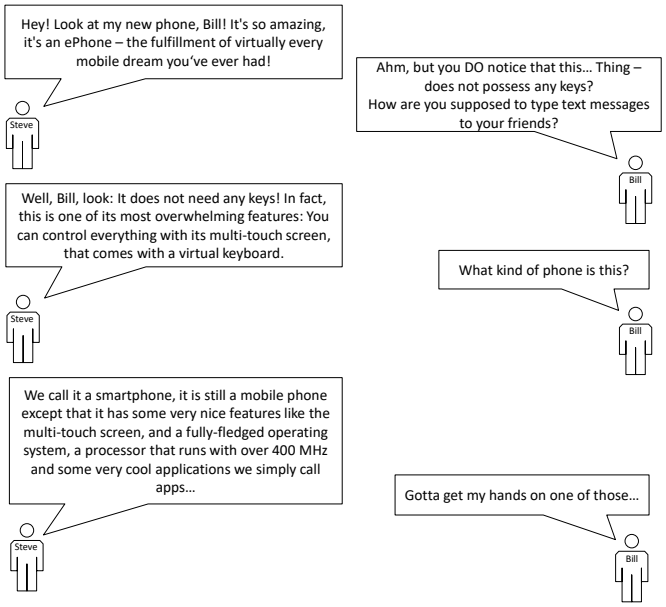


Figure 22: A demonstrative conversation between Steve and Bill.

In this short conversation Steve sets a stimulus in Bill by showing him his amazing new mobile phone. Bill's first response is reluctance to accept the new features of the phone. Then he starts gathering information and finally he does indeed want to have the mobile phone.

4.3 Opinion Leadership and Buying Influence

The spread of information through multiple channels in a social system is a key mechanism driving the diffusion of innovations [125]. The distribution of "news" spread fast through a small world network which follows a power law distribution, because the average distance of two nodes measured in nodes in the shortest connecting path is comparatively small [126].

The *information search* phase of a consumer can be structured as follows: A consumer may acquire information about a product from his neighbourhood. If the person contacted cannot answer the request adequately he can, alternatively, delegate a request to one of his neighbours. The process of passing information from person to person (e.g. about a product) is often called *word-of-mouth*.

A common model for the phase of *information search* (see Section 4.2) follows the so called opinion leadership [124, pp. 8-9]. Among social structures, influence depends on an individual's expertise, status or power. In social networks which do have a small world like structure hubs often act as opinion leaders [127, p. 129]. According to Katz who first coined the concept of opinion leadership, opinion leaders evolve because of their values, their competence or their social relations. The latter is modelled for a node with a high in-degree indicating its important role in the community (hub).

> "Opinion leaders are said to be most influential in the word-of-mouth process." [128, p. 6]

Influence is not necessarily mutual and the level may vary depending on individuals. Therefore it is described in a non-symmetric *influence matrix*. Just as the adjacency matrix the influence matrix is indexed with the graphs nodes in row and column. The entry in cell i, j is zero if i and j are not connected and contains a measure for j's influence on i otherwise. A hub's influence on others is probably higher because of the social status that is ascribed to such a position assuming that the influence of a person on another person depends on popularity. Instead of working with an influence matrix with predefined values, structural aspects to compute a popularity factor for each node are used. Of course someone who is popular among other popular members of a community is more important than someone who is only popular in the eyes of the wallflowers. This line of argument leads directly to the page rank algorithm published by Page et al. The hyperlink matrix L is the transpose of the adjacency matrix where entries are divided by the sum of the entries in the corresponding column. This matrix obviously is a Markov matrix and Eigenvalues and Eigenvectors can be calculated iteratively. To guarantee convergence and to allow random effects the following iteration is used [129]:

$$p^{k+1} = \frac{1-\alpha}{n}e + \alpha L p^k$$

Starting with a vector that contains a valid probability distribution the limit of the sequence will yield a probability distribution over the set of agents whose values can be used to compute the influence matrix. In the formula α is a damping factor between 0 and 1, e the vector with 1 in each

component, and n the number of nodes. Nodes with out-degree 0 will get a value of $\frac{1}{n}$ in each corresponding cell in matrix L. Otherwise the matrix would not be a Markov matrix.

This influence matrix where the individual weights of influence can be properly represented was the main focus of the investigation for the reference scenario *opinion leadership* (see Section 7.1.1). Parts of this section have been published in the MATES 2014 Springer Lecture Notes in Computer Science (LNCS) [130].

4.4 Personal Preferences

General market segments consist of buyers and sellers who demand and offer competing products. Consumers compare attributes of products (i.e. price or technical features of available products) and try to rank them according to their personal preferences (see [131, pp. 202-204] and [132]). Since consumers often have specific multi-dimensional requirements on products (e.g. price, product weight, product quality, etc.) a multi-criteria comparison for product properties is considered.

To enable multi-criteria comparison quantifiable attributes are normalised to points for comparison using the span between the highest and the lowest value that appears among the described products of one kind. Let $a_1, ..., a_n$ be attributes of an object and p_i the corresponding calculated values. The *weighted preference value* of that object is the sum $\sum_{i=1}^{N} w_i \cdot p_i$ where $\sum_{i=1}^{N} w_i = 1$. By definition it lies between 0 and 1.

For example: let the camera resolution values within a fictive mobile phone market segment range from a minimum value of 4.1 megapixels to a maximum of 20.7 megapixels. The normalised value of a camera resolution of

15.9 megapixels is then calculated as follows: the actual difference between 15.9 and 4.1 ($15.9 - 4.1 = 11.8$) is divided by the difference between the maximum value of 20.7 and the minimum value of 4.1 ($20.7 - 4.1 = 16.6$): $\frac{11.8}{16.6} = 0.7108$. With this calculated value a camera resolution with 15.9 megapixels can be estimated to lie in the upper third quantile. But obviously consumers will base their buying decision not only on one attribute. Each consumer weighs different characteristics of a product with different importance. To take these individual preferences into account the criteria are weighted with weighting factors between 0 and 1 which sum up to 1. In the given example the camera resolution may be weight with a factor of 0.3 leaving 0.7 for other attributes. The calculated value of 0.7108 is multiplied by the individual comparison factor of 0.3: $0.71 \cdot 0.3 \approx 0.213$. Figure 23 illustrates the calculation of the example.

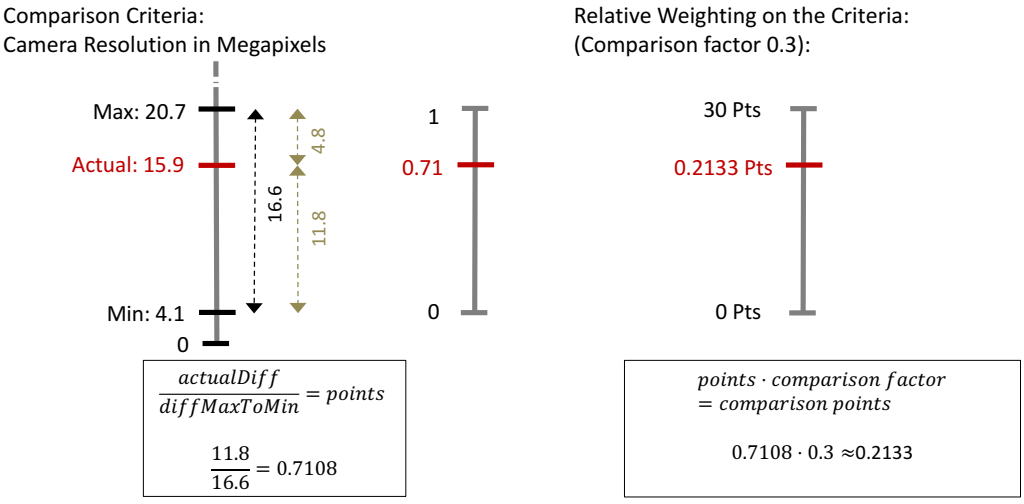


Figure 23: Calculation of points for comparison.

This multi-criteria comparison is used in the *alternative evaluation* phase. Each consumer can now attribute individual importance to each attribute which is relevant to his decision. This approach was used for the *personal*

preference scenario (see Section 7.1.2). A version of this section has been published in the PAAMS 2015 Springer Lecture Notes in Artificial Intelligence (LNAI) [133].

4.5 Summary of Chapter 4

Chapter 4 has given a brief overview of social structures, marketing theories and behavioural patterns relevant to the simulations used for the experiments.

Section 4.1 described social structures and introduced small-world networks (see section 4.1.1). A model that describes networks with power law degree distribution was given in Section 4.1.2 (i.e. the preferential attachment algorithm from Barabási and Albert).

Section 4.2 described the S-R model. A template of a rudimentary S-R agent is given and the decision-making process is described and demonstrated with a demonstrative conversation between Steve and Bill. The *information search* phase is discussed more in detail and the process of passing information from person to person (i.e. word-of-mouth) is introduced.

Section 4.3 shows a common model for the phase of *information search* (i.e. opinion leadership). Followed by a discussion that influence is not necessarily mutual an *influence matrix* was developed which is based on the page rank algorithm.

Finally, Section 4.4 a multi-criteria comparison for multi-dimensional consumer preferences on products is considered. Each consumer can then attribute individual importance to each product attribute.

5 Agents and Ontologies

This chapter describes combination of agent and ontology techniques. The architecture developed here can be extended into a comprehensive framework. The behaviour of agents can be modelled declaratively with semantic technologies.

5.1 Fundamentals

5.1.1 Agent Design Methodology

An important benefit of agent-based modelling is that interactions between agents can be simulated directly. With regard to Section 2.1 agents focus their activities on achieving given goals while acting according to available plans. Such a goal is a desired state that can be reached by use of these plans. An appropriate paradigm for the development of agents is the BDI concept. It is characterised by the implementation of an agent's **b**eliefs, **d**esires and **i**ntentions which can be used to model aspects of human behaviour [15]. The core concepts – beliefs, desires and intentions – are those which we as human beings naturally use to explain the reasoning of both ourselves and others. In a BDI model agents are endowed with beliefs about the environment and fellow agents in that environment, with intentions to execute actions structured into plans and desires, which represent the outcomes the agents want to achieve. A consistent subset of desires forms the agent goals, towards which plans should be developed.

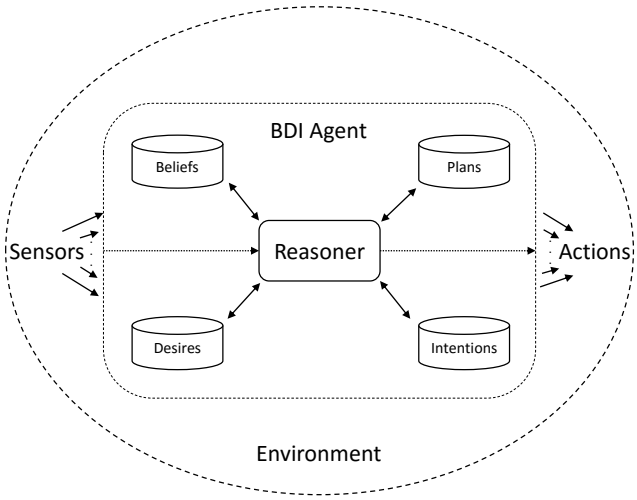


Figure 24: A high-level abstraction of a BDI agent (adopted from [134, p. 13]).

Fig. 24 illustrates the core elements of a BDI agent. They are as follows:

Beliefs represent the informational state, which comprise its knowledge about the world (knowledge base).

Desires represent the motivational state of the agent; in other words it defines the things that the agent may wish to achieve – goals.

Plans is a set of available actions. A goal can be achieved by using one or more plans.

Intentions of an agent are the particular plans that the agent has committed to performing in order to achieve its goals.

Reasoner is the engine which bundles together the previous four components. It receives data (e.g. communication), updates beliefs and goals, selects next agent actions.

The agent belief base stores everything an agent knows (or believes to know) about the environment it lives and acts in. Basically, modellers have to express knowledge (as well as desires, plans and intentions) in agent

frameworks in the respective programming language i.e. Java. This increases workload and potentially reduces re-usability, because the abstraction level of modelling such knowledge is often limited. Ontologies now offer a standardised way to represent knowledge in general. This knowledge can be used for specifying the things that exist and how these things are related to each other (i.e. domain knowledge). Furthermore, the ontology defines how conclusions can be drawn (i.e. inference knowledge) by using available reasoning instruments. Consequently, ontologies are a promising tool to model agent belief bases.

5.1.2 Rules

Rules extend the expressiveness of OWL. Such rules follow the form of *IF-THEN*-constructs and allow the expression of various kinds of complex statements. Semantic Web Rule Language (SWRL) is a language for the definition of rules which combines OWL with a subset of Rule Markup Language (RuleML)[7]. All rules are expressed in terms of OWL concepts (classes, properties, individuals). SWRL is an essential component of this work, therefore discussed in more detail at this point. It is a high-level abstract syntax for Horn-like rules. A Horn clause is a clause (a disjunction of literals) with at most one positive, i.e. unnegated, literal. SWRL atoms are defined as follows:

$$\mathcal{C}(i)|\mathcal{D}(v)|\mathcal{R}(i,j)|\mathcal{U}(i,v)|builtIn(p,v_1,\ldots,v_n)|i=j|i\neq j\rightarrow Atom$$

with:

$\mathcal{C}$ Class

$\mathcal{D}$ Data type

[7] http://ruleml.org

$\mathcal{R}$ Object property

$\mathcal{U}$ Data type property

i, j Object variable names or object individual names

$v_1, \ldots, v_n$ Data type variable names or data type value names

p Built-in names

Rules are of the form of an implication between an antecedent (b_s, body – the *if*-part) and a consequent (h, head – the *then*-part): $b_1, b_2, \ldots, b_n \rightarrow h$. The following example shows a SWRL rule where a person is classified as a *PersonWithChild* when a *hasChild* relation between two individuals exists. The question mark designated an object variable name or a data type variable name.

$$\text{Person(?x)} \wedge \text{Person(?y)} \wedge \text{hasChild(?x, ?y)} \Rightarrow \text{PersonWithChild(?x)}$$

Note that if domain and range of the object property *hasChild* are appropriately set to *Person* then the terms *Person(?x)* and *Person(?y)* are optional, because then the reasoner already ensures that the variables x and y are related to concept *Person*. The main SWRL-API developer O'Connor would call this rule as OWL syntactic sugar [135], because some SWRL rules can also be represented as DL axioms:

$$\text{Person} \sqcap \exists\ \text{hasChild}.\exists\ \text{Person} \sqsubseteq \text{PersonWithChild}$$

However, there are rules which cannot be formulated with OWL DL:

$$\text{Person(?x)} \wedge \text{Person(?y)} \wedge \text{hasParent(?x, ?y)} \wedge \text{hasSister(?y,?z)} \Rightarrow \text{hasAunt(?x,?y)}$$

The reason for this is that the consequent has two different variables ($?x, ?z$). Basically, the formulation of a SWRL rule which consists of more than one shared variable in antecedent and consequent translation to OWL DL is not possible (see [136, p. 204]). Note that SWRL rules can lead to undecidability if they are not DL-safely formulated. Whenever it is ensured that all variables in the consequent are used in the antecedent as well as the underlying rule interpretation system (i.e. reasoner) can assign facts to the variables, is an elementary understanding of DL-safe rules. The solution is then to apply DL-safe rules, wherein each variable must occur in a non-DL-atom in the rule body, i.e., DL-safe rules are SWRL rules restricted to known individuals. Considering the following rule as it may not be immediately obvious why variables in a SWRL rule would ever bind to anything other than known facts (cf. [137]):

$$\text{Agent(?x)} \wedge \text{Phone(?y)} \wedge \text{hasPhone(?x,?y)} \Rightarrow \text{AgentWithPhone(?x)}$$

This rule classifies any individual of concept *Agent* as an *AgentWithPhone* if it has an associated *hasPhone* object property with an individual which is belonging to concept *Phone*. Let the ontology be extended with the concept *Person* which is sub-concept of *Agent*. It has an associated restriction (*hasPhone* `some` *Phone*) formulated with DL. A single individual belonging to this concept in the ontology is defined. Because of the rule that states that an

agent that *hasPhone* belongs to the concept *AgentWithPhone* all individuals which belong to the concept *Agent* will be classified as *AgentWithPhone* as soon as the relation to a phone individual gets established. However, since there is no specific phone declared for the *hasPhone* property, a DL-safe implementation of a SWRL reasoner would not infer that the person is an *AgentWithPhone* meaning that the variable *?y* in the rule would be bound to an individual that is not explicitly known. Thus DL-safe rules may produce incomplete inferences, but this restriction of SWRL rules guarantees decidability which is more important. This limitation does not apply as long as the modeller is aware of this behaviour and restricts the rules to known facts. Current reasoners focus on a DL-safe implementation of SWRL. DL-safe rules look exactly like normal SWRL rules.

5.2 Incorporating Ontologies into BDI-Agents

Again, in classical implementations of multi-agent systems following the BDI architectural pattern built with frameworks such as Jadex [138], all aspects have to be coded using a conventional programming language (mostly Java) fitting into the hotspots of the framework. Building a BDI agent-based simulation all the agents must be populated with domain-specific knowledge of the world in which they act and live.

The default BDI reasoner of an agent receives sensory data, updates knowledge, forms intentions and selects the agent actions to perform. The following pseudo code describes a simple BDI agent reasoner (adopted from [139, p. 147]):

Algorithm 4 Simple BDI agent reasoner.

```
while unachieved goals do
    observe environment
    update beliefs
    prioritise intentions to achieve
    choose a plan for the intention
    execute and monitor the plan
end while
```

The algorithm consists of a loop of observing the world (sensors), updating its beliefs, prioritising on its intentions, choosing a plan to achieve its prioritised intention, then executing the plan and monitoring its progress. The while loop condition depends on there being unachieved goals. Note that the BDI reasoner is abstract and postpones detailed implementation decision. A goal may also be triggered through agent communication which can also have effects on the beliefs of the agent. Generally, a customisation of aspects of the agent has the consequence that the project has to be rebuilt which increases workload and potentially reduces reusability (see Section 2.2).

The use of a declarative rule language such as OWL allows the strict separation of concerns where ontological definitions and rules can be separated from behavioural patterns. On the one hand we have the world of ontologies with $\mathcal{O} =< \mathcal{C}, \mathcal{R}, \mathcal{I} >$ and on the other hand the world of agents with $Agent = < \mathcal{B}, \mathcal{D}, \mathcal{I} >$. Social aspects and information about internal aspects of the agent (e.g. its current state) are fully mapped to the ontology (see Fig. 25): the set of beliefs (i.e. knowledge), desires (i.e. goals) and intentions (i.e. plans of how to reach the goals). The modelling notation used for the next and following figures which includes OWL elements are listed in Appendix C.

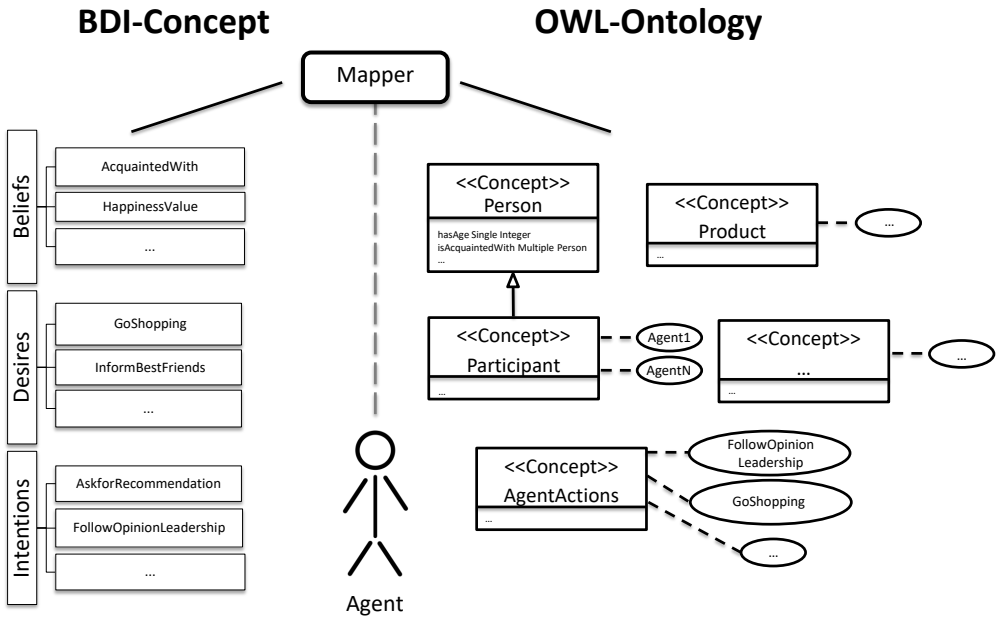


Figure 25: OWL-BDI-Mapping.

Factual knowledge of the world of an agent is represented by individuals and corresponding relations (i.e. set of beliefs). Relations can be restricted by specifying domain and range. Data properties link concepts and individuals to primitive data e.g. strings or numbers (i.e. the age of a person). Desires are represented as individuals of the concept *Desire* (i.e. goals) and intentions are represented as individuals of the concept *Intention* (i.e. a sub-set of the goals with an associated stack of plans for achieving them – the intended actions). The concrete actions an agent may carry out to reach its desire are described in plans (i.e. *AgentAction*). Thus desires have an object relation with domain set to *Desire* and range set to *AgentAction*.

Fig. 26 shows a rudimentary example of an BDI-OWL agent. The agent is represented as an individual of concept *Agent*. It has acquaintances to other agents linked through the object property relation *isAcquaintedWith*. A simple SWRL rule has set the object relation *hasAgentAction* to the

individual *GoShopping* of concept *AgentAction* for individual *myself*. The multi-agent framework has then to trigger the corresponding intention.

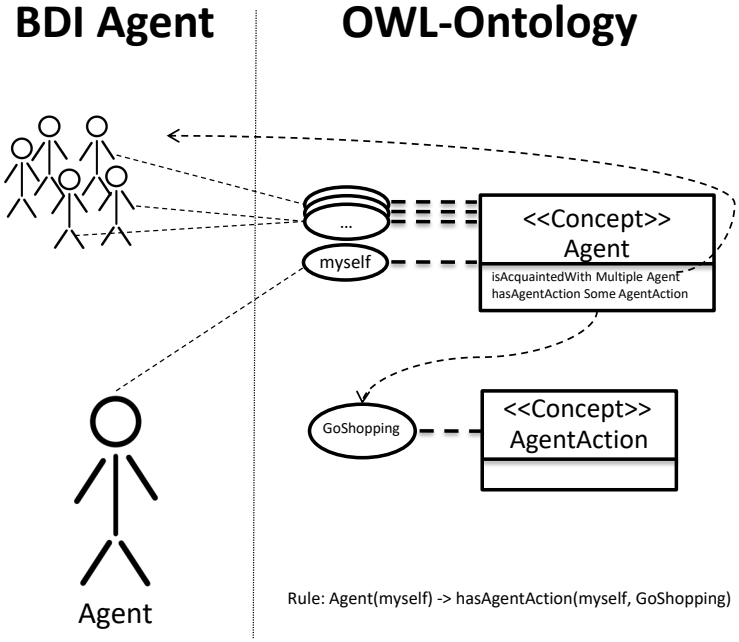


Figure 26: Small example of OWL / BDI agent model.

To summarise, the ontology and its inference mechanisms are used to determine the behaviour of an agent e.g. rules are used to determine plans and calculate actions. Each agent has its individual ontology while it is ensured that agents have a common understanding of the environment by providing commonly shared elements. This is implemented using a layered approach which will be discussed in the next section.

5.3 Layered-Ontology Model

This section is about the development of a universally applicable integration of semantic technologies and agent based systems. The layered approach is to construct a blueprint for an architecture that can easily be adapted to various simulation scenarios. Parts of this section were published in [130].

Fig. 27 shows a layered ontology model in which domain knowledge can be hierarchically separated by its degree of generality:

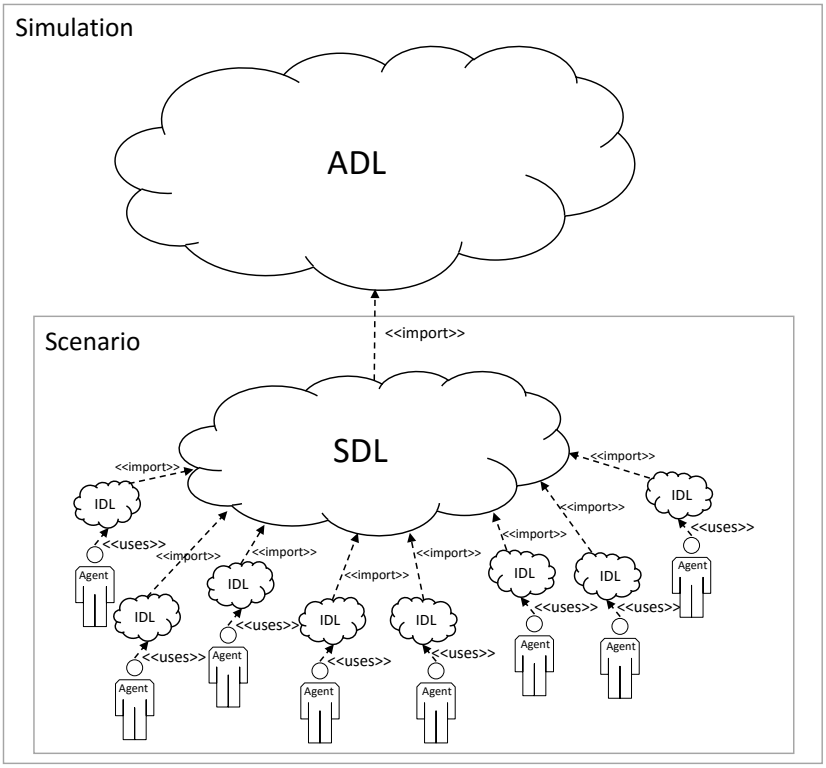


Figure 27: Three-layered ontology.

Abstract Domain Layer (ADL) is the top-level ontology which describes the domain-independent part and consists of general abstract concepts (see Fig. 27). The root concept of each ontology is *Thing*. It represents the basic knowledge i.e. concepts that cannot be assigned to a particular domain, and their relationships among themselves. In this context, the concept *ValuePartition* should be noted which is also related to the ADL. Value partitions are not part of ontology languages, they are a proven solution of a partition with disjoint subsets and domain independent. They can be created to refine a concept description. For example: *acquaintance* may describe the level of acquaintance between

individuals. The value partitions restrict the range of possible values to a list: *casual*, *friend* and *bestFriend*.

Specific Domain Layer (SDL) refines ADL by specialising abstract elements of ADL to fit the requirements of a specific domain. All concepts, relations and individuals on this layer are restricted to a specific product market e.g. mobile phone market which means that these elements are used as reference by agents and agent engineers.

Individual Domain Layer (IDL) expresses the individuality of each agent. It contains beliefs and definitions of individual behaviour of each agent (e.g. how an agent reacts to a certain stimulus).

Fig. 28 shows the three-layered ontology with example concepts on each layer. According to the concept hierarchy, the specialisation regarding its different layers becomes clear.

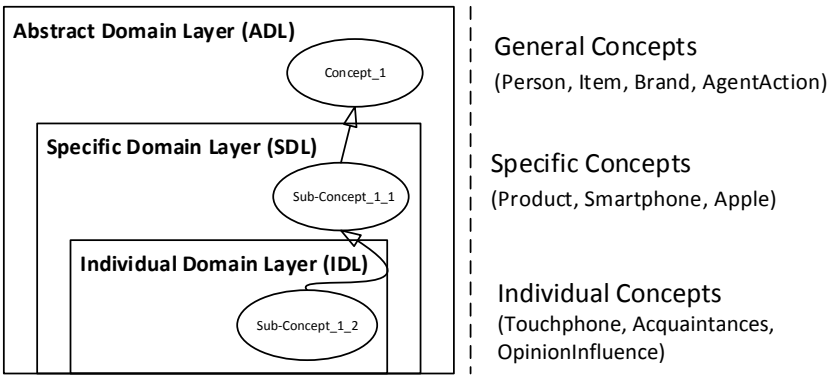


Figure 28: Three-layered ontology architecture example.

For example: the general concept *Item* is specialised in the environment of a mobile phone marked by *Product*. Since mobile phones within this domain are the relevant products, the concept *Product* is specialized by *Smartphone*. If an individual knows further specialisations (or synonyms), these are mapped within the IDL (in the example, the concept *Touchphone*).

By use of the `owl:import` statement ontologies can refer to another OWL ontology which contains definitions, whose meaning is considered to be part of the meaning of the importing ontology. The statement is transitive i.e. if ontology IDL imports SDL, and SDL imports ADL, then IDL imports both SDL and ADL. From a mathematical point of view the set of general concepts is a subset of the specific knowledge available to an agent. By separating knowledge into layers, general control of the simulation is kept independent of specific terms of a given scenario (see Section 7).

An important function of the ADL is to support broad semantic interoperability among the large number of individual market domains by providing a common starting point for the formulation of agents. In information science, the ADL would be then called as an upper ontology or top-level ontology. An upper ontology such as the ADL supports broad semantic interoperability among a large number of domain-specific ontologies by providing a common starting point [140] for the formulation of SDL ontologies as well as IDL ontologies.

The layered approach is mirrored into the Java application that implements the BDI concept (see Section 6).

5.4 Communication and Learning Capability

Interaction through communication between agents is one of the most important features of multi-agent systems. Usually, agents do not know *everything* in a multi-agent environment, but may extend their knowledge through social interaction i.e. communication.

While using ontologies as the main belief base, agents can communicate with each other by means of a common ontology. An ontology is machine

readable, and supports agent communication by defining and providing a shared vocabulary to be used in the course of communication, they also provide a definition of the terms that can be used in communication; ontologies also provide the definition of the world in which an agent grounds its actions. Different agents of a system can reach a shared understanding by committing to the same ontology. This enables them to make statements, communicate knowledge and make different queries. Use of ontology permits coherent communication and easier information sharing between agents, enabling agents to cooperate and coordinate their actions. The use of ontologies in message based communication gives meaning to the contents of messages sent between agents.

Agent knowledge is limited to what is defined in the hierarchy of ontologies possibly differing from what other agents know: Each agent is equipped with shared ontologies (common knowledge in ADL and SDL) both agents have a basic common understanding of the current domain. But the IDL is private and may differ from each agent. An agent may extend its knowledge base during a simulation, meaning that it has learning capability i.e. communications which refer to knowledge items that belong to the IDL layer. Therefore, agents can exchange information which contains concepts that may be new to the receiving agent. The receiving agent may then add new facts acquired through this information exchange into its belief base. When incorporating a new concept into its IDL the agent has to obtain all available information relating to that concept. Concepts with a direct superclass in ADL or SDL can easily be added to the IDL of the learning agent. If the concept does not have direct ancestors in ADL or SDL the super classes of the sending agent must also be included. Individuals and facts (properties)

about individuals can be added directly, if they are instances of a concept defined in ADL and SDL.

To summarise: Let o_1 and o_2 be individual ontologies. The intersection $o_1 \cap o_2$ is uncritical because it is obviously available to both agents. From the perspective of o_1 the set $o_2 \setminus o_1$ is critical, because it contains elements of $\mathcal{C}$, $\mathcal{R}$ or $\mathcal{I}$ which are relevant for the learning process.

Algorithm 5 Add information to knowledge base.

```
procedure LEARNING(Individual i)
    if ontology not contains i then
        addIndividualToIDL(i)
    end if
    for all dataProperties dp of i do
        if ontology not contains dp with i then
            addDPToIDL(dp,i)
        end if
    end for
    for all objectProperties op of i do
        if ontology not contains op with i then
            addOPToIDL(dp,i)
        end if
    end for
end procedure
```

Algorithm 5 shows a rudimentary but suitable learning strategy in pseudo code. Concepts, properties and individuals that do not exist in the ontology of the learning agent can be added to its private ontology directly. For the time being, the algorithm is restricted to directly adding unknown elements to the relevant ontology and it has to be ensured, that the sum of knowledge of all individual ontologies does not contain inconsistencies. Inconsistency is a severe error and the ontology cannot be instantiated. For example: An individual i is of type concept c_1. The learning agent receives the information

that individual i is of type c_2 as well while concept c_1 and concept c_2 are disjoint (see Fig. 29).

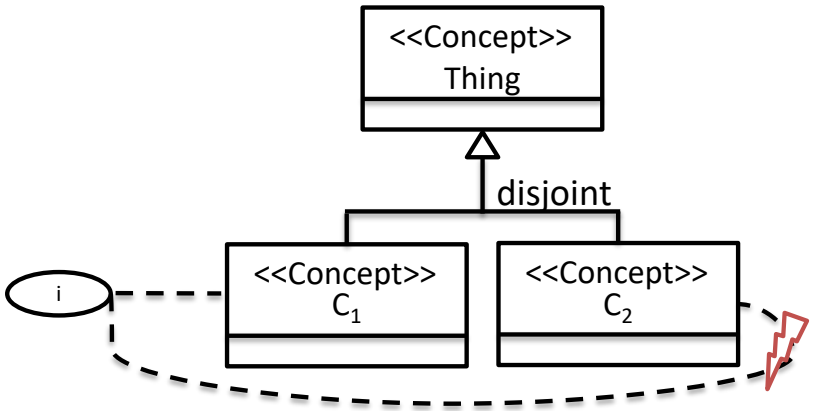


Figure 29: Inconsistency ontology example.

Here, the contradiction is quite obvious as long as the old information will be left in the ontology. It is required that the individual i has to be a member of class c_1 and class c_2 at the same time. For example, Schiemann as well as Noy et al. show variants to resolve such conflicts in their work (see [93] and [141]). However, in this thesis it is expected that every concept in IDL is a subconcept of concepts in ADL – possibly transitively. This is ensured by a Java routine that performs validation checks on the ontologies.

This learning capability has direct effects on the actions of agents e.g. their buying behaviour (see Section 7.1.2). The layered approach enables the possible learning capability described above.

5.5 Summary of Chapter 5

In this chapter, Section 5.1 described the fundamentals of the BDI concept, the concept of knowledge representation in ontologies and the SWRL language for the definition of rules which extends the expressiveness of OWL.

Section 5.2 considered the combination of agent and ontology techniques. A rudimentary example of an OWL-BDI agent model shows how the ontologies and its inference mechanisms are used to determine the behaviour of an agent.

The three-layered ontology model (i.e. ADL, SDL and IDL) is presented in Section 5.3. It shows that the domain knowledge is hierarchically separated by its degree of generality. Abstract general concepts (e.g. thing, person,...) are coded in the ADL while more specific elements (e.g. concepts of a certain market theory) belong to the SDL. Individual beliefs and desires are included in the IDL. Ontological commitment is ensured while ADL and SDL are shared by all agents.

Finally, Section 5.4 considered the ontology based communication between agents. Agents can exchange information that may be new for the receiving agent. A learning strategy is introduced which can be used when an agent has to incorporate new concepts into its private ontology.

6 Agent Software Framework

This chapter describes a concept of a practical integration of semantic technologies applied to an agent-based framework. The agent-based framework called AGADE is introduced that can run BDI agent simulations where agents communicate with each other, have knowledge of their environment and act in structured social environments. This chapter finishes with a demonstration of AGADE.

6.1 Ontology-based Business Simulations Framework

The challenge of running business simulations lies in representing social structures and marketing mechanisms appropriately. For this purpose, references to social structures and marketing theory (see Section 4) are made. When running business simulations, a *social structure* with mutual relations, *market mechanisms*, *individuals* with knowledge and behaviour and a configurable *tool* to incorporate all that are required. For this purpose, the framework AGADE was developed by the author. AGADE was demonstrated at the 13^{th} and the 14^{th} international conference on practical applications of agents and multi-agent systems (PAAMS). Achievements of the framework were presented at the PAAMS conference including two publications ([142], [143]). AGADE received the third award of scientific excellence (see Appendix D).

AGADE is a multi-agent simulation framework which leverages semantic technologies to facilitate a convenient modelling of the desired market context while empowering all actors to utilise their world knowledge for inferring implicit knowledge as well as deducing how to act in a specific situation. Agents are active parts of complex social structures, allowing them to not

only communicate with but also permanently learn from each other.

6.2 Architecture

This section demonstrates how ontologies can be integrated into the BDI concept utilising the Jadex agent framework. Jadex was the tool of choice because it provides a set of convenience tools (logging, monitoring,...), is Java based, and therefore can seamlessly connect to the reasoning mechanisms of the ontologies and its reasoners using the OWL API [144]. While the basic operations of agents are left in Java classes, certain aspects of the BDI agent are shifted into the ontology so that declarative rule languages such as the OWL (see Section 7) can be used.

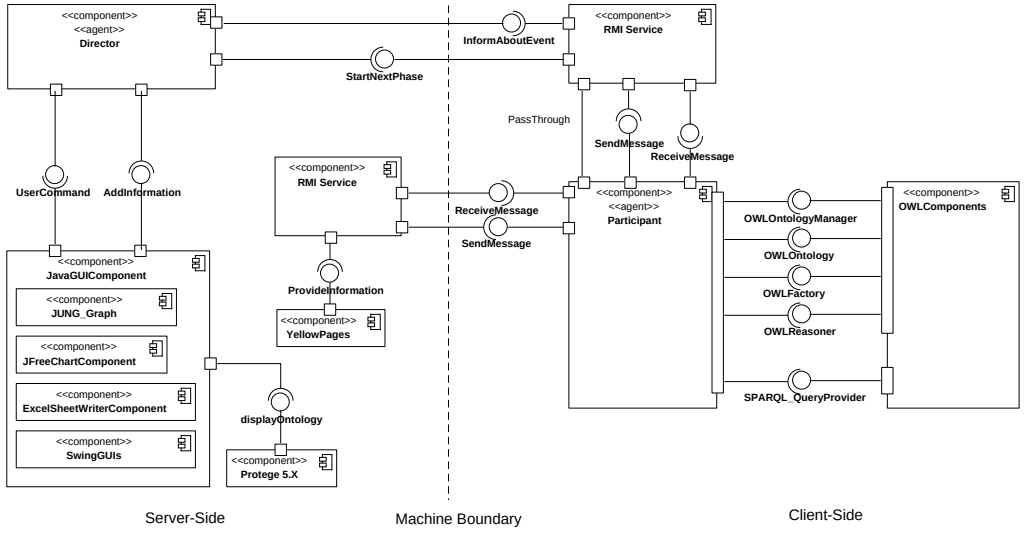


Figure 30: Abstract view on the AGADE architecture.

Fig. 30 shows an abstract component model of the proposed architecture. AGADE knows two different kinds of BDI agents: a director type agent and participant type agent. The director agent represents the central management entity i.e. the coordinator of the simulation. Participating agents comprise

any agent participating in the simulation (i.e. customer and seller). The director agent is connected with a graphical user interface where relevant information about the simulation is displayed (e.g. market share chart) and handles user commands (e.g. start and stop the simulation). The following sections describe the components and mechanisms in detail.

6.2.1 Distributed Environment

Simulation of real-world scenarios often requires a large number of agents. With increasing level of detail and resolution in the underlying models machine limitations both in the aspects of memory and computing power are reached. Even more when additional features like reasoning mechanisms of semantic technologies are used. First experiments have shown that the extensive use of ontologies results in high memory consumption due to the large number of `String` objects used in the reasoning process and caching mechanisms of the OWL API. Mengistu et al. claim that the support for schedulers for managing time, synchronising agents and data collections are crucial elements of multi-agent simulations. The component model as shown in Fig. 30 includes a mechanism that allows the scaling-up of simulations considerably: AGADE implements a Java RMI (Remote Method Invocation) based communication mechanism with which the agents can send and receive messages. RMI is directly based on socket communication and is therefore more efficient than alternative technologies like Web Services [145] or CORBA [146]. Running various benchmarks comparing the alternatives with results similar to what has been published before (e.g. [147]). For example: 1000 calls via Web Services on quad core cpu machines connected to a network with a transmission rate of 100 MBits took $\approx 203,37$ seconds.

Using RMI on the same machine settings 1000 calls were completed considerably faster in less than one second. The "cost" of constructing and decomposing the corresponding request via Web Services is significantly higher than using a socket based communication as RMI. As sending and receiving messages are basis elements of every multi-agent system, choosing an efficient technology for a distributed communication is crucial regarding simulation time consumption. However, when interoperability is a major requirement on a multi-agent system then Web Service is the technique of choice, because of the programming language-independence i.e. Web Services can be implemented using any programming language, and can be run on any platform. Because of the fact that Jadex as well as the OWL API are Java based, AGADE already relies on Java components and therefore RMI can be integrated. The source code which has been used for measuring the performance of the aforementioned technologies can be found in appendix E.

The communication between participating agents and between participating agent and director agent uses RMI based services both in distributed and in non-distributed mode ensuring a unified architecture for the framework.

The typical RMI flow of execution starts with an initial registration of objects in the RMI registry of a server using a unique name which makes the objects available for client access. Clients can now query the lookup service of the *RMI registry* to get a reference to the objects. The client can then invoke appropriately published methods. The roles of server and client are interchangeable i.e. each node in the network can act as client and as server thus allowing two way communication with asynchronous method invocation (see Fig. 31).

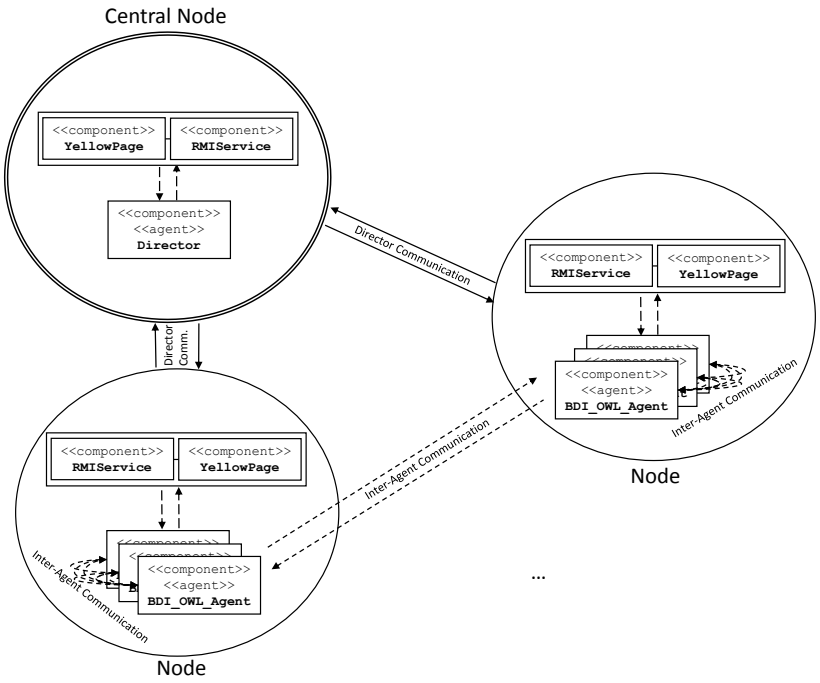


Figure 31: AGADE nodes model.

Each node provides a *YellowPages* service where agents are published to be used in inter-agent communication. Note that two nodes in the network do not necessarily have to be connected through yellow page entries as they may not have to communicate at all during the simulation. But at least one distinguished central node must be aware of all other nodes and can then act as a broker and enable communication if requested. Once communication between two nodes has been established mutual entries are made in the local yellow pages. This *lazy* set up of communication information reduces initial messaging efforts and provides direct links only if requested. The director agent and the GUI communication with the user obviously have to be placed on the central node as well. The delivery process of inter-agent communication is therefore implemented as follows:

- query local yellow pages on client-side and deliver message directly if

address information for both is available

- request broker on central node to provide necessary information and initiate communication

In the inter-client communication scenario, the client is querying the server for the client-address and will then deliver the message to the specific client directly. RMI message mechanisms use Java serialisation to deliver objects. OWL components are represented as `String`s and can therefore undergo the standard RMI serialisation process. Thus the content of each agent message is eventually a serialised instance of `String`.

The typical message content size of an OWL concept is about 850 Byte which is 0.0068 Mbit. Theoretically, a typical network bandwidth of 1000MBit/s allows the sending of $\sim 147,058$ messages simultaneously with respect to the content size without limitations neglecting connection overhead usage. To reduce the amount of network traffic between clients, the message content can be compressed by using *gzip* which reduces the size by roughly 50 percent.

The robustness of this architecture was successfully tested by simulating a homogeneous crowd of buyers acting in a mobile phone market where 100,000 agents were run over 100 rounds on 6 clients each connected with 100 MBits network speed and equipped with various hardware settings (RAM/CPU) (see Section 7.1.1).

6.2.2 Abstract OWL Agent

The layered approach (see Section 5.3) is mirrored into the Java implementation built according to the BDI concept. Class `AbstractOWLAgent` class corresponds to *ADL*. This abstract class defines the interface to the

OWL-Application Programming Interface (API) and connects elements of the ontology to the BDI architecture thus enabling an OWL-BDI agent to participate in AGADE simulations (see Fig. 32). References and methods to maintain ontologies and trigger plans are implemented here. Each agent is equipped with its own ontology reasoner and private ontology. AGADE supports OWL DL ontologies, which give maximum expressiveness while preserving computational completeness and decidability, which both are needed for modelling complex scenarios.

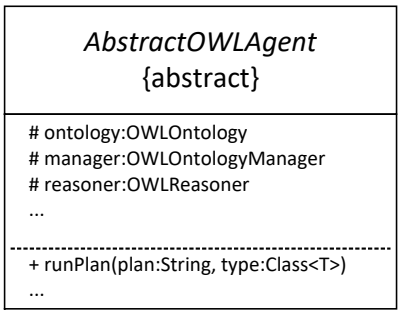


Figure 32: AbstractOWLAgent Java class.

Subclasses of `AbstractOWLAgent` are on the level of *SDL* and specify more concrete aspects of an agent (see Fig. 33). Each subclass references an *IDL* which in turn models the individual behaviour of an agent and describes the type of an agent. For example: when simulating a market place then one general market participant class has to be modelled which distinguishes between seller or customer in the individual ontologies used in the simulation. Additional Java subclasses are possible whenever more-specific categories are required.

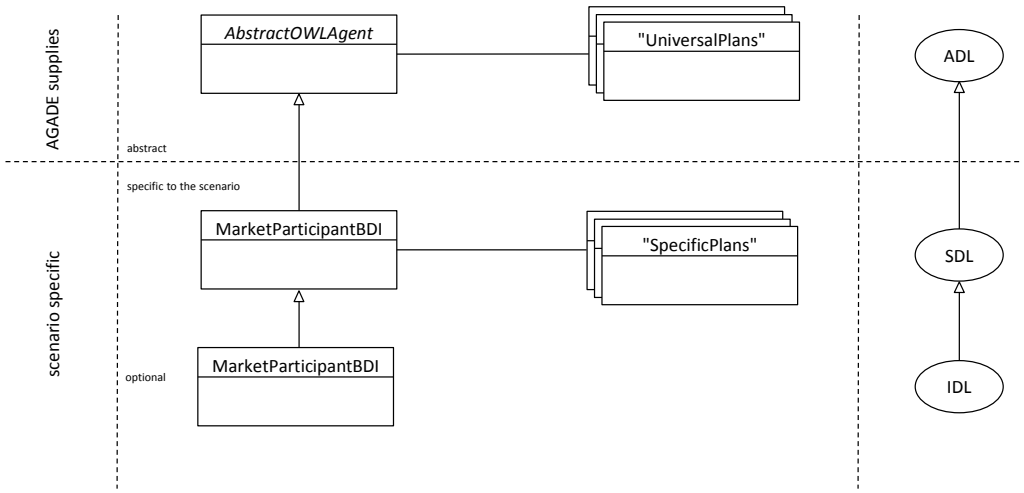


Figure 33: Interaction of layered ontology and agent classes.

6.2.3 Ontology-based Plan Selection

AGADE uses inference mechanisms of the semantic infrastructure (i.e. ontology reasoner) to determine the next actions an agent performs to reach its goals (i.e. desires). Note that while it is more natural to think that agents having goals which in turn are realised through tasks and actions, traditional BDI implementations do not support such explicit modelling of a goal behaviour. Goals are defined independently of other goals and mapped to one or more plans. A BDI plan consists of a sequence of plan steps which have to be expressed in the BDI framework. The decision which goal or at least which plan an agent has to use next is expressed in the ontology. Thus, plans (i.e. intentions) play a central role for BDI agents, because they encapsulate the recipe for achieving a goal. From a modelling perspective, ultimately, agents use plans to reach their goals (i.e. desires). The execution cycle for achieving a goal can be characterised as follows:

1. The selection of a plan (potentially more than one plan available) to

achieve the current goal

2. The execution of that plan and
3. The updating of agent state

Note that BDI agents do not always need to have a desire for running plans. A plan can also be directly considered whenever a condition of a plan (set at design-time by the BDI programmer) is set to true.

In compliance with the Jadex framework possible individual actions have to be denoted as plans. This is done in the Java code that implements the agent. Plans in Jadex can be represented as methods inside the Java class that implements the agent or alternatively as plan Java classes which have to provide a so called plan body method which is used to formulate the sequence of actions. The author recommends the coding of plans as Java classes to keep the agent behaviour pattern as flexible as possible, because plans written in Java classes can easily be made available to different agents by simply adding `@Plan` annotations to the specific agent class thus making them available in other simulations. AGADE can create plan pools out of available classes annotated as plans thus making them available in other simulations.

Each agent keeps a representation of itself in its private ontology. This is realised as an instance of Agent called myself. The ADL defines concepts *Desire* and *Intention* which are modelled as sub-classes of the concept *AgentAction* which is used to express that something that can be executed by some agent. Goals and intentions (sometimes called plans) are expressed as individuals of concept *Desire* and concept *Intention* respectively and can be linked to agents and actions through appropriate object properties. These are *nextAgentAction* (with domain *Agent* and range *AgentAction*)

and *hasIntention* (with domain *Desire* and range *Intention*). Assignment of elements in the range uses rule evaluation. The relevant steps of executing agent actions are shown in Fig. 34. It shows the principal workflow of selecting and executing the agent actions which are formulated in the private ontology and selected by a Java routine. The following paragraphs and sections in this chapter describe the workflow in more detail.

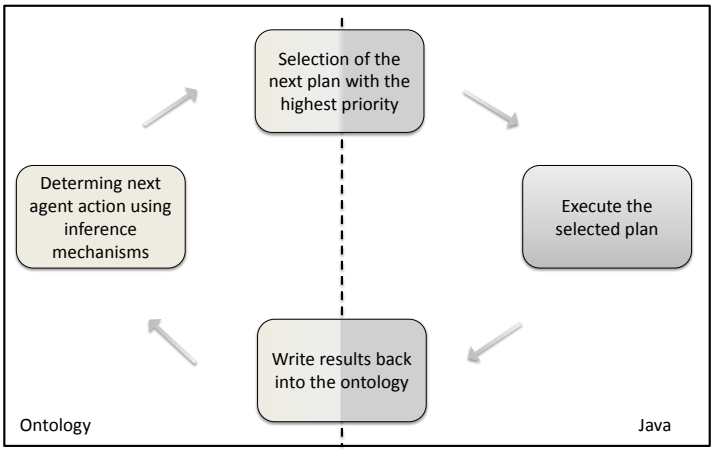


Figure 34: Ontology-controlled plan selection.

An agent may have more than one goal at the same time, which can also be achieved using a set of plans. Typically, the next plan is selected randomly if an agent has more than one next agent action. As SWRL does not support rule ordering, it cannot be used to express any arbitrary order of actions. If an order is required for plan execution (e.g. plan a should be run before plan b is executed), a data property called *hasPriority* with domain set to *AgentAction* and range set to `xsd:integer` can be used to express any particular order of agent actions. E.g. plan a has the priority set to 1 and plan b has the priority set to 2. For this, all next agent actions of *myself* will

be added to a *priority queue*. If a plan has no priority, it will be enqueued with the max-value of the data type `int` (see algorithm 6). Prioritisation is required because OWL works according to OWA meaning that something that is not explicitly known as *true* or vice versa known as *untrue* must be considered as *unknown*. It is therefore assumed that knowledge of the facts is simply missing. In order to illustrate the effect on the selection of the actions, the following situation is mentioned: Considering a condition y, an action a is to be set as the next action to be executed. It would be desirable if, in the event of a non-use of y, action b would be selected instead. However, according to OWA, it must be explicitly stated that y is not satisfied. Again, only the absence of the information on the validity of y does not allow to conclude that y is not satisfied and that action b is to be chosen. By assigning an individual priority value annotation to each action, this can be bypassed flexibly. The selection of a single next action to be performed per simulation round is computed, as described before, via the Java application. Here, the missing information about a condition y can be interpreted as not being applicable. For example: the absence of a priority specification can be interpreted as the lowest prioritisation which is done in a dedicated Java method (`addIntentions`). This then corresponds to the CWA.

Algorithm 6 Run next agent actions.

```
procedure SelectNextAgentAction
    PriorityQueue<OWLIndividuals> pq
    Set<OWLIndividuals> intentions
    Set<OWLIndividuals> agentActions ← nextAgentAction
    for all agentActions aa do
        if aa is member of concept desire then
            pq.addIntentions(aa.hasIntentions)
        else
            pq.addIntention(aa)
        end if
    end for
    for all pq do
        runPlan(pq.dequeue())
    end for
end procedure
```

In summary, Java classes annotated as plans have a corresponding member of concept `Intention` in the ontology. These links make facts and rules of the ontology accessible to the agents in the Java code. The object property `nextAgentAction` (with domain `Person`, which is basically equivalent to the set of agents, and range `AgentAction`) together with a rule (e.g. $Agent(myself) \rightarrow nextAgentAction(GoShopping)$ – the agent will permanently *GoShopping*) determines how the agent decides which plan to chose next. The next agent actions are periodically triggered by the round based management of AGADE.

6.2.4 Complex Calculations

The latest OWL 2 specification from 11th December 2012 does not provide syntax to express calculated values for data properties (see [5]). Iannone et al. published a proposal for enabling arithmetic computation in OWL-DL, but however, this contribution has not been integrated in the OWL specification

yet [148]. As an alternative, the available rule language SWRL provides math built-ins that enable OWL reasoners to perform simple mathematical operations [149]. For example, the atom such as `swrlb:add` uses the `add` built-in to add two numeric values. The statement `swrlb:add(?sum,5,4)` adds the value 5 and 4 and bind the sum ($9 = 5 + 4$) to the first argument `?sum`. Sánchez-Macián et al. present in their paper the groundwork for an extension to the SWRL language to overcome complex scenarios that include mathematical relationships and formulas that exceed current SWRL capabilities by enabling advanced mathematical support utilising the *OpenMath* libraries [150]. Wenzel et al. have put forth the approach but further work is still required for practical operations [151].

SWRL built-ins is limited and additionally may cause SWRL rules to lose decidability, which will let the reasoning mechanisms fail [152]. The author recommends the implementation of all numerical computations in Java (or any other procedural language) and making them accessible to the ontology through method calls. Consider, for example, the following SWRL rule:

$$Person(?p) \wedge hasAge(?p, ?age) \wedge swrlb : add(?newage, ?age, 1) \Rightarrow hasAge(?p, ?newage)$$

At first sight, this rule is going to increment the age of any person *?p* by one. However, this rule generates an infinite loop incrementing the individuals age, each age one greater than the previous age. This is because the built-in bind the argument *?age* of the data property *hasAge* in the antecedent and this property again is used in the consequent for overwriting

the current age, then rules may become undecidable i.e. this rule will never terminate the reasoning process.

6.2.5 Ontology Query Languages

Querying data through ontologies is an essential step when using ontologies as the main belief base for agents. Currently, several ways to query ontologies are available which can be used in the context of OWL based ontologies. However, in order to enable simulations with a very large number of agents, an efficient query language for OWL ontologies is essential. This section demonstrates well-established possibilities for retrieving data through ontologies using a test case and benchmarks their query performance in order to optimise response time when running multi-agent scenarios. The query should answer from the individual ontology of an agent (i.e. *myself*), the agent which has the highest opinion influence (i.e. *opinion leader* (see Section 4.3)) on *myself*. Fig. 35 shows the rudimentary ontology which is used for the test case.

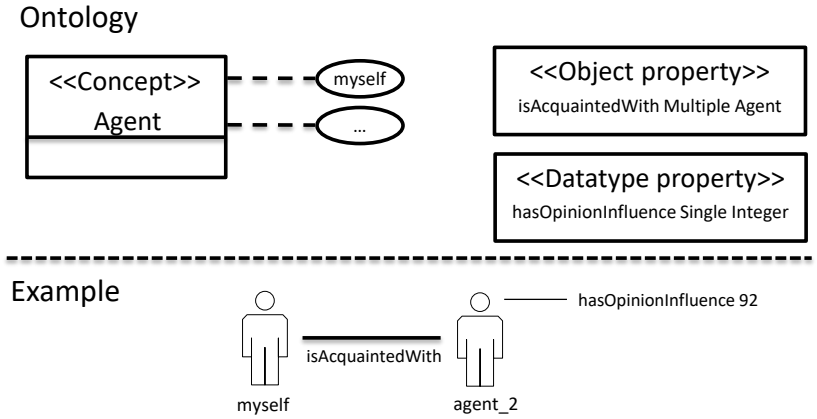


Figure 35: An example ontology used for the query variants.

6.2.5.1 SPARQL

SPARQL (SPARQL Protocol And RDF Query Language) is a W3C standardised query language, inspired by the well known database language SQL. With SPARQL, concepts, individuals and properties can be queried flexibly from ontologies. A select query provides all possible bindings for given variables. The query below shows the relevant SPARQL query to find the agent with the highest opinion influence which is acquainted with myself. The query consists of two parts: the `SELECT` clause identifies the variables to appear in the query results, and the `WHERE` clause provides the basic graph pattern to match against the data. Note that ontologies and their elements (i.e. concepts, properties and individuals) are identified using internationalised resource identifiers (IRI) (see [5]). Two IRIs are structurally equivalent if their string representations are identical (i.e. unique key). SPARQL allows the declaration of a prefix for abbreviating long IRIs in the query body with the aim of improving the readability which is used in the following query.

```
PREFIX pm: <http://www.thm.de/mnd/wbm/phonemarket#>
SELECT ?y ?o WHERE {
pm:myself pm:isAcquaintedWith ?y.
?y pm:hasOpinionInfluence ?o
FILTER (?o >=80)
}
ORDER BY DESC(?o) LIMIT 1
```

In addition to the readability of SPARQL queries the result set can be filtered, ordered and limited. The result set is in a descending order and the size of the result set is limited to one entry. Thus, this query result

has only a single entry with the highest opinion influence of all available acquaintances instead of a result set. There are currently two reasoners available for handling SPARQL queries: the reasoner Pellet (see [26]) as well as a SPARQL-supporting extension for the reasoner HermiT (see [25]). Both require the Apace Jena framework (see [153]), because the OWL API does not support direct use of SPARQL.

6.2.5.2 SPARQL-DL

SPARQL-DL is a query language based on SPARQL and specialised for OWL ontologies (see [154]). A query engine has been designed and developed for SPARQL-DL which is based on the OWL API and adds a common interface for each reasoner that is supported by the OWL API (see [155]). Just as SPARQL, all possible bindings of variables are returned to a select statement. Concepts, individuals and properties can be queried, as well as meta informations such as the disjointness of concepts or the transitivity of object properties. The following shows a SPARQL-DL query which selects the agents with their related opinion influence:

```
PREFIX pm: http://www.thm.de/mnd/wbm/phonemarket#
SELECT ?y ?o
WHERE {
PropertyValue(pm:myself, pm:isAcquaintedWith, ?y),
PropertyValue(?y, pm:hasOpinionInfluence, ?o)
}
```

In contrast to SPARQL, the value of the data property *hasOpinionInfluence* can not be limited by a minimum value. Furthermore, the maximum of the property values can not be queried directly. This has to be done by a

Java routine separately.

6.2.5.3 DL Query

DL queries allows the selection of super-concepts, sub-concepts or instances of a class. Only the last mentioned usage is relevant to the test case. A corresponding query engine for DL queries is not part of the OWL API, but is provided as a sample class (see [156]).

The test case, in which individuals have to be queried which have an acquainted relation to *myself*, shows a significant disadvantage in the usage of DL queries. If the query consists of an object property assertion, the object of this property cannot be a variable. The following query is, unlike SPARQL, not possible:

```
myself isAcquaintedWith ?y
```

An inverse formulated query, however, could solve this issue:

```
isAcquaintedWith myself
```

Note that this would force an *isAcquaintedWith* to become a symmetric relationship. However, such a modelling would not correspond to reality, because the relationship is not mutual.

Alternatively, there might be an inverse object property of *isAquaintedWith* which is named *isKnownBy*. This work around is used for the test case in order to enable DL queries. It should be noted, however, that an individual in reality may not know that the other individual knows him. The agents would thus be able to conclude such knowledge about the inverse object properties. The corresponding DL query has the following form:

```
(isKnownBy value myself) and                                    1
(hasOpinionInfluence some integer[>=80])                        2
```

Again, the maximal value of the property values of the result set have to be determined by a Java routine since order and limit the result set is not supported directly.

6.2.5.4 OWL API

Alongside available query language another possibility is to access the elements of the ontology directly using the OWL API. In this case, all individuals have to be selected which are linked with the object property *isAcquaintedWith* (line 1). After that, the resulting set can be sorted and limited to 1 entry by using a Java routine (line 2):

```
Set<OWLIndividual> valuesSet = getObjectPropertyValues(         1
   myself, opIsAcquaintedWith);
OWLIndividual highestInfluencePerson =                          2
   searchForHighestDataPropValue(dpHasOpinionInfluence,
   valuesSet, 80);
```

The individual with the highest opinion influence (minimum level is set to 80) is then mapped to the attribute `highestInfluencePerson`.

6.2.5.5 Benchmarking

The following benchmark was used to explore the performance of the aforementioned ontology query languages. Again, in order to enable simulations with a very large number of agents, an efficient query language for OWL ontologies is essential, especially when running a business simulation game

where participants expect an appropriate response time. Each of the query variants shown in the previous sections is called 100 times via a separate Java thread. Each of these threads creates its own object of the queries class, which in turn includes its own ontology object and relevant reasoner instance. This setting shows that each agent is running in different threads and uses its own ontology and reasoner.

The computation time, which is required to answer the queries, is the decisive criterion in this scenario. Table 3 shows the computation time for each query variant. It was run on a quad core CPU and 32GB RAM. Each query language was run with its own Java virtual machine and using the flag `Xmx` and `Xms` set to $30GB$ which specifies the maximum memory allocation pool in order to avoid incorrect measurements regarding the time consuming mechanisms of the Java garbage collector.

Table 3: Benchmark test results (duration time). Averages after 100 runs (100 method calls for each run) for each query language.

Type	Averages in ms
SPARQL	11357
SPARQL-DL	18398
DL Query	152124
OWL API	5135

The most time spent is when using the DL query. The fastest variant is the OWL API followed by SPARQL and SPARQL-DL. The author recommends avoiding DL queries for that reason. Not important in the selection of the variants, but still an advantage is using OWL API directly, as no further APIs are required. A disadvantage is that queries with several

restrictive conditions require a significantly larger amount of coding effort. However, this disadvantage is accepted in the perspective of the improvement of performance.

Again, the computation-time is an crucial factor for the scenarios especially when a scenario is used as a business game where participants usually want to get feedback of their decisions immediately. The queries are therefore implemented directly via the OWL API.

6.2.6 Round-based Management

Round-based simulations enable the definition of discrete time steps. The director agent triggers the beginning of each new round. Participating agents send control messages after having finished their task. According to round-based simulations the next round starts after receiving all control messages (i.e. initialising the social network or start-stop the simulation). The round-based approach allows a synchronisation point whereas agents can report their current state. This ensures that all agents are synchronised when running a simulation in a distributed environment with heterogeneous computing power.

Each round (or time step) of the simulation is structured into four phases (see Fig. 36) with defined functionality and integration into the BDI paradigm.

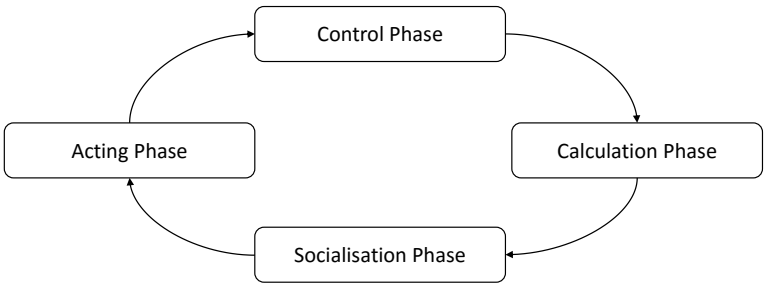


Figure 36: Four phases simulation

The *control phase* is modelled for user interactions and user information meaning the director agent processes commands issued by the user during execution of the last round and relevant GUI components (e.g. statistical graphs) are updated. In the *calculation phase* the director agent tells each agent to make necessary calculations to update its internal state. Agents update their mutual relations and possibly build new connections to other agents in the *socialisation phase*. Next agent actions are triggered in the *acting phase* depending on the individual state and rules evaluation. These four phases have proved to be appropriate for running economic simulations after intensive test-runs. However, the existing structure can be easily extended with additional phases if required.

6.3 AGADE Overview

AGADE provides a wizard for creating a new simulation. The first thing which has to be done is to specify the general simulation settings (i.e. simulation name, number of agents, number of rounds, ...) as well as setting the location of the SDL which is shared for all participating agents and ensures ontological commitment (see Fig. 37).

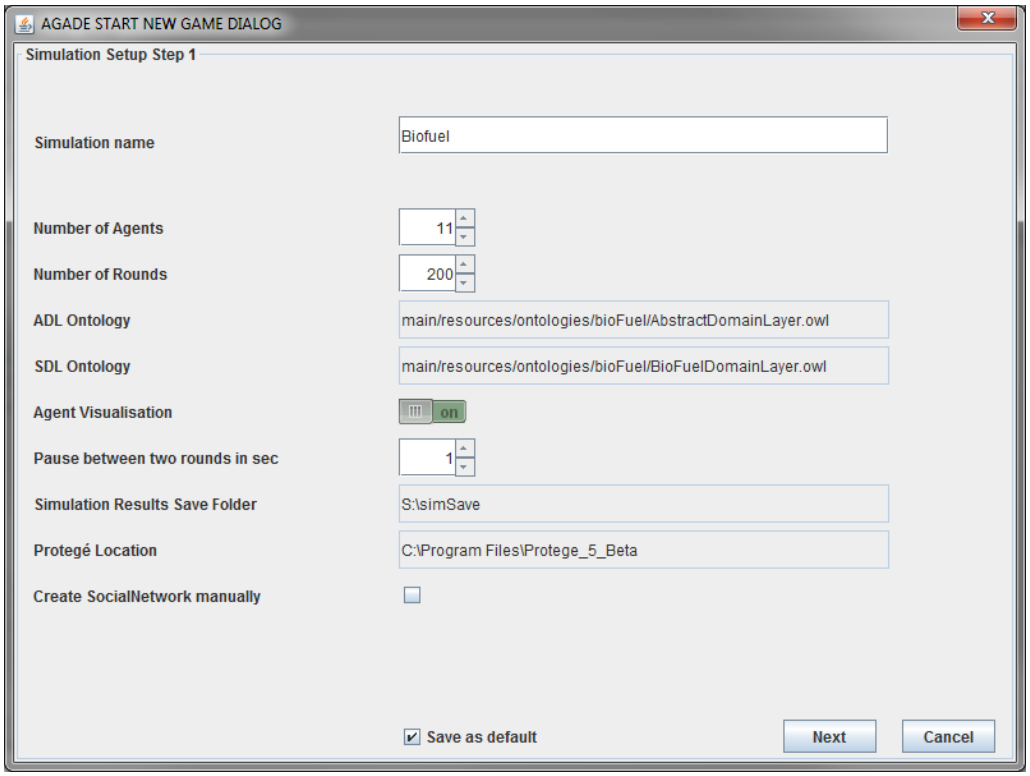


Figure 37: AGADE start new simulation dialog step I.

The mapping between agents and ontologies must be defined in the second dialog (see Fig. 38). The agent classes and the relevant IDL ontolgies have to be selected. This dialogue allows the specification of how many agent instances for each agent type have to be created and each agent instance will be equipped with its individual ontology.

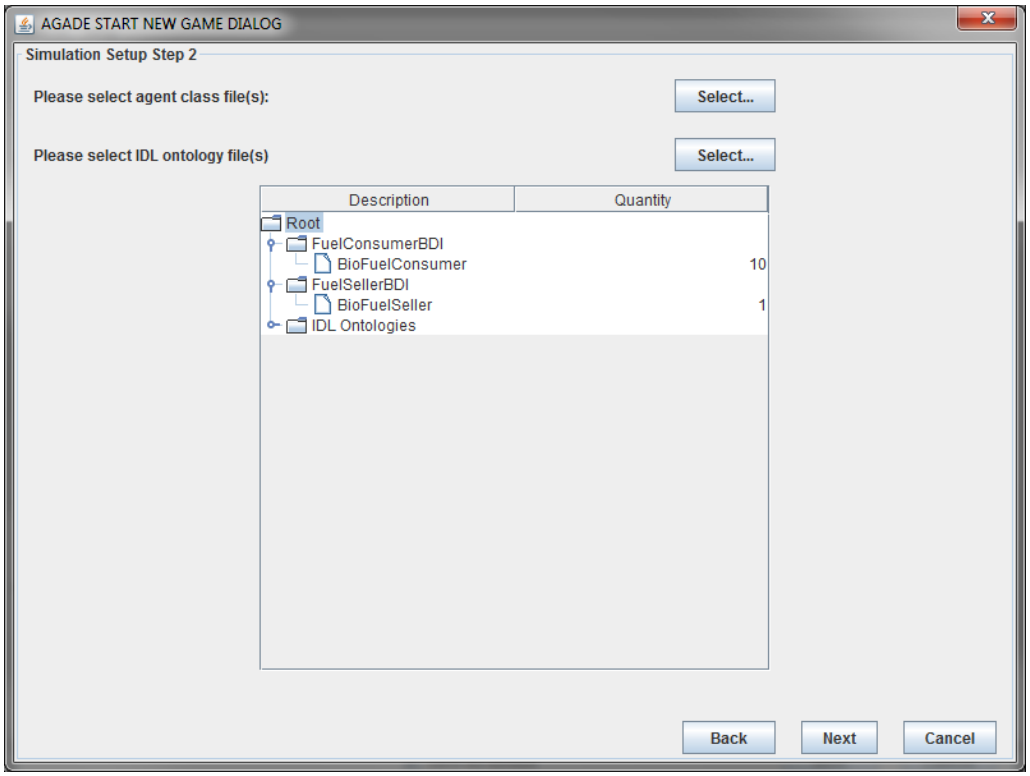


Figure 38: AGADE start new simulation dialog step II.

Before starting the actual simulation, the social structure has to be defined, comprised of the mutual relations of all agents. The adjacency matrix can be added directly, the groundwork of the social structure is laid by defining who knows whom (see Fig. 39). On top of this groundwork, an arbitrary number of additional relational aspects (each with its own adjacency matrix and influence matrix respectively (see the register on top of Fig. 39) can be built, e.g. by defining the degree of technical understanding agent a attributes to agent b or simply the degree to which one agent is affected by another. The algorithms presented in Section 4.1.2 and Section 4.3 can be used here. Social aspects and information about the current state of the agent are mapped to the ontology. The social aspects will be transferred to properties to the relevant agent (i.e. the edge value between two nodes).

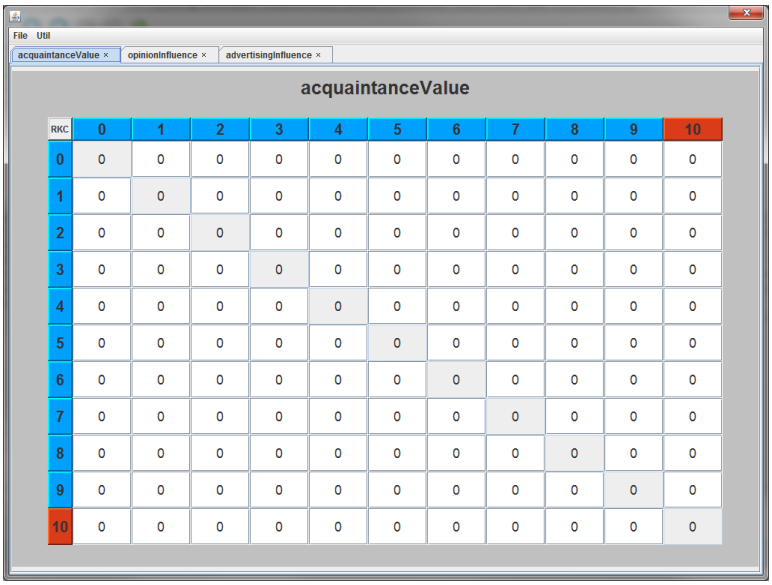


Figure 39: Adjacency matrix setup.

The simulation itself can be controlled using the GUI displayed in Fig. 40. The graphical representation of the social network can be deactivated. This is recommended when running a simulation with a large number of agents when a graphical representation is not reasonable.

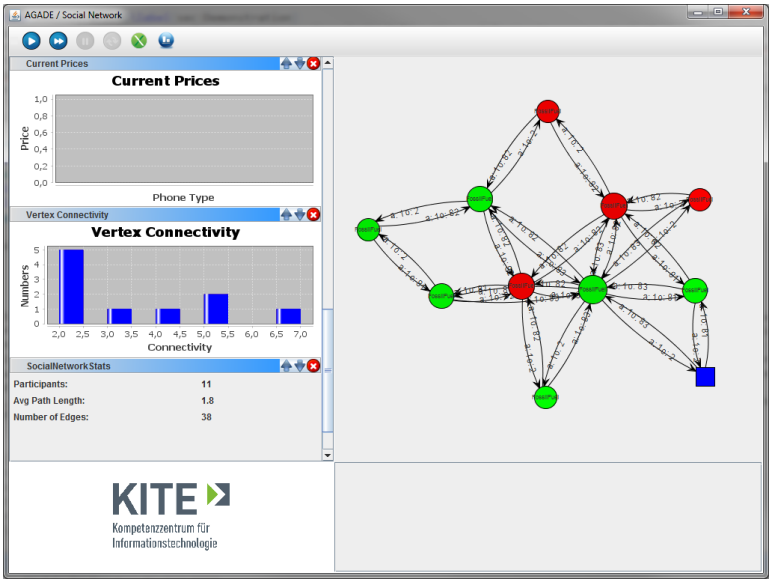


Figure 40: AGADE simulation GUI.

On the top of the GUI the control buttons are located: between any two time steps a simulation can be halted so that further inspections of the current state of affairs are possible. Data describing crucial aspects of the current simulation is displayed continuously. The right hand side of the screen is a graphical display of the social structure formed by all participating agents using the Java universal network graph framework Jung (see [157]). The vertices of the graph represent the agents using different shapes for different agent types (i.e. consumer and seller). Size and colour of the vertices can be used to display additional information on the respective agent (e.g. the individual state of the respective agent). The edges between the vertices depict the relations of the agents giving a precise description of each relation as they are labelled with the respective relation indices. However, the social graph can also be used to advance into the very mind of each agent, as it offers a view on the current private ontology state belonging to an agent of interest. Fig. 41 shows the current state of the private ontology of a participating agent using the open source ontology editor Protégé (see [158]).

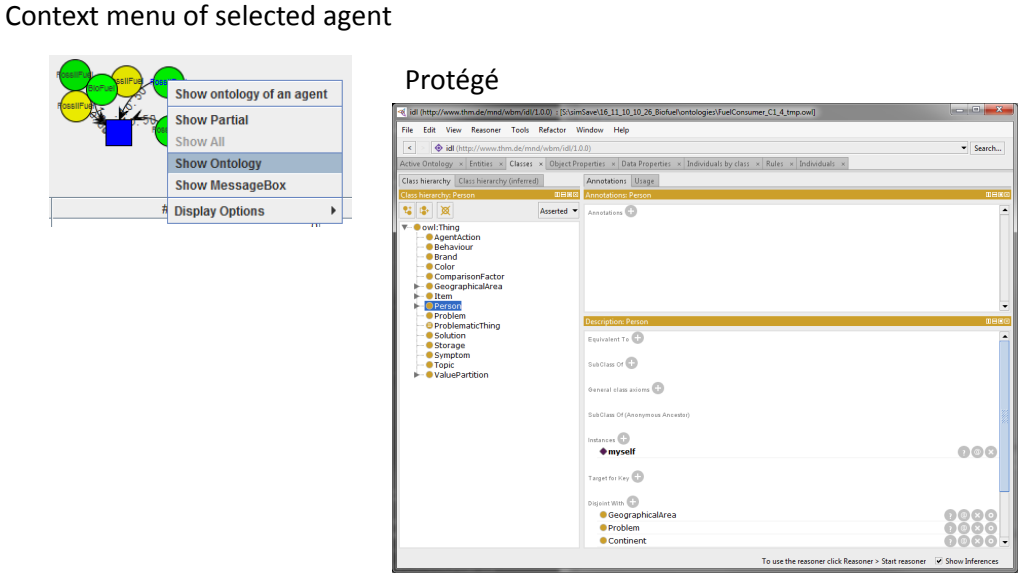


Figure 41: Current ontology state of an participating agent.

6.4 Summary of Chapter 6

In this chapter, the concept of a practical integration of semantic technologies applied to an agent-based framework has been shown (i.e. AGADE).

Section 6.2 has demonstrated how ontologies can be integrated into the BDI concept utilising the Jadex agent framework. The high memory consumption when incorporating many ontology instances has been discussed, Section 6.2.1 having shown an architecture which can be used to run simulations in a distributed environment. Section 6.2.2 demonstrated an `AbstractOWLAgent` class (on the level of *ADL*) that describes corresponding elements of an OWL-BDI agent that enable it to participate in AGADE simulations. The ontology-based plan selection of an BDI agent has been described in Section 6.2.3 where agent actions are expressed in the private OWL ontology. Section 6.2.4 discussed the current limitations of SWRL bult-ins followed by a recommendation to implement numerical computations

in Java or any other procedural language and make them accessible to the ontology through method calls. Having described the well-established query languages for OWL ontologies in Section 6.2.5, a benchmark has shown that the OWL API outperforms current available ontology query languages. Ensuring that all participating agents are synchronised when running a simulation in a distributed environment with heterogeneous computing power, the definition of discrete time steps (i.e. round-based) has been considered in Section 6.2.6.

Finally, the last section has given an overview of AGADE which supports the development and calibration of dynamic business scenarios.

7 Results and Validation

This chapter demonstrates how the three-layer ontology architecture can be used in various simulation scenarios. Different market mechanisms (e.g. opinion leadership) are implemented in the interactive round based multi-agent simulation framework (AGADE). The process of applying the underlying AGADE mechanisms to simulation scenarios will be described. The ontologies are used to make world knowledge available to the agents which can then determine their actions in accordance with this knowledge. The experiments give proof-of-concept and are used to validate the proposed three-layer model. Results obtained using AGADE are compared with simulation results published earlier and with data collected through online surveys. This chapter concludes with a case study where a scenario has been applied as a business game using AGADE.

The following text passages have been part of the publications [130], [133], [143] and [159].

7.1 Mobile Phone Market Scenario

The first scenario demonstrates how BDI agents can be used to model individuals as participants in social structures where they act as potential buyers in a mobile phone market simulation. Running a mobile phone market simulation was inspired by TOPSIM (see Section 3.1) and its mobile phone market scenario which is used in classrooms at the Technische Hochschule Mittelhessen.

The scenario is modelled in two-stages: The reference scenario *opinion leadership* is used to verify the architecture, the *personal preferences* mecha-

nism extends this scenario with a more heterogeneous structure of market participants.

7.1.1 Opinion Leadership

7.1.1.1 Summary

Opinion leadership is a well understood marketing mechanism that has gained new attention with the advent of social networks [128]. The first scenario models a rudimentary consumer mobile phone market. Each agent represents a single consumer and acts in a social structure which is constructed according to network theoretic algorithms (see Section 4.1). For simplification, a product is identified by brand name (single dimension). Individuals who want to buy a new mobile phone can get information from their neighbourhood. The disposition of a person to buy a new mobile phone is expressed with the so called *happinessValue* (see Section 4.2): If this value is below a given threshold the person will make amends by initiating the purchase of a new mobile phone. Individuals who are directly connected to an opinion leader will be directly influenced in their purchasing decision and follow the opinion leader's advice by buying the same product. Individuals who are not connected to an opinion leader look for a *friend*. If they are not linked to any person, they will not buy a new phone because of the lack of information.

7.1.1.2 Mobile Phone Market Ontology

While the ADL ontology describes the abstract elements of an BDI agent and can be used in different scenarios without any changes, the SDL has to be modelled for the mobile phone market scenario with its product and specific elements. Furthermore, the individual aspects have to be implemented in

the IDL.

The process of modelling includes the specialisation of abstract concepts (see Fig. 42).

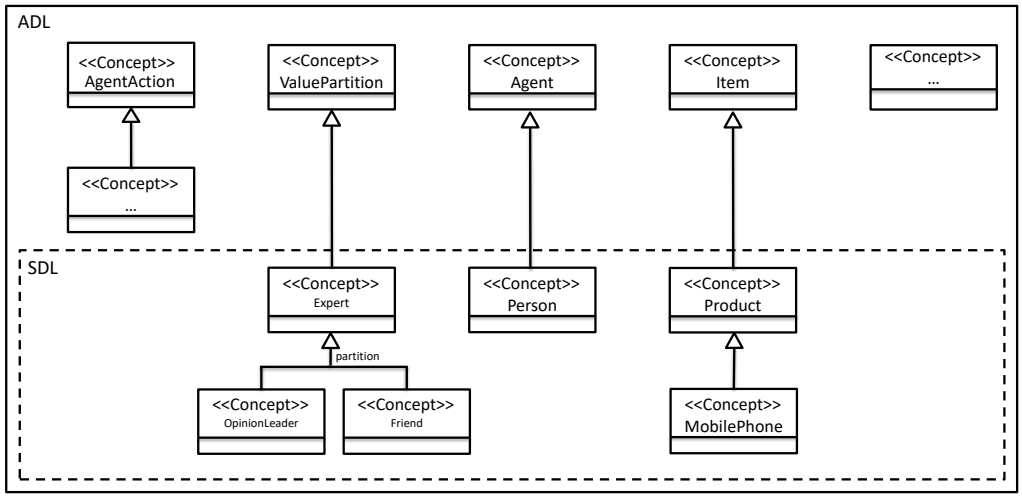


Figure 42: Sub-concepts of the mobile phone market SDL.

Consumers represent people or individuals and these are represented by agents in the market. Therefore, the concept *Person* is modelled which gets the sub-concept of *Agent*. Mobile phones are traded in the market which are represented by the concept *MobilePhone* which is the specialisation of *Product* which again is a sub-concept of *Item*. For reasons of simplification, there are no further specialisation of concept *MobilePhone*. Since individuals are directly influenced by individuals with high opinion influence (i.e. opinion leader) in their buying decision, the opinion leaders are represented as members related to the concept *OpinionLeader* in the ontology model. In general, opinion leaders have wide knowledge and understanding of a specific product category and can be seen as experts. The concept *Expert* (modelled as a super-concept) represents this fact. In this scenario, individuals who are not connected to an opinion leader are going to ask a friend for advice

who can be seen as an expert in this context as well. *Expert* is therefore modelled as a value partition i.e. the set of experts is the union of the two disjoint subsets *OpinionLeader* and *Friend*. Thus opinion leaders and friends will considered as experts. The following describes how the classification of *OpinionLeader* and *Friend* can be done before the relevant SWRL rules are shown.

The ontologies of the agents are architectured in a way that they can only be influenced by a peer agent if its respective opinion influence value is greater than a specific threshold denoted as β. To make sure that each hub possesses an opinion influence value in the range of $[\beta, 100]$ each hub is given β as a basic value. Its pagerank is multiplied by $(1 - \beta)$ and added to the base value, leading to the following formula: Let A_C be the set of all consumer agents, ρ the page rank function that assigns the page rank to each agent $a \in A_C$ (according to its position in the associated sociogram), then the opinion influence ϑ of a is calculated as follows:

$$\vartheta(a) = \begin{cases} \beta + \rho(a) \cdot (1 - \beta) & \text{if } a \text{ is a hub} \\ 0 & \text{else} \end{cases}$$

Consequently, agents can be ranked according to their influence value. While hubs are always ranked at the top, ordinary non-hub agents follow in the ranking. Thus, the *opinionInfluence* is an element of $\mathcal{R}$ with domain *Person* and range literal `xsd:Integer`. This value is what really makes an opinion leader together with the following rules:

1. Person(myself) $\wedge$ isAcquaintedWith(myself, ?x) $\wedge$
   integer[$>= \beta$](?y) $\wedge$ opinionInfluence(?x,?y)

⇒ OpinionLeader(?x)

2. Person(myself) ∧ isAcquaintedWith(myself, ?x) ∧
   integer[$< \beta$](?y) ∧ opinionInfluence(?x,?y)
   ⇒ Friend(?x)

The first rule will handle individuals (variable $?x$) which are related to the agent *myself* as *opinion leaders* whenever the opinion influence value to the agent is higher than a given value β. If the value is less than or equal to β, the second rule will handle individuals as friends. However, both sets (hubs and non-hubs) are sorted. It is concluded that each existing non-hub relationship is a *friend* relationship with medium opinion influence. This behaviour can be modified by adding an appropriate value for the classification of the relationship. The person will contact neighbouring hubs and will choose the one with the highest opinion influence value first. If the set of opinion leaders is empty, then the person is going to ask a friend (selected randomly).

The IDL describes the individual factual knowledge of a person (e.g. which mobile phone is he familiar with) as well as the individual behaviour (e.g.: if you are unhappy, ask someone popular for advice). The following rule expresses how the person decides which plan to chose next:

$$\begin{aligned} &\text{Agent(myself)} \wedge \text{hasHappinessValue(myself, ?hvalue)} \wedge \\ &\text{hasHappinessValueThreshold(myself, ?threshold)} \wedge \\ &\text{lessThan(?hvalue, ?threshold)} \wedge \text{isAcquaintedWith(myself, ?p)} \wedge \\ &\text{Expert(?p)} \Rightarrow \text{nextAgentAction(myself, opinionLeadershipPlan)} \end{aligned}$$

To summarise: An agent a_1 which is considered as *myself* ($myself \in \mathcal{I}$)

and *myself* has a *hasHappinessValue* ($hhv \in \mathcal{R}$). If hhv is below the *hasHappinessValueThreshold* ($hhvt \in \mathcal{R}$) and if a_1 *isAcquaintedWith* ($iaw \in \mathcal{R}$) with another agent a_2 ($a_2 \in \mathcal{I}$) and a_2 is an expert ($a_2 \in Expert \in \mathcal{C}$) the next agent action of a_1 is set to *opinionLeadershipPlan* ($folp \in \mathcal{I}$ and $folp \in AgentAction \in \mathcal{C}$) by using ontology reasoning techniques. This action implements the process in which a consumer is selected with the highest opinion influence value from the set of experts.

Fig. 43 shows the relevant relations at the level of the IDL-layer for each consumer agent.

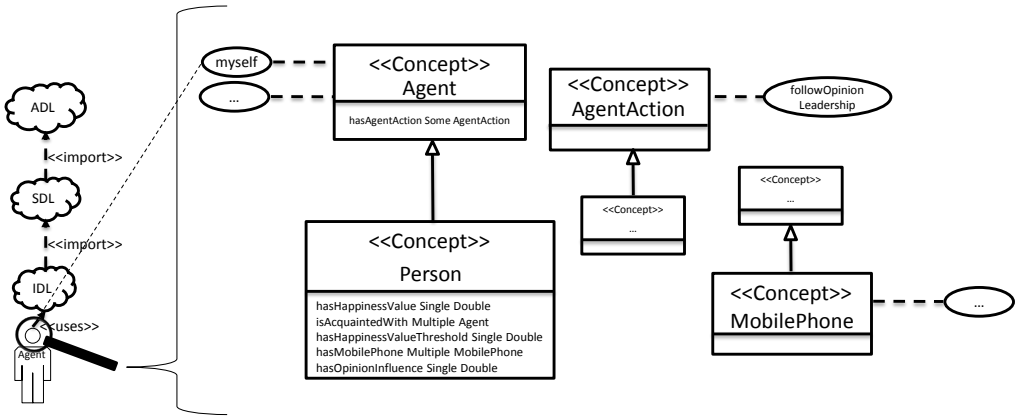


Figure 43: Rudimentary mobile phone market IDL.

The relevant opinion leadership plan which is triggered whenever the next agent action is set to *opinionLeadership* is implemented in the framework as follows (see Algorithm 7: BLAH

Algorithm 7 Opinion leadership plan.

```
1: procedure OPINIONLEADERSHIPPLAN(myself)
2:     expert = queryOpinionleader(beliefbase)
       ▷ select opinion leader with highest influence value
3:     if expert is empty then
4:         expert = queryFriend(beliefbase)      ▷ select a friend randomly
5:     end if
6:     if expert is empty then
7:         quit                ▷ terminate plan (not connected to an expert)
8:     end if
9:     phone = askForRecommendation(expert)
       ▷ trigger a communication plan and send a message and ask for a
   recommendation
10:    if phone is empty then
11:        quit              ▷ terminate plan (received no recommendation)
12:    else if beliefbase does not contain phone then
13:        learning(phone)          ▷ learning mechanism (see Section 5.4)
14:    end if
15:    GoShopping(phone)                          ▷ activate GoShopping goal
16:    increase happiness value                          ▷ (see Appendix F)
17: end procedure
```

To simplify the simulation, the model does not have any budget restrictions, different products and only one seller agent which has an unlimited storage and no commercial interest.

7.1.1.3 Parameters and Results

Agents and their relations form a small world network of 1000 nodes built with preferential attachment. Any node that is connected to more than 100 nodes is considered as a hub. The weights of all edges connecting to a node to one of these hubs is set to an opinion influence of $\beta = 80$ which results in a range from 80 to 100 making the hubs to opinion leaders. Various simulation experiments have shown that this is a sensible value for β. These opinion leaders are equipped with a mobile phone (same type of phone).

Agents have a happiness factor which they aim to maximise and this factor deteriorates continuously over time representing the internal stimuli (i.e. need for change) (according to the Stimulus-Response (S-R) model, see Section 4.2 and Appendix F).

Experiments have shown that the structure of the graph scales up with a rising number of nodes. Fig. 44 shows a line graph with the number of phones sold after a simulation of 100 rounds over 100 times. The x-axis describes the number of rounds and the y-axis describes the distribution in percent of mobile phones. The number of phones increases rapidly at the beginning and runs into saturation. This demonstrates that innovation spreads from hubs through the network.

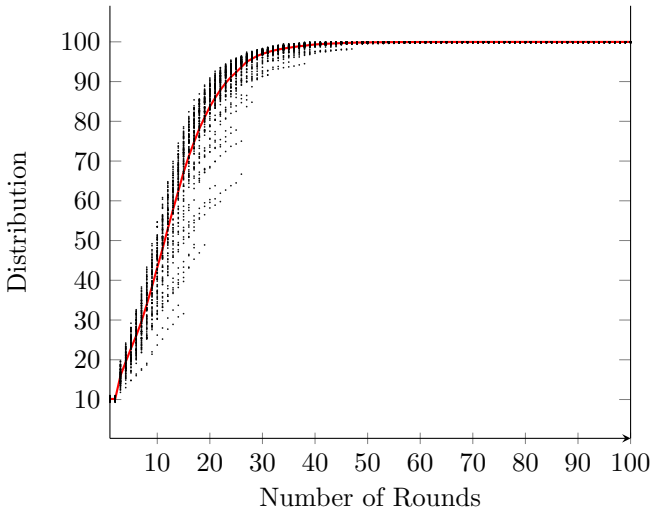


Figure 44: Phone distribution after 100 rounds (line indicates averages, points mark results of single simulation runs).

The distribution of the product corresponds to data published by James S. Coleman et al. [160]. They studied the adoption rate of a new antibiotic (tetracycline – code-named Gammanym) by doctors in the field. They

analysed prescription information and the spread of the use of that new drug and collected data about the mutual influence of the doctor's opinions. They detected that there were early adopters, most of them well connected to other doctors (namely hubs) and that the use followed the social network structure.

Although one might argue that comparing medical drug distribution to the spreading of a specific phone type has some drawbacks, the line of argumentation that this data makes comparable, is the decision process of adopting a product (adoption of a new drug by doctors and mobile phone distribution described here). Both are based on a social structure including hubs (i.e. inter-connected doctors) and follow a pattern where personal influence (consulting-intensive) plays a significant role.

Comparing the data of the adoption rate (Fig. 45) with the data created by the prototype (see Fig. 44) shows significant similarity.

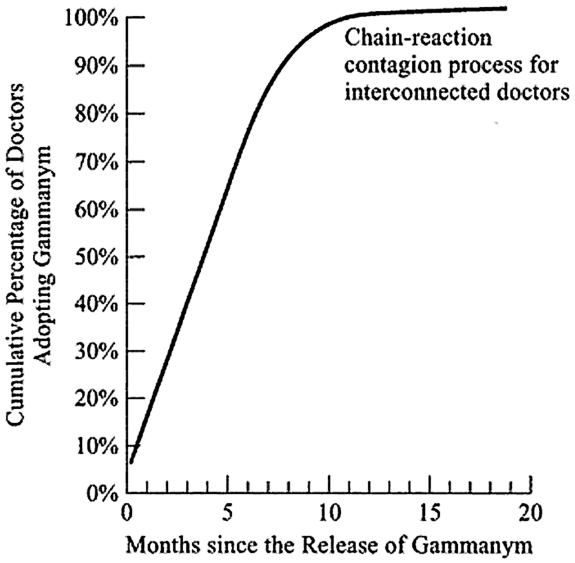


Figure 45: The rate of adoption of Tetracycline by doctors [161, p. 147].

7.1.2 Personal Preferences

7.1.2.1 Summary

According to Section 4.4 consumers compare attributes of products i.e. price or technical features of available products and try to rank them according to their personal preferences. This mechanism is an extension of the mobile phone market simulation with the aim of modelling more complex scenarios with a more heterogeneous structure of market participants. Beside buying a new mobile phone with the opinion influence mechanisms, consumers can gather information about mobile phones i.e. all technical data and determine the mobile phone with the highest weighted preference value out of the set of mobile phones collected. One way to gather information about mobile phones is picking agents from the social environment who are already equipped with mobile phones and asking them for advice. Alternatively, agents can delegate a request to one of their neighbours i.e. all agents the consumer is connected with, or alternatively contact sellers directly to get available products instead of asking other consumers.

7.1.2.2 Mobile Phone Market Ontology

The ADL is reused and the SDL (both from previous scenario (see Section 7.1.1)) is extended with a concept *PersonWithItem* which is a subconcept of Person ($PersonWithItem \subset Person$). Agents are represented as members of concept *Person* again and everything that may be owned in some way or other by a person belongs to concept *MobilePhone* using the property *hasProduct* (domain set to *Person* and range set to *MobilePhone*). The properties *isAcquaintedWith* and *hasOpinionInfluence* are elements of $\mathcal{R}$

with domain and range *Person* respectively.

If an agent a_1 has a *isAcquaintedWith* relation to another agent a_2 and a_2 *hasProduct* p and $p \in MobilePhone$ and p has attributes of a *MobilePhone*, a_1 can conclude that a_2 is a person who owns something that is a mobile phone. The following SWRL classifies a_2 as a member of concept $PersonWithMobilePhone \in Person$ by using ontology reasoning techniques:

$$isAcquaintedWith(myself, ?a) \wedge hasProduct(?a, ?p) \wedge MobilePhone(?p) \Rightarrow PersonWithMobilePhone(?a)$$

Note that p does not have to be defined as a mobile phone as the OWL reasoner will conclude this from properties of p.

Data property relations are used for describing technical data of mobile phones quantified by numerical values (see Section 4.4). The calculations are triggered by the rule evaluation process during the calculation of an agent's personal preferences. Relevant data will be retrieved from the agents private ontology and gets updated immediately with the results calculated in Java (see Fig. 46).

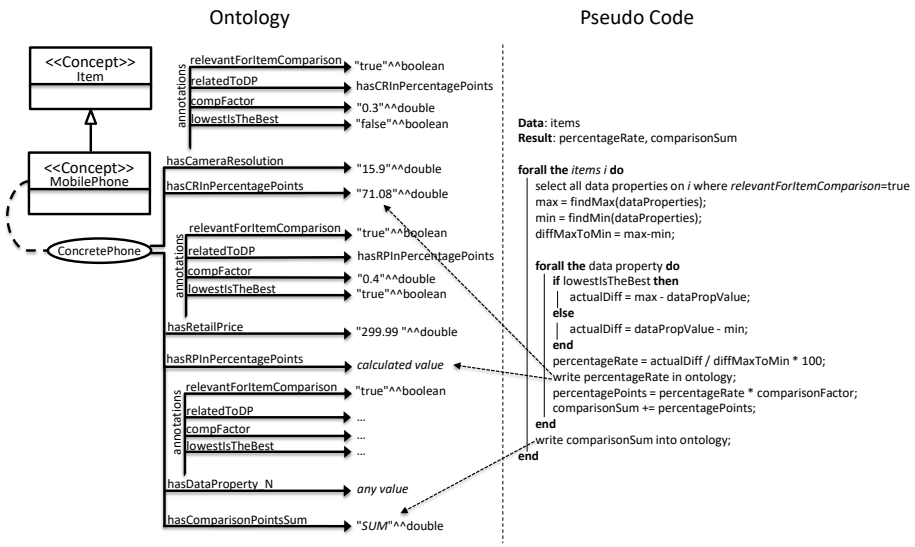


Figure 46: Calculation of scores of phone attributes.

The sum of each calculated comparison point is stored in a data property *hasComparisonPointsSum* related to the relevant item in the relevant IDL of an agent. The results are available to the reasoning process immediately. Mathematical operations are controlled by annotations in the ontology. Each element can be annotated with instructions of how it will be handled during rule evaluation. In particular a *comparison annotations* is designed which is used to define how data properties will be evaluated during the calculation of personal preferences: *relevantForItemComparison*, *compFactor*, *relatedToDP* and *lowestIsTheBest*. They can be used for data properties of individuals of concept *Item*. While *relevantForItemComparison* carries a boolean value that indicates whether the property should be included in the calculation, *compFactor* represents the weighting factor. The *relatedToDP* annotation names another data property which stores the calculated percentage points. The *lowestIsTheBest* annotation changes the orientation of the comparisons: a lower value is considered better than a higher value. This applies to

attributes such as retail price or product weight.

After the comparison process is finished the agent is able to decide which item to buy. Additionally, minimal requirements for a mobile phone can be defined e.g. the mobile phone must have a battery life span that is at least as good as a given value. Such minimal requirements can be easily expressed with SWRL at the IDL layer as they do not require complicated calculations. Only those products which satisfy all given minimal requirements, are classified as members of concept *ItemAccordingPreferences* and are then ranked according to personal preferences. If there is no item that matches the minimal requirements, the agent can search for further products by starting information gathering or alternatively reduce minimal requirements. The following SWRL rule shows an example of how a minimal requirement can be expressed:

$$\begin{aligned} &\text{MobilePhone(?x)} \wedge \text{double[>= 8.0](?y)} \wedge \\ &\text{hasCameraResolutionInMegapixels(?x,?y)} \\ &\Rightarrow \text{ItemAccordingPerferences(?x)} \end{aligned}$$

Modelling personal preferences and including them in buying plans shows how individual market behaviour can be expressed in an OWL ontology. The ontology is again the main basis for the decision-making process for agents. Integration of Jadex agents and elements of the ontology is reached by use of annotations.

7.1.2.3 Parameters and Results

For an appropriate simulation scenario, data was collected by running an online survey on a restricted group of individuals (72 students and staff from Edinburgh Napier University). The survey include a quantitative analysis of the brand distribution, the brand loyalty of a person and the personal buying behaviour. The model is simplified according to Holland by restricting it to a subset of available data [162, pp. 45–46]. This setting aims at clarity and predictability by concentrating on a reduced set of facts. Survey data is used to set comparison factors and minimal requirements for the products. The first question regarding to identify personal buying behaviour "Why did you choose this brand?" had seven possible answers, the following four were named most often:

1. Decision based on test reports (34 individuals)
   ↳ *personal preferences*
2. Followed recommendation of friends and family (11 individuals)
   ↳ *opinion leadership*
3. In-store consultation (3 individuals)
   ↳ specialisation of *opinion leadership* (i.e. seller is modelled as expert)
4. Owned by many of my friends (socialisation) (2 individuals)
   ↳ new plan: find the maximum of the sum of products of the set of friends

Coming from a technically oriented organisation most of the participants used test reports as their main source of information. Test reports typically list all technical features of a product.

Based on the given answers the following four behaviour patterns are

modelled:

Decision based on test reports

Every phone the agent is aware of is measured by personal preferences with respect to minimal requirements defined and the best is selected. If the agent only knows one phone or none it will gather information about phone models from every agent it is in social contact with and will then apply personal preferences. The following SWRL rule, located in the private ontology, triggers the corresponding plan:

$$\begin{aligned} \text{Agent(myself)} \wedge \text{hasHappinessValue(myself, ?hvalue)} \wedge \\ \text{hasHappinessValueThreshold(myself, ?threshold)} \wedge \\ \text{lessThan(?hvalue, ?threshold)} \wedge \\ \Rightarrow \text{nextAgentAction(myself, personalPreferencesPlan)} \end{aligned}$$

Opinion Leadership

The agent will chose the phone that is possessed by the socially most important agent in its community (hub). The following SWRL rule, located in the private ontology, triggers the corresponding plan:

$$\begin{aligned} \text{Agent(myself)} \wedge \text{hasHappinessValue(myself, ?hvalue)} \wedge \\ \text{hasHappinessValueThreshold(myself, ?threshold)} \wedge \\ \text{lessThan(?hvalue, ?threshold)} \wedge \text{isAcquaintedWith(myself, ?p)} \wedge \\ \text{Expert(?p)} \wedge \\ \Rightarrow \text{nextAgentAction(myself, opinionLeadershipPlan)} \end{aligned}$$

In-store consultation

An agent following this plan will contact a seller and apply personal preferences to each phone the seller recommends. The following SWRL rule located in the private ontology triggers the corresponding plan:

$$\begin{aligned} &\text{Agent(myself)} \wedge \text{hasHappinessValue(myself, ?hvalue)} \wedge \\ &\text{hasHappinessValueThreshold(myself, ?threshold)} \wedge \\ &\text{lessThan(?hvalue, ?threshold)} \wedge \text{isAcquaintedWith(myself, ?p)} \wedge \\ &\text{Seller(?p)} \\ &\Rightarrow \text{nextAgentAction(myself, clientCounsellingPlan)} \end{aligned}$$

Owned by many of my friends (social affiliation)

The agent will chose the phone that most of the agents he is socially connected to possess. The following SWRL rule located in the private ontology triggers the corresponding plan:

$$\begin{aligned} &\text{Agent(myself)} \wedge \text{hasHappinessValue(myself, ?hvalue)} \wedge \\ &\text{hasHappinessValueThreshold(myself, ?threshold)} \wedge \\ &\text{lessThan(?hvalue, ?threshold)} \wedge \text{isAcquaintedWith(myself, ?p)} \wedge \\ &\text{Person(?p)} \\ &\Rightarrow \text{nextAgentAction(myself, socialAffiliationPlan)} \end{aligned}$$

Note that agents cannot take all information of available mobile phone models directly from test reports as the simulation would converge very fast without influence of the environment. Social influence is considered as very

important and therefore the test reports are a source of detail information only. Modelling these behavioural patterns and respective individual personal preferences resulted in 67 different IDLs (each IDL relates to one survey response). The survey in which participants were asked to order eight available attributes according to its subjective importance for them. The first four attributes were then weighted with the factors 0.4, 0.3, 0.2 and 0.1 meaning that only these four had an effect in the simulation. Non-quantifiable factors cannot be used in the computation of the overall comparison factor and were therefore expressed as SWRL rules as in the following, that states that a phone should have an Android operating system:

$$Android(?y) \wedge Smartphone(?x) \wedge hasOperatingSystem(?x, ?y) \Rightarrow ItemAccordingPreferences(?x)$$

If non-quantifiable attributes were found among the first four attributes in an individual ordering, the weighting factors were shifted so that the highest quantifiable factor received the value 0.4.

The mobile phones modelled were taken from a web portal hosted by a popular German computer magazine[8]. Each brand in the simulation is represented by the product that was ranked highest by that portal (one for each brand). The properties used for the comparison are listed in table 4. Smartphones are distributed uniformly over all agents at the beginning of the simulation.

[8]Available at `http://www.chip.de/bestenlisten/Bestenliste-Handys--index/detail/id/900/`, Accessed on January 10, 2015.

Table 4: Properties relevant for the comparison.

Property	Range	Lowest is the best
hasSpeedValue	xsd:double	false
hasRecommendedRetailPrice	xsd:double	true
hasCallingQuality	xsd:double	false
hasHandlingValue	xsd:double	false
hasDesignValue	xsd:double	false
hasCameraResolution	xsd:double	false
hasBatteryOnlineLifetime	xsd:double	false
hasOperatingSystem	OperatingSystem	—
hasBrand	Brand	—

Agents and their relations form a small world network of 1006 (uniform distribution – $1006 = 67 * 15 + 1$; 15 agents for each of the modelled IDL and one seller agent) agents built with preferential attachment with parameters set according to Barabási's (see Section 4.1). Again, the *happinessValue* is used to trigger the buying process.

Fig. 47 shows the brand distribution after a simulation of 100 rounds. While the x-axis describes the number of rounds the y-axis shows the number of phones for a point in time.

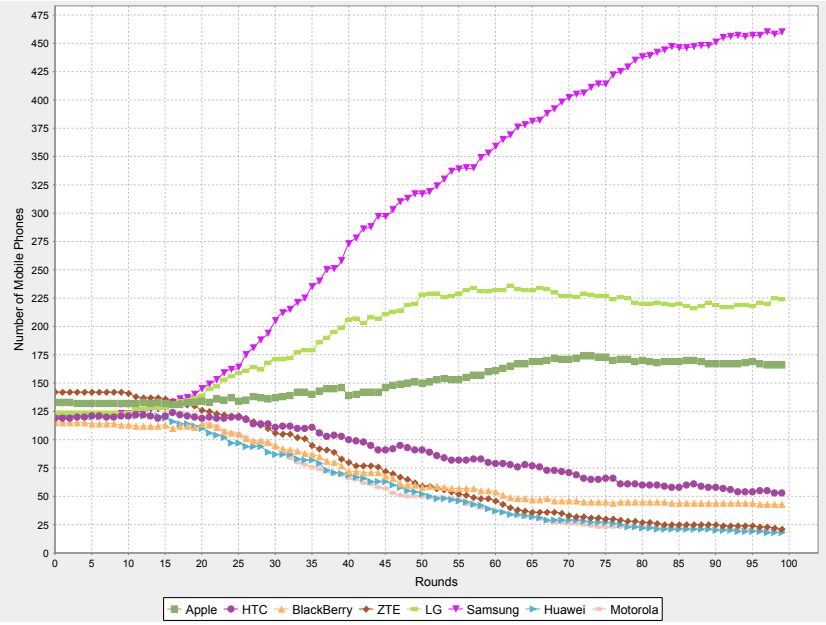


Figure 47: Brand distribution chart after 100 rounds generated by AGADE.

The following section discusses the similarities of the simulation result as shown in the following table:

Table 5: Brand distribution in survey compared with brand distribution in AGADE after 100 rounds.

Brand	Distribution in Survey	Distribution in AGADE	Difference
Samsung	40.58%	45.77%	5.19%
Apple	21.74%	16.52%	-5.22%
LG	13.04%	22.29%	9.25%
HTC	10.14	5.27%	4.87%
Motorola	5.80%	1.99%	-3.81%
BlackBerry	4.35%	4.28%	-0.07%
ZTE	2.90%	2.09%	-0.81%
Huawei	1.45%	1.79%	0.34%

Looking at Apple, HTC, and LG differences can be observed. The author can see an explanation in the battery life span where there is a difference in the relevant models. 15 individuals chose this attribute among the three most important criteria which caused a relatively high influence on the buying decision. As reality is simplified by only modelling one phone per brand details may have been missed that can cause this effect. Another aspect might be that an apparently rational buying decision based on facts and figures may mask rather subconscious elements of the decision that were not mentioned in the survey. However, when concentrating on the upper third, it consists of the same brands (top-ranked) as well as the brands BlackBerry, ZTE and Huawei (lower-ranked) are very close to each other.

7.2 Fuel Market Scenario

The fuel market scenario models the launch of a new product into an existing marketplace. It demonstrates how the proposed three-layer ontology (ADL, SDL and IDL) architecture can be applied to another market domain and how user interaction enables the use of AGADE within business games. The results obtained using AGADE are compared with simulation results published earlier. This section is part of a publication in the journal of artificial societies and social simulation [159].

7.2.1 Summary

Kiesling et al. describe an agent-based model for the diffusion of a second generation biofuel product into the Austrian market [163]. High initial investments in infrastructure are necessary to launch biofuel into the fuel market, therefore considerable financial resources are at stake before a

sufficiently large share of the market can be conquered. This is why the simulation focuses on the crucial questions of whether biofuel will be accepted by the market at all and to what extent consumers will actually use biofuel, abandoning their traditional choice. Kiesling prefers an agent-based model over alternative approaches because existing mathematical models "do not distinguish between the individual characteristics of consumers and thus neglect consumers' heterogeneity with regard to preferences and behaviour".

In the simulated fuel market each product is characterised along multiple dimensions using a variety of attributes such as price, quality, or environmental friendliness. Consumers estimate product characteristics based on their limited individual level of information which is either fed by personal experience or by communication with other individuals or advertising. Each agent represents a single consumer and acts in a social structure which is constructed according to network theoretic algorithms e.g. preferential attachment. Kiesling characterises the participating agents by a number of individual parameters among which we find the so called *innovative threshold*. This is used to model individual innovativeness meaning the willingness to adopt innovations e.g. prefer new products or new brands over established ones. Even if they would profit from using biofuel instead of conventional fuel, consumers may still refuse to purchase it just because it is simply too new. They would rather wait for others and their experiences. This fact is modelled by means of the innovative threshold. According to Roger's model of innovativeness [125], this threshold can be used to categorise the consumers into innovators, early adopters, early majority, late majority, and laggards. More parameters are available that describe the role of the individual as a target of communication and its individual behaviour as a

consumer. Each agent keeps an *information level* (short: $info(p, a)$, for product p and agent a) that may increase through communication with fellow agents or by being target of marketing activities. This value automatically decreases over time as information is obsolescing if not updated. The driving behaviour of an agent is modelled through a parameter that determines fuel consumption and in consequence the frequency of stops at the filling station. Furthermore, the agent has a *utility threshold* that describes the level of utility a product must have to be considered as a valid alternative. Product quality is modelled by the *product quality value* (short: $ppq(p, a)$, for product p and agent a) which reflects consumer experiences with the product. The *utility value* describes the subjective utility a consumer ascribes to a product and determines product selection.

Agents decide to buy a new product only if the *utility value* of that product is higher than the *utility threshold* of the agent. The *utility value* of a product p for an agent a is calculated with the following *utility function* (adopted from [164, p. 67]):

$$u_t(a,p) = (1-price_t(p))\cdot w^a_{1,t} + price_t(p)\cdot w^a_{2,t} + ppq_t(p,a)\cdot w^a_{3,t} + info_t(p,a)\cdot w^a_{4,t}$$

where $w_i, t^a \geq 0$ and $\sum_i = 1^4 w^a_{i,t} = 1$ for each agent a and each time period t. For brevity the index t is omitted meaning that the proposition holds for each value of t. The weights are related to the price (w_1 and w_2), the preference for high quality (w_3), and the readiness to buy renewable energy (w_4) which depends on information.

The set of consumers is partitioned into four segments which is reflected in the four weights of the utility function:

- price-sensitive consumers with emphasis on low prices (largest segment, 70 percent) have a high value for w_1, $w_2 = 0$ and low values for w_3 and w_4
- consumers with focus on high-quality (15 percent) high value for w_3, $w_2 = 0$ and low values for w_1 and w_4
- eco-consumers for which price or quality are less important than environmental friendliness (10 percent) high value for w_4, $w_2 = 0$ and low values for w_1 and w_3
- snob buyers looking for exclusive products taking high prices as a signal for high quality (smallest segment, 5 percent) high value for w_2, $w_1 = 0$ and low values for w_3 and w_4

Note that the sizes of the segments are estimations based on the description given by Kiesling [163]. Agents are assigned randomly to the segments.

The dynamics of the permeation process of biofuel in the fuel market is modelled with three types of events that can occur during a simulation: *communication events*, *need events* and *experience events*. The events are either triggered stochastically or periodically depending on their nature.

Communication events

Communication events are generated on the set of agents and are used to model the diffusion of information in the agent society. An agent is selected randomly and then decides again at random which of its relations it chooses to communicate with. The number of such mutual communication events in a single round of the simulation is proportional to the number of neighbours of the agent chosen. During the communication the agent with the lower *information level* adjusts

this value according to that of the communication counterpart following an *influence factor* defined on the connection in the social network weighing the influence.

Need events

A *need event* occurs whenever an agent encounters a deficit triggering a purchasing process. This deficit is indicated when the fuel level falls below the fuel threshold of the agent. If the agent does not have any knowledge about biofuel at all, conventional fuel is bought. Otherwise, the utility value of biofuel is calculated and if the resulting value lies above the individual utility threshold, the agent chooses biofuel.

Experience events

In between two need events an experience event occurs. Experience events model personal experiences with a product by directly updating the quality value of the product.

Marketing activities are scheduled periodically to introduce new products and to spread information about established products. A seller agent is modelled as a super hub which can send messages to each available agent. The agents are selected randomly and have their information level updated. Note that besides personal communication which follows the structure of the social environment (word of mouth and opinion leadership) marketing activities are the second means to spread knowledge in the community.

7.2.2 Fuel Market Ontology

The ADL used in previous simulations (see Section 7.1 and Section 7.2) is reused without any changes, the SDL has to be adapted to describe the fuel market with its products and specific elements. Furthermore, specific aspects

of agent behaviour have to be implemented in the individual ontologies of seller and consumer. In this process abstract concepts from ADL have to be specialised by sub-concepts (see Fig. 48).

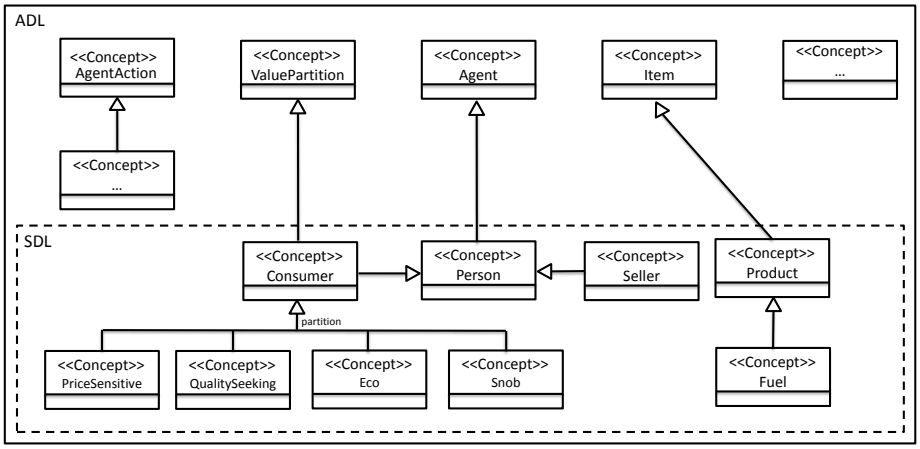


Figure 48: Sub-concepts of the biofuel market SDL.

The concept *Person* (which again is a sub-concept of *Agent* gets sub-concepts *FuelSeller* and *FuelConsumer*. The concept *Fuel* represents the available products in this market, thus it is modelled as a sub-concept of *Product* which again is a sub-concept of *Item*. The four consumer segments are modelled as a value partition i.e. the set of consumers is the union of the four disjoint subsets *Price-sensistive* consumer, *Quality-seeking* consumer, *Eco* consumer and *Snob* consumer that are again modelled as sub-concepts of *Consumer* (taken from SDL) and *ValuePartition* (taken from ADL). Relevant variables such as fuel level, travel behaviour, and utility value are modelled as data properties having the respective concepts as range (see Fig. 49).

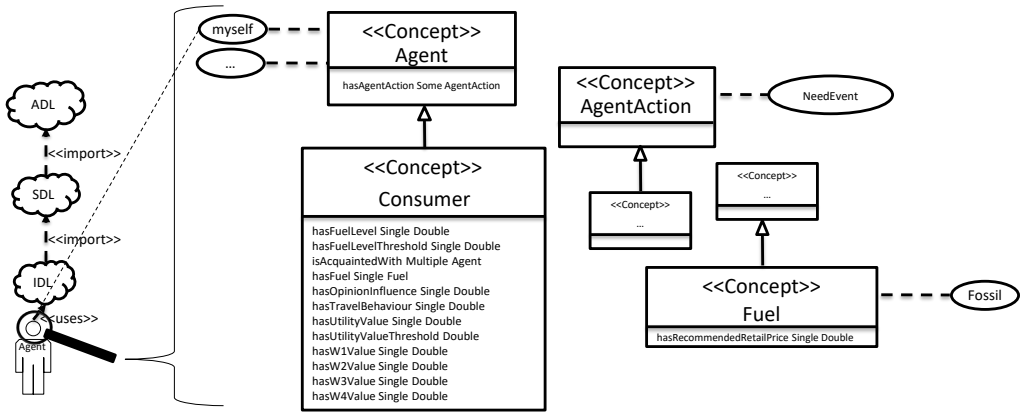


Figure 49: Consumer IDL biofuel.

Fig. 50 shows the IDL for the seller agent which focuses on relevant activity (i.e. marketing activities). The seller agent is aware of both available products on the market (fossil fuel and biofuel).

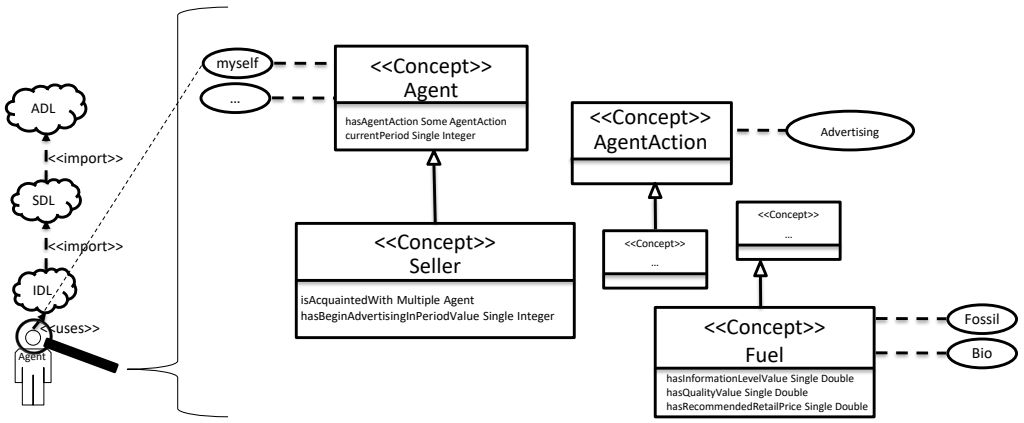


Figure 50: Seller IDL biofuel.

Besides containing the concepts of the market domain ontology, the IDLs define all relevant facts to model the state as well as the individual behaviour of the agents making extensive use of SWRL rules. The following SWRL rules are used to classify each individual agent into one of the four consumer segments (see previous section):

Price-sensitive consumer

The following rule determine the agent as a price sensitive consumer when the value of $w_1 = 0.6$, $w_2 = 0$ and $w_3, w_4 = 0.2$.

$$\begin{aligned} \text{Consumer(myself)} \wedge \text{hasW1Value(myself, ?w1)} \wedge \text{equal(?w1, 0.6)} \wedge \\ \text{hasW2Value(myself,?w2)} \wedge \text{equal(?w2, 0)} \wedge \\ \text{hasW3Value(myself,?w3)} \wedge \text{equal(?w3, 0.2)} \wedge \\ \text{hasW4Value(myself, ?w4)} \wedge \text{equal(?w4, 0.2)} \\ \Rightarrow \text{PriceSensitive(myself)} \end{aligned}$$

Product-quality consumer

The following rule determine the agent as a quality seeking consumer when the value of $w_1, w_4 = 0.2$, $w_2 = 0$ and $w_3 = 0.6$.

$$\begin{aligned} \text{Consumer(myself)} \wedge \text{hasW1Value(myself, ?w1)} \wedge \text{equal(?w1, 0.2)} \wedge \\ \text{hasW2Value(myself,?w2)} \wedge \text{equal(?w2, 0)} \wedge \\ \text{hasW3Value(myself,?w3)} \wedge \text{equal(?w3, 0.6)} \wedge \\ \text{hasW4Value(myself, ?w4)} \wedge \text{equal(?w4, 0.2)} \\ \Rightarrow \text{QualitySeeking(myself)} \end{aligned}$$

Eco consumer

The following rule determine the agent as a eco consumer when the value $w_1, w_3 = 0.1$, $w_2 = 0$ and $w_4 = 0.8$.

$$\text{Consumer(myself)} \wedge \text{hasW1Value(myself, ?w1)} \wedge \text{equal(?w1, 0.1)} \wedge$$
$$\text{hasW2Value(myself,?w2)} \wedge \text{equal(?w2, 0)} \wedge$$
$$\text{hasW3Value(myself,?w3)} \wedge \text{equal(?w3, 0.1)} \wedge$$
$$\text{hasW4Value(myself, ?w4)} \wedge \text{equal(?w4, 0.8)}$$
$$\Rightarrow \text{Eco(myself)}$$

Snob consumer

The following rule determine the agent as a snob consumer when the value $w_1 = 0$, $w_2 = 0.6$ and $w_3, w_4 = 0.2$.

$$\text{Consumer(myself)} \wedge \text{hasW1Value(myself, ?w1)} \wedge \text{equal(?w1, 0)} \wedge$$
$$\text{hasW2Value(myself,?w2)} \wedge \text{equal(?w2, 0.6)} \wedge$$
$$\text{hasW3Value(myself,?w3)} \wedge \text{equal(?w3, 0.2)} \wedge$$
$$\text{hasW4Value(myself, ?w4)} \wedge \text{equal(?w4, 0.2)}$$
$$\Rightarrow \text{Snob(myself)}$$

The SWRL rule which is used to determine the appropriate plan of a consumer agent for the next simulation round is formulated as follows:

$$\text{Consumer(myself)} \wedge \text{hasFuelLevel(myself, ?hfl)} \wedge$$
$$\text{hasFuelLevelThreshold(myself, ?hflt)} \wedge \text{lessThan(?hfl, ?hflt)}$$
$$\Rightarrow \text{hasNextAction(myself, needEventPlan)}$$

If the *fuel level* value of consumer falls below its individual threshold, a need event is triggered i.e. the next agent action will be set to *eedEventPlan* by using ontology reasoning techniques. This plan implements the process in which the product is chosen.

The following SWRL rule is used to determine the corresponding marketing activity plan of the seller agent. The agent is going to run the *advertisingPlan* when the period value is greater than or equal to a given value (i.e. 80 which corresponds to advertising setting to Kiesling).

$$\begin{aligned} &\text{Seller(myself)} \wedge \text{currentPeriod(myself, ?period)} \wedge \\ &\text{hasBeginAdvertisingInPeriodValue(myself, ?adv)} \wedge \\ &\text{greaterThanOrEqual(?period, adv)} \\ \Rightarrow\ &\text{hasNextAction(myself, advertisingPlan)} \end{aligned}$$

Again, agents communicate with other agents (e.g. they exchange information about product details) and this communication refers to knowledge items that belong to the IDL layer. In this scenario, agents exchange information about biofuel which may be totally new for the receiving agent. The agent then add new facts (e.g. information level of biofuel or price value) acquired through this information exchange into its belief base. When incorporating a new concept into its IDL (e.g. introducing biofuel through a marketing campaign), the agent can add the concept into its IDL directly because of sharing SDL and ADL (see Section 5.4).

7.2.3 Parameters and Results

Agents and their relations form a small world network of 30000 consumer agents and one seller agent built with preferential attachment using the Barabási algorithm which builds the network iteratively by adding agents to an initial graph (see Section 4.1.2). The overall number of agents was chosen due to limitations of the available hardware (two servers, each equipped with two quad core CPUs and 128GB RAM). The number of connections of

a new agent was set to 225. Any agent that is connected to more than 3000 agents is considered a hub. Due to the parameter settings of the network construction, approximately 1 percent of agents are hubs. The weight of the edges connecting nodes to a hub is randomly set to an opinion influence value between 0.8 and 1. This lets hubs act as opinion leaders who strongly influence buying decisions of other consumers through communication events (see Section 4.3). *Experience events* for an agent occur randomly in each round after the first time the agent has bought biofuel.

The travel behaviour of an agent is represented by a stochastic variable which is initially calculated for each agent and based on a normal distribution with a mean of 0.3 and a standard deviation of 0.05. In each time period the fuel level is reduced by this value. The individual threshold that leads to a stop at the filling station (see above *fuel level threshold*) is set to a random number between 0 (exclusive) and 0.20 (inclusive). The price value of biofuel is set to 1.20 € and the price value of fossil fuel is set to 1.00 €.

Initially, fossil fuel is the only fuel consumers can buy. At the beginning of the simulation the consumers are not even aware that there is an alternative product such as biofuel. The only fuel which is modelled in the private ontology is fossil fuel. When biofuel is launched into the market, the seller agent initiates marketing activities to spread information about the new product. Agents add this new information to their private IDL (see Section 5.4). The timing of the product launch can be chosen arbitrarily. For comparison reasons with Kiesling, the point in time is to time period 85. However, this has no significant effect on the curve progression, but only shifts the graph on the time axis.

Marketing activities focus on increasing the information level of each

agent within reach of the activity. A marketing activity contains the variable *information content value*. This is set to a random value between 0 and 1 and models the amount of information that is carried by the activity. The amount of marketing activities is limited to 1% of the overall number of participating agents in each period. Whenever an agent is reached by a marketing activity, the *information content value* is added to biofuel information level of the agent. If an agent is affected by more than one marketing activity in one time period, the maximum of the information content values of all activities will be added. As described above the decision of buying biofuel instead of fossil fuel finally depends on the individual utility values and the individual thresholds.

The time horizon is set to 400 rounds (i.e. time periods) and a total of 100 runs with varying random seeds. As expected, the results (see Fig. 51 and Fig. 52) are very similar to Kiesling's results. Differences occur because not all of Kiesling's parameter settings were published with exact values. Reasonable assumptions were made here.

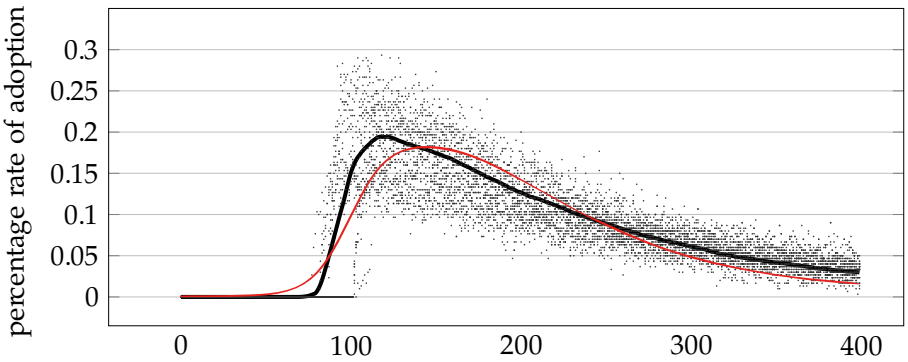


Figure 51: Biofuel adoption curve (points mark results of single simulation runs, thick line indicates averages by AGADE and the thin line is the average line published by Kiesling).

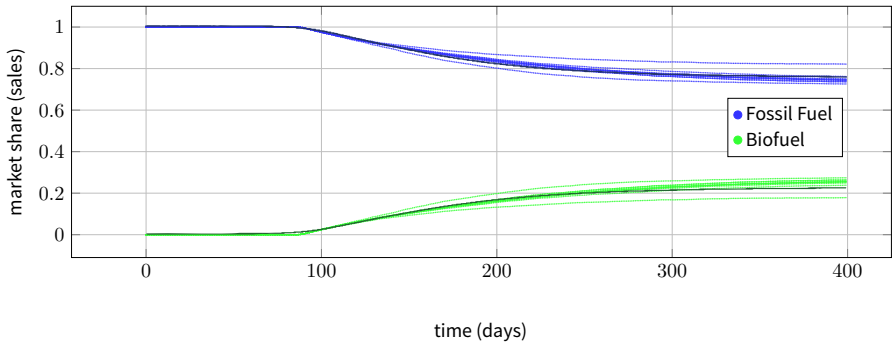


Figure 52: Unit sales volume market share (data generated by AGADE).

7.3 Business Game Experiments

The previous section described a personal preference scenario where marketing activities are crucial for placing a new product into a saturated market. Here, challenges certainly lie in a limited marketing budget and the unpredictability of the effects of applied marketing means. The overall goal is to increase revenues. In this case, business games will train the ability to plan effective campaigns that make optimal use of a given budget.

With regard to Section 3.1, business games are a specific subspecies of what is called a serious game. The concept combines business simulations and games to support management and entrepreneurial training [14, p. 3]. Business games are used as an educational means to go beyond mere theoretical contemplation and to simulate practice. They provide students with the opportunity to develop decision-making skills and improve confidence in situations of risk and uncertainty through the simulation of management practices across the enterprise in ever more complex real-life contexts.

This section shows how the fuel market scenario can be applied as a business game using AGADE.

7.3.1 Summary

Providing a user interface through which parameter settings can be modified during the simulation, AGADE lets users intervene in the simulations. Users can experiment with various settings and investigate strategies that may accelerate the adoption of biofuel by the market and thus increase the market share. After each period, users will receive feedback that describes the evolution of the market and can then draw conclusions. Users can analyse and contemplate results and decide what to do in the next period. Note that existing equation-based simulations usually do not present results to participants with real-time animations, as they can only display the final output of equations at the end of each round. An example for a possible action is cutting consumer prices which of course will have an impact on the sales figures and revenues. This impact will immediately be reflected by the system and the calculations of the next round. A possible hypothesis to be tested is "If you win opinion leaders, you will convince their community as well". This hypothesis can be tested by simulating target marketing [165, pp. 133-154], which can be done by identifying hubs in the social network and observing how word-of-mouth mechanisms let information diffuse through the network. Social network analysis can identify hubs by simply looking for agents with a high number of connections to fellow agents. As finding opinion leaders in real-world communities is difficult and expensive, it is well worth the effort to examine the effects of opinion leaders in a business simulation before starting to locate them in the community. Without simulations this would have to be done with field experimentation (compare [160]).

The simulation itself can be controlled through the GUI (see Section

6.3). The user interface control is displayed at the start of the simulation and then after a predefined number of steps (see Fig. 53). At each stop the user can specify the amount of money to be invested in marketing activities in the next rounds and select a marketing strategy (i.e. target marketing). Beyond that, the price of the product under observation (here biofuel) can be adjusted. The feature that the social graph is continuously updated visualises the dynamics of information diffusion.

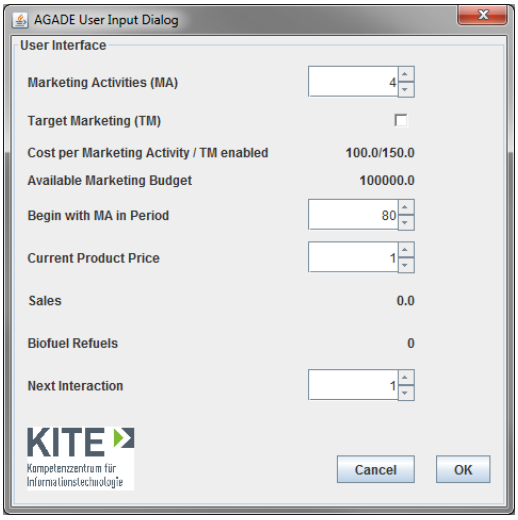


Figure 53: AGADE user input dialog.

An experiment was conducted with two restricted groups of students in the classroom (10 first year bachelor's program students and 20 second year master's program students, both business informatics) to investigate *how simulation settings respond to business decisions* at the Technische Hochschule Mittelhesen, Germany. Two groups of students with different levels of knowledge of the topic "marketing" and with different levels of experience in the use of business games were selected. While the first year bachelor students had never actively taken part in business games in the classroom before, the master program students already had used them and

were experienced in their use and effect (e.g. Topsim and Simultrain (see Section 3.1)). The experiment was divided into three phases: instructions, pre-experiment test and post-experiment test. The instructions provided in the first phase described the purpose of the experiment and gave instructions on how to use AGADE. The pre-experiment test consisted of questions about the law of demand and supply and other questions regarding marketing effects. The intention here was to establish a mental preparation of the test individuals towards the scenario of the game and eventually to assess the subjects' knowledge about particular economic concepts. The post-experiment was a questionnaire in which the reaction of the subjects and the user experience made after running the simulation was to be measured. For each question, the usual 5-point Likert scale from strongly disagree to strongly agree was used (see [166]). The overall duration of the experiment was 90 minutes. Table 6 shows the list of questions used.

Table 6: Survey questions

ID	Question
q1	The simulation course meets my expectations
q2	The simulation appears to be realistic
q3	Playing the simulation gave me new insights into market dynamics
q4	The simulation explained possible effects of marketing activities
q5	The simulation responds to my decisions
q6	The scenario was easily comprehensible
q7	The scenario was complex
q8	The scenario is suitable for a business game
q9	The scenario is suitable for a marketing training
q10	My theoretical marketing knowledge was adequate
q11	Practical relevance
q12	Encourages teamwork

All students played an individual game using the biofuel scenario (see Section 7.2). Player inputs were directly sent to the seller agent by using the user interface control as shown in Fig. 53. The following section discusses the survey responses.

7.3.2 Results and Discussion

This section deals with the results of the questionnaire in Table 6 from both student groups. Fig. 54 compares the mean values of the answers for both groups.

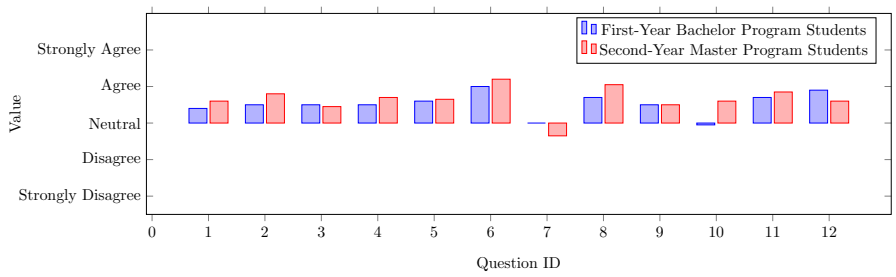


Figure 54: Results of the questionnaire

Both student groups responded with similar answers with the exception of question 10. The significant lower value of question 10 is due to the fact that one group was a first-year student group which had not visited a marketing lecture yet. Overall, the students reacted as was expected before the experiment (see questions 1, 2 and 5). The survey results show that the scenario is principally applicable for business simulations (q3, q4, q6, q7, q8, q9, q11, q12). Despite the different levels of knowledge of both groups there are only slight differences in the results. Note that both groups were made familiar with a necessary minimum of marketing theory before the experiments. All in all, the simulation of the consumer market was accepted as a valid model of reality and shows the applicability of the concept of using ontology based BDI agents to model consumer behaviour and communication. A particular positive aspect is that this approach allows to inspect the current state of the private ontology of an agent of interest at any time. The master student group used this quite often and highlighted such a possibility, as it gives insight into the agents' decisions and behaviour. Master students have used this possibility, because they have been made familiar with basic OWL concepts in a knowledge management lecture before, which the bachelor students have not.

7.4 Summary of Chapter 7

Chapter 7 has demonstrated how the three-layer ontology architecture can be used in various simulation scenarios. Section 7.1.1 has shown a simulation where agents act as potential buyers in a mobile phone market using the opinion leadership marketing mechanism. The agents were directly influenced from their neighbourhood through communication. The resulting distribution chart (generated by AGADE) corresponds to data published earlier. An extension of the mobile phone market with a more heterogeneous structure of market participants has been given in Section 7.1.2 (i.e. personal preferences). Simulation data was collected by running an online survey on a restricted group of individuals (students and staff from Edinburgh Napier University) and each survey response has been related to one private ontology. The brand distribution (generated by AGADE) was compared with brand distribution in the survey.

The three-layer ontology architecture has been applied to another market domain (i.e. fuel market) which has been described in Section 7.2. This scenario models the launch of a new product (i.e. biofuel) into an existing marketplace. Results were compared to data published earlier.

Finally in Section 7.3, the fuel market scenario has been used in an experiment with two restricted groups of students in the classroom where users have modified the parameters during the simulation. Survey responses have shown that the scenario is principally applicable to business simulations and the consumer market was accepted as a valid model of reality.

8 Conclusions and Future Research

In this chapter, the main findings of this thesis are summarised and reflected upon with regard to the research questions stated in Section 1.4. General conclusions are drawn and recommendations for future research are derived.

8.1 Conclusion on Ontology Based Business Simulations

This thesis has described a generic architecture to establish a connection between the world of agents and the world of ontologies. Multi-agent systems consist of agents that mutually interact and exist in an environment (see Section 2.1). Therefore, they are a natural instrument for modelling complex dynamic systems as they allow to directly describe components and interactions of such a system. Ontologies provide a means to equip agents with knowledge of their internal and external world thus extending modelling capabilities (see Section 2.2).

Business simulations are tools that can be applied to analyse and to understand complex structures e.g. organisations or markets (see Section 3.1). Therefore, the integration of ontologies and agents is suited to create business simulations. The agent-based framework AGADE (see Chapter 6) facilitates the creation of simulations with semantically advanced technology in the context of business simulations. Ontologies implement the knowledge base of each agent representing knowledge about their environment and their relations and their individual behaviour. Standardised formal languages like OWL (see Section 2.3) deliver a promising tool for the implementation

of these knowledge bases. This idea consequently follows [16], but the techniques developed here are unique to this thesis.

8.1.1 Research Question 1

To what extent can software agent methodologies be utilised in the context of a business game to enhance realistic gaming behaviour?

Unlike previous research into classical business game simulations built mathematically on systems of equations, AGADE shows an integration of a multi-agent based system and semantic technologies modelling individuals as core components of the simulations. Agents are parts of an overall given social structure (e.g. small-world network – see Section 4.1.1) which is not necessarily transparent for each agent in its full extent. The part of the structure the agent is aware of, is encoded in an OWL ontology available to each agent (i.e. IDL). Mutual influence of agents also is encoded there.

This thesis has demonstrated how different principal behaviours (see Chapter 4) of agents can be implemented by means of semantically enhanced BDI agents. Word-of-mouth dynamics and personal preference behaviour have been implemented in homogeneous as well as inhomogeneous communities of agents (see Section 7.1). Real-world communities were modelled and the results of the simulations compared with data drawn from surveys in this community. Furthermore, a scenario in which a new product is launched into a saturated market shows the potential of the approach to be applied in more complex contexts (see Section 7.2). Relevance of the model and the results has been checked through comparison with previously published material and field experiments with students in which AGADE has been used successfully as a rudimentary business game (see Section 7.3).

Accompanying surveys were used to measure the users' satisfaction with handling and results of the simulation.

Equipping each agent with its own individual IDL, as was done in the mobile phone scenario, becomes too expensive in large scale simulations. For the implementation of the personal preference scenario (see Section 7.1.2), the behavioural patterns and respective individual personal preferences resulted in 69 different IDLs which is a time-consuming task which can be reduced significantly by grouping agents according to relevant characteristics thus reducing the number of different ontologies to be implemented and by reusing available ontologies through import.

8.1.2 Research Question 2

How can ontologies handle and influence the activities of BDI agents?

Ontologies are used as a knowledge base for agents and allow access to powerful standardised inference engines that offer leverage for the agents' decision processes. Each agent is equipped with a personal instance of a reasoner and a personal private ontology (see Section 5.2). Personal behaviour, e.g. decision making, and effects of social communities are part of the model which is the basis for business games that simulate effects of decision making in communities. Proof of concept was given in a case study with a scenario (opinion leadership – see Section 7.1.1) in which agents are part of a typical social structure (small world network). The information about this structure is made available to the agents so that they may be aware of their position and their importance in their social environment.

The personal preference scenario (see Section 7.1.2) shows a more heterogeneous distribution of market participants in the same market segment

(mobile phone market) where consumers not only base their buying decision on opinion leaders, but also try to find the appropriate product according to their personal preferences. The process of collecting information about products is shown using message transfer and simple learning mechanisms (see Section 5.4). In this scenario, 67 different private IDL ontologies model the individual behaviour of consumers which may choose products from a pool of real-world phones and brands.

The third scenario (fuel market scenario – see Section 7.2) models the launch of a new product into an existing marketplace, namely biofuel into the saturated motor fuel market of Austria. A previously published agent-based model was reproduced using AGADE with its semantic components. Assessing the implementation processes of theses scenarios one important aspect can be emphasised: The use of ontologies considerably reduces coding efforts as behaviour can be declared with rules rather than with Java code.

8.1.3 Research Question 3

How can ontologies be adapted to different simulation scenarios? OWL ontologies allow a modular structure with reuse as existing ontologies can be imported. In their paper Heijst et al. present a number of ways how ontological descriptions of the contents of domain knowledge can be constructed and used to improve the knowledge engineering process [167]. They underpin that an application-specific ontology should be constructed early in the knowledge engineering process. In this thesis, a layered ontology is derived in which domain knowledge is hierarchically separated by its degree of generality. The three-layer ontology model approach (ADL, SDL and IDL – see Section 5.3) allows agents to share knowledge and create a basic common

understanding of their environment while enabling reuse of fundamental concepts and simple learning mechanisms. The generic approach allows the simulation of different scenarios.

This thesis has demonstrated how the three-layer ontology architecture can be applied to different simulation scenarios. While the ADL can typically be reused unchanged, given SDL structures have to be adapted to the relevant domain e.g. a market of a specialised type of product. The IDLs define all relevant facts to model the state as well as the individual behaviour of each agent making extensive use of SWRL (see Section 5.1.2) rules.

AGADE combines a multi-agent based system with semantic technologies for modelling each aspect of individual behaviour in different scenarios e.g. a dynamic market environment. Furthermore, classroom experiments in which students had to implement and run typical multi-agent scenarios as pharmaceutical supply chain (see [168]) and credit risk (see [169]). The results of the projects and the students' feedback have been used to confirm and improve the generic approach. The results of all simulation scenarios prove the principal applicability of the three-layer ontology architecture. Although, the scenarios can also be reproduced with other agent simulation frameworks, the flexibility and versatility of AGADE justify this approach. AGADE can potentially support arbitrary simulations as long as the underlying structure of the scenario can be modelled by means of ontologies (individual aspect) and social structure.

8.2 Future Work on Ontology Based Business Simulations

One principle area for further investigation within ontology based business simulations, is the creation of ontologies of a higher degree of generalisation. Here more general scenarios and behavioural patterns can be modelled. Basically, this means creating what is usually called an upper ontology. Both ADL and SDL can be generalised to support additional marketing theories and behavioural patterns (e.g. the price/value hypothesis, supply and demand model or customer loyalty) as well as scenarios beyond that (e.g. organisational behaviour). Available upper ontologies such as the Base Formal Ontology, Cyc (or OpenCyc) and to a certain extend WordNet are a starting points for these investigations (see [170]).

Modelling heterogeneous structures with a high number of individual ontologies is time-consuming and requires tedious work on details. The efficiency of the approach may be increased by using larger building blocks and by using a domain specific language designed to both accelerate and facilitate the development of powerful AGADE components. This language can improve the readability allowing developers to focus on what is important and making development less error-prone.

A Published Work

2016

Farrenkopf, T., Guckert, M., Urquhart, N., Wells, S. (2016). Ontology Based Business Simulations. Journal of Artificial Societies and Social Simulation, 19(4):14, 2016.

Farrenkopf, T., Guckert, M., Urquhart, N., Wells, S. (2016). AGADE - Scalability of Ontology Based Agent Simulations In: Y. Demazeau, I. Takayuki, J. Bajo, J.M. Escalona (Eds.) Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection: 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Proceedings, Lecture Notes in Computer Science Vol. 9662, pp 256-259. Springer International Publishing.

2015

Farrenkopf, T., Guckert, M., Urquhart, N. (2015). AGADE - Using communities of agents to provide realistic feedback in business simulations. In: Y. Demazeau, K.S. Decker, J. Bajo Pérez, F. De la Prieta (Eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability, Lecture Notes in Artificial Intelligence Vol. 9086, pp 93-106. Springer International Publishing.

Farrenkopf, T., Guckert, M., Urquhart, N. (2015). AGADE. In: Y. Demazeau, K.S. Decker, J. Bajo Pérez, F. De la Prieta (Eds.) Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability, Lecture Notes in Artificial Intelligence Vol. 9086, pp

271-274. Springer International Publishing.

2014

Farrenkopf, T., Guckert, M., Hoffmann, B., Urquhart, N. (2014). AGADE - How individual guidance leads to group behaviour and how this can be simulated. In: Muller, J., Weyrich, M., Bazzan, A. (Eds.) Multiagent System Technologies, Lecture Notes in Computer Science Volume 8732, pp 234-250. Springer International Publishing.

2013

Farrenkopf, T., Held, T. (2013). Talk to agents. In: Integration und Konnexion, AKWI, pp 59-59. News & Media, Berlin.

Farrenkopf, T., Willems, M. (2013). Business Games in Economy - State of Play and Way Forward. In: Integration und Konnexion, AKWI, pp 108-119. News & Media, Berlin.

B Description Logic Symbols

In logic, a set of symbols is commonly used to express logical representation. The following table lists the basic logic symbols.

Table 7: Basic logic symbols (adopted from [171]).

OWL Constructor	**Notation**	**Semantics**
owl:Thing	$\top$	is a special concept with every individual as an instance i.e. Thing
owl:Nothing	$\bot$	empty concept
intersectionOF	$C_1 \sqcap ... \sqcap C_n$	$Human \sqcap ... \sqcap Male$
unionOf	$C_1 \sqcup ... \sqcup C_n$	$Male \sqcup ... \sqcup Female$
complementOf	$\neg C$	$\neg Male$
oneOf	$\{x_1\} \sqcup ... \sqcup \{x_n\}$	$\{bob\} \sqcup ... \sqcup \{alice\}$
allValuesFrom	$\forall R.C$	$\forall hasParent.Human$
someValuesFrom	$\exists R.C$	$\exists hasParent.Doctor$
maxCardinality	$\geqslant nR$	$\geqslant 2hasParent$
minCardinality	$\leqslant nR$	$\leqslant 1hasParent$
subClassOf	$C_1 \sqsubseteq C_2$	$MobilePhone \sqsubseteq Product$
equivalentClass	$C_1 \equiv C_2$	$Woman \equiv Human \sqcap Female$
subPropertyOf	$P_1 \sqsubseteq R_2$	$hasDaughter \sqsubseteq hasChild$
equivalentProperty	$R_1 \equiv R_2$	$cost \equiv price$
transitiveProperty	$R^+ \sqsubseteq R$	$anchestor^+ \sqsubseteq ancestor$
type	$a : C$	$Alice : HappyMother$
property	$\langle a, b \rangle : R$	$\langle Alice, Bob \rangle : hasChild$

C Ontology Modelling Notation

There is no standard graphical ontology representation in the literature so far. HADZIC and colleagues presented a modelling notation syntax that is inspired by UML diagrams [17]. The following shows the relevant ontology modelling notations which are used in this thesis.

Table 8: List of modelling notations.

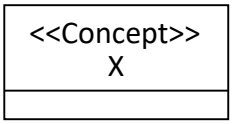
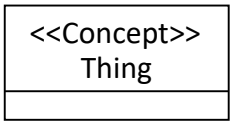
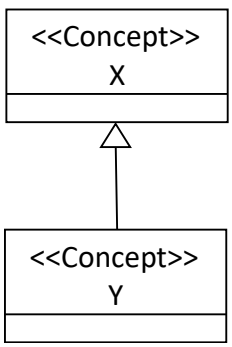
Concept	Notation	Semantics
Class		Ontology class *X*.
Thing Class		All classes in ontology are subclasses of class *Thing*.
Generalisation		To express either class-subclass or property-sub property.
		Class *Y* is subclass of class *X*.

Table 8: List of modelling notations.

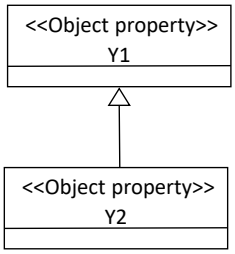
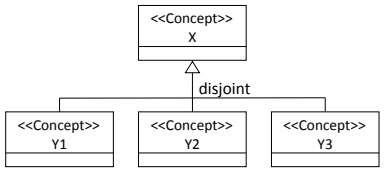
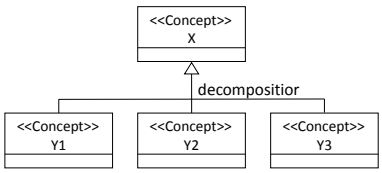
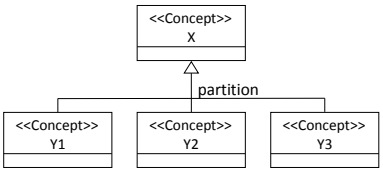
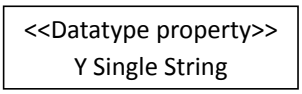
Concept	Notation	Semantics
		Object property *Y2* is sub property of object property *Y1*.
		Disjoint classes *Y1*, *Y2*, *Y3*.
		Decomposed classes *Y1*, *Y2*, *Y3*.
		Partition classes *Y1*, *Y2*, *Y3*.
Data type property		Data Type property *Y* which is functional and its type is `String`.

Table 8: List of modelling notations.

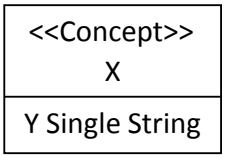
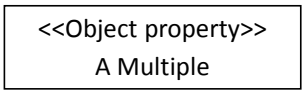
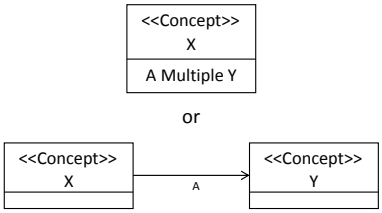
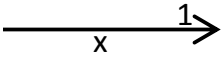
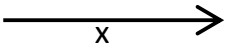
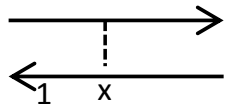
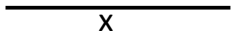
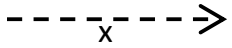
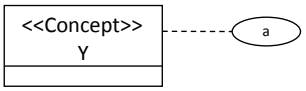
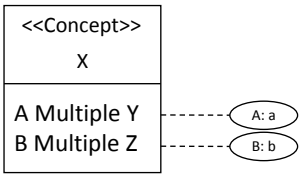
Concept	Notation	Semantics
		Class X has data type property Y which is functional and its type is `String`.
Object property		Object property A which is non-functional.
		Class X has relations with class Y. The relation considers as an object property named A which is non-functional property.
Property characteristic		
- Functional property		A functional property X.
- Non-functional property		A non-functional property X.

Table 8: List of modelling notations.

Concept	Notation	Semantics
- Inverse functional property		A inverse functional property *X*.
- Symmetric property		A symmetric property *X*.
- Transitive property		A transitive property *X*.
Class instance		*a* is individual of class *A*.
Property		*a* is individual of property *A* and *b* is an individual of property *B*.

D Certificate

PAAMS includes a special track for demonstrations. Demonstrations are intended to exhibit practical applications of multi-agent systems. The demo presentation was awarded with an IBM demonstration award. AGADE receives the third award of scientific excellence (see Fig. 55).

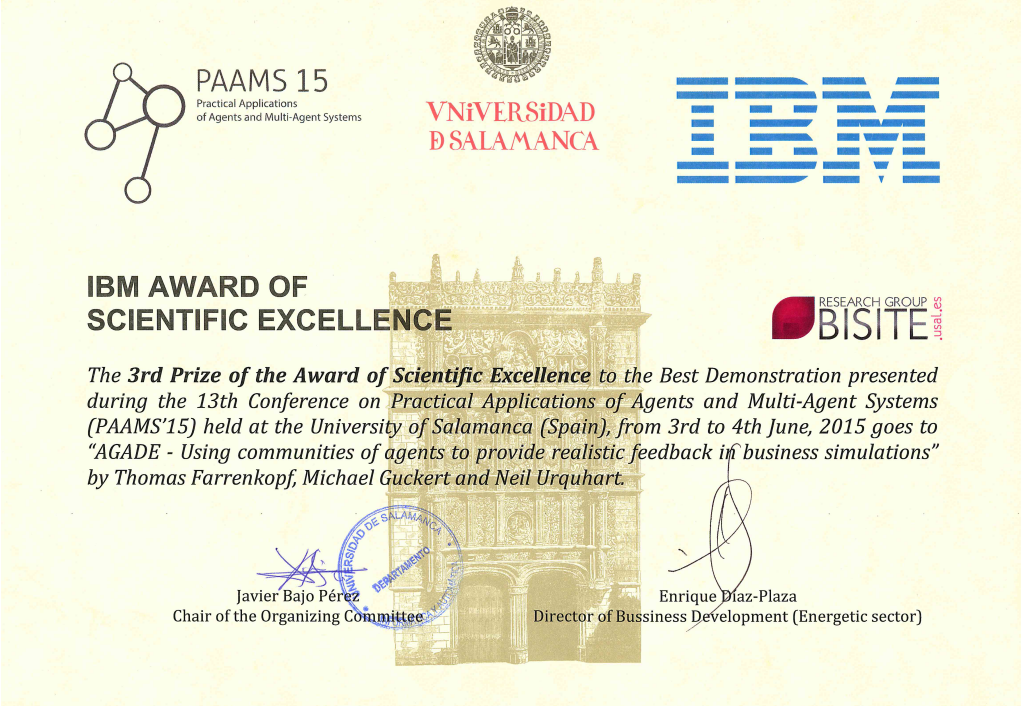


Figure 55: IBM Award of Scientific Excellence.

E RMI vs. Web Services

This section shows the source code used for the benchmark written in Java.

Web Services:

```
public class WebServiceServer {

    /**
     * Starts a simple server to deploy the web service.
     */
    public static void main(String[] args) {
        String bindingURI = "http://localhost:9898/md5WebService";
        MD5WebService webService = new MD5WebService();
        Endpoint.publish(bindingURI, webService);
        System.out.println("Server started at: " + bindingURI);
    }
}

@WebService
public class MD5WebService {
    @WebMethod
    public String hashString(String input) {
        try {
            MessageDigest msgDigest = MessageDigest.getInstance("MD5");
            byte[] inputBytes = input.getBytes();
            byte[] hashedBytes = msgDigest.digest(inputBytes);

            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < hashedBytes.length; i++) {
                sb.append(Integer.toString((hashedBytes[i] & 0xff) + 0x100, 16)
                        .substring(1));
            }

            return sb.toString();
        } catch (NoSuchAlgorithmException ex) {
            ex.printStackTrace();
            return "";
        }
    }
}

public class WebServiceClient {

    /**
     * Starts the web service client.
     */
    public static void main(String[] args) {
        MD5WebServiceService client = new MD5WebServiceService();
```

```
        long t = System.currentTimeMillis();
        MD5WebService md5Webservice = client.getMD5WebServicePort();
        for (int i = 0; i < 1000; i++) {
            md5Webservice.hashString("admin");
            // System.out.println("MD5 hash string: " + hash);
        }
        System.out.println(System.currentTimeMillis() - t);
    }
}
```

RMI:

```
public class ServerEndpoint {
    public ServerEndpoint() {
        try {
            System.setProperty("java.rmi.server.hostname", "localhost");
            // System.setProperty("java.rmi.transport.tcp.maxConnectionThreads",
            // Integer.MAX_ VALUE);
            Registry registry = LocateRegistry.createRegistry(8888);
            registry.rebind("Server", new ServerImpl());
        } catch (RemoteException ex) {
            System.out.println(ex.getMessage());
        }
        System.out.println("Server is running");
    }
}

public interface IServer extends Remote {

    public String hashString(String s) throws RemoteException;

}

public class ServerImpl extends UnicastRemoteObject implements IServer {

    protected ServerImpl() throws RemoteException {
        super();

    }

    public String hashString(String s) throws RemoteException {
        MessageDigest msgDigest = null;
        try {
            msgDigest = MessageDigest.getInstance("MD5");
        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        byte[] inputBytes = s.getBytes();
        byte[] hashedBytes = msgDigest.digest(inputBytes);
```

```
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < hashedBytes.length; i++) {
            sb.append(Integer.toString((hashedBytes[i] & 0xff) + 0x100, 16).substring(1));
        }

        return sb.toString();
    }

}
public class Client {

    private IServer server;

    public IServer getServerConnection() {
        if (server == null) {
            try {
                server = (IServer) Naming.lookup("//" + "127.0.0.1" + ":8888/Server");
            } catch (MalformedURLException | RemoteException | NotBoundException e) {
                e.printStackTrace();
            }
        }
        return server;
    }

    public static void main(String[] args) throws RemoteException {

        Client c = new Client();
        long t = System.currentTimeMillis();
        IServer serverConnection = c.getServerConnection();
        for (int i = 0; i < 1000; i++) {
            serverConnection.hashString("HelloWorld");
        }
        System.out.println(System.currentTimeMillis() - t);

    }

}
```

F Calculation of Happiness Value

The following pseudo code shows the mathematical operation of the happiness value used in the mobile phone market scenario (see Section 7.1). This algorithm has proved to be appropriate for running the mobile phone scenarios after intensive test-runs.

Algorithm 8 Update happiness value.

```
Ensure: triggered after each round
  function updateHappinessValue(currentHappinessValue)
    if rand < 0.5 then
      if currentHappinessValue > 5 then
        happinessValue = currentHappinessValue - 5
      end if
    else
      if currentHappinessValue > 30 then
        happinessValue = currentHappinessValue - randValue(30)
      else if currentHappinessValue > 15 then
        happinessValue = currentHappinessValue + randValue(70)
      else
        happinessValue = 0
      end if
    end if
    return happinessValue
  end function
```

Bibliography

[1] C. Reviews, *Effective Training*. Cram101, 2016, ISBN: 9781619066427.

[2] T. Grüne-Yanoff, "The explanatory potential of artificial societies", *Synthese*, vol. 169, no. 3, pp. 539–555, 2009.

[3] A. Waldherr and N. Wijermans, "Communicating social simulation models to sceptical minds", *Journal of Artificial Societies and Social Simulation*, vol. 16, no. 4, p. 13, 2013.

[4] T. Berners-Lee, J. Hendler, O. Lassila, *et al.*, "The semantic web", *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[5] W3C OWL Working Group, *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 11 December 2012, Available at `http://www.w3.org/TR/2012/REC-owl2-overview-20121211/`.

[6] O. Lassila, R. R. Swick, W. Wide, and W. Consortium, *Resource description framework (rdf) model and syntax specification*, 1998.

[7] J. Hendler and D. L. McGuinness, "The darpa agent markup language", *IEEE Intelligent systems*, vol. 15, no. 6, pp. 67–73, 2000.

[8] D. L. McGuinness, F. Van Harmelen, *et al.*, "Owl web ontology language overview", *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[9] J. H. Holland, "Studying complex adaptive systems", *Journal of Systems Science and Complexity*, vol. 19, no. 1, pp. 1–8, 2006.

[10] J. W. Forrester, "System dynamics, systems thinking, and soft or", *System Dynamics Review*, vol. 10, no. 2-3, pp. 245–256, 1994.

[11] C. Gros, *Complex and adaptive dynamical systems*, ser. Springer Complexity. Berlin, Heidelberg: Springer, 2008, ISBN: 978-3-540-71873-4.

[12] U. Wilensky and W. Rand, *Introduction to agent-based modeling: Modeling natural, social, and engineered complex systems with NetLogo*. Cambridge, Mass.: The MIT Press, 2015, ISBN: 978-0262731898.

[13] E. Kaynak, J. Wolfe, and J. Keys, *Business Simulations, Games, and Experiential Learning in International Business Education*. Taylor & Francis, 2012, ISBN: 9781136371516.

[14] N. Baldissin, S. Bettiol, S. Magrin, and F. Nonino, *Business game-based learning in management education*. Business Game srl, 2013, ISBN: 978-1-291-32255-2.

[15] M. Bratman, *Intention, plans, and practical reason*. Cambridge, MA: Harvard University Press, 1987, ISBN: 9780674458185.

[16] T. R. Gruber, "A translation approach to portable ontology specifications", *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[17] M. Hadzic, P. Wongthongtham, T. Dillon, and E. Chang, "Notations for the integrated ontology and multi-agent system design", in *Ontology-Based Multi-Agent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 143–163, ISBN: 978-3-642-01904-3.

[18] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice", *The knowledge engineering review*, vol. 10, no. 02, pp. 115–152, 1995.

[19] G. Weiss, *Multiagent systems: A modern approach to distributed artificial intelligence*. Cambridge and Mass: MIT Press, 1999, ISBN: 0262232030.

[20] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995, ISBN: 0-13-103805-2.

[21] L. R. Grimm, "Psychology of knowledge representation", *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 5, no. 3, pp. 261–270, 2014.

[22] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?", *AI magazine*, vol. 14, no. 1, p. 17, 1993.

[23] T. Gruber, *What is an ontology?*, 1992.

[24] F. Baader, *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.

[25] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang, "Hermit: An owl 2 reasoner", *Journal of Automated Reasoning*, vol. 53, no. 3, pp. 245–269, 2014.

[26] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51 –53, 2007, Software Engineering and the Semantic Web.

[27] W3C, *Owl web ontology language*, 2004.

[28] J. v. Leeuwen, *Algorithms and complexity*, 1. ed., 2. impr. Amsterdam: Elsevier, 1992, ISBN: 0444880712.

[29] M. Dean, G. Schreiber, S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "Owl web ontology language reference", *W3C Recommendation February*, vol. 10, 2004.

[30] W. Recommendation, *Owl web ontology language overview*, 2004.

[31] M. Horridge, S. Jupp, G. Moulon, A. Rector, R. Stevens, and C. Wroe, *A practical guide to building owl ontologies using protégé 4 and co-ode tools*, The University of Manchester, Ed., 2007.

[32] P. V. Biron and A. Malhotra, "Xml schema part 2: Datatypes second edition", *W3C Recommendation*, 2004.

[33] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner", *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[34] H. Gomaa, *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge, and New York: Cambridge University Press, 2011, ISBN: 978-0521764148.

[35] S. Cranefield, S. Haustein, and M. Purvis, "Uml-based ontology modelling for software agents", in *Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, 2001.

[36] S. Cranefield and M. Purvis, *Ontologies for interaction protocols*, 2002.

[37] S. Cranefield, J. Pan, and M. Purvis, "A uml ontology and derived content language for a travel booking scenario", in *Ontologies for Agents: Theory and Experiences*, ser. Whitestein Series in Software Agent Technologies, S. Cranefield, V. Tamma, T. W. Finin, and S. Willmott, Eds., Basel: Birkhäuser-Verlag, 2005, pp. 259–276, ISBN: 3-7643-7237-0.

[38] S. Cranefield, M. Purvis, and M. Nowostawski, *Is it an ontology or an abstract syntax? modelling objects, knowledge and agent messages.* 2000.

[39] R. W. Collier, C. F. B. Rooney, and G. M. P. O'Hare, "A uml-based software engineering methodology for agent factory", in *Sixteenth International Conference on Software Engineering and Knowledge Engineering*, 2004, pp. 20–25.

[40] Object Management Group, *Ontology definition metamodel*, 2009.

[41] G. Hillairet. (2007). Atl use case - odm implementation (bridging uml and owl). Accessed on January 10, 2016.

[42] S. Groppe, *Data Management and Query Processing in Semantic Web Databases*. Springer Berlin Heidelberg, 2011, ISBN: 9783642193576.

[43] S. Greco and C. Molinaro, *Datalog and Logic Databases*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2015, ISBN: 9781627051149.

[44] C. Baker and K. Cheung, *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences*, ser. SpringerLink: Springer e-Books. Springer US, 2007, ISBN: 978-0-387-48438-9.

[45] J. W. Forrester, *World Dymamics*, 2nd ed. Cambridge: Wright-Allen Press, 1971, ISBN: 978-1563270598.

[46] J. D. Sterman, *Business dynamics: Systems thinking and modeling for a complex world*. Boston [etc.]: Irwin/McGraw-Hill, 2000, ISBN: 0072311355.

[47] R. A. Wells, "Management games and simulations in management development: An introduction", *Journal of Management Development*, vol. 9, no. 2, pp. 4–6, 1990.

[48] F. D. Laramée, *Game Design Perspectives*, ser. Advances in Computer Graphics and Game Development Series. Charles River Media, 2002, ISBN: 9781584500902.

[49] J. A. Wolfe and B. Keys, *Business simulations, games and experiential learning in international business education*. New York: International Business Press, 1997, ISBN: 9780789000415.

[50] R. Sauter, "La modélisation des facteurs humains dans la gestion de projets", PhD thesis, EPFL, 1996.

[51] TATA INTERACTIVE SYSTEMS GmbH, *Topsim - ecommerce seminarleiterhandbuch*, TATA INTERACTIVE SYSTEMS GmbH, Ed., 2007.

[52] C. H. Hommes, "Heterogeneous agent models in economics and finance", *Handbook of computational economics*, vol. 2, pp. 1109–1186, 2006.

[53] A. Belyanin, "Daniel Kahneman and Vernon Smith: Nobel prize for the feeling of reality", *Vorposy Economiki*, vol. 1, 2003.

[54] The Royal Swedish Academy of Sciences, *Press release: Psychological and experimental economics*, 2002.

[55] V. Smith, *Vernon smith on markets and experimental economics: Podcast interview*, 2007.

[56] G. N. Gilbert, *Agent-based models*. Los Angeles: Sage Publications, 2008, ISBN: 9781412949644.

[57] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems", *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7280–7287, 2002.

[58] S. Chauhan, *Microeconomics: An Advanced Treatise*, ser. Eastern Economy Edition. Prentice-Hall Of India Pvt. Limited, 2009, ISBN: 9788120338609.

[59] K. Steiglitz, M. L. Honig, and L. M. Cohen, *A computational market model based on individual action*, 1996.

[60] G. Orcutt, "A new type of socio-economic system", *International Journal of Microsimulation*, vol. 1, no. 1, pp. 3–9, 2007.

[61] P. Williamson, *Editorial*, International Journal of Microsimulation, Ed., 2007.

[62] Y. Chen, X. Su, J. Rao, and H. Xiong, *Agent-based microsimulation of economy from a complexity perspective*, 2000.

[63] C. O'Donoghue, H. Redway, and J. Lennon, "Simulation migration in the pensim2 dynamic microsimulation model", *International Journal of Microsimulation*, vol. 3, pp. 65–79, 2010.

[64] M. G. Mueller and P. d. Haan, "How much do incentives affect car purchase? agent-based microsimulation of consumer choice of new cars—part i: Model structure, simulation of bounded rationality, and model validation", *Energy Policy*, vol. 37, no. 3, pp. 1072–1082, 2009.

[65] B. Raney, N. Cetin, A. Vollmy, M. Vrtic, K. Axhausen, and K. Nagel, "An agent-based microsimulation model of swiss travel: First results", *Networks and Spatial Economics*, vol. 3, pp. 23–41, 2003.

[66] G. Iori, "A microsimulation of traders activity in the stock market: The role of heterogeneity, agents' interaction and trade frictions", *Journal of Economic Behaviour & Organization*, vol. 49, pp. 269–285, 2002.

[67] D. K. Gode and S. Sunder, "Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality", in *Journal of Political Economy*, University of Chicago Press, Ed., vol. 1, Chicago: University of Chicago Press, 1993, pp. 119–137.

[68] D. Cliff and J. Bruten, *Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets*, 1997.

[69] ——, *Minimal-intelligence agents for bargaining behaviors in market-based environments*, 1997.

[70] R. Das, J. E. Hanson, J. O. Kephart, and G. Tesauro, "Agent-human interactions in the continuous double auction", in *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'01, Seattle, WA, USA: Morgan Kaufmann Publishers

Inc., 2001, pp. 1169–1176, ISBN: 1-55860-812-5, 978-1-558-60812-2 [Titel anhand dieser ISBN in Citavi-Projekt ÃČÂijbernehmen].

[71] D. Cliff, *Evolutionary optimization of parameter sets for adaptive software-agent traders in continuous double auction markets*, 2001.

[72] A. Malucelli, A. P. Rocha, and E. Oliveira, "B2b transactions enhanced with ontology-based services", in *in ICETE'04 - 1st International Conference on E-business and Telecommunication Networks*, 2004, pp. 10–17.

[73] H. Stuckenschmidt and I. J. Timm, "Adapting communication vocabularies using shared ontologies", in *Proceedings of the Second International Workshop on Ontologies in Agent Systems, Workshop at 1st International Conference on Autonomous Agents and Multi-Agent Systems*, 2002, pp. 15–19.

[74] A. Malucelli, D. Palzer, and E. Oliveira, "Combining ontologies and agents to help in solving the heterogeneity problem in e-commerce negotiations", in *International Workshop on Data Engineering Issues in E-Commerce*, IEEE, 2005, pp. 26–35, ISBN: 0-7695-2401-X.

[75] R. Mendoza and M.-A. Williams, "Ontology based object categorisation for robots", in *Australasian Ontology Workshop (AOW 2005)*, T. Meyer and M. A. Orgun, Eds., ser. CRPIT, vol. 58, Sydney and Australia: ACS, 2005, pp. 61–67.

[76] P. Krysta, M. Li, T. R. Payne, and N. Zhi, "Mechanism design for ontology alignment", in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1587–1588.

[77] M. P. Singh, P. E. Cannata, M. N. Huhns, N. Jacobs, T. Ksiezyk, K. Ong, A. P. Sheth, C. Tomlinson, and D. Woelk, *The carnot heterogeneous database project: Implemented applications. distributed and parallel databases*, 1996.

[78] J.-i. Akahani, K. Hiramatsu, and K. Kogure, "Coordinating heterogeneous information services based on approximate ontology translation", in *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*, ACM Press, 2002, pp. 10–14.

[79] K. Hiramatsu, N. Communicationware, and B. Benjamin, "Map-based user interface for digital city kyoto", in *In Proc. of INET2000 The Internet Global Summit*, 2000.

[80] L.-K. Soh, "Collaborative understanding of distributed ontologies in a multiagent framework: Experiments on operational issues", in *Ontologies for Agents: Theory and Experiences*, V. Tamma, S. Cranefield, T. W. Finin, and S. Willmott, Eds. Basel: Birkhäuser Basel, 2005, pp. 95–120, ISBN: 978-3-7643-7361-0.

[81] S. Deen and K. Ponnamperuma, "Dynamic ontology integration in a multi-agent environment", in *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, IEEE, 2006, 6 pp–378, ISBN: 0-7695-2466-4.

[82] T. Finin, R. Fritzson, D. McKay, and R. McEntire, *Specification of the kqml agent-communication language*, 1993.

[83] M. Genesereth, R. E. Fikes, R. Brachman, T. Gruber, P. Hayes, R. Letsinger, V. Lifschitz, R. Macgregor, J. McCarthy, P. Norvig, and R. Patil, *Knowledge interchange format version 3.0 reference manual*, 1992.

[84] Foundation for Intelligent Physical Agents, *Fipa acl message structure specification*, 2002.

[85] ——, *History of fipa*, 1997.

[86] T. Finin, *What ever happened to kqml?*, 2006.

[87] J. L. Austin, *How to do things with words*, ser. William James lectures. Clarendon Press, 1962.

[88] J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969, ISBN: 9780521096263.

[89] ——, *Intentionality: An Essay in the Philosophy of Mind*, ser. Cambridge Paperback Library. Cambridge University Press, 1983, ISBN: 9780521273022.

[90] J. R. Searle and D. Vanderveken, *Foundations of Illocutionary Logic*. University Press, 1985, ISBN: 9780521263245.

[91] B. Schiemann and U. Schreiber, "Owl dl as a fipa acl content language", in *Proceedings of the Workshop on Formal Ontology for Communicating Agents*, Malaga, Spain, 2006.

[92] Á. F. Moreira, R. Vieira, R. H. Bordini, and J. Hübner, *Agent-oriented programming with underlying ontological reasoning*, 2005.

[93] B. Schiemann, "Vereinigung von owl-dl-ontologien für multi-agentensysteme", PhD thesis, Universität Erlangen-Nürnberg, Universitätsstraße. 4 and 91054 Erlangen, 2010.

[94] A. Fuhrmann and H. Rott, *Logic, Action, Information: Essays on Logic in Philosophy and Artificial Intelligence*. De Gruyter, 1996, ISBN: 9783110139945.

[95] K. Kravari and N. Bassiliades, "A survey of agent platforms", *Journal of Artificial Societies and Social Simulation*, vol. 18, no. 1, p. 11, 2015.

[96] E. Mangina, *Review of software products for multi-agent systems*, Agentlink, Ed., UK, 2002.

[97] S. Khalique, M. Jamshed, H. Suguri, H. Ahmad, Arshad Al, and M. Awan, "Assessment of owl and fipa-sl as semantic language", in *Proceedings of the IEEE Symposium on Emerging Technologies, 2005*, IEEE, 2005, pp. 536–541, ISBN: 0-7803-9247-7.

[98] F. Bellifemine, G. Caire, and D. P. A. Greenwood, *Developing multi-agent systems with JADE*. Chichester: Wiley, 2007, ISBN: 978-0-470-05747-6.

[99] Emorphia Limited, *Fipa-os agent toolkit*, 2000.

[100] Reticular Systems Inc., *Agentbuilder reference manual: An integrated toolkit for constructing intelligent software agents*, 2004.

[101] N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas, *Jack intelligent agents (tm) - summary of an agent infrastructure*, 2001.

[102] M. J. Huber, *Jam: A bdi-theoretic mobile agent architecture*, 1999.

[103] A. S. Rao, *Agentspeak(l): Bdi agents speak out in a logical computable language*, 1996.

[104] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007, vol. 8.

[105] D. M. Da Silva and R. Vieira, "Argonaut: Integrating jason and jena for context aware computing based on owl ontologies (short paper", in *Proceedings of the Workshop on Agents, Web Services, and Ontologies–Integrated Methodologies (AWESOME'007) held as part of MALLOW'007, Durham 3–7 September*, Citeseer, 2007.

[106] A. Freitas, R. H. Bordini, F. Meneguzzi, and R. Vieira, "Towards integrating ontologies in multi-agent programming platforms", in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2015 IEEE/WIC/ACM International Conference on*, IEEE, vol. 3, 2015, pp. 225–226.

[107] G. Boella, R. Damiano, J. Hulstijn, and L. van der Torre, "A common ontology of agent communication languages: Modeling mental attitudes and social commitments using roles", *Applied Ontology*, vol. 2, pp. 217–265, 2007.

[108] G. Weiß, *Agentenorientierte Softwareentwicklung: Methoden und Tools: Gerhard Weiß ; Ralf Jakob*, ser. Xpert.press. Berlin, Heidelberg, and New York: Springer, 2005, ISBN: 3-540-00062-3.

[109] G. Caire and D. Cabanillas, *Jade tutorial: Application-defined content languages and ontologies*, 2002.

[110] MadKit Team, *The madkit team*, 2012.

[111] Jacques Ferber and Olivier Gutknecht, *Aalaadin: A meta-model for the analysis and design of organizations in multi-agent systems*, 1997.

[112] O. Gutknecht and J. Ferber, "The madkit agent platform architecture", in *In Agents Workshop on Infrastructure for Multi-Agent Systems*, 2000, pp. 48–55.

[113] FIPA-OS, *Fipa-os v2.1.0 distribution notes*, 2001.

[114] ——, *Fipa-os developers guide*, 2001.

[115] I. Horrocks, *Owl: A description logic based ontology language for the semantic web*, 2006.

[116] A. Radcliffe-Brown, *Structure and Function in Primitive Society: Essays and Addresses*. Free Press, 1952.

[117] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*, ser. Princeton studies in complexity. Princeton University Press, 1999, ISBN: 9780691117041.

[118] J. Travers and S. Milgram, "An experimental study of the small world problem", *Sociometry*, vol. 32, no. 4, pp. 425–443, 1969.

[119] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks", *Rev. Mod. Phys.*, vol. 74, pp. 47–97, 1 2002.

[120] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks", *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[121] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph", *CoRR*, 2011.

[122] A. Polynikis, *Random walks and scale-free networks*, 2006.

[123] G. P. Lantos, *Consumer Behavior in Action: Real-Life Applications for Marketing Managers*. M. E. Sharpe Incorporated, 2010, ISBN: 9780765629111.

[124] C. Fill and G. Hughes, *CIM Coursebook Marketing Communications 07/08*. Taylor & Francis, 2013, ISBN: 9781136419638.

[125] E. Rogers, *Diffusion of Innovations, 5th Edition*. Free Press, 2003, ISBN: 9780743258234.

[126] D. J. Watts, *Small worlds: The dynamics of networks between order and randomness*, ser. Princeton studies in complexity. Princeton, N.J, and Woodstock: Princeton University Press, 2004, ISBN: 0691117047.

[127] A.-L. Barabási, *Linked: The new science of networks*. Cambridge: Perseus, 2002, ISBN: 0452284392.

[128] P. Lerud, Molander-Hjorth, and F. Söderstjerna, "Opinion leaders and word-of-mouth: A case study of masai barefoot technology shoes", PhD thesis, Lund University, Sweden, 2007.

[129] Arvind Arasu, Jasmine Novak, and John Tomlin, *Pagerank computation and the structure of the web: Experiments and algorithms*, 2002.

[130] T. Farrenkopf, M. Guckert, B. Hoffmann, and N. Urquhart, "Agade", in *Multiagent System Technologies: 12th German Conference, MATES 2014, Stuttgart, Germany, September 23-25, 2014. Proceedings*, J. P. Müller, M. Weyrich, and A. L. C. Bazzan, Eds., Cham: Springer International Publishing, 2014, pp. 234–250, ISBN: 978-3-319-11584-9.

[131] M. Baker, *The Marketing Book*. Taylor & Francis, 2012, ISBN: 978-1-136-35691-9.

[132] Y. Kwark, J. Chen, and S. Raghunathan, "Online product reviews: Implications for retailers and competing manufacturers", *Information systems research*, vol. 25, no. 1, pp. 93–110, 2014.

[133] T. Farrenkopf, M. Guckert, and N. Urquhart, "Agade using personal preferences and world knowledge to model agent behaviour", in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection: 13th International Conference, PAAMS 2015, Salamanca, Spain, June 3-4, 2015, Proceedings*, Y. Demazeau, K. S. Decker, J. Bajo Pérez, and F. de la Prieta, Eds., Cham: Springer International Publishing, 2015, pp. 93–106, ISBN: 978-3-319-18944-4.

[134] E. J. Norling, "Modelling human behaviour with bdi agents", PhD thesis, University of Melbourne, 2009.

[135] M. O'Connor. (2009). The semantic web rule language.

[136] M. Krötzsch, *Description Logic Rules*, ser. Ciencias (E-libro–2014/09). IOS Press, 2010, ISBN: 9781614993421.

[137] M. O'Connor, *Swrl language faq*, 2016.

[138] A. Pokahr, L. Braubach, and K. Jander, "The jadex project: Programming model", in *Multiagent Systems and Applications*, ser. Intelligent Systems Reference Library, M. Ganzha and L. C. Jain, Eds., vol. 45, Springer Berlin Heidelberg, 2013, pp. 21–53, ISBN: 978-3-642-33322-4.

[139] L. Sterling and K. Taveter, *The Art of Agent-oriented Modeling*, ser. Intelligent robotics and autonomous agents. MIT Press, 2009, ISBN: 9780262013116.

[140] I. Niles and A. Pease, "Towards a standard upper ontology", in *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, ACM, 2001, pp. 2–9.

[141] N. F. Noy, M. A. Musen, *et al.*, "Algorithm and tool for automated ontology merging and alignment", in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*, 2000.

[142] T. Farrenkopf, M. Guckert, and N. Urquhart, "Demo paper: Agade", in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection: 13th International Conference, PAAMS 2015, Salamanca, Spain, June 3-4, 2015, Proceedings*, Y. Demazeau, K. S. Decker, J. Bajo Pérez, and F. de la Prieta, Eds., Cham: Springer International Publishing, 2015, pp. 271–274, ISBN: 978-3-319-18944-4.

[143] T. Farrenkopf, M. Guckert, N. Urquhart, and S. Wells, "Demo paper: Agade", in *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection: 14th International Conference, PAAMS 2016, Sevilla, Spain, June 1-3, 2016, Proceedings*, ser. Lecture Notes in Computer Science, Y. Demazeau, T. Ito, J. Bajo, and M. Escalona, Eds., Springer International Publishing, 2016, pp. 256–259, ISBN: 9783319393247.

[144] Matthew Horridge and Sean Bechhofer, "The owl api: A java api for owl ontologies", *Semantic Web Journal 2*, vol. 2, no. Special Issue on Semantic Web Tools and Systems, pp. 11–21, 2011.

[145] I. Singh, S. Brydon, G. Murray, V. Ramachandran, T. Violleau, and B. Stearns, *Designing Web Services with the J2EE 1.4 Platform: JAX-RPC, XML Services, and Clients*. Pearson Education, 2004.

[146] R. Orfali and D. Harkey, *CLIENT/SERVER PROGRAMMING WITH JAVA AND CORBA, (With CD)*. John Wiley & Sons, 2007.

[147] N. A. B. Gray, "Comparison of web services, java-rmi, and corba service implementation", in *Fifth Australasian Workshop on Software and System Architectures*, 2004.

[148] L. Iannone and A. Rector, *Calculations in owl*, University of Manchester, 2008.

[149] I. Horrocks *et al.*, *Semantic Web Rule Language (SWRL)*. W3C Member Submission, 21 May 2004, Available at `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.

[150] A. Sánchez-Macián, E. Pastor, J. E. de López Vergara, and D. López, "Extending swrl to enhance mathematical support", in *Web Reasoning and Rule Systems: First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007. Proceedings*, M. Marchiori, J. Z. Pan, and C. d. S. Marie, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 358–360, ISBN: 978-3-540-72982-2.

[151] K. Wenzel and H. Reinhardt, "Mathematical computations for linked data applications with OpenMath", in *24th OpenMath Workshop*, Bremen, Germany, Jul. 2012.

[152] F. Yuzhang, L. Yang, and L. Yuan-Fang, "Discovering anomalies in semantic web rules", *Secure System Integration and Reliability Improvement*, vol. 0, pp. 33–42, 2010.

[153] A. Jena, "A free and open source java framework for building semantic web and linked data applications", *Available online: jena. apache. org/(accessed on 28 April 2015)*, 2015.

[154] E. Sirin and B. Parsia, "Sparql-dl: Sparql query for owl-dl.", in *OWLED*, vol. 258, 2007.

[155] ——, "Sparql-dl: Sparql query for owl-dl", in *In 3rd OWL Experiences and Directions Workshop (OWLED-2007*, 2007.

[156] I. Palmisano, *Dl queries with a real reasoner*, (Accessed on November 08, 2016), 2015.

[157] J. F. D. Team. (2010). Jung - java universal network/graph framework. (Accessed on December 04, 2016), (visited on 12/04/2016).

[158] M. A. Musen, "The protégé project: A look back and a look forward", *AI Matters*, vol. 1, no. 4, pp. 4–12, Jun. 2015.

[159] T. Farrenkopf, M. Guckert, N. Urquhart, and S. Wells, "Ontology based business simulations", *Journal of Artificial Societies and Social Simulation*, vol. 19, no. 4, p. 14, 2016.

[160] J. S. Coleman, E. Katz, H. Menzel, and Columbia University. Bureau of Applied Social Research, *Medical innovation: a diffusion study*, ser. Advanced Study in Sociology. Bobbs-Merrill Co, 1966.

[161] R. Cross, A. Parker, and L. Sasson, *Networks in the Knowledge Economy*. New York: Oxford University Press, 2003, ISBN: 9780195347883.

[162] J. H. Holland, *Signals And Boundaries*. MIT, 2012.

[163] E. Kiesling, M. Günther, C. Stummer, R. Vetschera, and L. M. Wakolbinger, "A spatial simulation model for the diffusion of a novel biofuel on the austrian market", in *ECMS*, A. Bargiela, S. Ali, D. Crowley, and E. Kerckhoffs, Eds., ser. Proceedings of the 24th European Conference on Modelling and Simulation (ECMS 2010), Kuala Lumpur, Malaysia, 2010, pp. 41–49, ISBN: 978-0-9564944-1-2.

[164] M. Günther, C. Stummer, L. M. Wakolbinger, and M. Wildpaner, "An agent-based simulation approach for the new product diffusion of a novel biomass fuel", *Journal of the Operational Research Society*, vol. 62, no. 1, pp. 12–20, 2011.

[165] A. Weinstein, *Handbook of Market Segmentation: Strategic Targeting for Business and Technology Firms, Third Edition*. Taylor & Francis, 2013, ISBN: 9781135185657.

[166] R. Likert, *A Technique for the Measurement of Attitudes*, ser. A Technique for the Measurement of Attitudes Nr. 136-165. publisher not identified, 1932.

[167] G. Van Heijst, A. T. Schreiber, and B. J. Wielinga, "Using explicit ontologies in kbs development", *International journal of human-computer studies*, vol. 46, no. 2-3, pp. 183–292, 1997.

[168] G. Jetly, C. L. Rossetti, and R. Handfield, "A multi-agent simulation of the pharmaceutical supply chain", *Journal of Simulation*, vol. 6, no. 4, pp. 215–226, 2012.

[169] S. Jonsson, "Credit risk: An agent-based model of post-credit decision actions and credit losses in banks", *Journal of Simulation*, vol. 6, no. 4, pp. 253–266, 2012.

[170] V. Mascardi, V. Cordì, and P. Rosso, "A comparison of upper ontologies.", in *WOA*, vol. 2007, 2007, pp. 55–64.

[171] I. Horrocks, "Owl: A description logic based ontology language", in *International Conference on Logic Programming*, Springer, 2005, pp. 1–4.