ADAPTATION TO UNEXPECTED CHANGES: WHERE ECOSYSTEMS AND
MULTI-AGENT SYSTEMS MEET

A thesis submitted to The University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Humanities

2010

Cesar MARIN PITALUA

MANCHESTER BUSINESS SCHOOL
Contents

Abstract 13

Declaration and copyright 14

About the author 17

1 Introduction 18
   1.1 Problem description and research goal 18
   1.2 The research approach and methodology 19
   1.3 Hypotheses and contributions to knowledge 20
   1.4 Thesis structure 21

2 Ecosystem Domains 23
   2.1 Domains with the need for adaptation to unexpected changes 23
      2.1.1 Business ecosystems 23
      2.1.2 Digital service ecosystems 24
      2.1.3 Manufacturing 25
      2.1.4 Transport 26
      2.1.5 City modelling 26
   2.2 Adaptation in ecosystem domains 27

3 Adaptation Approaches in Multi-Agent Systems 29
   3.1 Agent and multi-agent system 29
   3.2 Dimensions for analysing adaptation approaches 30
   3.3 Classification of adaptation approaches 32
      3.3.1 Adaptation as an automaton 33
      3.3.2 Adaptation as a control system 34
      3.3.3 Adaptation as semi-isolated evolution 35
      3.3.4 Adaptation as complex interactions 36
      3.3.5 Adaptation as an ecosystem 37
   3.4 Adaptation principles for MAS development 38
## CONTENTS

3.5 Adaptation properties of complex adaptive systems ................. 40
3.6 Characteristics of natural ecosystems as CAS ......................... 41
  3.6.1 Biosphere as CAS ........................................... 41
  3.6.2 Hierarchical structures ..................................... 42
  3.6.3 Food-web dynamics .......................................... 42
  3.6.4 Mechanics of complex aggregates ................................ 42
  3.6.5 Interactions and complexity .................................. 43
  3.6.6 Circular flow of materials .................................... 44
  3.6.7 Ecodynamics ................................................ 45
  3.6.8 A thermodynamic principle .................................... 45
3.7 Other classification frameworks ...................................... 45
  3.7.1 Adaptation approaches by intelligence dimensions .................. 46
  3.7.2 Adaptation approaches by their research scope .................... 46
3.8 Discussion .................................................................. 47

4 A Formal Model of a Dynamic Agent-based Ecosystem: DAEM ............ 49
4.1 The role of CAS adaptation properties .................................. 49
  4.1.1 Property Prop1. Aggregation .................................... 49
  4.1.2 Property Prop2. Diversity ...................................... 50
  4.1.3 Property Prop3. Flow .......................................... 51
  4.1.4 Property Prop4. Non-linearity ................................... 52
4.2 An overview of DAEM .................................................. 53
  4.2.1 Agent interactions .............................................. 53
  4.2.2 Environment mediation and agent senses .......................... 54
  4.2.3 From agent interactions to complexity and adaptation .......... 55
  4.2.4 Supporting ecosystem domains .................................. 56
4.3 Formalising DAEM ........................................................ 56
  4.3.1 Definition of a resource ......................................... 57
  4.3.2 Definition of an agent and its classification ........................ 57
  4.3.3 Agent capability to forget ....................................... 59
  4.3.4 Resource evaluation and feedback .................................. 60
  4.3.5 Definition of the environment ..................................... 62
  4.3.6 Situatedness in the environment .................................. 63
  4.3.7 Agent displacements in the environment .......................... 65
  4.3.8 Environment-mediated local interactions .......................... 67
  4.3.9 Environment-mediated surrounding awareness ...................... 68
  4.3.10 Agent foraging behaviour ...................................... 72
  4.3.11 Linking producers and consumers .................................. 75
  4.3.12 Link chaining .................................................. 77
6.2.4 Representatives of the business ecosystem .......................... 115
6.3 A practical execution example in the automotive business ecosystem 117
   6.3.1 Foraging for suppliers .............................................. 117
   6.3.2 Service evaluation and feedback .................................. 118
   6.3.3 Engaging suggested interactions .................................. 119
   6.3.4 Competition of suppliers .......................................... 119
   6.3.5 Rejecting suggested interactions .................................. 121
   6.3.6 Authorising interaction ............................................. 122
   6.3.7 A final analysis of the business ecosystem case ............... 123
6.4 Execution context of a digital service ecosystem ....................... 124
   6.4.1 User layer ............................................................. 125
   6.4.2 Digital service layer ................................................ 126
   6.4.3 DAEM layer ........................................................... 126
6.5 A Web service ecosystem: an example .................................... 126
   6.5.1 A new service provider enters the ecosystem .................. 127
   6.5.2 Request for a better service ....................................... 128
   6.5.3 A final analysis of the Web service ecosystem example ...... 130
6.6 Discussion .................................................................. 130

7 Comparison With Other Ecosystem Approaches ............................. 132
   7.1 Degrees of adaptation .................................................... 132
   7.2 Ecosystem models ......................................................... 133
      7.2.1 ECHO ................................................................. 133
      7.2.2 Evolved biological ecosystems ................................... 133
      7.2.3 Algebra and visual notation for service interfaces .......... 133
      7.2.4 Ecological calculus ................................................. 134
      7.2.5 Population-based adaptive system .............................. 134
      7.2.6 Coordination model to long-term transactions .............. 135
   7.3 Ecosystem architectures ................................................ 135
      7.3.1 DIET ................................................................. 135
      7.3.2 DBE ................................................................. 136
      7.3.3 DAEDALUS .......................................................... 136
      7.3.4 Open negotiation environment .................................... 136
      7.3.5 Request-based virtual organisations ............................ 137
      7.3.6 Architecture for unanticipated environments ............... 137
      7.3.7 Tuple based ........................................................ 137
   7.4 Comparison .............................................................. 138
8 Conclusions

8.1 Contributions ........................................... 140
8.2 Future work ............................................... 141
8.3 Final remarks ............................................ 143

Bibliography .................................................. 144

A Literature Review of Adaptation in Multi-Agent Systems 156

A.1 Automaton .................................................. 156
   A.1.1 A social reasoning for adaptation .................. 156
   A.1.2 The AMAS theory ...................................... 157
   A.1.3 Adaptive heuristic for opponent classification (AdHoc) .... 157
   A.1.4 Reusable software components ..................... 158
   A.1.5 Memory-based adaptation ............................ 158
   A.1.6 Environment adaptation .............................. 158
   A.1.7 A generic component-based architecture ............ 159
   A.1.8 Protocol composition ................................. 159
   A.1.9 A social reasoning framework InFFrA ................. 160

A.2 Control system ............................................ 160
   A.2.1 Computational ecosystems ............................ 160
   A.2.2 Organisational adaptation simulations ............... 160
   A.2.3 An adaptive organisational framework ............... 161
   A.2.4 An Artificial Immune System ARTIS ................. 161
   A.2.5 An economic model .................................... 161
   A.2.6 An adaptive organisational policy .................. 162
   A.2.7 Adaptive web site agent .............................. 162
   A.2.8 Emergence of adaptive searching rules .............. 163
   A.2.9 Monitoring interdependence graphs .................. 163
   A.2.10 Economic firm simulation ............................ 164
   A.2.11 A dynamic opponent modelling ..................... 164
   A.2.12 Meta-rules in adaptive economic firms ............. 165
   A.2.13 Cooperative information-sharing .................... 165
   A.2.14 Dialogue adaptation ................................. 165

A.3 Semi-isolated evolution ................................. 166
   A.3.1 Genetic learning ...................................... 166
   A.3.2 The evolution of MAS structure ..................... 166
   A.3.3 The evolution of behavioural rule-sets ............... 167
   A.3.4 Anthill framework ..................................... 167
   A.3.5 Ethogenetics .......................................... 167
   A.3.6 The DIET toolkit experiments ........................ 168
| A.3.7 Biological ecosystems                      | 168 |
| A.3.8 An advice-exchange mechanism              | 168 |
| A.3.9 Evolution of strategies                   | 169 |
| A.3.10 Bottleneck neural networks               | 169 |
| A.4 Complex interactions                       | 169 |
| A.4.1 Emergent design system                    | 169 |
| A.4.2 Coordination artefacts                    | 170 |
| A.5 Ecosystems                                  | 170 |
| A.5.1 ECHO                                      | 170 |
| A.5.2 The DIET toolkit                          | 171 |
List of Figures

1.1 Chapter inter-dependency diagram. ........................................ 22
3.1 Adaptation as an automaton. ................................................. 34
3.2 Adaptation as a control system. ............................................. 35
3.3 Adaptation as semi-isolated evolution. ................................... 36
3.4 Adaptation as complex interactions. ...................................... 37
3.5 Adaptation as an ecosystem. ............................................... 38
4.1 Agent interactions. ............................................................. 54
4.2 Agent senses in the environment. .......................................... 55
4.3 DAEM at the ecosystem level. .............................................. 56
4.4 Reference model for supporting ecosystem domains. ............... 57
5.1 DAEM simulation prototype architecture. .............................. 83
5.2 DAEM message hierarchy. ................................................... 84
5.3 Experimental ecosystem. ...................................................... 85
5.4 Accumulated number of simulation cycles. .............................. 90
5.5 Accumulated number of changes of preference links. ................ 91
5.6 Confidence to obtain 5 preference links. ............................... 92
5.7 Stability of the number of preference link changes. ................. 93
5.8 Convergence of preference links with $f = 6.0$. ...................... 94
5.9 Convergence of preference links with $f = 8.0$. ...................... 95
5.10 Preference changes with an environment factor $f = 6.0$. .......... 97
5.11 Preference changes with an environment factor $f = 8.0$. .......... 98
5.12 Convergence of preference links in a centralised approach. ..... 100
5.13 Preference changes in a centralised approach. ...................... 101
5.14 DAEM simulation snapshots. .............................................. 103
6.1 DAEM architecture. ......................................................... 106
6.2 Macro level of the business ecosystem execution context. ........ 114
6.3 Micro level of the DAEM execution context. ............................ 115
6.4 Agent $p$ forages for services. ................................. 118
6.5 Service evaluation and feedback. ............................... 119
6.6 Engaging suggested interaction. ............................... 120
6.7 Changes are detected through service evaluations. .......... 120
6.8 Rejecting suggested interactions. ............................. 121
6.9 Authorising suggested interaction. ............................ 122
6.10 Macro level of the digital service ecosystem execution context. . 125
6.11 A new service provider is found in the ecosystem. .......... 128
6.12 Inter layer communication when there is a new service ...... 128
6.13 Interlayer communication when there is a new request ...... 129
List of Tables

1.1 Hypotheses, tests, and contributions to knowledge. 20

3.1 Analysis of adaptation approaches in MASs. 33
3.2 Classification framework of adaptation in MASs. 34

5.1 Parameter summary for best DAEM results. 102

7.1 Approaches to unexpected changes. 138

8.1 Contributions and validated hypotheses. 141
List of Algorithms

6.1 Agent behaviour. .................................................. 109
6.2 Environment dynamics. ........................................... 112
Abstract

Unexpected changes occurring in complex and dynamic domains render supporting systems unsuited to the new conditions. Examples of such domains include business ecosystems, digital service ecosystems, manufacturing, transport, and city modelling. These are regarded as ecosystem domains. Multi-agent systems are seen as an appropriate technology for their support, yet they lack the required ability to adapt to unexpected changes.

The research presented in this thesis aims to create a multi-agent system based in-silico model endowed with the capability of adaptation to unexpected changes occurring in ecosystem domains. The approach taken consists of applying adaptation properties of complex adaptive systems, such as natural ecosystems, to multi-agent systems to create one which can cope with unexpected changes.

A dynamic agent-based ecosystem model called DAEM is formalised by combining characteristics of natural ecosystem and principles of adaptive multi-agent systems. A set of experiments is presented using a DAEM prototype to demonstrate its resilience to unexpected changes in a hypothetical ecosystem. A comparison is made against a simulated typical solution for showing how DAEM is more resilient to unexpected changes than the typical approach. This supports the claim of this thesis that DAEM represents a significant contribution to knowledge.

A software embodiment of DAEM to drive adaptation in ecosystem domains is presented and placed in an execution context evaluated by two practical examples of ecosystem domains. These show how DAEM suggests interactions to the supporting system of the execution context, and incorporates taken decisions into the ecosystem regarding interactions with other individuals. This supports the claim that the DAEM software embodiment is suitable for providing adaptation support in ecosystem domains, thus representing another significant contribution of this thesis.

The contributions to knowledge of this thesis are then a) a formal model of a dynamic agent-based ecosystem called DAEM; and b) a software embodiment of DAEM, called DAEM layer, to support adaptation in ecosystem domains. Future work includes further tests to analyse patterns and make estimations in existing ecosystems, among others.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

1. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and s/he has given The University of Manchester the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

2. Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the John Rylands Library of Manchester. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.

3. The ownership of any patents, design, trade marks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of any copyright works, for example graphs and tables (“Reproductions”) which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

4. Further information on the conditions under which disclosure, publication and exploitation of this thesis, the Copyright and any Intellectual Property Rights and/or Reproductions described in it may take place is available from the Head of the Manchester Business School (or the Vice-President) and the Dean of the Faculty of Humanities.
Para Gisela,
más que mi compañera de vida,
eres mi amiga incondicional.

Gracias por partirme la naranja :-)
Many many thanks to

- Nikolay, for all the support you have given me throughout the PhD. Especially for giving me the freedom to do what I wanted :-)

- Consejo Nacional de Ciencia y Tecnología (CONACyT), for the funds provided through the sponsorship 1972197/218103. All the people I contacted at CONACyT were always kind and helpful.

- Iain, for your support and encouragement during the development of the contributions of this thesis.

- Giovanna and Babis (my PhD examiners), for their comments to improve this work. I enjoyed the Viva.

- Gisela, for sharing with me all the “joys” of the PhD; for listening (and actually paying attention) to all my thoughts and feelings.

- Marco, for all the endless discussions of science, research, and everything else.

- My parents and sister, for encouraging me all the time (and for asking me many times what my research was about).

- Microsoft, that interview back in 2003 helped me realise I wanted to do research. If you had offered me a position I would’ve probably declined it I think, but thanks for the rejection letter anyway. (I still keep it.)
About the author

Cesar A. Marin is currently a Research Associate at the Centre for Service Research, part of the Manchester Business School. His research is mainly about the applications of adaptive and self-organising approaches to service systems. He is currently coordinating the research contributions on semantic interoperability to the project COMMIUS, funded by the European Commission (EC) under the 7th Framework Programme.

Cesar’s background is on computer science focused on distributed artificial intelligence applied to business and organisation environments. He obtained both an MSc and a Diploma in Intelligent Systems, and a BS in Computer Systems Engineering at the Monterrey Institute of Technology, in Monterrey, Mexico.

Before joining The University of Manchester, Cesar was a Research Assistant at the Centre for Intelligent Systems at the Monterrey Institute of Technology, where he worked in research areas such as decentralised agent-based workflow systems, enterprise information and knowledge distribution, and behavioural adaptive modelling.
Chapter 1

Introduction

Software systems are increasingly complex and dynamic typically due to the increase of demand and the abundance of information, driving software developers to distribute, and diversify supporting systems. Consequently changes in the system frequently occur, sometimes unexpectedly, due to the inherent complexity.

Agent technology is seen as a suitable solution to build these supporting systems because of the agents’ ability to form distributed multi-agent systems (MASs) and their ability of autonomous problem-solving [Jennings et al., 1996]. Nevertheless MASs cannot easily cope with the frequent changes occurring in such environments. This is due to the way software systems are commonly built, where the software designer must envisage the complete set of behaviours and interactions in the system [Capera et al., 2003]. Likewise, in MAS development the designer has to foresee all potential situations and interactions an agent may be involved in, and determine the agents’ optimal behaviour beforehand [Alonso et al., 2003b].

One way to deal with the vicissitudes when encountering increasingly complex and dynamic domains is to furnish agents with the capability to adapt to unexpected changes [Capera et al., 2003]. Adaptation is seen as a pervasive property intrinsic to human behaviour which agents should possess in such environments [Hayes-Roth, 1995]; a true adaptive behaviour is a continuous and dynamic process [Maes, 1994]; an essential feature in MASs and not only an emergent characteristic [Guessoum, 2004] as it is usually conceived. The research reported in this thesis aims to provide MASs with the capability to adapt by themselves to unexpected changes in complex, dynamic environments.

1.1 Problem description and research goal

Not surprisingly, research interest has been attracted to the area of adaptation in MASs, cf. [Alonso et al., 2003a, Kudenko et al., 2005]. Diverse and fragmented ap-
1.2 The research approach and methodology

Approaches have been followed including evolutionary computation, control systems and machine learning. Nonetheless most of the approaches enable adaptation to already-known situations (cf. [van Splunter et al., 2003]) or assume that the environment will never change (cf. [O’Riordan, 2005]). These approaches still depend on the stability of the environment which make them unsuitable for dynamic domains. This motivates the search for more suitable adaptation approaches not depending on the environment stillness.

In nature there are systems formed by sets of individuals showing complex behaviours which are evidently not apprehended by the underlying individuals. These systems face continuous changes in their extremely dynamic environment and adapt accordingly. Examples of such systems are the human immune system which adapts to combat unexpected intruders; viruses mutate their structure in order to survive and propagate the population (disease); natural ecological systems (ecosystems) contain species adapting to alterations to resource availability.

All these are examples of complex adaptive systems (CASs), and possess properties permitting them to adapt to unexpected environment variations [Holland, 1995]. By transferring the properties governing CASs into MASs, it could be possible to attain applications adaptable to unexpected changes in dynamic and complex environments.

Therefore, this research project aims to create a multi-agent system based in-silico model endowed with the capability of adaptation to unexpected changes occurring in ecosystem domains. Intuitively, an unexpected change is a not-foreseen alteration of a significant impact to the system. The systemic nature of the change is not apprehended by the system underlying elements, yet they sense some localised effects.

1.2 The research approach and methodology

In order to accomplish the research goal described above, the approach taken in this research consists of analysing adaptation properties formulated within the field of CASs and applying them to MASs, providing a framework for developing MASs which can cope with unexpected changes occurring in complex, dynamic environments. Natural ecosystems have been chosen as the main object of study aiming to reveal core principles and structures of CASs, so that these can be transferred into MASs. This choice is motivated by the intuitiveness of core structure, and the wealth of available literature. The focus is on domains with a complex, dynamic environment resembling an ecosystem (to keep the analogy) such as a business ecosystem, manufacture, digital (service) ecosystem, transport, and city modelling.

The methodology followed in this research is the Design Science [Hevner et al.,
Introduction

Table 1.1: Hypotheses, tests, and contributions to knowledge.

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Tests</th>
<th>Contributions to Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1. It is possible to create a functional adaptive MAS based on ecosystem adaptation properties.</td>
<td>A prototype implementation.</td>
<td>A formal model of a Dynamic Agent-based Ecosystem (DAEM).</td>
</tr>
<tr>
<td>H2. The resulting MAS adapts to unexpected changes.</td>
<td>Experiments using unexpected changes.</td>
<td></td>
</tr>
<tr>
<td>H3. The resulting MAS is more resilient to unexpected changes than a centralised approach.</td>
<td>Comparison of experiments using a simulated central directory.</td>
<td></td>
</tr>
<tr>
<td>H4. The resulting MAS supports adaptation to unexpected changes which other approaches cannot in practical scenarios.</td>
<td>Examples of practical scenarios and a comparison of approaches.</td>
<td>A DAEM layer architecture for supporting ecosystems domains.</td>
</tr>
</tbody>
</table>

2004] due to its suitability to problems exhibiting complex interactions between the problem sub components and their solutions, such as in the targeted ecosystem domains mentioned above.

The methodology itself focuses on the process of discovering knowledge by developing an artefact and its subsequent testing [Hevner et al., 2004]. The artefacts produced in this research consist of an MAS model, based on a set of formal definitions, and a software architecture which provides a context for operationalising the model in practical situations. Furthermore, following the evaluation guidelines suggested by [Hevner et al., 2004] for this type of research, two relevant evaluation methods are used to test the artefacts: experimental simulation and descriptive scenarios, respectively.

Moreover, the nature of the methodology involves an iterative process alternating between designing and testing. The results presented in this thesis correspond to the last iteration of the process. All these aspects will be presented in further chapters of this thesis as described in Section 1.4.

1.3 Hypotheses and contributions to knowledge

The hypotheses considered in this research were tested according to the IT artefacts developed and their related evaluations. Such hypotheses and their testing methods are explained below showing how they contribute to the output of this research. A summary can be seen in Table 1.1.

**H1. It is possible to create a functional adaptive MAS based on ecosystem adaptation properties.** To test this hypothesis an IT artefact was created named the Dynamic Agent-based Ecosystem Model (DAEM). A prototype based on DAEM's underlying elements was implemented as a proof of concept.
H2. **The resulting MAS adapts to unexpected changes.** To test this hypothesis, the developed prototype was put in a controlled laboratory scenario where unexpected changes altered the whole system yet it managed to adapt itself accordingly.

H3. **The resulting MAS is more resilient to unexpected changes than a centralised approach.** To test this hypothesis, a centralised approach was simulated and put through unexpected changes as well. The results were compared analytically against the results obtained when using the ecosystem approach; and

H4. **The resulting MAS can support adaptation to unexpected changes which other approaches cannot in practical scenarios.** For this hypothesis the testing method was different in nature: a generalised software architecture based on DAEM was designed and run on “pencil and paper” scenarios. Additionally, a comparison to related approaches is also presented.

The testing of hypotheses demonstrates the validity of the main contributions to knowledge obtained from this research: a) an MAS ecosystem based model for complex, dynamic environments and capable to adapt to unexpected changes; and b) a software architecture reflecting the model’s capability of adaptation to unexpected changes.

### 1.4 Thesis structure

The structure of the rest of the thesis is explained below. A complementary diagram explaining the chapter inter-dependencies is shown in Figure 1.1.

**Chapter 2** provides an overview of domains in which the occurrence and frequency of unexpected changes demand an approach which is better suited to handling these changes than the existing ones. These domains include business ecosystems, digital service ecosystems, manufacturing, transport, and city modelling. Then common characteristics of these domains are drawn which help define the concepts of *unexpected change* and *adaptation*.

**Chapter 3** points out that software agents are a suitable technology for supporting the domains mentioned in the previous chapter. The work in the area of MAS adaptation is reviewed; approaches are classified, stating an opportunity area for achieving adaptation to unexpected changes, i.e. adaptation as an ecosystem. Then from a selected literature in ecosystem modelling, characteristics are drawn as essential to an ecosystem to function and adapt to environmental changes.
Chapter 4 develops the proposed approach of MAS adaptation by systematically aligning principles from MAS engineering and characteristics from natural ecosystem modelling. The proposed approach underpins a dynamic agent-based ecosystem model called DAEM for supporting adaptation to unexpected changes. A formal description of DAEM is also presented in this chapter.

Chapter 5 describes a prototype, which was developed to enable a number of experiments to test the capability of DAEM for adaptation. An analysis of the results obtained is also presented.

Chapter 6 provides details of a software architecture based on DAEM which considers existing supporting systems and incorporates them in the adaptation process. Moreover, two practical scenarios are presented where the software architecture supports adaptation to unexpected changes. A discussion is also included about how typical approaches cannot deal with unexpected changes in these scenarios.

Chapter 7 presents a literature review and a comparison to other ecosystem models and architectures. The comparison shows how DAEM is able to adapt to unexpected changes whereas other approaches cannot.

Chapter 8 summarises the obtained results, emphasises how this research impacts the state-of-the-art and gives suggestions for further developments in the field.
Chapter 2

Ecosystem Domains

This chapter provides an overview of example domains in which the occurrence and frequency of unexpected changes demand for a more suitable approach for their support. These example domains are business ecosystems, digital service ecosystems, manufacturing, transport, and city modelling. A definition of adaptation is then introduced as a process to overcome unexpected changes in these domains.

The chapter presents the aforementioned ecosystem domains in Section 2.1. Then the concept of adaptation is defined in Section 2.2.

2.1 Domains with the need for adaptation to unexpected changes

Complex and dynamic environments face unexpected changes rendering supporting systems unsuited to the new conditions. This section provides a brief description of example domains with such complex and dynamic environments. These domains are business ecosystems, digital service ecosystems, manufacturing, transport, and city modelling. The intention is not to review the literature related to them, but only to provide an overview of the domains which require supporting systems with the ability of adaptation to unexpected changes.

2.1.1 Business ecosystems

Current competitive markets drive companies to engage in partnerships and alliances to achieve both steadily growth and competitive advantage. The volatility of contemporary markets cause enterprise partnerships to form opportunistically. Thus in order to prosper, companies must continually adapt to the evolving markets. The essence of this domain is captured by the concept of business ecosystem.
A business ecosystem is a strategic planning concept introduced by [Moore, 1993] and is defined as a set of co-evolving companies developing capabilities in response to wide-ranging innovations such as new products, lower prices, better quality, etc. Those capabilities are typically in the form of other innovations, motivating the initial innovating company to improve a product or to introduce new ones as a response. This creates a never-ending cause-response process leading companies to co-evolve [Moore, 1993]. Innovations typically trigger a domino effect in the market such that an introduced innovation changes the market making other companies to innovate further which changes the market again and so on, creating an ever-changing dynamic environment [Peltoniemi and Vuori, 2005].

Typically a business ecosystem is formed by a set of companies forming supply chains and interact by consuming products from each other, by forming alliances, or by competing for a bigger share of the market. For instance consider the business ecosystem of the automotive industry in which a car company buys the individual car parts (doors, car seats, instrument panels, etc.) from different suppliers which in turn have their own suppliers and so on. At any point in the supply chains innovations are introduced by competing companies changing the offer in the market which could make interested companies to change suppliers. Other companies are affected by such changes when they notice variations in their purchase orders making them consider innovating their own products and so on.

All these changes occur unexpectedly because competing companies cannot predict when the changes will happen. So in order to cope with such dynamism, companies must detect and react promptly to unexpected changes which makes the whole business ecosystem to adapt continually to the ever-changing circumstances [Hol et al., 2007]. Approaches to business ecosystems in the literature mainly focus on other specific features of business ecosystems such as optimal partnership creation, cf. [Pappas et al., 2007, Wang and De Wilde, 2008], with no focus at all on adaptation to unexpected changes. Yet it is a property business ecosystems must possess.

### 2.1.2 Digital service ecosystems

A digital service ecosystem is a term used in this thesis to encompass similar concepts namely a digital environment, digital ecosystem, service ecosystem and variations of these. A digital environment has been loosely defined as a virtual space containing digital species such as software components, agents, services, business models, rules, etc. [Muntaner Perich and De la Rosa Esteva, 2007]. A digital ecosystem has been conceptualised as a software system for exploring properties of natural ecosystems in practical domains [Briscoe et al., 2007]. A (Web) service
2.1 Domains with the need for adaptation to unexpected changes

ecosystem is simply described in [Barros et al., 2005] as a logical collection of Web services, whereas [Quitadamo et al., 2007] considers a service ecosystem as a collection of autonomous software components providing services.

A digital service ecosystem is a complex and dynamic environment due to a number of reasons: a) there is a dense and increasing population of services/components/devices [Quitadamo et al., 2007, Zambonelli and Viroli, 2008]; b) the technology used is changing and becoming more complex [Viroli and Zambonelli, 2010]; and c) the applications are more difficult to manage [Hiel et al., 2008]. Consequently digital service ecosystems are evolving rapidly and without much control [Quitadamo et al., 2007]. Because of all this, there is a need for adaptation to unforeseen changes in this domain [Barros et al., 2005, Hiel et al., 2008].

Approaches in the area have proposed diverse ways to create digital service ecosystems, for instance by emphasising the need of predictability and reliability of services [Küster et al., 2008]. Other approaches focus on using virtual organisations, cf. [Svirskas et al., 2008], or software agents, cf. [Hiel et al., 2008], yet they assume that by simply using these technologies adaptation to (unexpected) changes is already covered, which does not happen. Clearly there is still a need for an approach to adaptation to unexpected changes.

2.1.3 Manufacturing

Manufacturing is a complex and dynamic environment due to the market demands and societal pressure to increase the production pace [Van Brussel et al., 1998]. Production itself requires more sophisticated system solutions to cope with frequent changes occurring in its environment, which becomes unstable because of changing demands of the markets themselves [Monostori et al., 2006].

In production scheduling and control, orders are placed to build and/or pack a (set of) product(s), involving a process composed by jobs/machines which consume resources. The resources are limited and the jobs contributing to an order may need to be executed in a specific sequence. Orders are diverse in the number of product units to complete, the number of involved jobs, their sequence and timing among other considerations. The load vary depending on the order being processed and the maximum capacity of the machines. Fluctuations in placed orders, resource availability, production site layout, among other variants make this environment complex and dynamic in which unexpected changes occur.

Likewise, inventory management have been pointed out as a complex environment [Wurman et al., 2008] for similar reasons. Because of the complexity in manufacturing, support for adaptation is considered a need yet to be satisfied [Monostori et al., 2006].
2.1.4 Transport

Transport deals with people and freight moving from one place to another in an urban environment. This domain is commonly seen as dynamic because of all the players and their interactions involved in it, for instance private cars, lorries, buses, trains and underground among others, in addition to the infrastructure: streets and sensors underneath, traffic lights, (electronic) signs, etc. Typically the need for transport is called demand and involves the flow of vehicles, pedestrians and freight, and the transport offer is called supply which involves the infrastructure and services [Bazzan et al., 2008].

The fluctuations in the transport demand during a period of time, unexpected accidents and faults, delays, energy consumption, and a limited transport supply make this domain increasingly complex and calls for the need of approaches to distribute the demand within the infrastructure and optimise its usage [Bazzan et al., 2008]. Take for instance people going to work by underground when a sudden fault at a station makes the light train no longer in operation. Then those people have to use other means of transport to reach their destinations. They could get a taxi or a bus, walk, or get to the next underground station. Another example is when people drive their cars and there is an accident blocking the road ahead. People then have to take alternative routes to reach their destinations. Examples like these are common in transport and demand for supporting system able to cope with unexpected changes.

Existing approaches focus on diverse and specific aspects of transport, for instance on traffic coordination at traffic management centres, cf. [Hernández et al., 2002], maximisation of (urban) network throughput by controlling the signalling at adjacent road intersections, cf. [Basile et al., 2004], or improvement of the traffic flow by optimising both the timing of traffic lights and routes selected by drivers, cf. [Bazzan et al., 2008]. Even when some approaches have shown improvements, there is still work to do especially to support the ability of dealing with unexpected changes, cf. uncertainty [Chen and Cheng, 2010].

2.1.5 City modelling

The modelling of cities in general aims at gaining long-term understanding of specific phenomena occurring in urban scenarios [Galán et al., 2009]. Some examples involve studying the dynamism of populations in residential areas cf. [Benenson, 2004], water management cf. [Galán et al., 2009], or more holistic approaches such as analysing the city metabolism involving transport, industrial processes, services, commercial activities, and other socio-economic factors cf. [Robinson et al., 2007].

Typically city modelling involves simulating scenarios in which the interactions
between the elements being modelled are taken into high consideration, e.g. people and social groups, and the way they or their decisions change during the simulation, cf. [Benenson, 1999]. However the interactions are rather simplistic not capturing the complexity the domain demands.

Adaptation then comes from the need to capture complex, macro level structures [Benenson, 1999] emerging from dynamic interactions and local conditions [Benenson, 2004] which cannot easily be captured by equations. Cities are not static and there is still a need to capture the self-organising evolutionary development of cities which calls for novel approaches to adaptation in complex systems [Robinson et al., 2007] where unexpected changes occur such as in actual cities.

2.2 Adaptation in ecosystem domains

In the domains described above there are common characteristics worth noticing: 

a) there is a number of diverse players to be represented or modelled. These refer to the companies in a business ecosystem, the services or components in a digital service ecosystem, the products and machines in a manufacturing domain, people and vehicles in the transport domain, and social groups in city modelling; 

b) there are interactions between the players. Such interactions tend to be frequent thus increasing the dynamism of the system; 

c) an object or information is exchanged or shared during the interactions. Then the result of the interactions dictate future interactions either between the same players or with others; finally 

d) unexpected changes occur which the overall system needs to cope with. That is, an individual player may not necessarily be the only one affected by the changes thus not the only one with the need to adapt. But it is the collection of players which need to adapt to unexpected changes.

In particular, unexpected changes and adaptation to them are the main focus of this thesis. Then the definitions of these are provided below.

A change is an alteration of an environment (cf. [Hol et al., 2007]) in such a way that it creates a cascade of changes which alter the environment further (inspired by [Peltoniemi and Vuori, 2005]).

An unexpected change is a change introducing an unknown environmental status (cf. [Hol et al., 2007]) such that the underlying system elements cannot apprehend the overall situation at once (cf. [Holland, 1995]), i.e. a change is unexpected because a) it cannot be predicted, b) its inducer is unclear, and c) the holistic effect in the system is unknown.

From the biological point of view, adaptation "is the process whereby an organism fits itself to its environment. Roughly, experience guides changes in the
organism’s structure so that as time passes the organism makes better use of its environment for its own ends” [Holland, 1995]. Inspired by it, the definition of adaptation used in this thesis reads as follows:

**Adaptation** is the process by which a system adjusts itself and reaches an equilibrium after having experienced a change, cf. homoeostasis [Levin, 1998].

The common characteristics of the domains presented in this chapter along with the definitions above maintain an analogy to natural ecosystems. Because of this the rest of this thesis refers to these domains as *ecosystem domains.*
Chapter 3

Adaptation Approaches in Multi-Agent Systems

This chapter explores the different approaches of adaptation in MASs. A classification framework of approaches is derived where the ecosystem adaptation class is highlighted as the most promising for supporting ecosystem domains. Natural ecosystems then are chosen as the means for reviewing examples of complex adaptive system modelling. Some of their characteristics are extracted for further usage in developing the approach to unexpected changes.

This chapter is structured as follows: Section 3.1 and Section 3.2 define relevant concepts as well as dimensions for analysing adaptation approaches, which are presented throughout Section 3.3. Then from Section 3.4 to 3.6, principles, properties and characteristic are abstracted from MAS development and natural ecosystems for supporting adaptation as an ecosystem. Section 3.7 then presents other approaches to classify adaptation before finishing the chapter with a discussion in Section 3.8.

3.1 Agent and multi-agent system

The definitions of agent and MAS vary in the research community. However, the one selected for this thesis is that from [Wooldridge, 2002]:

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

According to [Wooldridge and Jennings, 1995], an agent may also exhibit other capabilities such as social ability which allows an agent to interact with other agents by using an appropriate communication language; reactivity permits an agent to
perceive its surrounding environment and respond to such perceptions accordingly; and pro-activeness allows an agent to take the initiative to pursue its goal.

Now that the term agent has been defined, the definition of an MAS is given which is slightly adapted from [Panait and Luke, 2005]:

**An MAS** is a system in which there is more than one agent interacting with one another, and where environmental constraints may not at any given time allow the agents to know everything about the world.

Additionally, there have been attempts to define an adaptive agent. For instance, [Maes, 1994] defines it as an agent becoming better at achieving its objectives. Another example is that from [Sichman and Demazeau, 1995] who defines an adaptive agent as an agent taking different decisions whilst achieving its goals. Moreover, [Guessoum, 2004] defines it as a reactive agent which uses a symbolic representation of the world. However, the definition used in this thesis is that of Pitrat’s as cited by [Guessoum, 2004]:

**An adaptive agent** is an agent having knowledge of its own structure and evolutionary capacities, thus being able to modify its own structure in order to change its behaviour.

Combining the definitions of an MAS and of an adaptive agent, an adaptive MAS is defined as follows:

**An adaptive MAS** is an MAS situated in an environment and is capable to self-modify its structure and internal organisation according to its agents varying their interactions as a reaction to environmental changes\(^1\).

Notice that an adaptive MAS is not necessarily composed by adaptive agents. However, approaches in ecosystem domains already use MASs for their support due to the intrinsic characteristics of the domains themselves. Nevertheless using MASs only contribute to part of the solutions. There is still a need to tackle adaptation to unexpected changes. This motivates a further analysis of MAS approaches dealing with adaptation in diverse domains and applications.

### 3.2 Dimensions for analysing adaptation approaches

According to the definition of an adaptive MAS given above, there are three elements involved to have an adaptive MAS namely a) an MAS, b) an environment, and c) the relation between them due to the MAS “situatedness”.

\(^1\)This definition is revisited in Section 5.8.
These elements are used to determine the dimensions for analysing adaptation approaches in the area of MASs. Because the analysis considers the nature of adaptation itself, other MAS properties such as application domains, development techniques, or robustness are not taken into account as the analysis dimensions. In more details, the analysis dimensions are obtained as described below.

**MAS:** One of the main “guiding” [Maes, 1994] and engineering [Parunak, 1997] principles to enable adaptation is the interactions among agents. These interactions can be considered either static or dynamic. A static MAS is an MAS whose agent interactions are predefined. For instance, an agent is pre-programmed or designed to interact specifically with another agent. In contrast, a dynamic MAS is an MAS whose agent interactions are not restricted, allowing the agents to interact freely with whoever they encounter if they want to.

**Environment:** The main reason for adaptation is the changes occurring in the environment. The nature of these changes depends mainly on the way time and states are handled by the MAS. Therefore, one of the dimensions for environment characterisation established by [Russell and Norvig, 2003] says that the environment can be either discrete or continuous. A discrete environment is that whose events are discrete, i.e., changes occur in a drastic manner. Consequently, states are assigned to the environment. Typically, event types are known in advance, but the occurrence time might remain unknown. A continuous environment is that where events happen gradually, i.e. changes are not drastic. As a result, agents commonly model the environment as a function which they attempt to control, anticipate, or optimise. Furthermore, event types as well as occurrence time might remain unknown.

**MAS-environment relation:** According to [Parunak et al., 2000], an MAS is not independent from its environment. This is because the latter “provides the appropriate conditions that enable interaction among agents” [Odell et al., 2003]. However, this dependence may not be tight, therefore it is obvious to consider the degree at which environmental changes affect an MAS and its (re-)actions affect the environment back. Thus, the strength of this relation is used for characterising it as either strong or weak. A strong relation is that where one change in the environment affect the MAS almost immediately, and vice versa. A weak relation is that where a single change in the environment does not affect the MAS directly, but the effect is delayed till the accumulation of many changes cross a threshold; and vice versa.

These dimensions are used to analyse the different adaptation approaches
taken by other researchers in the field of adaptation in MASs. Examples of adaptation approaches and implementations in MASs have appeared in fragmented research streams aiming to enable adaptation for different purposes and domains. Streams go from deterministic systems to flexible systems, from evolutionary computation to complex systems, from games to organisational support. Therefore, the examples analysed were selected due to their claim to be adaptive, and not by their results quality nor the application domain.

In the following section, a classification of the approaches is presented derived from the examples found in the relevant literature. The examples themselves are described in Appendix A.

### 3.3 Classification of adaptation approaches

Table 3.1 shows the adaptation approaches in MASs sorted by the dimensions and forming five clusters. The first one is characterised by a discrete environment, a static MAS and a strong MAS-environment relation. Approaches in this cluster rely mainly on the discreteness and sometimes stillness of the environment.

The second cluster is similar to the first one, except the former considers a continuous environment. That is, it relies on the possibility of modelling the environment as a function whose values can be estimated by agents.

From the third cluster on, there is no much emphasis on the environment, thus approaches vary between a discrete environment and a continuous one. In particular, the third cluster differs mainly from the previous two in that it has a weak MAS-environment relation. Moreover, approaches in this cluster consider extensively evolutionary computation to achieve adaptation, in particular genetic algorithms.

The last two clusters are represented by only two examples each. They are characterised by having a dynamic MAS allowing flexible patterns of interactions between agents. In particular, the fourth cluster have a weak MAS-environment relation. Approaches in this category emphasise the emergence of complex behaviours by having dynamic agent interactions, but paying little attention to the environment.

Finally, the fifth cluster is characterised by a dynamic MAS as well, but with a strong MAS-environment relation. The emphasis here is that the environment is considered as a supplier of resources needed by the agents, and not only a place where agents interact.

Clusters shown in Table 3.1 characterise the approaches into five adaptation classes in MASs: a) automaton, b) control system, c) semi-isolated evolution, d) complex interactions and e) ecosystem.
### 3.3 Classification of adaptation approaches

Table 3.1: Analysis of adaptation approaches in MASs.

<table>
<thead>
<tr>
<th>Adaptation Approaches</th>
<th>Environment (Discrete, Continuous)</th>
<th>MAS Dynamics</th>
<th>Relation Weak</th>
<th>Emergence Strong</th>
<th>Cluster Number</th>
<th>Cluster Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sechen &amp; DeBono, 1998</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td>1</td>
<td>Automaton</td>
</tr>
<tr>
<td>Capos et al., 2003</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fovates et al., 2003</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>van Saunter et al., 2003</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lehman, 2004</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mertens et al., 2004</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amari &amp; Inoue, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Olea et al., 2009</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fovates et al., 2009</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glance et al., 1998</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carly, 1998</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fatima &amp; Umea, 1998</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hofmeyr &amp; Forrest, 2000</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schlenburg &amp; Ross, 2001</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fatima &amp; Woodrage, 2001</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fazzari &amp; Bixens, 2002</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noponreechokst et al., 2003</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guassoun et al., 2004a</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guassoun et al., 2004b</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Martin et al., 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fede &amp; Guassoun, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quah et al., 2006</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nguyen &amp; Wistuba, 2006</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grafiti et al., 1992</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vache &amp; al., 1998</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basset &amp; De Jong, 2000</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Montresor et al., 2002</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Landau &amp; Picardi, 2003</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marrow et al., 2003</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valdecastelle et al., 2004</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Napora &amp; Olivera, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O'Riordan, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pane &amp; Tan, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voss, 2005</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combos et al., 2004</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Holland, 1995</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marwede et al., 2001</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2 shows a different clarifying perspective on these clusters, as a classification framework. Such a classification helps to highlight gaps in the area of adaptation in MASs and to identify where research effort should be addressed to tackle adaptation to unexpected changes as in business ecosystems.

### 3.3.1 Adaptation as an automaton

The adaptation as an automaton approach comprises systems which use a static MAS with a strong relation with a discrete environment. In general, agents in this class have a fixed set of interactions, actions, and components, and a fixed set of potential states is assigned to the environment. Agent interactions and environ-
Adaptation Approaches in Multi-Agent Systems

Table 3.2: Classification framework of adaptation in MASs.

<table>
<thead>
<tr>
<th>MAS-Env Relation</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Discrete</td>
</tr>
<tr>
<td>Static</td>
<td>Strong</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Weak</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
</tr>
<tr>
<td>Dynamic</td>
<td>Strong</td>
</tr>
</tbody>
</table>

mental events bring the environment to determined states from another specific one, cf. an automaton.

Since agent interactions and environmental states are pre-determined, an MAS adapts its behaviour by choosing actions and interacting in a specific way according to either the current or desired environmental state. Figure 3.1 depicts how adaptation as an automaton works.

A common assumption in approaches within this class is that the environment has a determined number of states which an MAS is able to control or it knows the right response for them. Adaptation as an automaton assumes that only the expected states and event types will be encountered in the environment, cf. a closed system. Such an assumption is not suitable for a dynamic environment such as an ecosystem domain where unexpected events occur.

3.3.2 Adaptation as a control system

The adaptation as a control system approach comprises systems which use a static MAS whose relation with its continuous environment is strong. Generally, agents
3.3 Classification of adaptation approaches

3.3.3 Adaptation as semi-isolated evolution

The adaptation as a *semi-isolated evolution* approach comprises systems using a static MAS whose relation with its environment is weak. Approaches in this class

...
show no significant difference between a discrete environment and a continuous one. This class in general sees agents modifying their internal structure by means of evolutionary computation, in particular genetic algorithms. Agent interactions are used for evaluating MAS fitness to the current environmental conditions.

Consequently, two different stages operate: one for adapting the MAS using evolutionary computation, and the other for assessing MAS adaptations, i.e. evolutions. Figure 3.3 presents how adaptation as a semi-isolated evolution functions.

A feature common to approaches fitting in this category is that they all regard the environment as static, at least during the moment when the MAS is adapting. That is why this class is considered semi-isolated. In the end, adaptation as semi-isolated evolution assumes that the environment will not change whilst adapting. This assumption is questionable for supporting ecosystem domains because the environment is always changing, thus should not be considered stationary.

### 3.3.4 Adaptation as complex interactions

The adaptation as a *complex interactions* approach comprises systems using a dynamic MAS which has a weak relation with its environment. Approaches in this category show no substantial difference between a discrete environment and a continuous one because the focus is on the MAS itself. That is, the MAS adapts to changes created mainly by the agents themselves which do not seem to affect the environment to the point where any resource from it is considered unlimited.

This class targets adaptation to different situations and conditions created by the agents in an MAS through dynamic interactions, thus triggering the emergence of complex behaviours which do not seem to affect the environment. Figure 3.4 depicts how adaptation as complex interactions works.
3.3 Classification of adaptation approaches

The emergence of complex behaviours can be seen as an approach providing enough flexibility for achieving adaptation. Nevertheless there is little consideration to the impact onto the environment. In conclusion, adaptation as complex interactions makes two assumptions: a) the environment will provide unlimited resources to the MAS, and b) the environment is not needed for adaptation. These assumptions are far from realistic in ecosystem domains because the environment they consider indeed has limited resources.

3.3.5 Adaptation as an ecosystem

The adaptation as an ecosystem approach comprises systems which use a dynamic MAS with a strong relation to its environment. Approaches in this class show no substantial difference between a discrete environment and a continuous one. In general, adaptation as an ecosystem makes use of dynamic agent interactions to allow complex behaviours to emerge, in a manner similar to the adaptation as a complex interactions approach. The difference between the two approaches is in the way in which MAS adaptations impact the environment and vice versa. In the adaptation as an ecosystem approach they are continually affected by each other, similarly to a natural ecosystem, resulting in a strong MAS-environment relation. Figure 3.5 depicts this approach.

This adaptation class is suitable to support ecosystem domains due to mainly
these factors, a) the continual effects the MAS and its environment have on each other permitting the former to adapt to different environmental changes, which makes it promising to support adaptation to unexpected changes; and b) the strong relation between the MAS and its environment makes the former to consider the resources and limitations the environment might impose. This motivates the development of an approach using MASs portraying adaptation as an ecosystem for supporting ecosystem domains. In the next section a set of adaptation principles for MAS development is reviewed for this purpose.

3.4 Adaptation principles for MAS development

This section presents a set of guiding and nature-inspired engineering principles found in the literature for building adaptive agents and MASs. The purpose of including them in this thesis is to understand how adaptation as an ecosystem could be developed. The ADELFE methodology [Bernon et al., 2003] was not considered here because it is based on a theory (AMAS) classified under the adaptation as an automaton class of the presented framework. The principles are summarised below.

Prin1. Agents should be integrated entities such that any function, perception, or specialised algorithm should be encapsulated inside a single running entity
3.4 Adaptation principles for MAS development

[Maes, 1994], “living” in an environment [Parunak, 1997]. This principle basically establishes that in terms of software, an agent should be an independent running programme, cf. execution thread.

**Prin2. Agents should be small in mass**, i.e. they should be able to compose bigger systems [Parunak, 1997]. In other words, agents should be the building blocks of higher level systems such as an MAS.

**Prin3. Agents should be situated in an environment.** That is, an agent should be contained within a virtual space where it can perform some actions [Maes, 1994]. Environment containment or “situatedness” extends beyond the typical execution environment of a programme or an agent.

**Prin4. An agent belongs to a community** in which other agents deal with similar problems [Maes, 1994]. That is, agents are considered social entities solving related problems in such a way that they might cooperate or compete as appropriate. This implies interactions between agents as well as with the environment [Maes, 1994].

**Prin5. Agents should be small in time** in such a way that their actions should be influenced by recent information, i.e. only the latest information is kept in memory as time elapses [Parunak, 1997]. As a result, agents keep forgetting information to avoid having a global knowledge of the system.

**Prin6. Agents should share their knowledge**, i.e interacting agents should exchange the information they know [Parunak, 1997]. Typically, the amount of information to share depends on the situation and possibly to the strategy of the agent. Yet, some of it should be shared.

**Prin7. Diversity obeys to random and repulsive forces** [Parunak, 1997]. Typically, diversity refers to the difference of kinds, but it also refers to knowledge, solutions, or even preferences. Thus, randomness helps an agent to opportunistically try new solutions, whereas repulsion keeps the agent from those situations known as not favourable.

**Prin8. Agents need to perceive the environment**, therefore they need sensors as well as actuators to affect the environment back [Maes, 1994]. This principle emphasises the importance of a strong relation with the environment.

**Prin9. An agent needs to orient itself** in the environment by means of dissipative flow mechanisms [Parunak, 1997], i.e. an agent needs to perceive a notional direction in the environment, so that it knows where it is and decides where it wants to go.
Prin10. An agent community should be decentralised such that no way of centralised control is used by the agents [Parunak, 1997], i.e. there must not be an (omniscient) agent guiding or supervising the rest of the community.

Prin11. Agents should execute concurrently and independently in no specific order [Parunak, 1997], cf. software threads.

The principles mentioned above focus on what to consider when designing and developing an adaptive agent or MAS. Yet they do not provide an insight into the requirements or properties of adaptation itself. Therefore they are sought in another area.

### 3.5 Adaptation properties of complex adaptive systems

Indeed in nature there are systems composed by individuals exposing complex behaviours beyond the capabilities of the composing individuals, for example the human immune system which adapts itself to unexpected intruders in order to defend the human body; viruses which adapt their structure (mutate) in order to survive and spread the population (the disease); and ecosystems such as forests and oceans which adapt to environment resource changes and species migrations. Such systems are in continuous adaptations in their dynamic, ever changing environment, and correspond to complex adaptive systems (CASs). These are known to possess properties allowing them to adapt to unexpected environmental variations [Holland, 1995].

One of the hypotheses (H1) underpinning this thesis is that by capturing the properties underlying CASs and transferring them into MASs, adaptation to unexpected changes could be achieved in dynamic and complex environments. There are four essential properties widely accepted for making a CAS adaptable to different situations as proposed by [Holland, 1995]. These are used as the basis for developing the approach of this thesis, and are described below.

**Prop1. Aggregation** refers to the way individuals are grouped into populations which in turn are further grouped into species and so on to higher levels.

**Prop2. Diversity** refers to the number of different types of individuals or species existing in a collection of individuals.

**Prop3. Flow** refers to the exchange of energy, nutrients and/or information among individuals of the same population and among species.
Prop4. Non-linearity refers to the global behaviour emerging from the interactions of the underlying elements. Such a behaviour goes beyond the individuals' capabilities and understanding.

3.6 Characteristics of natural ecosystems as CAS

The study of CAS involves the analysis of how complicated structures and interaction patterns emerge from disorder via simple but powerful change guiding rules [Levin, 1998]. CASs have been studied by different research groups in different fields ranging from physics and biology to mathematics and economics. In this thesis CAS is studied from the point of view of biology, i.e. CAS as a natural ecosystem due to the analogy ecosystem domains. Thus, natural ecosystem modelling approaches are reviewed to obtain further insight into how an ecosystem works. The approaches described in the following subsections are the selected ones based on their support to the adaptation properties mentioned above in Section 3.5. Then the construction of the ecosystem approach to unexpected changes is presented in Chapter 4.

3.6.1 Biosphere as CAS

[Levin, 1998] proposes to study ecosystems and the biosphere as CASs. The author argues how these two systems function according to the adaptation properties described above thus they can be regarded as CASs. Moreover, the author presents challenges for studying CASs and in particular two characteristics typical of ecosystems and the biosphere which give further support to the adaptation properties. These are described below.

C1. Local natural selection. A natural selection process consists mainly of selecting a mate partner with the best desired features, which makes the undesired ones to fade out after some generations. By being a local process, natural selection delays the fading out of undesired features, which might not be that undesired in the future if preferences change, thus maintaining the adaptation property Prop2, diversity.

C2. History dependence refers to a multiplier effect commonly known as the “butterfly effect”, and states that a series of consecutive events could not have happened if the first one had not occurred in the first place. Typically individuals' interactions contribute to produce such an effect. Such a characteristic supports the adaptation property Prop4, non-linearity.
3.6.2 Hierarchical structures

[Kolaša, 2005] presents an approach of complex systems where hierarchical structures emerge from subcomponents' integration. According to Simon, as cited by [Kolaša, 2005], there would be no emergent systems as a whole nor integration if systems were completely decomposable into its primary elements. The author proposes a series of equations to represent the degree of integration of several hierarchical components before disassembling when facing a strong enough perturbation. The result can be summarised in the following characteristic of ecosystems.

C3. An integrated system behaves as a single entity. This characteristic refers to the system-subcomponents relation, such that as integration increases, the autonomy of the system’s subcomponents diminishes and the system complexity progressively disappears. This clearly reflects the adaptation property Prop1, aggregation.

3.6.3 Food-web dynamics

[Rossberg et al., 2005] presents a model for species evolution which considers a number of overlapping food chains called food-webs. The model represents food-web dynamics as species being both resources and consumers at the same time whilst participating in links within food chains. Thus when a species evolves due to a specialised evolution or adaptation, it affects all the connections with other species and new connections are created. The characteristics abstracted from the model are the following.

C4. Species form food chains in such a way that a species may function as a resource for another species and as a consumer for yet another one. Thus food chains are created by having a linear succession of species exchanging resources or interacting in some way. This characteristic supports the adaptation property Prop3, flow.

C5. An ecosystem resembles a network where energy flows vary over time, so that there can be connections, disconnections and re-connections. A connection in an ecosystem refers to choosing the resource to consume and thus maintain a flow of energy or nutrients from one species to the next, supporting in this way the adaptation property Prop4, non-linearity.

3.6.4 Mechanics of complex aggregates

[Maurer, 2005] explains the mechanics (cf. dynamics) of complex aggregates seen in ecosystems. The author argues that even though nature is full of uncertainty,
unpredictability, and contingency, there is certain order within, which is probabilistic due to physical and biological laws. Therefore, there is no absolute certainty that a precise behaviour will always be obtained from a complex system.

The author explains that aggregates are usually regarded as *macroscopic* entities or system level, whilst their components are considered as *microscopic* entities or individual level. Both entities have a state determined by a probabilistic distribution describing the range and frequency of states each entity may have. States are specified by different variables which all together represent the aggregate's behaviour. So, each macroscopic variable is associated with the probabilistic behaviour of the microscopic variables.

Moreover, the author presents a series of equations to model how the macroscopic entities are related to the microscopic entities. An interesting element of the model is that the species diversity is considered to exist over a geographical region. That is, a species population have the tendency to move to the regions with enough resources for its maintenance, which might be restricted by the existence of another species population in the same region. Therefore, diversity and even specialised evolution occur over a geographical space. For instance, if a sub population of a species is created and moves to a different region, then it will evolve differently from the original one because the sub population will follow a different evolution path due to different environmental conditions.

Thus, three characteristics are drawn from this model as described below.

**C6. Ecosystems have two levels of aggregation:** a macroscopic and a microscopic level, also called the system or individual levels respectively. This clearly support the adaptation property Prop1, aggregation.

**C7. Diversity occurs over a geographical space.** The latter maintains it or develops it further which in turn create a specialised evolution. It is evident that this characteristic supports the adaptation property Prop2, diversity.

**C8. Populations cluster near resources** needed for their own maintenance, i.e. a population moves as a whole (cf. migrates) to another area when there are not enough resources to consume at their current place. This characteristic support the adaptation property Prop1, aggregation.

### 3.6.5 Interactions and complexity

[Green and Sadedin, 2005] presents an analysis on how interactions function as a key process for emergence within ecosystems. The authors present background on complexity theory and make the parallel to ecosystems and CASs. In particular, the authors argue that the main component of complexity is the non-linearity, thus
for modelling it is necessary to view a system as a network whose connections (cf. interactions) depend on the medium, i.e. how the network is connected. Moreover, feedback loops are considered to increase the non-linearity because they can either stabilise or destabilise the network connectivity because feedback affects the interaction itself which in turn creates further feedback and so on, increasing the non-linearity. In summary, the main points to consider from the author's work are the following.

C9. Network connectivity depends on the interaction medium. The way a network is connected depends on how the interactions are made. Since networks are regarded as a component for non-linearity, this characteristic support the adaptation property Prop4, non-linearity.

C10. Feedback loops increase non-linearity. As mentioned above, it is because they can either stabilise or destabilise the network connectivity. Clearly this characteristic support the adaptation property Prop4, non-linearity.

3.6.6 Circular flow of materials

[Otsuka, 2004] tries to answer the question of how different organisms with different strategies form an ecological system. This problem is theoretically presented by a model in which there is a flow of materials or substances among producers, consumers and decomposers within an ecosystem.

Initially, the simplest system consists of autotrophic organisms playing a central role in the development of ecosystems, i.e. plants. They absorb nutrients and synthesize them into sugars by using photon energy. Occasionally they have phases in which the population increases due to reproduction and others in which the population decreases due to death. These phases maintain a stable community. After dying, their bodies disintegrate and the molecules are absorbed into the ground and become available to other living autotrophs. It is a cycle of materials or resources, cf. a food chain.

After considering this simple system, the author adds a set of decomposers to the cycle. The decomposers consume the dead bodies of the autotrophs. Eventually, the decomposer dead bodies are also disintegrated and absorbed into the ground to become available for plants. After this, more complexity is added to the cycle by considering animals who eat plants and then predators of different ranks, i.e. predators eating other predators eating other animals and so on. The author demonstrates how the material cycling is intensified by an increasing number of species composing the system making it dynamic even from the simplest case. This characteristic is abstracted as described below.
Material cycling keeps the dynamism in an ecosystem i.e. the exchange of materials or nutrients between the members of a food chain maintains the dynamism as long as the species exist. That is, such an exchange occurs over and over between species. Since this characteristic refers mainly to the flow of nutrients, it is allocated to support the adaptation property Prop3, flow.

Ecodynamics

[Ulanowicz, 2004] introduces the concept of autocatalysis referring to a positive feedback occurring across several links forming a loop upon which the feedback remains positive. Autocatalysis can be viewed as a circular set of interactions \( A \Rightarrow B \Rightarrow C \Rightarrow A \) with a positive outcome making the network activity to increase.

Additionally, more than two autocatalytic loops may share some of the interacting element. As a consequence, those interactions not providing enough benefit to the loop are discarded, encouraging this way a competition in the loops. In summary, the elements drawn from this work are summarised as follows.

Positive feedback encourages competition by increasing the activity in a network, specially when loops exist. Those interactions not providing enough benefit to the loop are discarded, thus supporting the adaptation property Prop4, non-linearity.

A thermodynamic principle

[Jørgensen and Fath, 2004] applies thermodynamic principles to explain observations on ecosystems as a way to make a link between observations and ecological theory. Some of the principles are chemistry related such as one saying that carbon-based life occurs only between 250 K and 350 K (kelvin degrees). And others have been already selected as ecosystem characteristic in previous subsections. Yet there is one that is worth selecting it as an ecosystem characteristic:

Ecosystem processes are irreversible in such a way that due to the multiplier effect one event has over the next one, it is difficult to assign states to the system, and thus nearly impossible to go back to a previous state. Therefore, it supports the adaptation property Prop4, non-linearity.

Other classification frameworks

Other classification of approaches have been derived considering other aspects related to adaptation. The following subsections analyse a couple of them.
3.7.1 Adaptation approaches by intelligence dimensions

In particular, [Hayes-Roth, 1995] classifies adaptation based on intelligent strategy dimensions of an adaptive intelligent system (AIS) namely perception strategy, control mode, reasoning tasks, reasoning methods, and meta-control strategy. These dimensions seem equivalent to those presented in Section 3.3. However, a deeper analysis of the assumptions shows how such dimensions are covered by the framework presented in this chapter.

1. Adaptation based on perception strategies refers to adaptation approaches in which strategies are chosen from a fixed set according to needs and limitations. This dimension is covered by the automaton class.

2. Adaptation of a control mode focus on the degree at which agents can interleave their actions according to constraints and environment uncertainty. This dimension is comprised by the control system class.

3. Adaptation of reasoning tasks focus on the allocation of reasoning tasks to dynamic objectives. However, this approach considers only the interrupting and resuming of the reasoning process according to external events. Therefore, this dimensions is covered by the automaton class as well.

4. Adaptation of reasoning methods refers to the well-known trade-off problem, i.e. balancing between exploring new models of the world and exploiting the models already constructed. Balancing clearly implies controlling variables, thus this is covered by the control system adaptation class.

5. Adaptation of meta-control strategy considers allocating computing resources among different configurations of competing and complementing tasks. In the end, this dimension tries to maximise the behaviour utilised. Therefore it can be easily mapped to the control system adaptation class.

A drawback of this classification approach is that adaptation is considered at the individual level, i.e. the notion of MAS is not considered at all. An MAS can render easily the adaptation issues considered by [Hayes-Roth, 1995], for instance resource allocation and diverse task execution due to the MAS distributed nature and diverse autonomous problem-solving capabilities.

3.7.2 Adaptation approaches by their research scope

Another classification framework is proposed by [van Splunter et al., 2003] where the research scope of adaptation approaches is suggested to be considered as the dimensions for a classification. In spite of not providing examples, the framework
envisages three classes namely agent's knowledge, agent's interface, and agent's functionality. Likewise, these classes can be mapped to those presented earlier in this chapter.

1. Research on adaptation of agent's knowledge covers user modelling, MAS coordination and learning. This category is very general to be mapped directly to a specific class of the classification framework. Nevertheless, some of the examples within both the automaton and the control system classes deal with the same issues. Therefore, research on agent's knowledge adaptation can be mapped to either the automaton or control system class depending on their details regarding the dimensions explained in Section 3.2.

2. Research on adaptation of agent's interface deals with the agent language when interacting with its execution platform. Typical examples of this view are those dealing with mobility. Yet again, this category is very general to be mapped to a particular class of the framework. Thus, when looking at the examples covering mobility it can be appreciated that they are assigned to different classes because for most of them mobility is not an issue for adaptation, but an element of their domain and application. Nevertheless, one approach in particular (see [Amara-Hachmi and Fallah-Seghrouchni, 2005]) deals with adaptation to different platforms. Since it was classified to the automaton class, adaptation of agent's interface is assigned to it as well.

3. Research on adaptation of agent's functionality includes agent adaptation by means of software engineering, i.e. reusing software components. These approaches depend on the availability of software components for adaptation. Thus, adaptation of agent's functionality can be mapped directly to the automaton class because adaptation depends on the foreseen situations the reusable components are developed for.

The adaptation categories [van Splunter et al., 2003] presents are quite general and does not consider the environment at all. In addition, it does not present any examples within those categories. Yet those views are covered by the framework presented earlier.

3.8 Discussion

Other classification approaches fail to capture the elements for an adaptive MAS, i.e. the MAS itself, its environment and the relation between the two, whereas the classification framework derived in Section 3.3 uses those elements to build the
framework upon. The analysis of approaches presented in Appendix A highlights these elements.

The benefit of the classification framework is not the capability to classify every approach on adaptation, but rather to identify advantages and disadvantages of types of approaches. Therefore, when considering the nature of a target domain it becomes evident which of the classes fits with the features of such a domain. In the particular case of ecosystem domains, the ecosystem class is the most promising for dealing with adaptation to unexpected changes.

The thirteen characteristics drawn from the ecosystem modelling approaches support the adaptation properties of CASs. These characteristics provide an insight into how a natural ecosystem functions. Yet for using them in MASs it is necessary to combine them with the guiding and engineering principles for developing adaptive agents and MASs. Indeed such is the approach taken in this thesis and it is developed in Chapter 4.

Summary

This chapter presented the definitions of unexpected change, adaptation, agent, and MAS. Using the main elements of an MAS, a classification framework of adaptation approaches in MASs was introduced. It shows five adaptation classes: automaton, control system, semi-isolated evolution, complex interactions, and ecosystem. From them, only adaptation as an ecosystem seems promising to support the systems in an ecosystem domain. Therefore, a set of guiding and engineering principles are considered to help in developing adaptive agents and MASs, yet they do not provide enough insight into the properties of adaptation itself.

This encouraged the search for inspiration from the field of CASs. Adaptation properties were introduced as an essential part of a CAS of which a natural ecosystem is considered a relevant example to draw inspiration from. Then ecosystem modelling approaches were reviewed and underlying characteristics supporting the adaptation properties were abstracted. In Chapter 4, all these principles, characteristics and properties will be combined to derive the approach used in this thesis.
Chapter 4

A Formal Model of a Dynamic Agent-based Ecosystem: DAEM

This chapter introduces a formal model of a dynamic agent-based ecosystem called DAEM. It is built on a foundation fusing principles for building adaptive agents and MASs (see Section 3.4), and ecosystem characteristics (see Section 3.6). The fusion is organised through the adaptation properties of CASs, which are transferred onto an MAS. This is used to derive DAEM, one of the contributions of this thesis.

This chapter is structured as follows: Section 4.1 builds the foundation of the approach. Section 4.2 presents an overview of DAEM before introducing the incremental exposition of DAEM in Section 4.3. Finally a discussion is presented in Section 4.4 before closing the chapter.

4.1 The role of CAS adaptation properties

Each of the following subsections presents a plan for transferring one CAS adaptation property onto an MAS. This is to be done by combining principles of adaptation and characteristics of natural ecosystem. Each step results in a set of DAEM elements.

4.1.1 Property Prop1. Aggregation

The adaptation property Prop1 aggregation refers to the capability of a set of individuals to form organised groups [Holland, 1995]. Such an organisation could be based on similar or complementary characteristics of the individuals. Typically, individuals are grouped into populations which are grouped into species and so on to higher levels or organisation.
Section 3.4 presents a set of principles for adaptive agents and MASs development. Those principles supporting the adaptation property of aggregation (Prop1) are listed below.

- Prin1. Agents should be integrated entities;
- Prin2. Agents should be small in mass;
- Prin3. Agents should be situated in an environment; and
- Prin4. An agent belongs to a community.

Throughout Section 3.6 characteristics of natural ecosystems were extracted from approaches of ecosystem modelling. Those characteristics supporting the adaptation property of aggregation (Prop1) are listed as follows.

- C3. An integrated system behaves as a single entity;
- C6. Ecosystems have two levels of aggregation; and
- C8. Populations cluster near resources.

By combining principles and characteristics, DAEM elements are obtained to support this adaptation property.

**E1. Integrated agents represent different level of aggregation.** Agents portray either individuals or a population of them. When representing a population an agent is considered to behave as a single integrated individual at the micro level. Yet because they are agents they are still small components able to be part of communities at the macro level.

**E2. Agents acquire resources** produced by other agents. Thus, they have a tendency to stay near their required resource.

**E3. Agents are situated in an environment,** i.e. agents share a virtual space where they wander around in their search for resources.

### 4.1.2 Property Prop2. Diversity

This adaptation property refers to the number of different types of individuals existing in a population [Holland, 1995]. This introduces the notion of categories or classes of individuals grouped according to similar characteristics.

See below the list of principles for adaptive agents and MASs development that support this adaptation property.

- Prin5. Agents should be small in time;
4.1 The role of CAS adaptation properties

- Prin6. Agents should share their knowledge; and
- Prin7. Diversity obeys to random and repulsive forces.

The natural ecosystem characteristics supporting this adaptation are listed below.

- C1. Local natural selection; and
- C7. Diversity occurs over a geographical space.

Combining principles and characteristics, a set of DAEM elements supporting this diversity are derived.

E4. **Agents are able to forget** in order to avoid global knowledge, thus keeping a population of agents with diverse knowledge.

E5. **Agents contrast one another by their preferences** on resources, i.e. agent diversity is based on preferences.

E6. **Agents move and interact locally in a space.** That is, agents move around in the environment, yet they decide who they want to interact with based on their preference.

E7. **Agents follow random and repulsive forces**, i.e. they move randomly in the environment and avoid areas with resource they do not prefer.

4.1.3 Property Prop3. Flow

The adaptation property of flow refers to the exchange of energy, nutrients and/or information among individual of the same population and among species [Holland, 1995]. Although interactions have been mentioned earlier to be part of DAEM, this property explicitly refers to an exchange of something possibly produced by the individuals themselves.

Below is the set of principles for adaptive agents and MASs development which support this adaptation property.

- Prin8. Agents need to perceive the environment; and
- Prin9. An agent needs to orient itself.

From natural ecosystems, these are the characteristics supporting the adaptation property of flow.

- C4. Species form food chains;
• C5. An ecosystem resembles a network; and
• C11. Material cycling keeps dynamism in an ecosystem.

As a result of combining principles and characteristics supporting flow, these are the elements obtained for DAEM.

**E8. Agents perceive the environment for orientation.** Agents use various sensors to perceiving their surrounding in the environment. Likewise, the environment provides with mechanisms for orientation such as pheromone-like marks left in the environment.

**E9. Agents form dynamic chains.** Like natural ecosystems, agents exchange resources thus making preference chains which form dynamically as preferences change or better resources appear. Nevertheless, chains are dynamic in such a way that they are created, changed, replaced, dissolved or re-created continually over time.

### 4.1.4 Property Prop4. Non-linearity

The adaptation property of non-linearity refers to the global behaviour that emerges from the interactions of the underlying individuals and goes beyond the individuals' capabilities [Holland, 1995]. This property is the one that provides complexity and dynamism to a system.

The principles for adaptive agents and MASs development supporting this non-linearity are the following.

• Prin10. An agent community should be decentralised; and
• Prin11. Agents should execute concurrently.

Likewise, those characteristics of natural ecosystem supporting this property are shown below.

• C2. History dependence;
• C9. Network connectivity depends on the interaction medium;
• C10. Feedback loops increase non-linearity;
• C12. Positive feedback encourages competition; and
• C13. Ecosystem processes are irreversible.

By combining the above principles and characteristics, a set of DAEM elements are derived as listed below.
4.2 An overview of DAEM

**E10. Agent interactions produce a feedback.** In more detail, an agent provides feedback to another one about what they just exchanged in such a way that it motivates the latter to compete with other agents the former agent interacts with.

**E11. The environment mediates agent interactions** in two ways, as the facilitator or medium for agent interactions to occur and as a restrictor of who an agent interacts with.

**E12. Agents are decentralised and concurrent.** This element refers to the software execution independence of each agent and the lack of a central control.

**E13. Events and actions are irreversible,** in such a way that once they occur it is not possible to undo their effects. As a result, all actions and events affect the agents and the environment triggering more actions and events and so on. Therefore, there is a dependence on past situations or history.

4.2 An overview of DAEM

The following subsections describe briefly the main internal mechanisms of DAEM in order to capture the “big picture” to facilitate its understanding when the formalisms are introduced. The main mechanisms described below are agent interactions and environment mediation. Moreover, details of how the elements supporting DAEM are given further on in Section 4.3.

4.2.1 Agent interactions

An agent is seen as the producer of a resource and the consumer of another. Thus, an agent offers its resources to potential consumers and receives offers from potential producers. As a consumer, resource offers are evaluated to determine who is more convenient to consume from, and therefore, to tag the producer as the preferred one. Resource evaluations are sent back to the potential producers, so that they know how good their resources are perceived. These evaluations are normalised (i.e. they have values within the range \([0, 1]\)) according to the best resource offer recently evaluated. Any consumer sending a normalised evaluation of 1 is telling the producer that its resource offer is considered the best one so far. Thus, the producer knows it is the preferred one for a specific resource for that specific consumer.

Likewise, producers determine who are their preferred consumers according to received evaluations. Because these evaluations are normalised, the closer the evaluation is to 1 the better that consumer is preferred over others. Therefore,
any agent knowing it is the preferred producer of its preferred consumer will try to increase the interaction frequency with its counterpart; and vice versa. Preferred producer and preferred consumer together constitute a link in the ecosystem.

The interaction frequency, i.e. re-sending resource offers, permits the producer to know as soon as possible whether a consumer has changed its preference since the last time they interacted. In such a case, the producer has to improve the resource it provides to become the preferred producer again. Normalised evaluations give an idea of how good a resource is perceived when compared to the best one a consumer has recently found. This information is useful to determine how much improvement the producer needs to best the preferred producer. Figure 4.1 depicts agent interactions.

### 4.2.2 Environment mediation and agent senses

The environment is a virtual surface where agents wander across and encounter others in order to interact. It mediates agent interactions and provides agents with “senses”, namely positioning, proximity, sight, and smell. The first one allows an agent to know where in the environment it is located and decide where it wants to go next. The next one permits an agent to identify who are in its vicinity and the resources being offered and consumed, so that only those agents close enough are able to interact with it. The sight sense lets an agent to “see” beyond its vicinity and to identify others within a certain distance. This is mainly for deciding whether to come closer or not for interacting depending on the resources the agent is interested in.

In addition, all agents leave a trail of evaporative marks on the environment.
4.2 An overview of DAEM

These marks contain information about the resources the agent offers and consumes, but not information about the agent itself. Thus, when agents move across the environment they can be tracked down by means of "smelling" the mark trail. The smell sense lets an agent to detect a notional gradient on the environment and guide its exploratory behaviour towards where it seems to be something of its interest. These senses permit agents to forage for resources consumed and produced by those who have given the best evaluation and have offered the best resource, accordingly. Figure 4.2 shows agent senses in the environment.

4.2.3 From agent interactions to complexity and adaptation

In general, when any agent $i$ is the preferred producer for agent $j$ and at the same time $j$ has given the highest evaluation $i$ has received, they will forage for each other. If no other agent with a better resource comes across, $i$ and $j$ will eventually increase the frequency of their interactions, creating and strengthening in this way a link in the ecosystem. Thus, a chain is a collection of links going from the producers offering the basic resources up to the final consumer acquiring a transformed resource.

Additionally, participating in more than one different chain by offering and/or consuming more than one resource, agents become nodes in an ecosystem network. Due to chain links formed by individual preferences on producers and consumers, and on initial environment positions, two non-exclusive situations might arise: a) competitors will be attracted closer to the links in their attempt to be part of the chain; and b) one or more separate, similar, competing chains might emerge along with its competitors swarming around the links. Consequently when an agent improves its resource, or a new agent enters the ecosystem, or an existing agent offers a new resource, it will trigger changes to local preferences at the individual level: better producers are preferred over others. Then at system level some links disappear and others are (re-)created while the ecosystem tries to find a new
configuration for its chains. That is, when unexpected changes occur at the individual level, the system level immediately re-organises itself by replacing missing links thus adapting to unexpected changes. Figure 4.3 depicts the ecosystem level behaviour.

4.2.4 Supporting ecosystem domains

The final objective of DAEM is to help supporting systems in ecosystem domains to deal with unexpected changes by adaptation. In a practical scenario, DAEM sits as a supporting layer to that of the supporting, existing system as in Figure 4.4.

Without going into the details, supporting systems of ecosystem domains are assumed to exist one way or another and possessing the means to interact with the physical world. Then DAEM reflects the different elements/components of the supporting system using agents and an environment. Activities in the supporting system are informed to DAEM as updates which may cause changes in the ecosystem making it to adapt by itself. DAEM suggests actions to the represented individuals of the ecosystem as a way to cope with changes, some of them unexpected.

DAEM is an ecosystem layer which abstracts interactions from an ecosystem domain supporting system into an ecosystem. And it is at this level where the adaptation as an ecosystem approach is played out. A more concrete example of an execution context using the reference model is presented in Chapter 6.

4.3 Formalising DAEM

The formal exposition of DAEM is presented in an incremental way through the following subsections. For the sake of compactness and simplicity, DAEM uses expressions such as $\prod_{j=1}^{M} a_j$ to signify an ordered list, and expressions like $\prod_{i=1}^{N} (a, b, \ldots, c)_i$ to signify a table containing the $i$-th tuple as the $i$-th row. Additionally, the use of superscripts distinguishes multiple use of terms especially to denote
4.3.1 Definition of a resource

The section starts by defining resources as the core underlying element of DAEM.

**Definition 1.** A resource is represented by the pair \((rs, val)\) where \(rs\) is a resource description and \(val\) is its specific value. The resource description specifies the basic details of the resource, i.e. a resource type. The value discriminates between two resources with the same description.

Notice that resources cannot be uniquely identified despite having a value \(val\). This is for allowing the existence of duplicates. The internal structure of both the resource description and its value is not discussed here. It is simply assumed that an agent is able to understand them and use them to support its own decisions. Moreover, it is considered possible that an agent can change the resource value.

Resources are necessary for agents to acquire, which conform with the element E2 of aggregation. The formal definition of agents is provided next.

4.3.2 Definition of an agent and its classification

DAEM has a micro level and a macro level of aggregation according to element E1. Depending on the ecosystem domain, the micro level is obtained when an agent represents an individual or a set of them behaving as an integrated system (cf. a population) from the supporting system (see Section 4.2.4). The macro level then is obtained by considering the collection of agents as the ecosystem which then can
be seen as an integrated (eco)system. This representation supports the element E1 of aggregation.

Formally, an agent in DAEM is defined as follows:

**Definition 2.** The tuple \((aid, R, A, M)\) is used to describe an agent \(i\) and related interactions as follows:

- **aid** is the identifier of an agent \(i\). For the sake of simplicity, the expression agent \(i\) is used to refer to an agent with identifier \(aid_i\).

- \(\{IRS, ORS\} \in R\) is the resource set, where
  - \(IRS = \prod_{r=0}^{i-1}(rs, val)^{ir}\) is a list containing all the resources agent \(i\) consumes, i.e. input resources.
  - \(ORS = \prod_{or=0}^{i-1}(rs, val)^{or}\) is a list containing all the resources agent \(i\) produces, i.e. output resources, where \((rs, val)^{ir} \neq (rs, val)^{or}\) for all resources in \(IRS\) and \(ORS\) respectively.

- \(\{sendof, sendev\} \in A\) is a set of actions available to the agent:
  - \(sendof (aid_i, aid_j, (rs, val)^{or})\) signifies that agent \(i\) offers the resource \((rs, val)^{or}\) to agent \(j\).
  - \(sendev (aid_i, aid_j, (rs, val)^{or}, ev)\) signifies that agent \(i\) informs to agent \(j\) that the evaluation of resource \((rs, val)^{or}\) is \(ev\). In this case such a resource was previously offered by agent \(j\), i.e. it is \(j\)'s output resource.

- \(\{IMP, IMC\} \in M\) is the interaction memory set, where
  - \(IM = \prod_{r=0}^{ms-1}(aid_j, (rs, val), ev)\) is an interaction table recording the last \(ms\) interactions that an agent \(i\) had with any agent \(j\), specifying its identifier \(aid_j\), the resource \((rs, val)\) the interaction was about and the corresponding evaluation \(ev\). The interaction table size \(ms\) is specific to each interaction table.
  - \(IMP = \prod_{ir=0}^{g-1}im^{ir}\) is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the input resource \((rs, val)^{ir} \in IRS\),
  - \(IMC = \prod_{or=0}^{h-1}im^{or}\) is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the output resource \((rs, val)^{or} \in ORS\).
The agent identifier $aid$ is unique to an agent and helps to single it out from any other agent. However, the identifier by itself does not offer any sort of classification as suggested by the adaptation property of diversity. Nevertheless the resource set $R$ helps to provide it by allowing the agent to produce a set of output resources ($ORS$) and to consume another different set of input resources ($IRS$). Thus, when some agents produce the same resources (possibly with different values) they all belong to the same producer class. Likewise, when some agents prefer and consume the same resources it is said that they belong to the same consumer class. That is, the resource description $rs$ of a resource is used to categorise agents.

In a natural ecosystem, individuals belonging to the same consumer class are considered competitors because they target the same producers/resources. This classification based on resource preference, consumption and production corresponds to the element E5 of diversity. Furthermore, resource consumption itself supports the element E2 of aggregation.

### 4.3.3 Agent capability to forget

According to element E10 of non-linearity, agents need to interact and provide feedback on such interactions. In this regard, DAEM simplifies an ecosystem interaction and only considers resources being offered, evaluated, and a feedback of such evaluations. That is, agents do not consume (i.e. “eat”) resources nor other agents. Instead DAEM focuses on the preference of consumers/producers based on evaluations.

Interactions are recorded in the interaction memory set $M$, specifically in the interaction memories $IMP$ and $IMC$. The former is used to store the information related to interactions with producers, i.e. the input resource the interaction was about, the absolute evaluation, and the identifier of the agent the interaction was with.

Likewise, $IMC$ is used to record the information related to interactions with consumers, i.e. the output resource the interaction was about, the normalised evaluation received from the consumer, and the identifier of the agent the interaction was with. Each interaction memory contains a list of dedicated interaction tables $im$ corresponding one-to-one to either an input or an output resource (see Definition 2).

In order to implement the capability to forget, it is necessary to limit the amount of information to be stored in the memory. This is achieved by limiting the interaction tables to store the $ms$ most recent interactions only. This is defined as follows:

$$ms = s \cdot |Ag_{int}|$$  \hspace{1cm}  (4.1)
where the $s$ is a factor determining the memory size according to the number of potential peers $A_{set}$ this agent may interact with about a specific resource of interest. The actual value of $s$ (and thus $ms$) is discussed in Chapter 5, here it is only said that $ms$ is a size limiting the memory in such a way that not all interactions can be recorded.

Thus, the evaluation $ev$ sent back to agent $i$ is normalised to the highest evaluation agent $j$ has given in the last $ms$ interactions regarding the resource $(rs, val)_{or}$, i.e. the maximum absolute evaluation $ev$ found in (“remembered” by) the agent $j$’s interaction table $im_{ir} \in IMP$. Memory limitation provides the agents with the ability to forget, which corresponds to the element E4 of diversity.

### 4.3.4 Resource evaluation and feedback

Actions $sendof$ and $sendev$, contained in $A$, are used to offer a resource and to provide an evaluation or feedback on such a resource, respectively. For instance, when agent $i$ offers a resource to agent $j$, using action $sendof(aid_i, aid_j, (rs, val)_{or})$, agent $j$ immediately verifies whether the description $rs$ of the offered resource corresponds to one of its input resource descriptions, i.e. $rs$ in the tuple $(rs, val)_{ir} \in IRS$. If such is the case, then agent $j$ proceeds to evaluate the offered resource.

Notice that the element $val$ from the resource $(rs, val)$ has different meanings according to whether it refers to an output or an input resource. In the former, $val$ refers to the resource value commonly offered, cf. a reference value for all output resources. In the latter, $val$ refers to the minimum expected value of an offered resource, cf. a quality standard for all input resources.

Thus when agent $j$ analyses the offered resource, the agent evaluates it by comparing it to its reference value $val$ in $(rs, val)_{ir}$, and then normalises it according to the best recent offer. Afterwards, agent $j$ returns to agent $i$ a positive normalised evaluation $ev$, using action $sendev(aid_j, aid_i, (rs, val)_{or}, ev)$. Otherwise, a negative value is sent back meaning that agent $j$ does not require (or is not interested in) that particular resource as an input.

Normalisation uses the information of the most recent interactions. Therefore the normalised value of $ev$ sent to a producer agent $i$ depends on the resource evaluation function particular to agent $j$ and on its most recent experience. This is shown in the following equation:

$$
ev = \begin{cases} 
\frac{1}{\text{eval}((rs, val)_{or})} & \text{if } \text{eval}((rs, val)_{or}) \geq \text{ev}_{\max} \\
\text{ev}_{\max} & \text{otherwise}
\end{cases}$$

(4.2)

where $\text{ev}_{\max}$ is the $ev$ with the maximum value contained in the tuples within the
corresponding memory table \( im_{or} \) in the input memory \( IMP \), e.g. the \( ev \) obtained from the interaction with an agent \( h \), i.e. \( (aid_h, (rs, val)_h, ev) \in im_{or} \in IMP \). Notice that the subscript \( or \) is used here to denote that agent \( i \)'s output resource description is found in agent \( j \)'s input memory table \( im \).

Function \( eval \) evaluates resources according to the agent’s preferences. Thus, a resource from the same producer could be evaluated differently by two independent consumers. As mentioned above, the evaluation \( ev \) received through other agent’s action \( sendev \) and stored in an interaction table \( im_{ir} \in IMP \) is always within the range \([0, 1]\) because of the normalisation. Whereas the evaluation \( ev \) recorded in \( im_{or} \in IMC \) have an open range \([0, \infty)\) because of the absolute result coming directly from function \( eval \).

Normalised evaluations are a reflection of the relative difference in the perceived resource quality offered by different producers based on the recent resource offer’s highest evaluation. Therefore, a normalised evaluation functions as a measure for the minimum relative improvement required in a resource for an agent to become a preferred producer. Thus, resource improvements can be noticed when new evaluations are made causing partner preference changes, for instance, by raising standards and thus expectations.

For example, consider an agent \( i \) who grants \( x \) as the absolute evaluation for agent \( j \)'s offered resource \((rs, val)^j_i\). Say \( x \) turns out to be the highest evaluation \( i \) has recently granted for that \( rs \), i.e. \( ev_{\text{max}} = x \). Then agent \( i \) sends a normalised evaluation of \( ev = 1 \) to agent \( j \) who now knows it is the preferred producer for agent \( i \).

Say a similar situation occurs with agent \( h \): it offers resource \((rs, val)^h_i\) to agent \( i \) who concedes \( y \) as the corresponding absolute evaluation. Say \( y \) turns out to be greater than \( ev_{\text{max}} \), i.e. \( y > x \). Thus \( ev_{\text{max}} = y \) and agent \( h \) receives \( ev = 1 \) as the normalised evaluation. Likewise, agent \( h \) now knows it is agent \( i \)'s preferred producer.

However, when agent \( j \) interacts again with agent \( i \) and the latter compares agent \( j \)'s resource with the new value of \( ev_{\text{max}} = y \), agent \( i \) sends a normalised evaluation of \( ev = x/y \) to agent \( j \) where \( x/y < 1 \) after Equation 4.2. Finally, agent \( j \) knows it has been replaced as a preferred producer.

In summary, the resource evaluation and feedback mechanisms encourage competition by providing an idea of how much improvement agents require to make in order to be the preferred producer. This complies with the element \( E10 \) of non-linearity. Furthermore, improvements can be considered as independent adaptations whose effects can be detected by the differences in resource evaluations, both absolute and normalised, once adaptation is achieved. DAEM does not include individual agent adaptation but its effects can be detected.
4.3.5 Definition of the environment

An environment is a fundamental element for natural ecosystem. In [Odell et al., 2003] an environment was roughly defined as the provider of conditions under which an agent exists. However, for incorporating the elements of the adaptation properties identified in Section 4.1 the environment is defined as a virtually observable mediating surface where agents move around and encounter others in order to interact. It is a common environment as identified by [Omicini et al., 2004]. The environment supports capabilities such as a sense of positioning and displacement, proximity-based interactions and surrounding awareness.

**Definition 3.** An environment \( env \) is a grid characterised using the tuple \((es, Ads, D)\) where

- \( es \) is the environment size, i.e. the grid is \( es \times es \) where each side goes from 0 to \( es - 1 \),

- \((eid, epos, EIRS, EORS)_i\) is an agent description tuple specifying agent \( i \)'s identifier \( eid \) (cf. \( aid \) in Definition 2), position \( epos = (x, y) \) in the environment, and its resource description lists \( EIRS = \prod_{ei=0}^{g-1} rs_{ei} \) and \( EORS = \prod_{eo=0}^{h-1} rs_{eo} \) (cf. IRS and ORS in Definition 2).

- Let \( Ag \) denote the set of agents currently existing in the environment, then \( Ads = \prod_{j=0}^{|Ag|-1} (eid, epos, EIRS, EORS)_i \) is a table describing agents, their positions and resources, existing in the environment. The number of description tuples depends on the number of agents \(|Ag|\) currently in the environment.

- \( \{itr\} \in D \) is the agent perception radii set containing the agent interaction radius \( itr \) where \( itr \geq 1 \).

The environment shape is a torus formed by having the \( es \times es \) grid seamed at the edges: both \( x \) and \( y \) axis run from 0 to \( es - 1 \). Then the lines \( y = 0 \) and \( y = es - 1 \) are contiguous as well as the lines \( x = 0 \) and \( x = es - 1 \). The result is a torus in which an agent can wander continually without “falling off”, yet it is the two-dimensional surface the one used by the agent where Euclidean distances can be calculated. The environment size \( es \) depends directly on the number of agents \(|Ag|\) inhabiting it, i.e.

\[
es = f \cdot |Ag|
\]  

(4.3)

where \( f \) is a factor determining the environment size according to the currently existing number of agents in the environment \(|Ag|\). The actual value of \( f \) is discussed on Chapter 5, here let us say that the environment size \( es \) must be large enough
for the agents to avoid to come across to each other highly frequent if they moved entirely random on it. The existence of an environment as virtual space for moving around supports element E3 of aggregation.

### 4.3.6 Situatedness in the environment

The table $Ads$ contains descriptive information of each agent currently existing in the environment. Such information includes the identifier, the agent current position on the environment, and related resource description lists for input and output resources. When the environment holds such information of an agent it is said that the agent exists or inhabits the environment. Nevertheless, for the environment to hold the aforementioned information and to provide with situatedness it is necessary to define the relationship between an agent and the environment.

The environment uses the agents’ information and exposes it to the rest of the agents in the environment, cf. a common environment [Omicini et al., 2004]. Definitions 2 and 3 show elements in common, namely $aid$, $IRS$, $ORS$ and $eid$, $EIRS$, $EORS$ respectively. These environment elements are reflections of the agent elements into the environment itself, i.e. information exposition.

**Definition 4.** Reflection is an environment property which denotes that some values are exposed from the agent into the environment in such a way that if a change occurs to the original one, it is reflected on the counter part. However, it depends on the agent to permit values to be reflected in the environment. Reflection is represented with the relation $\rho$ saying that the first argument is reflected on the second one for agent $i$’s values, given that $i$ has allowed it.

Therefore agent identifier $aid$ and agent resource lists $IRS$ and $ORS$ are reflected in the environment as $eid$, $EIRS$, $EORS$, respectively. This results in the following reflections—using an infix notation—between an agent $i$ and the environment. Read $\leftarrow$ as “gets”.

\[
aid \rho_i eid := eid_i \leftarrow aid_i \tag{4.4}
\]
\[
IRS^i \rho_i EIRS := EIRS_i \leftarrow \{ \exists ir \in IRS_i \mid rs_{ir} \}^i \tag{4.5}
\]
\[
ORS^i \rho_i EORS := EORS_i \leftarrow \{ \exists or \in ORS_i \mid rs_{or} \}^i \tag{4.6}
\]

where $EIRS$ is a set containing only the resource descriptions of each resource contained in agent $i$’s input resource set $IRS$. Likewise, $EORS$ is a set containing only the resource descriptions of each resource contained in agent $i$’s output resource set $ORS$. The equations shown above apply for all agents $i \in Ag$ existing in the environment $env$. Both $IRS^i$ and $ORS^i$ contents are chosen by the agent itself according to what the agent wants to expose to the environment.
Situatedness is an environment characteristic identified as element E3 of the adaptation property of aggregation. However it is now necessary to add new capabilities to the agents for them to make use of the environment and fully support this element. The following definition expands the previous agent definition for this purpose.

**Definition 5.** The tuple \((aid, R, A, M)\) is used to describe an agent \(i\) and related interactions where additional or revised items (in bold) are described as follows:

- \(aid\) is the agent's identifier.
- \(\{IRS, ORS\} \in R\) is the resource set, where
  - \(IRS = \prod_{i=0}^{n-1}(rs, val)_{ir}\) is a list containing all input resources.
  - \(ORS = \prod_{o=0}^{h-1}(rs, val)_{or}\) is a list containing all output resources, where \((rs, val)_{ir} \neq (rs, val)_{or}\) for all resources in \(IRS\) and \(ORS\) respectively.
- \(\{sendof, sendev, walk, subscr, unsubscr\} \in A\) is the (revised) set of actions available to the agent, where
  - \(sendof(aid_i, aid_j, (rs, val)_{or})\) signifies that agent \(i\) offers the resource \((rs, val)_{or}\) to agent \(j\).
  - \(sendev(aid_i, aid_j, (rs, val)_{or}, ev)\) signifies that agent \(i\) informs to agent \(j\) that the evaluation of agent \(j\)'s resource \((rs, val)_{or}\) is \(ev\).
  - \(walk(dest)\) signifies that the agent changes its current position on the environment towards the point \(dest = (p, q)\).
  - \(subscr(aid, IRS', ORS')\) conveys the agent's willingness to reflect agent values namely, the agent identifier \(aid\), and the resource descriptions found in the lists \(IRS'\) and \(ORS'\) into the environment, where \(IRS' \subseteq rs IRS\) and \(ORS' \subseteq rs ORS\).
  - \(unsubscr(aid, IRS', ORS')\) conveys the agent's willingness to stop reflecting agent values namely, agent identifier \(aid\), and resource descriptions found in the lists \(IRS'\) and \(ORS'\) into the environment, where \(IRS' \subseteq rs IRS\) and \(ORS' \subseteq rs ORS\).
- \(\{IMP, IMC\} \in M\) is the interaction memory set, where
  - \(im = \prod_{r=0}^{ms-1}(apos, aid_j, (rs, val), ev)\) is an interaction table recording the last \(ms\) interactions that took place with any agent \(j\), specifying its identifier \(aid_j\), location \(apos\) (cf. epos) where agent \(i\) was when the interaction took place, the resource \((rs, val)\) the interaction was about and
the corresponding evaluation $ev$. The interaction table size $ms$ is specific to each interaction table.

- $IMP = \prod_{ir=0}^{g-1} im_{ir}$ is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the input resource $(rs, val)_{ir} \in IRS$,

- $IMC = \prod_{or=0}^{h-1} im_{or}$ is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the output resource $(rs, val)_{or} \in ORS$.

Actions $subscr$ and $unsubcr$ express the agent’s willingness to disclose (or stop disclosing) itself and related resources into the environment, complementing Definition 4. Effectively, it is a mechanisms for registering to the environment to permit it to disclose related resources and to track agent movements. But before an agent can move in the environment, it is necessary that it knows where it is positioned in the environment. This piece of information comes from the position $epos$ in an agent description tuple $(eid, epos, EIRS, EORS)_i \in Ads$. However the agent gets to know it not by the reflection property, but by a positioning sense as described below:

**Definition 6.** An agent $i$ senses its position on the environment by receiving the pair $apos = (x, y)$ corresponding to $epos$ in the agent description tuple $(eid, epos, EIRS, EORS)_i \in Ads$, where $Ads$ is part of the environment $env$.

The pair $(x, y)$ is updated with every change regarding agent movement in the environment. Thus regardless of the agent position on the environment, the agent senses its current position.

The environment characteristics such as recording existing agents and their positions, the agents sensing their current positions on the environment, and the environment reflection property correspond to the element $E3$ of the adaptation property of aggregation.

### 4.3.7 Agent displacements in the environment

A straightforward usage of agent positioning is agent displacements. In order to express that agents move in the environment, action $walk(dest)$ has been added to the agent definition (see Definition 5). Such an action transforms the environment every time it is performed because it re-locates the agent to a nearby position from the current one. Therefore, by consecutive re-locations towards a destination point $dest$ and acquiring the corresponding position pair $apos = (x, y)$, the agent has a sense of displacement in the environment. The environment transformation is
represented in the following way:

$$\omega(i, epos, dest) := epos_i \leftarrow epos_i + \text{ang}$$

(4.7)

$\text{ang} = (u, v)$ is a pair corresponding to a direction-vector which represents an unambiguous indication of an agent’s next position. Using the 1-norm (cf. Manhattan norm) commonly used in the research community, an agent has four possible directions to move to from any position on the environment due to the torus. However, $\text{ang}$ depends on the final destination $dest$ the agent wants to walk to.

Randomness is part of element E7 supporting the adaptation property of diversity, agent movements are determined by randomly selecting the agent’s final destination as follows:

$$\text{dest} = ([\text{rand()} * \text{es}], [\text{rand()} * \text{es}])$$

(4.8)

where $\text{rand()}$ is a function returning a random number within the range $[0, 1]$.

Thus when the agent moves in the environment, performing action $\text{walk}($dest$)$, the environment calculates the agent’s direction-vector $\text{ang}$. For this purpose it is necessary to consider the surface continuity.

For instance, having an environment of size $\text{es} = 100$ and an agent wanting to reach the point $\text{dest} = (97, 30)$ from position $(5, 30)$ it could walk by either moving further in the positive region of $x$, i.e. $x = x + 1$ and eventually reaching the destination or moving towards the negative region of $x$, i.e. $x = x - 1$ and then appearing on $(99, 30)$ because of the torus and continuing moving towards the negative region. The latter option is the shortest path and the most convenient for the agent to move to.

The environment considers the shortest path and calculates the direction-vector $\text{ang}$ accordingly. This figuring is performed as described in the equations below:

$$\text{ang} = \begin{cases} \frac{-\text{\text{p-x}}}{|p-x|} & \text{if } |p - x| \geq \frac{\text{es}}{2} \\ \frac{p-x}{|p-x|} & \text{if } |p - x| < \frac{\text{es}}{2} \end{cases}, \begin{cases} \frac{\text{\text{q-y}}}{|q-y|} & \text{if } |q - y| \geq \frac{\text{es}}{2} \\ \frac{q-y}{|q-y|} & \text{if } |q - y| < \frac{\text{es}}{2} \end{cases}$$

(4.9)

where $(p, q)$ is the destination point $\text{dest}$ (randomly selected) and $\text{es}$ is the size of the environment $\text{env}$.

The direction-vector $\text{ang} = (u, v)$ is then adjusted to decide whether to re-locate the agent to a nearby position along the x-axis or along the y-axis, i.e. diagonal displacements are not allowed due to the 1-norm. Thus the possible values of
4.3 Formalising DAEM

$\text{ang} = (u, v)$ are $(-1, 0), (1, 0), (0, -1), (0, 1)$, and $(0, 0)$. The latter is not precisely a direction-vector but it does not go against the 1-norm. The adjustment is randomly made according to the distance the agent would need to walk on each axis as depicted below:

$$\text{ang} = \begin{cases} 
(u, 0) & \text{if } \text{rand}() \ast (dx + dy) \leq dx \\
(0, v) & \text{otherwise}
\end{cases} \quad (4.10)$$

where the random function iteratively allows a “diagonal” displacement in the environment towards $\text{dest}$, and $dx$ and $dy$ are obtained as follows:

$$dx = (u(p - x) + es \mod es) \quad (4.11)$$

$$dy = (v(q - y) + es \mod es) \quad (4.12)$$

The operator $\mod$ and the use of $es$ in the equation maintain the torus continuity. Once the direction-vector $\text{ang}$ is obtained, the new agent position $\text{epos}$ calculated by the environment according to Equation 4.7 becomes

$$\text{epos} \leftarrow \text{epos} + \text{ang} := ((x + u + es \mod es), (y + v + es \mod es)) \quad (4.13)$$

where the operator $\mod$ and the use of $es$ help to maintain the torus continuity.

For instance, if an agent $i$ is located at $\text{epos} = (0, 6)$ on an environment of size $es = 100$, and walks ($\text{walk(dest)}$) towards e.g $\text{dest} = (84, 7)$ randomly selected, then the new position will be $\text{epos} = (99, 6)$. Agent $i$ then knows its current position by obtaining the pair $\text{apos} = (99, 6)$ by the positioning sense explained in Definition 6. Agent random displacements in the environment contributes to support the elements $E6$ and $E7$ of the adaptation property of diversity.

4.3.8 Environment-mediated local interactions

As can be appreciated in Definition 5, the interaction table $im = \prod_{r=0}^{ms-1} (\text{apos}, \text{aid}_j, (rs, val), ev)_r$ now contains the element $\text{apos}$ in its tuples, allowing an agent $i$ to store in its memories $\text{IMP}$ and $\text{IMC}$ the location at which an interaction with an agent $j$ took place in the environment. Agents walk randomly in the environment and because of that, agent interactions occur randomly as well. But before that, any two agents have to approach to each other to a distance suitable for interacting. The environment definition specifies a “close enough” distance for interaction by using the interaction radius $itr \in D$ which delimits a circle of influence around each agent. This interaction radius constrains the agent to have interaction only with those agents within such a circle of influence. It is defined as follows:
Definition 7. An agent $i$ senses the proximity of another agent $j$ within a distance $itr$ by receiving a proximity table $T$ containing agent $j$’s identifier, position and related resource descriptions from the environment $env$:

$$T = \{ \forall |e_{posi} - e_{posj}| \leq itr \; j \in Ads \mid (eid, e_{pos}, EIRS, EORS) \}$$  \hspace{1cm} (4.14)

where $|e_{posi} - e_{posj}|$ is the Euclidean distance between agent $i$ and agent $j$ and $itr \in D$ is the proximity radius.

The proximity table $T$ is updated with every change occurring in the environment. Receiving the proximity table $T$ is the mechanism all agents have for starting interactions with potential consumers nearby. The agents use it to select with whom to interact based on the resources produced and consumed by their peers in the surrounding area. Moreover, the environment constrains agent interactions by enabling agent action $sendof$ only between agents within each other’s proximity area. Thus, the proximity sense forces local interaction in the environment complying with element E6 of the adaptation property of diversity. Moreover, the environment mediates agent interactions, thus conforming with element E11 of the adaptation property of non-linearity.

4.3.9 Environment-mediated surrounding awareness

So far DAEM supports agent interactions between randomly encountered agents in the environment. However, they can hardly gain structure if they continue interacting randomly, consequently no structured community such as an ecosystem can surface.

In natural ecosystems, individuals move around in the environment in their search for food. They track down potential preys by perceiving trails which go beyond their reach. They usually make use of “senses”, e.g. sight, smell, etc., and follow a notional gradient in the environment which guides their exploratory behaviour [Parunak, 1997, Jørgensen and Fath, 2004]. By giving the agents these mechanisms they could use them for guiding their exploratory behaviour towards the agents producing the resources they are interested in. In particular, agents might “see” beyond their interaction space and “smell a trail” left by other in the environment. For this purpose, the environment definition is modified to support such “senses.”

Definition 8. An environment $env$ is a grid characterised using the tuple $(es, Ads, Mks, D)$ where additional and revised items (in bold) are defined as follows:

- $es$ is the environment size,
4.3 Formalising DAEM

- \((eid, epos, EIRS, EORS)_i\) is an agent description tuple specifying agent \(i\)'s identifier \(eid\), position \(epos = (x, y)\) on the environment, and its resource description lists \(EIRS = \prod_{i=0}^{k-1} rs_{ei}\) and \(EORS = \prod_{eo=0}^{h-1} rs_{eo}\).

- \(Ads = \prod_{i=0}^{Ad_{j=0}} (eid, epos, EIRS, EORS)_i\) is a table describing agents, their positions and resources, existing in the environment.

- \((epos, EIRS, EORS, ods)\) is a mark description tuple specifying that an agent left an evaporative mark at position \(epos = (x, y)\) and containing resource description lists \(EIRS\) and \(EORS\). Additionally, marks have an odour strength of \(ods\) which evaporates gradually one unit at a time.

- \(Mks = \prod_{m=0}^{ods_{max}} Ads = \prod_{m=0}^{ods_{max}} (epos, EIRS, EORS, ods)_m\) is a table where all marks are described. The number of mark description tuples depends on the maximum odour strength \(ods_{max}\) a mark can have and the number of subscribed agents in the environment. This is to allow each agent to have as many evaporative marks as the odour strength. The actual value of \(ods_{max}\) is discussed further in Chapter 5. Duplicates are allowed to exist.

- \(\{itr, sgr, smr\} \in D\) is the perceptions radii set where \(itr\) is the interaction radius, \(sgr\) is the sight radius, and \(smr\) is the smell radius; where \(smr > sgr > itr \geq 1\).

An additional element to the new environment definition is the sight radius \(sgr\) which allows an agent to identify other agents within a distance \(sgr\), but not to make any sort of contact. The agent would need to come closer (within a distance \(itr\)) and interact either to know the value of a resource or to offer a resource. This sense expands the perception area an agent has in the environment and it is defined as follows:

**Definition 9.** An agent \(i\) sees another agent \(j\) beyond a distance \(itr\) and up to \(sgr\) by receiving a sight table \(G\) containing agent \(j\)'s identifier, position and resource descriptions:

\[
G = \{itr < |epos_i - epos_j| \leq sgr \mid (eid, epos, EIRS, EORS)_j\} \quad (4.15)
\]

The sight table \(G\) is updated with every change occurring in the environment. Receiving the sight table \(G\) is the mechanism all agents have for identifying others at a maximum distance \(sgr\) and decide whether to approach or not to interact based on the agents’ resource descriptions. Agents use this table to analyse from a grater distance than \(itr\) whether is worth approaching or not for interaction.

Additionally, the new environment definition now includes evaporative marks represented by the table \(Mks\). Notice that there is no information about any agent
identifier, but the position where the agent was when the mark was dropped. Additionally, marks contain information such as resource description lists $EIRS$ and $EORS$ related to the agent who dropped the mark, and an odour strength $ods$. The latter is a diminishing value specifying the mark’s remaining strength which indirectly indicates how long ago the agent passed by, i.e. the lower the odour strength the longer time has passed since the agent was in that location.

Moreover, two or more marks with different odour strengths $ods$ (and the same resource list contents) form a trail in the environment, which helps the agents to create a notional gradient and figure out a direction on the grid of where the source might be. For this purpose, the smell radius $smr$ allows an agent to perceive marks left on the environment within a distance of $smr$, and follow a notional gradient or trail of where to look for the resources the agent is interested in.

In particular, the idea of sensing evaporative mark trails (cf. pheromones) is inspired by the concept of stigmergy (cf. [Parunak, 1997]); the purpose of mark trails in DAEM is to create an indirect communication medium for guiding the agent exploratory behaviour to encourage agent encounters. Indeed, there has been other approaches inspired by the same concept and used for the same purpose, but with a different application.

For instance, digital pheromones are used to guide agent displacement through a space, either in a virtual environment [Parunak et al., 2005] or in the real world [Mamei and Zambonelli, 2007]: dropped pheromones produce disparate concentration levels on the environment and create gradient fields, which allows an agent to perform a hill climbing search in order to move either towards or away the agent producing the pheromones.

In such an approach, agents sense the pheromone concentration on the spot. Therefore, for determining any gradient field direction at least two pheromone concentrations have to be read in different locations. That is, agents have to navigate first in the environment to get some orientation, and then decide what direction to take. Once a direction is figured out, the agent have to follow the actual pheromone trail all the way to the source agent regardless of any sudden direction change.

Whereas in DAEM, agents have a smell sense which allows them to perceive evaporative marks in a bigger area making the agents capable to detect partial trails along with their gradient at once without navigating first. Furthermore, DAEM agents can react readily to direction changes of the trail source and then take a straight path (within an area delimited by $smr$) to the new direction without the need to follow the actual trail all the time.

Another stigmergy-inspired approach, called cognitive stigmergy [Ricci et al., 2007], considers annotations (cf. pheromones) to keep information about mediating artefacts. These contain values to represent pieces of information denoting an
ontology. Annotations are created by agents and diverse mediating artefacts to provide feedback to other artefacts on session information, number of artefact user agents, etc. Even though the approach is stigmergy-inspired as well, its use and final purpose is different from that in DAEM.

Yet another difference between the aforementioned approaches and DAEM is that, in the former the agents are responsible for dropping the pheromones on the environment, thus leaving them with the choice of not to drop any pheromone at all. This might confuse agents whilst tracking down a trail source and possibly obtaining an undesired system behaviour.

In DAEM the environment is in charge of creating (“dropping”) the marks on behalf of the agents due to being an environment property. Therefore, mark existence (i.e. creation and evaporation) is managed by the environment for all agents registered in \( Ads \). A similar approach in which the environment manages pheromone-like tuples by means of rules is presented in [Mamei and Zambonelli, 2009].

To represent the environment property of mark dropping, the function \( \kappa(Ads, Mks) \) is used signifying that agent information contained in \( Ads \) is projected as new marks in \( Mks \) and left in the environment:

\[
\kappa(Ads, Mks) := Mks \leftarrow \{ m \in Ms, \text{ods}_{\text{max}} \mid (\text{epos}, EIRS, \text{EORS}, \text{ods}_{\text{max}})_m \} \\
Ms \leftarrow \{ \forall i \in Ads \mid (\text{epos}, EIRS, \text{EORS})_i \}
\] (4.16)

where \( \text{epos}, EIRS, \) and \( \text{EORS} \) come from the agent description tuples contained in \( Ads \) and \( \text{ods}_{\text{max}} \) is the maximum odour strength a mark can initially have.

The evaporation process makes the odour strength \( \text{ods} \) to diminish over time until the mark disappears. This implies an additional transformation to reflect this process. Function \( \tau(Mks) \) represents such a function by specifying that existing marks vanish as time elapses:

\[
\tau(Mks) := Mks \leftarrow TMk \backslash \{ \forall_{\text{ods}=0} t \in TMk \mid (\text{epos}, EIRS, \text{EORS}, \text{ods})_t \} \\
TMk \leftarrow \{ \forall m \in Mks, \text{ods} \mid (\text{epos}, EIRS, \text{EORS}, \text{ods} - 1)_m \}
\] (4.17)

Finally, to make use of the marks for orientation the agent needs to perceive them. Such a perception if defined as follows:

**Definition 10.** An agent \( i \) smells mark \( m \) beyond a distance \( sgr \) and up to \( smr \) by receiving a smell table \( L \) containing mark \( m \)'s position, resource descriptions and odour strength:

\[
L = \{ \forall_{sgr < |\text{epos}_i - \text{epos}_m| \leq smr} m \in Mks \mid (\text{epos}, EIRS, \text{EORS}, \text{ods})_m \}
\] (4.18)

Likewise, the smell table \( L \) is updated with every occurring environmental trans-
formation $\kappa$ and $\tau$. Marks contain only resource descriptions, thus for identifying
the source agents it is necessary to approach to a distance of $\text{sgr}$ which is the sight
radius. Marks existing within the sight radius are not received in the table $L$ (i.e.
not perceived) because there is no pragmatic reason to duplicate information, i.e.
within the sight radius resource descriptions can also be detected as well as identify
agents, whereas with marks there is no agent identification.

Receiving a smell table $L$ is the mechanism all the agents have for sensing a
notional gradient for guiding their exploratory behaviour in the direction where it
seems there is an agent producing something of their interest. In summary, the
environmental property to create evaporative marks in an agent's path in the en-
vironment, the agents' ability to see agents and smell marks related to resources
produced and consumed by others, and specially the agent capability to orient itself
and guide a foraging behaviour refer to the element $E_8$ of the adaptation property of
flow. Also, the element $E_{11}$ of the adaptation property of non-linearity is strength-
ened.

### 4.3.10 Agent foraging behaviour

Foraging requires a target to track down which is a preferred consumer or a pre-
ferred producer. For this purpose, an additional element in the agent definition is
added as shown below:

**Definition 11.** The tuple $(\text{aid}, R, A, M)$ is used to describe an agent $i$ and related
interactions where additional or revised items (in bold) are described as follows:

- $\text{aid}$ is the agent's identifier.
- $\{\text{IRS}, \text{ORS}, \text{rsp}\} \in R$ is the resource set, where
  - $\text{IRS} = \prod_{ir=0}^{\text{rs}-1}(\text{rs}, \text{val})_{ir}$ is a list containing all input resources.
  - $\text{ORS} = \prod_{or=0}^{\text{rs}-1}(\text{rs}, \text{val})_{or}$ is a list containing all output resources, where
    $(\text{rs}, \text{val})_{ir} \neq (\text{rs}, \text{val})_{or}$ for all resources in $\text{IRS}$ and $\text{ORS}$ respectively.
  - $\text{rsp}$ is a pointer to the description of the resource being the foraging
target alternating between $\text{IRS}$ and $\text{ORS}$.
- $\{\text{sendof}, \text{sendev}, \text{walk}, \text{subscr}, \text{unsubscr}\} \in A$ is the set of actions available to
  the agent, where
  - $\text{sendof(aid}_i, \text{aid}_j, (\text{rs}, \text{val})_{or})$ signifies that agent $i$ offers the resource
    $(\text{rs}, \text{val})_{or}$ to agent $j$.
  - $\text{sendev(aid}_i, \text{aid}_j, (\text{rs}, \text{val})_{or}^{aid}_j, \text{ev})$ signifies that agent $i$ informs to agent
    $j$ that the evaluation of agent $j$'s resource $(\text{rs}, \text{val})_{or}$ is $\text{ev}$.  

4.3 Formalising DAEM

- \textit{walk}({\textit{dest}}) signifies that the agent changes its current position on the environment towards the point \textit{dest} = (p, q).

- \textit{subscr}({\textit{aid}}, {\textit{IRS}}, {\textit{ORS}}) conveys the agent’s willingness to reflect agent values namely, agent identifier \textit{aid}, and resources found in the lists {\textit{IRS}} and {\textit{ORS}} into the environment, where {\textit{IRS}} \subseteq {\textit{IRS}} and {\textit{ORS}} \subseteq {\textit{ORS}}.

- \textit{unsubscr}({\textit{aid}}, {\textit{IRS}}, {\textit{ORS}}) conveys the agent’s willingness to stop reflecting agent values namely, agent identifier \textit{aid}, and resources found in the lists {\textit{IRS}} and {\textit{ORS}} into the environment, where {\textit{IRS}} \subseteq {\textit{IRS}} and {\textit{ORS}} \subseteq {\textit{ORS}}.

- \{IMP, IMC\} \in M is the interaction memory set, where

  - \textit{im} = \prod_{r=0}^{m-1} (apos, \textit{aid}j, (rs, val), ev)\textsubscript{r} is an interaction table recording the last \textit{ms} interactions that took place with any agent \textit{j}, specifying its identifier \textit{aid}j, location apos where the interaction took place, the resource (rs, val) the interaction was about and the corresponding evaluation ev. The interaction table size \textit{ms} is specific to each interaction table.

  - IMP = \prod_{ir=0}^{g-1} \textit{im}ir is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the input resource (rs, val)ir \in {\textit{IRS}},

  - IMC = \prod_{or=0}^{h-1} \textit{im}or is an interaction memory containing a list with all the interaction tables corresponding one-to-one to the output resource (rs, val)or \in {\textit{ORS}}.

All agents have a principal goal when they forage: to find the preferred producer and consumer for each of their resources. A preferred producer is the agent with the highest absolute evaluation contained in the respective interaction table within IMP. On the other hand, a preferred consumer is the agent who provided the highest normalised evaluation stored in the respective interaction table within IMC. Then, according to the interaction tuple with the highest evaluation regarding the resource description pointed by \textit{rsp}, a destination point \textit{dest} is selected. Notice than \textit{rsp} points to a resource description continually alternating between {\textit{IRS}} and {\textit{ORS}}.

Agents possess two foraging behaviours: a direct foraging and a semi-random foraging. The former is activated when \textit{rsp} points to a resource description, e.g. \textit{rs} in (rs, val)ir \in {\textit{IRS}}, and the agent \textit{i} tries to reach the position where the last interaction with the preferred producer took place, as recorded by the corresponding interaction memory \textit{im}ir \in IMP. If such an agent is seen, then agent \textit{i} follows it
either until interacting or until running into a better producer in the way. After such an interaction, the pointer \( rsp \) switches to another resource description to forage in the other resource list, e.g. \( rs \in (rs, val)_{or} \in ORS \) and the process starts over with a different target.

For the particular case when an agent does not have either any input or any output resource to forage for (due to being a basic producer or a top consumer), then the initial position on the environment (obtained when the agent first registered) is used as a foraging destination to avoid agents to stick to a producer or consumer, and to continue providing dynamism to the environment.

However, if agent \( i \) gets to the destination and the target producer agent is not there nor at sight reach, then the semi-random foraging behaviour starts to functions. From this moment on, agent \( i \) tries to follow any mark trail related to the resource description being foraged for, i.e. the resource description pointed by \( rsp \). Once the trail source is found and the interaction takes place, \( rsp \) switches to another resource description in the other resource list, e.g. \( rs \in (rs, val)_{ir} \in ORS \), then the process starts over in a direct foraging behaviour.

This particular interaction occurs regardless of whether the trail source was the target agent or not. Nonetheless, if the resource description targeted by \( rsp \) cannot be smelled near by, then agent \( i \) starts foraging randomly in the environment until the mark is detected and follows such a trail. Notice that any trail found referring to the target resource description could be produced by a non-preferred agent. Even in that case the agent will follow the trail.

The resource pointer \( rsp \) indicates the resource description to look for, yet the specific agent being targeted comes from the memory according to the resource (indicated by \( rsp \)) with the highest evaluation, i.e. the preferred agent. Therefore non-preferred agents are not foraged for, but that does not mean there are no interactions with them. If they come within the proximity area an interaction is likely to occur and will be recorded in the involved agents’ memory. Yet they will not be a priority thus the place where the interaction occurred will be ignored, until their resources are the highest evaluated, cf. repulsion.

DAEM does not consider any mapping between performing a task or gaining knowledge in either the physical world or the supported system, and walking, foraging, seeing, smelling, or being in the proximity area of another agent in the environment (see Figure 4.4). The environment is a place for exploration, interaction and adaptation by abstracting only the ecosystem elements from a complex and dynamic domain, i.e. an ecosystem domain.

In summary, a resource pointer is used for resource foraging which exploits agent senses to track down preferred consumers and producers by following interest gradients. Foraging supports the element E7 of the adaptation property of
diversity. Moreover, the foraging behaviour along with the effect produced by resource evaluation feedback support the element E10 of the adaptation property of non-linearity.

4.3.11 Linking producers and consumers

Agents forage for resources produced or consumed by those agents with the highest evaluation. Agents interact with those agents found in the foraging path regardless of whether they are the foraging target. When a producer agent is preferred by its preferred consumer agent they forage for each other, i.e. they will return to the area when they had their last interaction (as long as they remain in each other’s short memory). As long as no other agent with a better resource interferes, mutually preferred producer and consumer will keep on foraging for each other, thus creating and strengthening a “producer-consumer” link. Because links are formed by “the best” producer and consumer, they may become an interest attraction point to other agents because of wanting to interact with the best agents, thus creating and encouraging competition. Since agent interactions depend on individual preferences, more than one separate link may appear resembling natural ecosystems where different links between the same species exist. The notion of link formation is formalised as described below.

Definition 12. A link is formed when the output resource of an agent provides the input resource to another agent, and the agents prefer each other:

\[ l = i \odot j \]  

(4.19)

where \( i \) and \( j \) are agents and one of agent \( i \)’s output resource descriptions corresponds to one of agent \( j \)’s input resource descriptions \( rs_i \equiv rs_j \) where \( (rs, val)_i \in ORS_i \) and \( (rs, val)_j \in IRS_j \), i.e. agent \( i \) produces a resource consumed by agent \( j \). Notice that the operator \( \odot \) is not commutative.

The symbol \( \odot \) is used to emphasise that the link formed by two reciprocally preferred agents represents a resource flow, i.e. it resembles both the production and consumption of a resource, though as mentioned earlier the actual resource consumption is not considered part of the model.

A logical deduction is that an agent can be part of more than one link if more than one output resource is available. That is, \( i \odot j \), and \( i \odot k \) are valid links if the relations \( rs^1_i \equiv rs^0_0 \) and \( rs^1_k \equiv rs^0_0 \) exist where \( \bigcap_{h=1}^2(rs, val)_h \in ORS_i \) and \( (rs, val)_0^1 \in IRS_j \) and \( (rs, val)_0^k \in ORS_k \).

Extending the notion of a link, a path refers to a succession of links defined as follows:
**Definition 13.** A path from an agent \(i\) to an agent \(k\) is the minimum set of links required to connect agent \(i\) to agent \(k\); thus

\[
s^n(k) = l_0 \odot l_1 \odot \ldots \odot l_{n-1}
\]  

where \(n\) is the number of links involved in the path, \(l_0\) begins with agent \(i\) and \(l_{n-1}\) ends with agent \(k\). It is possible to write \(i \sigma^n k\) to indicate that there exists a path from \(i\) to \(k\) consisting of \(n\) links. Notice that \(i \sigma^1 k\) is equivalent to \(l_0\) which in turn is equivalent to \(i \odot k\). When \(n = 0\) the expression \(s^0(k)\) is valid indicating an empty path to \(k\).

As a consequence, any agent \(i\) can participate in as many paths as links it contributes to, i.e. for a path \(i \sigma^n k\) there is a link \(i \odot j\) where agent \(j\) is in the path to \(k\) as well, so that \(j \sigma^{n-1} k\) for \(n > 1\). The length \(n\) of a path \(s^n(k)\) depends directly on the number of agents registered in the environment. Therefore, the maximum value \(n\) may take is \(n = |Ag|\) where all agents participate in the same, single path.

Typically in ecosystems a species feeds from more than one other species and contribute to compound link in the food chain. Likewise in DAEM one agent can have more than one input resource and contribute to a compound link in the same path. By extending Definition 12 it is possible to take into account such a compound input:

**Definition 14.** A compound link comprises a subset of agents \(I\) whose some of their output resources all together match (some of) the input resources of an agent \(j\). A compound link is denoted as follows:

\[
\ell = I \odot j
\]  

where \(I \subseteq Ag\), \(j \in Ag\), \(j \notin I\) such that there is a link \(i \odot j\) for all agents \(i \in I\). In the specific case where \(I\) consists of only one agent \(i\), a compound link \(\ell\) is equivalent to a link \(l\) such that \(I \odot j = i \odot j\).

Likewise, any agent \(i\) could participate in as many compound links \(\ell\) as links \(l\) it forms part of. Before introducing the next level of linking, a couple of supporting concepts are introduced.

**Definition 15.** The notion of predecessor and successor is used to determine a relative position in a path. Let \(i, j \in Ag\) be agents, it is said that “\(i\) is a predecessor of \(j\)” or “\(j\) is a successor of \(i\)” if there exists a path from \(i\) to \(j\), i.e. \(i \sigma^n j\) is a valid path where \(n \leq |Ag|\).

**Definition 16.** A layer is the collection of predecessors at the link \(l_0\) of all paths \(s^n\) where agent \(j\) is the common successor. The layer \(n\) belonging to agent \(j\) is
4.3 Formalising DAEM

represented as follows:

\[ \psi^n(j) = \{ \forall \sigma_n i \in Ag \} \]  

(4.22)

The definition above conveys a compound element to be used as a high level building block to a chain, i.e. to connect consecutive layers. This is achieved by extending Definition 14 as shown below.

**Definition 17.** A layer link \( \lambda^n(j) \) consists of the connection between two consecutive layers \( \psi^n(j) \) and \( \psi^{n-1}(j) \) where agent \( j \) is the common successor; it is the collection of individual links \( l_0 \) and \( l_1 \) in all paths \( \sigma^n(j) \) to agent \( j \) where \( n > 1 \). This is represented as follows:

\[ \lambda^n(j) = \psi^{n-1}(j) \oplus \psi^n(j) \]  

(4.23)

where \( n > 0 \). Notice that \( \lambda^1(j) = \ell \) corresponds to the compound link \( I \odot j \) where \( I \subset Ag \) and \( j \notin I \).

Link formation both at the single and compound level represents a notion of flow between interacting agents, which are allowed to form part of many flows (links) at the same time. This resembles what occurs in natural ecosystem where species may be part of many food chains. By capturing this notion DAEM complies with the element E9 of the adaptation property of flow.

### 4.3.12 Link chaining

Using the elements explained above, now it is possible to formalise how a chain is formed. This formalisation will be used later to formally represent an ecosystem.

**Definition 18.** A chain to the agent \( j \) is the set of layer links connecting all different producers from a primary layer \( \psi^0(j) \) to the final consumer agent \( j \). This is expressed in the following way:

\[ \Sigma^n(j) = \lambda^1(j) \oplus \lambda^2(j) \oplus \ldots \oplus \lambda^n(j) \]  

(4.24)

where \( n > 1 \). Similarly, \( \Sigma^2(j) = \lambda^1(j) = \ell \) is the compound link \( I \odot j \) where \( I \subset Ag \) and \( j \notin I \).

The definition above uses existing layers to represent a chain. Thus for describing how these are formed it is necessary to define the basic case: the empty chain.

**Definition 19.** An empty chain is that which contains a single agent not forming part of a chain, and is represented as follows:

\[ \Sigma^1(j) = j \]  

(4.25)
Notice that any agent can potentially be part of as many links (and thus chains) as output resources it produces. Hence any agent \( i \) can belong to a chain \( \Sigma^n(j) \) of agent \( j \) due to one of its output resources, and at the same time the same agent \( i \) can belong to an empty chain \( \Sigma^1(i) \) because it has not found a preferred consumer for the other of its output resources. An empty chain is the basic element for describing the process of chaining.

**Definition 20.** The process of chaining for agent \( j \)'s chain consists of attaching agent \( i \)'s chain to it at the primary layer \( \psi^0(j) \). This process is represented in the following form:

\[
\Lambda(\Sigma^q(i), \Sigma^n(j)) := \Sigma^q(i) + \Sigma^n(j)
\]

such that agent \( i \) produces a resource consumed by an agent \( k \in \psi^0(j) \) and both agents prefer each other. The new length of agent \( j \)'s chain is simply \( n + q \), i.e. the new chain is represented by \( \Sigma^{n+q}(j) \).

As can be easily deducted, the simplest case consists of two agents \( i \) and \( j \) not linked to any other agent and thus forming a single link, i.e. \( \Lambda(\Sigma^1(i), \Sigma^1(j)) \equiv i \cdot j \).

Chaining at both single and compound level resembles some network dynamics namely connections, disconnections and re-connections. This notion strengthens the support on the element E9 of the adaptation property of flow.

### 4.3.13 Changes, adaptation and the ecosystem

Changes occur when an agent introduces a new resource, stops producing one, or improves a resource. Either of these events provokes a preference changes which may cause a disruption in a chain: better producers are typically preferred over others. Nonetheless, these preference changes occur at the single link level between agents making a portion of a path (therefore of a chain as well) to be cut at that point. It can be seen as the reverse of the chaining process. More formally, changes in DAEM are represented in the following way:

**Definition 21.** A change in a chain \( \Sigma^n(j) \) of agent \( j \) is a disruption in the chain continuity caused by an agent \( i \) in the chain who changed preference to another producer or consumer. It is represented in the following way:

\[
\Pi(\Sigma^q(i), \Sigma^n(j)) := \Sigma^n(j) - \Sigma^q(i)
\]

where \( n > q > 1 \), and \( i \) was in the layer \( \psi^{n-q}(j) \).

The definition above assumes the presence of competitors at the disrupted link because a preference change cannot happen if there is no substitute agent. Furthermore, it is said that a change is *unexpected* because the agents involved in a
change cannot anticipate it nor apprehend the effect such a change may cause. Thus, the “re-chaining” process, or adaptation is defined as shown below:

**Definition 22.** The process of adaptation in a chain consists of regenerating a disrupted link by substituting the missing agent by another one. It is represented in the following form:

\[
\Delta \left( \Sigma^n(j) \right) := \Lambda \left( \Sigma^w(i), \Sigma^n(j) \right) \quad \text{at } t_1 \\
\Pi \left( \Sigma^q(k), \Sigma^n(j) \right) \quad \text{at } t_0
\]

given that \( n > q > 1 \) and \( n > w > 1 \), and agent \( k \) was in the layer \( \psi^{n-q}(j) \) at the moment of disruption. Moreover, the “re-chaining” considers that agent \( i \) produces a resource consumed by an agent \( k \in \psi^{n-q}(j) \) and both agents prefer each other.

Notice that changes occur at the agent level when preference changes. Yet at the system level adaptation occurs when chains re-assemble after an alteration regardless of the agents forming the chain. Furthermore, the degree at which changes occur depend on the frequency of resource improvements, which may vary according to agent interactions, and new resources being offered possibly to new agents entering the ecosystem. Finally, combining the processes defined above it is possible now to formally define an ecosystem.

**Definition 23.** An ecosystem comprises a set agents, a set of chains, a set of competitors not included in the chains, and an environment. All together they represent links, chains, and competition whilst contending for survival to changes in the ecosystem. It is represented in the following form:

\[ \Gamma = (Ag, C; \{ \forall j \in C \Sigma^n(j) \}, P, env) \]

where \( env \) is the environment, \( Ag \) is the set of all agents, \( C \) is the set of all final consumer agents at the top of the food chain \( C \subseteq Ag \), and \( P \subset Ag \) is the set of all the competitors trying to participate in any of the chains due to producing resources that might be of the interest of any agents \( \{ \forall j \in C \Sigma^n(j) \} \).

Adaptation in DAEM is achieved by replacing missing agents in broken or disrupted single links, a process not apprehended at the individual level. This is due in part by the decentralisation and concurrency of the agents, i.e. by having no central, guiding control agents contribute to the complexity of the system, thus representing the element E12 of the adaptation property of non-linearity.

Finally, the assignment of states to the ecosystem is complicated because of the high complexity provided by the combination of all agent displacements, agent messaging, marks on the environment at different degrees of evaporation, supply chain
creation, change, and adaptation occurring at non-deterministic intervals. Such a feature is commonly found in natural ecosystems making it almost impossible to go back to a situation in the past due to the non-determinism, thus creating history. This process is called *transformational evolution* [Levin, 1998] and comprises the element E13 of the adaptation property of non-linearity.

### 4.4 Discussion

Ecosystems have been classified as CASs [Holland, 1995] which are objects of study for their inherent self-organising capabilities [Hassas et al., 2006]. The same applies to ecosystem domains such as those mentioned in Chapter 2. Yet because DAEM is based on CAS adaptation properties it does not necessarily imply that DAEM is capable of self-organisation.

Self-organisation is defined as a process or mechanism in which a system adjust its internal organisation in order to adapt to changes, either internal or external, with no additional guiding [Di Marzo Serugendo et al., 2005]. This concept is indeed similar to adaptation as defined in Section 2.2, yet the difference lies in that adaptation does not imply any adjustment in the internal organisation of the system as long as it reaches an equilibrium again. On the other hand, self-organisation makes no assumption that the resulting behaviour is a desirable one. Therefore, self-organisation and adaptation are overlapping concepts in the sense that both are processes by which a system adapts to changes.

By definition DAEM comprises with the capability of adaptation thus being an adaptive system. A set of properties of self-organising systems is presented in [Di Marzo Serugendo et al., 2005], namely no external control, decentralisation, and dynamism. DAEM complies with these properties thus in principle it possesses the capability of self-organisation as well, at least by definition. Notwithstanding, at this point it is not clear whether it shows emergent behaviours, the typical characteristic of self-organisation. This point will be discussed again in Section 5.8 when discussing the experimental results.

Finally, DAEM represents one of the contributions of this thesis, yet the testing of its hypotheses is presented in Chapter 5. A comparison to other ecosystem models is presented in Chapter 7.

### Summary

This chapter presented the formal exposition of DAEM, a model for adaptation in ecosystem domains. Then it shows how characteristics of ecosystem modelling
and principles of adaptive MAS development can be integrated under the adaptation properties of CASs. DAEM comprises agents, their interactions mediated by a perceivable environment, and chain formation, its changes, and the adaptation process. Finally, it is discussed that, by definition, DAEM is an adaptive system with properties of self-organisation. Yet the latter is discussed again after presenting the experimental results in Chapter 5.
Chapter 5

Experiments On The Adaptive Capacity of DAEM

This chapter presents the experiments carried out on a DAEM prototype to test its capability to overcome unexpected changes. The changes consists of unexpected resource innovations triggering preference changes whilst using different sets of parameters. Furthermore, a comparison with a centralised resource discovery approach (cf. a central resource directory) is presented. The results demonstrate how DAEM reacts and adapts better to changes in the environment than a centralised approach. Such experiments and comparisons represent the tests to the hypotheses H1, H2 and H3 of this thesis (see Section 1.3 for more details).

The chapter is structured as follows: Section 5.1 describes the DAEM prototype. Then the simulation setup is explained describing how the experiments were carried out and the parameters that were used. The first experiments are presented in Section 5.5 where the analysis presented are mainly on convergence and confidence. In addition, a comparison against a centralised approach is presented as well in Section 5.6. The experiment results then are summarised in Section 5.7 before finishing with a discussion in Section 5.8.

5.1 DAEM prototype

A running prototype of a model represents a proof of concept that the model can be developed. In addition, the prototype can be used for further testing to evaluate whether it performs as the model indicates and possibly to detect conditions and issues that where not considered beforehand. The DAEM prototype presented in this chapter is the result of several iterations between modelling, implementing, and testing, as expected when following the Design Science methodology explained in [Hevner et al., 2004].
5.1 DAEM prototype

The DAEM prototype developed consists of a number of components namely a set of independent agents, an environment, a communication medium for helping the environment to mediate communication, and a simulation setup. The latter uses the Multi-Agent Development Kit (Madkit) micro kernel to manage the agent execution time. Figure 5.1 shows more details about the prototype.

Initially, the setup component launches the agents according to the configuration parameters such as the number of top consumers, the maximum length of a chain, and the maximum number of competitors at a link. Using this information the agents are created and input and output resources are assigned to them. Further details will be given in the next section.

Once the agents are launched and their resources set, the setup module creates two entities: the environment and the Communication Medium. The latter substitutes the typical agent ACL messaging (cf. [Wooldridge, 2002]) to effectively re-route agent communication through the environment. The environment creates a graphical interface for visualising what is happening in the environment surface. Finally, information about the current environment situation is produced as output every certain number of simulation steps (to be detailed further on). This information is later used to analyse the overall DAEM behaviour as will be explained further on in this chapter.

5.1.1 Communication medium

The Communication Medium plays an important role: it helps the environment to mediate both agent-to-agent and agent-to-environment communication. Communication in DAEM is based on Act Messages which is the Madkit representation of ACL messages. Such a message was extended to create three other message types namely a) agent messages for agent-to-agent communication; b) environment actions for agent-to-environment action execution; and c) perceptions for

\[http://www.madkit.org/\]
environment-to-agent sensing results. The message hierarchy is depicted in Figure 5.2.

In detail, agent messages consist of resource offers between agents. For providing feedback on resources the same offer is sent back with the evaluation attached to it (see Section 4.3.4). Environment actions correspond to actions walk and subscribe as presented in Definition 5. Perceptions represent all environment sensing as defined throughout Section 4.3 namely positioning, proximity, sight, and smell.

Therefore the Communication Medium component catches all agent messages and environment actions sent by the agents and then passes them to the environment, which after processing them generates perceptions for each agent according to what they perceive in their surroundings. In addition, the environment filters agent messages according to whether the recipient is within the interaction area (see Section 4.3.8). Perceptions and filtered agent messages are then delivered to the corresponding agent by the Communication Medium component. This process is performed back and forth at runtime.

The Communication Medium component supports the Madkit setup in running the simulation: it helps to pass communication messages during a simulation cycle, whereas Madkit controls the time steps, sets the simulation parameters, and introduces unexpected changes in the simulation run. The simulation details are explained from the following section on.

5.2 The simulated ecosystem

For the experiments, a hypothetical, simple, yet realistic ecosystem is considered in which a chain breeding environment consists of six sets of five agents each. One of these sets acts as the pool for choosing the top consumer, another one acts as a pool for selecting the basic producer, and the remaining four sets function to
5.3 Experimental setup

select the intermediaries. Furthermore, hypothetical resources are created so that at each link the same resource was offered and evaluated. Therefore, chains of five links (composed by six agents each) may emerge, which is empirically considered complex enough to test the prototype. This setting is created by the Madkit setup component depicted in Figure 5.1.

The resources have a numeric value randomly assigned which the consumers use to evaluate offers. Their resource evaluation function also considers a private criteria for evaluating offers (see Section 4.3.4) which for the sake of simplicity is the same resource value. Therefore, at each link all consumers tend to prefer the producer whose resource has the highest value among its competitors. As a result of this and the fact the producers tend to prefer the consumer who grants the highest normalised evaluation, there is a tendency to one link for each of the five resource types. Creating this tendency ensured that for each resource there is only one link all agents would like to participate in. Therefore, the expected number of existing links at all times is five, yet the agents have to converge to it by reciprocal preference. This helped to validate whether the system converged to the expected number of links, see Figure 5.3.

Notice that the emergence of chains is not being evaluated, but the convergence to a number of links which may or may not form a chain. This is because links are the basis for chains, thus the most basic case is used for the evaluation. Further details are given below.

5.3 Experimental setup

In Section 2.2, an unexpected change is defined as an alteration creating a ripple effect altering the whole system, where the produced reaction cannot be appre-
hended by the entities at the micro level. The experiments carried out using the DAEM prototype demonstrate DAEM's capability to find an equilibrium and its ability of resilience by converging back to an equilibrium after introducing an unexpected change.

One simulation corresponds to twenty runs. Each of them consists of allowing DAEM to run for a determined amount of simulation cycles in three consecutive periods. Initially, no unexpected changes are introduced to analyse whether DAEM converges to the expected number of links. The second period starts with a sudden introduction of an unexpected change at each link, causing one randomly-selected producer to improve its resource sufficiently to become the new preferred producer of the link (see Figure 5.3), yet the other agents have to detect it. The third period is similar except for using a different producer randomly selected at each link.

A change introduced at a link creates resource evaluations to vary, changing the preference of the agents thus foraging for different peers in different areas of the environment. This causes further changes because then other agents cannot find their preferred peers thus forage for others and eventually forgetting the preferred ones, causing more preference changes and so on. By doing this at every link at once, the whole ecosystem is being torn down emphasising the effect of the unexpected changes. Yet the ecosystem should adapt to these by converging back to the expected number of links every time.

Empirically, 30,000 simulation cycles are considered as long enough to perceive (both visually and in output data) any behavioural pattern. Nevertheless, DAEM was let to run for 60,000 simulation cycles before introducing any unexpected change. Thus, one run lasted 180,000 simulation cycles in which sets of unexpected changes were introduced at 60,000 and 120,000 simulation cycles.

The rationale for the first set of unexpected changes is straightforward: to test whether DAEM actually converges to the expected number of links after the first set of changes is introduced. The rationale for the second set of unexpected changes consists of testing whether DAEM is still resilient even after two changes, suggesting robustness to detect and adapt to more than two sets of unexpected changes.

5.3.1 Simulation parameters

One experiment consists of a set of simulations where the same set of parameters were used except for one whose value was varied to explore the effects of such variations. This was the memory size, which in Equation 4.1 is determined by the memory factor $s$ and the number of peers interacting for the same resource, i.e. all the available consumers or producers for a particular resource. For the experiments, the number of available producers and consumers was known in advance,
5.3 Experimental setup

Empirically, values for \( s \) between 0 and 5 have shown some significant differences in the perceived (both visually and in output data) behavioural pattern. Notwithstanding, the memory factor \( s \) was varied for each experiment from 0 to 10 at intervals of 0.4, i.e. considering five available agent (either consumers of producers) the memory size used ranged from 1 to 50 at intervals of 2.4 for all agents. For the particular case of \( s = 0 \), a memory size of 1 was used to allow agents to at least remember the last interaction and thus have a reference for resource evaluation.

Moreover, five sets of experiments were run where an additional parameter was varied to explore further the overall behaviour of DAEM. This was the environment size. In Equation 4.3 the environment size is determined by the environment factor \( f \) and the total number of agents inhabiting the environment. Empirically, values for \( f \) between 0 and 4 have shown significant differences in the perceived (both visually and in output data) behavioural pattern. Nevertheless, the environment factor \( f \) was varied between 0 and 8 at intervals of 2. Thus, considering a total of thirty agents the environment size \( es \) ranged from 15 to 240 at intervals of 60. For the particular case of \( f = 0 \), 0.5 was used instead since DAEM requires an environment to work with, i.e. producing an environment size \( es = 15 \), i.e. a grid of 15 × 15.

Other parameters consist of sense radii, which were fixed throughout the experiment sets to simplify the exploration of the variables’ range. These include a smell radius of 16 cells, a sight radius of 8 cells, and a proximity radius of 4 cells. The smell radius was chosen based on the case where the minimum environment size was used, i.e. an environment of 15 × 15 in which they could sense almost all the environment from any point\(^2\). Then it was fixed for all environment size variations. The proximity radius was determined according to the minimum space empirically seen where agents can have a full interactions, i.e. sending an offer and receiving an evaluation. The sight radius was simply determined as half the smell radius in order to allow an agent enough space to use its foraging behaviour, yet avoiding a similar value as the proximity radius so that the sight sense can help to identify agents before deciding to come closer to interact.

The last parameter fixed was the maximum odour strength (see Definition 8 and Equation 4.16). This was set to a quarter of the environment size \( ods_{max} = \frac{es}{4} \) because it was seen empirically that the semi-random foraging behaviour functions roughly the same amount of time than the directed one, thus both are roughly equally used.

In summary, five sets of experiments were run, each of which consisted of twenty-six simulations, which in turn were conformed by twenty runs each giving

\(^2\)This characteristic is used later to simulate a centralised approach.
a total 2,600 simulation runs. The metrics used to validate the output information is explained in the following section.

5.4 Evaluation metrics

This section explains the metrics used for analysing DAEM's behaviour for which three sets of metrics are used for comparing the expected number of links by using accumulated distribution, confidence, and convergence. Moreover, these metrics consider each simulation by the median data of all twenty runs, i.e. a simulation is represented by the median values over time. The median is used for such a purpose because it does not assume any particular distribution in the data thus not letting outliers to affect the central representation. These are explained in the following subsections.

5.4.1 Accumulated distribution

These metrics condense DAEM's overall behaviour and allow one to analyse it for different combination of values of $s$ and $f$. Nevertheless, they do not provide information on how DAEM behaves over time. Therefore, they are used only to highlight those combination of values which require further analysis.

**Accumulated time** is the total number of simulation cycles during which the expected number of links is found, i.e. five, in a simulation. Because the interest is on a high convergence to the expected number of links, the higher the value the better. It gives an idea on the overall system convergence.

**Accumulated number of changes** is the total number of changes in preference links found in between sampling times in a simulation. Because the interest is on a low instability (high stability), the smaller the number of preference link changes the better. This gives an idea on the overall system stability.

5.4.2 Confidence

These metrics provide a more detailed analysis of DAEM's behaviour because they consider time. They are used for comparison of different values of either $s$ or $f$ when the other is fixed. Furthermore, these metrics concentrate on accumulated values thus highlighting any behavioural tendency.

**Confidence** is the accumulated proportion of time (simulation cycles) in which the expected number of preference links is found in a simulation. With this met-
5.5 Experiments on adaptation to unexpected changes

ric the interest is on a high time proportion and how it changes during the simulation. Thus the higher and more stable value the better.

**Stability of changes** refers to the variation of the accumulated number of changes of preference links in a simulation. The interest with this metric is on a low accumulation of changes and how it changes during the simulation. Therefore, the lower and more stable the accumulation is the better.

5.4.3 Convergence

This set of metrics allow an even more detailed analysis of DAEM’s behaviour. They have fixed values of $s$ and $f$ thus permitting a direct comparison between any two simulations. In general, they focus on the convergence to the expected number of links without ignoring the preference changes occurring at the agent level.

**Number of preference links** refers to the actual number of preference links existing at the sampling time. It helps to analyse how the system behaves according to how close it is to the expected number of preference links in the system.

**Number of preference changes** refers to the number of changes in preference that occurred in between sampling times. It is used to determine how much and how often DAEM reacts to preference changes at the agent level especially after facing unexpected changes.

5.5 Experiments on adaptation to unexpected changes

As explained earlier in this chapter, for the experiments both the memory and the environment factors were varied to find a set of values for which DAEM demonstrated a high adaptability to unexpected changes. In the hypothetical ecosystem built for the experiments, five preference links were determined as the expected number to be found in the system at all time. The analysis of the results are presented in a general-to-particular format, i.e. first an overall view of the behaviour across all values of $s$ and $f$, then a deeper analysis on some of the values.

5.5.1 Selecting an optimal memory factor

For the initial analysis of DAEM’s behaviour across the range of $s$ and $f$, a distribution contour graph is used showing a projection of a 3D graph over the $xy$ plane where the $z$-axis is shown as a colour map over the plane. Moreover, contour lines help to visualise surface differences by surrounding areas with higher values on the
Figure 5.4: Accumulated number of simulation cycles.

The figure shows how the accumulated number of simulation cycles is distributed across combinations of both environment and memory factors. The environment factor $f$ ranging from 0.5 to 8 in the $x$-axis, the memory factor $s$ ranging from 0 to 10 on the $y$-axis, and a colour code bar assigning colours to different accumulated number of simulation cycles. The closer the colour is to yellow the higher the accumulation is. Likewise, the closer the colour is to black the lower the accumulation is. The purpose of this graph is to determine for which combination of values for both the memory and the environment factors the accumulated number of simulation cycles to obtain 5 preference links is higher.

As can be appreciated in Figure 5.4, partially delimited by the contour lines in the centre-left area of the graph, there is a cluster encompassing the highest accumulated number of simulation cycles, i.e. higher that 150,000. These values are found between memory factor values of 2.8 and 3.4, and for all environment factor values. This suggests that within this memory factor range it is likely to find the highest convergence to the expected number of preference links. Yet it does not say how stable the system is.

Figure 5.5 presents a distribution contour graph showing the accumulated number of preference link changes in a simulation. Opposing the previous contour...
5.5 Experiments on adaptation to unexpected changes

Figure 5.5: Accumulated number of changes of preference links.

graph, here the interest is on the lowest values of preference link changes. It can be appreciated that for memory factor values below 2 for all environment factors, and for memory factor values above 4 when the environment factor values are below 4, the number of preference changes are notably higher. Yet when looking at the area identified above of a memory factor between 2.8 and 3.2, it can be seen that a memory factor value of 3.2 shows the lowest accumulated number of preference link changes for all environment factors.

Indeed, a memory factor of 4 and an environment factor of 8 actually show the lowest accumulated number of preference link changes, yet this combination of factors is not within those showing the highest accumulated time for obtain 5 preference links during a simulation. As a result, a memory factor of 3.2 seems to be the better value since it shows the highest accumulated probability and at the same time the lowest number of preference changes for all values of the environment factor. Yet, it does not say how the system behaves at runtime. A further analysis is made in the following sections concentrating only on a memory factor $s = 3.2$.

5.5.2 Selecting an optimal environment factor

Having selected the memory factor $s = 3.2$, now it is possible to compare DAEM’s behaviour over time for all values of the environment factor $f$. Figure 5.6 shows a
Experiments On The Adaptive Capacity of DAEM

Figure 5.6: Confidence to obtain 5 preference links.

comparison of the confidence level to obtain 5 preference links during a simulation for different environment factors. The figure clearly shows that for all environment factors there is a period of time in which the confidence tends to go high then suddenly drops to a value below 0.15 except for \( f = 0.5 \) which drops to 0.72 after having reached an initial confidence of 1. After such a period, the confidence of all environment factors recovers and climbs over 0.70 at different paces though.

The particular cases of \( f = 6.0 \) and \( f = 8.0 \) show a slight decline in the confidence just after introducing the first set of unexpected changes at simulation cycle 60,000. Yet for the second set of unexpected changes at simulation cycle 120,000, only the environment factor of 4.0 shows a decline in the confidence suggesting that more than one unexpected change affects its behaviour negatively. From all the environment factors, only when \( f = 0.5 \) DAEM has a confidence of 0.91 nearly from the beginning of the simulation. Moreover, it seems that unexpected changes do not affect its confidence, which makes it unclear whether it oscillates so frequently that unexpected changes are never detected. Thus, the system stability is then analysed.

Figure 5.7 shows a comparison of the stability of the number of changes of preference links during a simulation for different environment factors. The figure depicts three principal behaviours. For the environment factors \( f = 0.5 \) and \( f = 2.0 \) the accumulated number of preference link changes keep raising seemingly at a constant
5.5 Experiments on adaptation to unexpected changes

Figure 5.7: Stability of the number of preference link changes.

linear rate regardless of any unexpected change. This means that even though they have a high confidence to obtain 5 preference links they keep oscillating quite frequently to other values without stabilising, which is not a desirable behaviour.

Another behaviour appreciated in Figure 5.7 is that of \( f = 4.0 \), which after the first set of unexpected changes at simulation cycle 60,000 seems to stabilise. Yet, after the second set of unexpected changes at simulation cycle 120,000, the accumulated number of preference link changes raises suddenly such that it even goes at a higher pace than \( f = 0.5 \) and \( f = 2.0 \). This behaviour is not desirable because it clearly behaves worst when facing more than one set of unexpected changes.

Finally, for the environment factors \( f = 6.0 \) and \( f = 8.0 \) DAEM clearly shows stability after introducing each of the sets of unexpected changes. Although \( f = 8.0 \) seems more stable than \( f = 6.0 \), a more detailed analysis of these two cases is necessary for determining how the system is converging to the expected number of preference links.

5.5.3 Analysis of the system convergence

This section analyses the number of preference links found at runtime and compares it with the expected one, i.e. five links when the memory factor is 3.2. Figure
5.8 reports the number of preference links found at runtime for the environment factor $f = 6$. The behaviour DAEM portrays shows a clearer tendency to the expected number of links and seems more stable when compared against the previous cases.

For instance, during the convergence (cf. latency) period (i.e. the first simulation cycles in which there is no clear tendency) DAEM reports in increasing number of links until it reaches 8 links at cycle 2,400, then coming down to 7 and 6 links at cycles 4,100 and 6,000 cycles, respectively, with two small oscillations. Then from cycle 12,700 it remains in 5 links with occasional variations to 6 links. The convergence period occurs because the agents are moving mainly randomly and are still exploring the environment. In principle, the larger the environment is the longer this period lasts and the less frequent the agent interactions occur.

Additionally, DAEM seems to stabilise once it converges to the expected number of 5 preference links with some sporadic short oscillations in which DAEM reported 6 links. Furthermore, it becomes notorious that DAEM seems to react to the unexpected changes just after they were introduced to the system. After oscillating due to the first set of unexpected changes, DAEM converges back to 5 links at 79,600 and keeps reporting the same number of links until cycle 119,000, where it reports 6 links for the next 200 cycles. Then the system reacts to the second set of unexpected changes at cycle 120,500 and keeps on oscillating sporadically mostly between 6 and 5 links, and sometimes 4 links. In spite of oscillations, using an
environment factor $f = 6$ DAEM reacts to unexpected changes and recovers from them showing a stable behaviour throughout the simulation.

Figure 5.9 reports the number of preference links found at runtime for an environment factor $f = 8$. DAEM shows a behaviour with an apparent even clearer tendency to the expected number of links and seems even more stable than the previous case. For example, the convergence period reaches only 7 links (not 8 as in previous case) at cycle 3,900 and it tries to stabilise there for a short while, reporting 6 links at cycle 7,000 lasting 2,100 cycles, and then going back to 7 links. Eventually, at cycle 10,800 DAEM reports 6 links and seems to stabilise there with occasional oscillations to 7 and 5 links. Finally, at cycle 29,000 it reports the expected number of links, i.e. 5 and it stabilises there, but with sporadic oscillations between 5 and 6 links.

In addition, there are two things to notice during the convergence period. First, it reaches its peak at 7 links, one link short of that reported for the same period with an environment factor $f = 6$. This apparent behaviour is the result of an increase of scatter of the reported number of links in each simulation run, reporting from 2 links up to more than 10, making 7 links the median for the simulation at the peak of the convergence period. Such a dispersion in the reported number of links is due to such a large environment size causing a) the agents to struggle to find others for interacting, hence a low reported number of links; and b) once they find
a few peers, they struggle to keep finding more for comparing the resources being offered thus creating preference links without considering most of the offers, hence a high reported number of links. Secondly, DAEM converges consecutively to two different links, first 6 then 5 links. This behaviour was not seen in the previous case. It is caused by the agents requiring more time to interact with others due to the environment size.

Furthermore, DAEM seems slightly more stable throughout the simulation than the previous case, specially after each set of unexpected changes. For instance, the first set of unexpected changes is detected at cycle 60,700 and then DAEM reports 6 links almost continuously until cycle 69,200. Then it converges back to 5 links and remains so except for two short oscillations to 6 links before the second set of unexpected changes is introduced. The latter is detected at cycle 122,700 causing two short oscillations to 6 links and then it comes back to report 5 links for the remaining of the simulation. Contrasting with the previous case, here unexpected changes are not detected immediately after they are introduced, but some hundred simulation cycles afterwards. Therefore, it is not clear whether DAEM reacts late to the changes or whether they are not detected at all.

Finally, when using an environment factor \( f = 8 \) DAEM reacts to unexpected changes and remains more stable throughout the simulation. Yet it is not clear whether the stability is either due to a good adaptation process or to not detecting properly the unexpected changes.

### 5.5.4 Analysis of preference changes

This section analyses how the preference changes are affected during a simulation, specially just after introducing unexpected changes to the system. Figure 5.10 illustrates the preference changes reported at runtime for an environment factor \( f = 6 \). During the first 100 simulation cycles, DAEM reports 4 preference changes. Then it oscillates between 1 and 2 changes for the next 1,700 cycles dropping to 0 only once. After that period, DAEM reports between 0 and 1 changes until reaching cycle 8,200, from which it reports 0 preference changes except for some sporadic occasions in which it was reported 1 change, and just after the unexpected changes were introduced.

Furthermore, after introducing unexpected changes into the system, DAEM immediately notices them as can be appreciated by the raise of the reported preference changes to 2 and a short period in which the changes oscillate between 0 and 1 changes, before going back again to 0 changes.

This environment factor makes DAEM to report mainly 0 preference changes throughout the simulation. This behaviour is due to the environment size balancing
between not so frequent agent interactions yet still close enough to detect unexpected changes immediately after they occur. This result combined with the one shown in Figure 5.8, support this value of the environment factor as one which produces the desirable behaviour of DAEM, i.e. converging to the expected number of links, reacting promptly to unexpected changes, and then converging back to the expected number of links.

Figure 5.11 shows the preference changes reported at runtime for an environment factor $f = 8$. The behaviour DAEM shows here differs slightly from the previous case. During the first 100 simulations cycles DAEM reports 2 preference changes. Then it oscillates between 0 and 1 changes for the next 7,200 cycles. After that period, DAEM reports 1 change in only one other occasion: just after the second unexpected change.

In can be appreciated in the same figure that DAEM does not report preference changes as often as in the previous cases, which makes it seem more stable. Stability is a desirable property, yet changes in agent preferences should be detected when introducing unexpected changes in the system. However, only one preference change was detected after introducing the unexpected changes. This suggests that the environment size is so large that preference changes are hardly detected even after introducing unexpected changes.

The behaviour DAEM shows here occurs because of the environment size
which makes the agents to explore more surface to find others to interact, thus interactions and hence any preference changes occur less frequently. As a result, when unexpected changes are introduced to the system the agents cannot detect them easily. Figure 5.11 shows that unexpected changes are hardly noticed when using an environment factor \( f = 8.0 \), which is not a desirable behaviour of DAEM.

### 5.5.5 Experiments summary

Throughout the analysis of these results, it is shown that DAEM indeed converges to the expected number of 5 preference links whilst being stable enough and resilient to unexpected changes only when using a memory factor of 3.2 and an environment factor of 6. This behaviour complies with the definition of adaptation presented in Section 2.2 and with the underlying formalisms presented in Section 4.3. Furthermore, it proves true the hypotheses H1 and H2 described in Section 1.3 supporting one of the contributions of this thesis: DAEM.

### 5.6 Comparing against a simulated centralised approach

A simulated centralised approach for ecosystems is compared against DAEM. Such an approach uses a simulated central discovery mechanism in which peers are
sought in order to interact as opposing to the environment.

In ecosystems domains, it is common to find the notion of a centralised directory for finding peer to interact. For instance, consider a digital service ecosystem in which a central service directory (cf. environment) is used for finding services either to consume them or for composing complex services (cf. [Briscoe and De Wilde, 2006, Fragidis et al., 2007]). Ideally, a service consumer may see all available services in the directory and decide which of them to consume according to own standards, e.g. maximum price, reliability, etc. Moreover, a service consumer may subscribe to a service to receive notifications of any update the service may have. Furthermore, a service consumer may subscribe to the service directory as well to obtain notifications of any new service being available.

However, typical service providers do not make their services available to all member of the central directory, but only to a few consumers due to, e.g. policies. Technical difficulties such as service definition may restrict a service to be considered for consumption or composition. This motivates the service consumer to constantly search for available services. In summary, a typical service provision hardly achieves the ideal centralised approach, yet still it is commonly used.

DAEM was used to simulate a pragmatic version of the centralised approach due to its resemblance to a constant search for peers when using a small environment (i.e. environment size $es = 15$). This makes all agents to virtually perceive all other agents at once similar to a central directory. Due to the high frequency of unordered interactions a small environment induces, a memory size of 60 (i.e. memory factor $s = 10$) was used to give the agents the opportunity to forget the minimal number of interactions, not guaranteed though.

### 5.6.1 Expected number of preference links

Figure 5.12 shows the number of preference links found at runtime for the simulated centralised approach. Such an environment size makes the agents to perceive virtually all the others from any position. As a result, the agents cluster immediately for interacting. Nonetheless, there is no clear tendency to the expected number of preference links, i.e. five. Throughout the simulation it keeps on oscillating between 4 and 5 links, and 3 and 6 links in rare occasions. Moreover, there is no clear reaction to the introduced changes due to the high frequency of interactions induced by such a small environment.

### 5.6.2 Preference changes

Figure 5.13 shows the preference changes reported at runtime for the simulated centralised approach. As can be seen, mainly between 10 and 14 preference
changes occur throughout the simulation. Moreover, at the beginning of the simulation and just after introducing unexpected changes there are considerable drops in the reported number of preference changes.

For the first 100 cycles DAEM reported 45 preference changes due to the agents suddenly “meeting” others. Afterwards, the preference changes drops to 2 then goes to the range $[10, 14]$. Similarly, after introducing the first set of unexpected changes the number of preference changes rises to 19, then drops to 3 before recovering and continuing within the range $[10, 14]$. Likewise, after introducing the second set of unexpected changes the preference changes rise to 19, then drops to 1 just before returning to the aforementioned range.

The first raise in preference changes being reported occurs due to the agents initially not knowing others for interacting. The other two raises happen because of the unexpected changes triggering changes which produce situations not contained in the agents’ memory. However, the drop in preference changes in the three cases occur due to the agents’ memory begin still filled with information about the new resource situation. Once the memory gets saturated, it keeps on changing preferences due to their proximity in the environment.
5.7 Final remarks on the experiments

Table 5.1 shows a summary of all the parameters used for which DAEM converges to the expected number of links and stabilises there even after facing unexpected changes. In particular to such a simulation, DAEM shows a confidence of 0.75 just before the first unexpected change (see Figure 5.6). Immediately after that, the confidence drops to 0.72 and recovers within the next 8,700 simulation cycles until
reporting 0.82 just before the second unexpected change.

At the end of the simulation, DAEM reports a confidence level of 0.86 demonstrating its resilience to unexpected changes. Moreover, Figure 5.7 shows that using these parameters the system is stable yet it still detects and reacts to both preferences changes and unexpected changes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment factor</td>
<td>6.0</td>
</tr>
<tr>
<td>Memory factor</td>
<td>3.2</td>
</tr>
<tr>
<td>Proximity radius</td>
<td>4</td>
</tr>
<tr>
<td>Sight radius</td>
<td>8</td>
</tr>
<tr>
<td>Smell radius</td>
<td>16</td>
</tr>
<tr>
<td>Maximum odour strength</td>
<td>$es/4$</td>
</tr>
</tbody>
</table>

Table 5.1: Parameter summary for best DAEM results.

The experiment results presented in this chapter demonstrate the following points: a) a proof of concept for the approach followed in this thesis; b) the set of parameter values necessary for the prototype to perform according to the underlying properties of the model; and c) the suggestion that it can perform better than typical, realistic centralised approaches in terms of convergence and adaptation to unexpected changes.

Finally, the results prove true the hypotheses H2 and H3 presented in Section 1.3 and support DAEM as one of the contributions of this thesis.

5.8 Discussion

DAEM is capable of adaptation to unexpected changes, as commonly occur with CASs, since DAEM is based on CAS adaptation properties. Yet, is DAEM capable of emergence and self-organisation as typical of CASs? An emergent characteristic seen in any run when using a memory factor of 3.2 and an environment factor of 6 is depicted in Figure 5.14. Agents are represented by tiny dots enclosed by three concentric circles representing the perception radii. Consecutive dots represent mark trails left by the agents. These help to visually figure out the direction agents are travelling.

The emergent behaviour consists of common places in the environment where agents with similar interests come together to interact. That is, one or two resources are offered in that area creating a niche, which makes interested agents to return regularly to that spot. However, these points are not fixed. They appear, disappear and sometimes they move to other areas in the environment. The number of agents converging to this point changes each time and they hardly synchronise to meet
5.8 Discussion

(a) Common point of interaction.  
(b) Moving away from interaction.  
(c) Moving to another point of interaction.

Figure 5.14: DAEM simulation snapshots.

there. Nevertheless, it still commonly occurs.

For instance, Figure 5.14(a) shows in the lower right area how some agents form a small cluster to interact. Figure 5.14(b) depicts in the lower right area how the agents move away to find other peers to interact with. Since agents move semi-randomly from time to time, they are not synchronised, thus when they come back to interact they do not arrive at the same time. This is the cause for niche dissolution. However, sometimes they catch the smell of the agent with the resource they are looking for, i.e. a mark trail. Therefore, they follow it and move the niche to another area in the environment. This can be appreciated in Figure 5.14(c). Niche creation is not defined nor designed in DAEM. It emerges from agent interactions and clearly suggests that DAEM does exhibit an emergent behaviour.

Additionally, the convergence to the expected number of preference links even after encountering unexpected changes implies a re-organisation of the system
structure driven by changes in interaction preferences at the individual level. DAEM is defined in this way on purpose and the experimental results confirm such a behaviour. Therefore, DAEM is also capable of self-organisation. It is not the purpose of this thesis to perform an analysis on self-organisation itself, which is interesting doing but is left as material for future work.

In Section 3.1 an adaptive MAS was defined as an MAS capable to self-modify its structure and internal organisation as a reaction to environmental changes. As seen in the experiments, DAEM complies with this definition which suggests a one-way connection between being an adaptive MAS and a self-organising system. That is, an adaptive MAS is that which exhibits adaptation by a self-organising process. However, notice that this does not imply that a self-organising system is also an adaptive MAS.

In particular to the experimental results, the number of preference links was fixed in order to know whether DAEM converges to an expected value. Indeed, now that it is known with which set of parameters DAEM performs better, the straightforward exploration path is to test it in a less controlled setting. For instance, using different evaluation functions per agent will allow the emergence of more than one preference link per resource type, which in turn might permit the emergence of more than one chain; another example of potential tests is that of using data from an existing ecosystem to feed the system and tests whether it can make estimations of future situations; or use those data to fine tune the behaviour according to an existing ecosystem. All these exploration paths are out of the scope of this thesis and as such are left for future work.

Summary

This chapter presented the results of the experiments carried out to test the DAEM capability to adapt to unexpected changes. The changes consist of unexpected resource innovations triggering preference changes, using different sets of parameters. Furthermore, a comparison with a centralised resource discovery approach (cf. a central service directory) was presented which clearly shows that DAEM adapts to unexpected changes and stabilises better than the centralised approach. The results function as a proof of concept of the thesis approach. Moreover, they confirm validity of three of the hypotheses of the thesis, H1, H2 and H3 described in Section 1.3 and thus support the claim of this thesis that DAEM represents a valid contribution to knowledge.

Next chapter presents an execution context for DAEM, showing an architecture and practical examples describing how DAEM can support adaptation to unexpected changes in practical ecosystem domains.
Chapter 6

Using DAEM within an Execution Context

This chapter presents a DAEM software architecture for supporting ecosystem domains. It is based on the definitions derived in Section 4.3. An execution context is presented as a way to exemplify the use of the architecture in two practical examples: a business ecosystem and a digital service ecosystem. The examples show how DAEM incorporates actions and decisions from the physical world into the dynamics of the ecosystem. In turn DAEM suggests who to interact with in order to help to cope with unexpected changes occurring in the ecosystem.

The chapter begins with the description of the DAEM architecture and related components in Section 6.1. Then the DAEM layer is presented along with other supporting components as part of a business ecosystem execution context in Section 6.2, followed by an execution example in Section 6.3. Likewise, a digital service ecosystem execution context is presented in Section 6.4 followed by an execution example in Section 6.5. Finally a discussion is presented in Section 6.6 before closing the chapter with a summary.

6.1 DAEM architecture

The DAEM architecture is informed by the formalisms of DAEM itself which already defines the essential elements and certain functionality, making the translation to software components straightforward. Thus, no methodology for MAS development was necessary for this process.

As defined in Chapter 4, DAEM comprises three main elements namely resources, agents, and the environment. The resources represent the information being exchanged by the agents through the environment. Therefore, only the agents and the environment are considered as components in the architecture, whereas
Using DAEM within an Execution Context

Figure 6.1: The DAEM architecture shows the environment as a separate entity from the agents, and mediating their interactions.

the resources are exchanged within the interactions. Figure 6.1 depicts the architecture.

Definition 11 defines an agent with the tuple \((aid, R, A, M)\) where

- \(R\) refers to the agent’s resources and a resource pointer. This is translated as the resource manager component. Moreover, \(R\) contains a pointer to the resource being foraged for which basically dictates the agent behaviour. Therefore, a separate behaviour component represents such a pointer;

- \(M\) is the agent memory which stores interaction related information. It is represented by two components: an interaction memory and an interaction manager; and

- \(A\) contains the agent actions which is represented as an actuator component. Likewise, the tables representing the sensed information from the environment are received by a sensor component (see Definitions 6, 7, 9, and 10).

Because of its nature, the agent identifier \(aid\) is not considered as a component. Similarly, Definition 8 describes the environment with the tuple \((es, Ads, Mks, D)\) where

- \(es, Ads, Mks\) are in essence the registries of the virtual space the agents inhabit. Thus a virtual surface component is envisioned encompassing other three components namely an agent registry, a mark registry, and a map;
• $D$ contains the radii supporting agent perceptions and constraining agent interactions. Therefore, they are directly devised as a perception manager and a communication manager respectively; and

• a Dynamics component controlling the environment reflections $\rho^i$ (Definition 4), and the environment processes namely agent displacements $\omega(i, epos, dest)$ (Equation 4.7), and mark evaporation $\tau(Mks)$ (Equation 4.17).

Repositories, supporting platforms, transmission media along with further details of the above components are presented in the following subsections.

6.1.1 Resource manager

The resource manager portrays the resource set $R$ in Definition 11 and functions as a manager of the resources the agent offers and consumes. It evaluates received resource offers according to internal standards (cf. Section 4.3.4) and it determines the added value in an resource to offer. Moreover, the resource manager operates as a scheduler for assisting the behaviour component in resource foraging, see Section 6.1.4 further on.

6.1.2 Interactions

A DAEM agent has two components dealing with agent interactions: the interaction memory and the interaction manager. The former represents the element $M$ according to the agent definition and manages the interaction tables storing the details of agent interactions namely resource evaluation, environment location where the interaction occurred, and the identifier of the agent participating in the interaction.

Moreover, the interaction memory keeps one interaction table per resource existing in the resource manager in order to have dedicated memory tables according to the resource an interaction is about. Furthermore, the interaction memory implements the forgetting mechanism by limiting the size of the interaction tables, thus removing the information of the oldest interaction whenever a new interaction is stored. See more details in Definition 2 and Section 4.3.3.

The interaction manager works as a gateway for all agent-to-agent communications. It portrays the actions $sendof$ for sending resource offers and $sendev$ for sending resource feedback in the set $A$. Incoming resource feedbacks are passed to the interaction memory for their storage, incoming resource offers are passed to the behaviour component for their evaluation. Outgoing resource offers and feedback of received offers are sent through this component, i.e. agent-to-agent communication passes through this component, which interacts with the Communica-
tion Manager on the environment side due to being in charge of the actual delivery of messages to other agents according to their proximity (see details further on).

6.1.3 Actuator and sensor

A DAEM agent contains two components for acting in the environment itself: the actuator and the sensor. The former carries out some actions defined in set $A$ namely $\text{walk}$ for moving in the environment, and $\text{subscr}$ and $\text{unsubscr}$ for registering and unregistering resources to the environment. It is not mandatory for an agent to register all of the resources it offers or consumes. However, not doing so will avoid other agents to perceive what the agent is interested in and what it offers.

The sensor works as the receptor of perception messages coming from the environment. These perceptions correspond to receiving a perception pair and various perception tables namely $\text{apos}$ for the agent position, $T$ for the proximity sense, $G$ for the sight sense, and $L$ for the smell sense as defined throughout Section 4.3. Notice that agents do not decide what to perceive, they simply receive perception messages about all their senses. It is up to the agent to decide what to do with that information.

6.1.4 Behaviour

The behaviour component corresponds to the agent intelligence based on both the direct and semi-random foraging behaviours explained in Section 4.3.10. It basically decides what to do next based on the perceptions received from the sensor, the last interactions recorded in the interaction memory, and the next resource to forage for according to the resource manager. Once a decision has been taken, this component instructs the actuator to perform an action on the environment and/or directs the interaction manager what to communicate to a particular agent. Notice that the agent who provided the best feedback to an output resource or the agent who has offered the best resource as input, is typically referred to as the best agent.

Algorithm 6.1 shows how the behaviour component functions. It receives a list of resource offers from the interaction manager and evaluates them. It sends the list of evaluated offers to the interaction memory for storage, who then sends back a list of the most recent interactions (both received evaluations and evaluated offers) and a reference to the best interaction for each resource. If one of the interactions was about the resource being foraged for, two situation could happen: a) if the semi-random foraging behaviour is activated, then it is deactivated and the behaviour component asks the resource manager for the next resource in the schedule to forage for; or b) if the list shows an interaction whose evaluation or feedback is the best one in the memory, then the behaviour component simply asks the resource
Algorithm 6.1 Agent behaviour.

while alive do
    obtain list of recent resource offers
    evaluate received offers
    obtain full list of recent interactions
    if interacted about resource being foraged for then
        if semi-random foraging is activated then
            deactivate semi-random foraging
            select next resource to forage for
        else if obtained the best evaluation then
            select next resource to forage for
        end if
    end if
    obtain perceptions
    if potential consumers are near then
        send resource offer to all near consumers
    else if best agent of foraged resource is seen then
        move towards best agent
    else if at best interaction place of foraged resource OR semi-random foraging is active then
        activate semi-random foraging
        if producer/consumer of foraged resource is seen then
            move towards closest producer/consumer at sight
        else if mark of foraged resource is smelled then
            move towards strongest-odour mark
        else
            move randomly
        end if
    else
        move towards location of the best last interaction of foraged resource
    end if
end while

manager for the next resource in the schedule to forage for. Otherwise no changes are made to the current foraging behaviour nor target.

Next, perception information is obtained from the sensor component and used to determine the agent’s destination in the environment. Such a destination is selected according to set of consecutive conditions: a) if potential consumers are found in the vicinity (proximity area), then send them resource offers. This action implies not moving anywhere in an attempt to remain close to them whilst waiting for feedback; else b) if the best agent for the foraged resource is at sight, then move towards it; else c) if the semi-random behaviour is already activated or the agent is currently at the location where the last interaction with the best agent occurred for the resource being foraged, then enter the semi-random behaviour explained below; otherwise
Using DAEM within an Execution Context

d) simply move towards the location where the last interaction with the best agent happened for the resource being foraged.

The semi-random behaviour is activated when arriving at the place where the last interaction with the best agent occurred, but such an agent was not found. From this moment on, the agent moves away from such a location by following its sight and smell senses, thus it follows the trail of something of the agent’s interest: a) first, if an agent with the resource being foraged for is at sight, then move towards such an agent; else b) if a mark referring to the foraged resource is smelled, then move to the mark with the strongest odour; otherwise c) move randomly.

6.1.5 Virtual surface

The virtual surface comprises three components which maintain a record of what is happening in the environment all the time. Such components are two registries for storing information of agents, their resources, and marks, and a map for locating agents and marks on the virtual torus.

The agent registry represents the description set $A_{ds}$ by maintaining references of all the agents registered to the environment as well as their resource descriptions. Thus, when an agent wants to enter the ecosystem, it has to register with action $\text{subscr}$ and has to specify its related resource descriptions. In addition, any resource no longer produced has to be unregistered by performing action $\text{unsubscr}$. These actions are received by the map component, then passed to the agent registry for processing. See Definition 5 for details on registration to the environment.

The mark registry represents the mark description set $M_{ks}$ by maintaining all marks existing in the environment. It uses information from the agent registry to create new marks. Moreover, the agents do not interact with this component because they do not decide whether to drop marks or not in the environment as defined in Equation 4.16. This is an environment property, thus managed by the environment itself.

The map component represents the surface itself by keeping a location reference of all the agents and marks in the environment. Such references point to both the agent and the mark registries. Consequently, it collects displacement instructions from the agents, i.e. actions $\text{walk}$. Registration-related actions are delegated to the agent registry, yet the map is notified whether a new agent needs to appear in the map. Thus, a reference to the new agent is randomly placed in the surface. On the other hand, if an agent unregisters itself, then its reference in the surface is simply removed. Additionally, the map is used by the perception manager to query locations for calculating distances on the torus, and ultimately for calculating agent perceptions.
6.1.6 Mediation and awareness

There are two main components that deal with mediation and awareness: the perception manager and the communication manager. These are based on the senses defined in Sections 4.3.6, 4.3.8, and 4.3.9 namely positioning, proximity, sight, and smell.

The perception component calculates the perceptions of all agents registered to the environment using the radii set $D$. This is done by querying the map component to obtain the current position of all agents and all marks in the environment. With this information it calculates what each agent perceives according to the different radii. As a result, the perception manager generates three different lists one for each of the senses namely a proximity list $T$, a sight list $G$, and a smell list $L$ as well as the position pair $apos$. The collected information is then sent to the respective agent regardless of whether it has performed an action or not.

The communication manager functions as a filter of agent communication in such a way that it only permits the message passing between those communicating agents within their proximity area. To obtain this information, it consults the last generated proximity list $T$ of the message sender to the perception manager. If the agents are within the proximity area, then the message is delivered. Otherwise, the message is simply discarded.

Such an environment mediation complies the interaction mediation supporting level of MAS environments [Weyns et al., 2007]. Moreover, by limiting agent communication, local and dynamic interactions are encouraged which are fundamental for building ecosystems [Green and Sadedin, 2005].

6.1.7 Dynamics

The dynamics component controls how the virtual surface is updated by working on the map component and the mark registry; it controls agent consecutive displacements on the torus according to the agent’s final destination and using the equations defined in Section 4.3.7. Moreover, it directly manipulates the mark registry in order to manage the life cycle of marks, i.e. creation, gradual evaporation, and deletion. Additionally, it regulates the timing of the whole environment by updating the map according to agent actions at fixed intervals, and by allowing the perception component to query the map after agent actions have been incorporated.

Algorithm 6.2 shows how this component functions. First, it prevents querying on the map in order to update the environment according to agent actions. Then the dynamics component manipulates the map to process all agent actions received up the current moment: if it is a displacement action ($walk(dest)$), then it re-locates the agent to a position closer to the desired destination $dest$; otherwise it delegates the
Using DAEM within an Execution Context

Algorithm 6.2 Environment dynamics.

```
loop
  lock the map for avoiding the perception manager to query the map
  for all actions received up to this moment do
    if action = walk(dest), then
      move agent i one position closer to dest in the map
    else {that is, either subscr or unsubscr}
      delegate action to the agent registry
    end if
  end for
  for all marks in the mark registry do
    reduce mark odour strength one unit
    if a mark has no odour left then
      remove mark from both the registry and the map
    end if
  end for
  for all agents in the map do
    create a new mark at the current agent position
  end for
  allow the perception manager to query the map to calculate perceptions
end loop
```

action to the agent registry because it is related to registration.

The rationale for the dynamics component to perform these tasks on the map is to maintain the displacement pace and logic, e.g. the “diagonal” displacement in the environment (see Equation 4.10), and to keep a control on the logic when other components operate and query the map.

Then the dynamics component manipulates the mark registry by reducing the odour strength to all marks in the environment. Then, those marks with no odour left are removed from the map. Afterwards, a new mark with the maximum odour strength is created at the current position for each agent in the environment. Then, the perception manager is allowed again to query the transformed map.

6.1.8 Repositories, supporting platforms and transmission media

The DAEM architecture in Figure 6.1 depicts three components additional to the environment and the agents: repositories, supporting platforms, and transmission media. Their implementation details are out of the scope of the thesis. Yet the purpose of these components is to emphasise that agents and the environment are completely independent entities.

The role of the repositories consists of providing persistence to the supported components independently, i.e. a repository for each individual agent and another one (or a set of them) for the environment. The repository configuration (if any)
depends entirely on the supported component.

The supporting platform refers to the set of settings to run the supported component, such as the setting in the operating system or whether the component is distributed over a network (cf. the environment in [Briscoe and De Wilde, 2006]). Yet again, this depends on the supported component.

The transmission media refers to the communication channels between the environment and the agents, i.e. how the messages, actions, and perceptions are transmitted to the recipient. These could be handled by only one transmission medium or by individual channels depending on the implementation conditions and considerations.

6.2 A business ecosystem: an example

This section introduces a business ecosystem as an execution context in which DAEM could be applied to. A discussion on the practical utility if it were implemented is presented in Section 6.3.7. The term service is used in this section and in Section 6.3 to generalise what a company produces/offers, i.e. either a service or a product.

A business ecosystem comprises a set of interacting companies [Moore, 1993]. Therefore a business ecosystem supporting software needs to consider the interactions between and among the software systems supporting the individual companies [Fragidis et al., 2007]. The reference model in Figure 4.4 is used as a basis to support business ecosystems. Three levels of abstraction are envisaged: the DAEM level, the ecosystem domain supporting system, and the physical world being represented. These are translated to a business ecosystem supporting system as follows.

Initially, two conceptual layers are assumed to exist in an already supported business ecosystem: a) a business layer where service descriptions (cf. resource descriptions in Section 4.3.1) and business rules exist. This layer is the “physical world” represented by the supporting system; and b) an agency layer where a set of software agents (possibly arranged as an MAS) supports and performs the transactions of the business layer. The agency layer is the equivalent to the ecosystem domain supporting system.

Then the DAEM level is simply seen as a layer where the ecosystem elements of DAEM are played out, updates from the agency layer are captured here and incorporated in the adaptation process. This layer is based on the DAEM architecture shown in Figure 6.1.

Figure 6.2 shows the execution context comprising many companies interacting at three conceptual layers. These are explained further in the following subsections.
6.2.1 Business layer

The business layer is where business rules exist and where companies interact by fulfilling physical services such as the actual delivery of goods. In this layer the users interact with underlying software systems by performing tasks or giving them instructions to perform automated tasks.

Companies specify their services here and propagate the service descriptions to the underlying layer for carrying out transactions and service execution in the agency layer. Actions and decisions made in this layer are sent to the DAEM layer for inclusion in the ecosystem dynamics.

6.2.2 Agency layer

The agency layer consists of software agents, typically arranged in an MAS, undertaking diverse tasks to support a company. The MAS interacts and negotiates with another MAS of a different company in order to fulfil services and to engage in transactions.

In case a transaction requires the user’s attention, e.g. for requiring an authorisation, the MAS sends the issue to the business layer for further consideration. Additionally, it receives service descriptions from the business layer.

6.2.3 DAEM layer

The DAEM layer is where MASs from the agency layer are projected as a single agent in order to participate in the ecosystems. Thus participating companies have...
their individual agent in DAEM for finding partners and customers whilst coping with changes happening in the business ecosystem.

Service descriptions coming from the agency layer are received by the corresponding agent for service offering (cf. resource offer) in the business ecosystem. New customers (cf. consumers) and suppliers (cf. producers) are suggested to the agency layer for carrying out the actual transactions and service fulfilment.

### 6.2.4 Representatives of the business ecosystem

The related representatives in the three layers constitute an interacting component in an open, collective system forming the business ecosystem. Figure 6.3 shows how any company $N$ interacts with any other company $M$ in the ecosystem. A company’s system is delimited by the company boundary itself. The same figure depicts the representatives: various users at the business layer, diverse software agents and legacy systems at the agency layer, independent agents but only one shared environment within the DAEM layer.

Notice that the figure only shows two companies for the sake of simplicity. Yet there could be more than two companies in the execution context, all of the sharing the same environment within the DAEM layer. The following subsections explain the different representatives within the figure and how they interact.
Users and business rules

The user and the business rules within a company represent the business layer. Typically, the users conform the company’s “know-how” and fulfil services such as designing products or providing consultancy. Furthermore, they take decisions when necessary by following the business rules.

Additionally, users interact with system interfaces for providing information to automate a task, or instructing the underlying system on specific operations such as solving conflicts or authorising activities.

Software agents and legacy systems

The software agents represent the agency layer within the company. They negotiate and engage in transactions with other company agents. Moreover, agents possibly in the form of an MAS manage supplies and require services from partner suppliers, and operate on legacy systems providing additional support to the business layer. Such legacy systems, if present, vary from company to company, and depend on the industry and the company size.

When some tasks require the user’s intervention, software agents notify the relevant user in the business layer and ask for their input. Otherwise, software agents simply inform the status of their activities. Such notifications maybe combined with the legacy system outputs, yet this depends on how they are configured.

Moreover, software agents interact with their representative in the DAEM layer and try to engage in interactions with another company’s agents according to the suggestions made by DAEM for a potential partner. In addition, any instruction coming from the users in the business layer (possibly a new business rule) is passed to the DAEM layer to be incorporated. Such instructions could refer to avoid a specific partner due to bad experiences, or to prefer a specific partner because of a recommendation.

Agents and the environment

The agents and the environment represent the DAEM layer as mentioned earlier. They receive service descriptions from the agency layer in order to find potential partners and customers and detect changes in the business ecosystem. Moreover, the agents could also receive instructions from the agency layer (possibly coming from the business layer) to avoid interactions with a company, or to prefer a supplier over another one.

Once these instructions are incorporated, evaluations of service offers trigger changes which propagate to the rest of the business ecosystem. As a result, pref-
When a car manufacturer is designing a new car, many partners are needed to provide all the different parts to assembly. For each of the parts there could be various suppliers to choose from depending on how they can fulfil the requirements whilst meeting standards. Likewise, each part supplier needs other partners to provide the required components to produce a specific part, and so on.

One of the car parts is the instrument panel (IP) composed by many other components such as microcontrollers. This example concentrates on an IP designer company looking for a provider of microcontrollers matching certain design specifications. Therefore, the IP designer company subscribes to the DAEM-based ecosystem hosted by the car manufacturer. For the sake of simplicity, the IP designer company and related supporting systems are called $p$.

Initially, company $p$ requests entry to the DAEM environment and a new agent is assigned to it. Agent $p$ receives the description of the services the company offers as well as the description of the services the company requires to produce IPs. The details of this initial setup is out of the scope of this thesis. The intention here is only to show that when a new agent is created the company needs to feed it with the company’s service details before setting it loose in the ecosystem.

Agent $p$ then subscribes its service descriptions to the environment by performing action `subscr` on the Map component, which delegates this task to the Agent registry. Then a reference to the agent is placed in the Map itself and the Dynamics component uses the new information to produce related marks in the environment.

### 6.3.1 Foraging for suppliers

After registering to the environment and being assigned to an initial random location, agent $p$ starts foraging for input services (cf. input resources), i.e. microcontroller suppliers.

Initially, agent $p$ will forage randomly in the environment. Eventually it will detect, e.g. three sets of marks in the environment informing about the services other agents require and offer. As shown in Figure 6.4, agent $p$ smells trail $m$ belonging to an agent offering car seats; trail $n$ belongs to an agent requiring laser units; and trail $q$ belongs to an agent providing microcontrollers. Because the later is what agent $p$ is foraging for, it follows trail $q$ until coming to an interaction reach.
In terms of component interactions, the agent walks in the environment by having the Actuator, coordinated by the Behaviour, interacting with the Map on the environment side. The agent’s position on the Map is updated with every move the agent performs. Moreover, after the Dynamics component in the environment updates the Map according to all the actions received, the Perception manager queries the Map to calculate each agent’s perception. This way, agent \( p \) eventually detects through the Sensor the trails depicted in Figure 6.4. This information is fed into the Behaviour to determine where to move next in the environment.

### 6.3.2 Service evaluation and feedback

Once the agent \( q \)’s Sensor informs the Behaviour about agent \( p \) being within the proximity area, an agent interaction occurs (see Figure 6.5(a)). Agent \( q \) offers the service \((mcr-ctlr, val)\) to agent \( p \) where \( mcr-ctlr \) represents the service description and \( val \) is the service value. In this example the former consists of a microcontroller specification and the latter is the related price.

As shown in Figure 6.5(b), agent \( p \) evaluates the offer which results in an absolute score of 400 according to internal standards. The value itself does not mean anything special, it is simply a value for further internal comparisons. Then it turns out the evaluation is the greatest agent \( p \) has recently given, thus it stores it as \( ev_{\text{max}} \). Finally, agent \( p \) provides a normalised feedback based on the greatest evaluation the agent has in memory. Since agent \( q \)’s service is the greatest, agent \( p \) sends 1.0 as the normalised evaluation. When agent \( q \) receives such an evaluation it immediately knows that according to agent \( p \)’s standards, it is the preferred supplier for the microcontrollers.

To bring about this process, first agent \( q \)’s Behaviour with the help of the Resource manager, passes a resource offer to the Interaction manager for delivery to agent \( p \). The environment’s Communication manager catches the offer and asks the Perception manager whether agent \( p \) is in the proximity area of agent \( q \). Since this is true, the Communication manager delivers the offer to agent \( p \). Agent \( p \)’s
6.3 A practical execution example in the automotive business ecosystem

(a) Agent \( p \) tracks down agent \( q \).

\[ \text{sendof}(p, q, (\text{mcr-ctlr}, \text{val}')) \]
\[ \text{eval}((\text{mcr-ctlr}, \text{val}')) = 400 \Rightarrow \text{ev}_{\text{max}} = 400 \]
\[ \text{sendev}(p, q, (\text{mcr-ctlr}, \text{val}), 1.0) \]

(b) Agent interaction details.

Interaction manager receives the offer and passes it to the Behaviour for their evaluation, which is then sent to the Interaction memory for its storage. The feedback is then sent back through the Interaction manager and passed to the environment’s Communication manager. The latter delivers the message to agent \( q \), after consulting the Perception manager.

6.3.3 Engaging suggested interactions

Once agent \( p \) has found a suitable supplier, it passes agent \( q \)’s details to its associated software agent \( p \) in the agency layer. In turn, software agent \( p \) tries to engage a transaction with its counterpart software agent \( q \) and get a shipment of microcontrollers.

After the transaction is successful, the software agent \( p \) notifies the relevant user at the business layer within company \( p \) about the transaction. Figure 6.6 depicts this process across the conceptual layers of the execution context showing how the three layers of the execution context are involved in the practical scenario.

6.3.4 Competition of suppliers

Continuing with the example, in the DAEM layer agent \( p \) comes across agent \( r \) who happens to be a microcontroller supplier. It then sends the offer \((\text{mcr-ctlr}, \text{val}')\) to agent \( p \) but with a different price \( \text{val}' \). Figure 6.7(a) shows that agent \( p \)’s evaluation score if 500 according to its standards resulting in the best evaluation agent \( p \) has granted or at least the best as far as it remembers. When agent \( r \) received a
normalised evaluation of 1.0 it immediately know it has bested other competitors and that it is the preferred supplier for agent $p$ for microcontrollers.

Eventually agent $q$ finds again agent $p$ in the environment and they interact again. Agent $q$ sends the offer $(mcr-ctlr, val)$ which agent $p$ evaluates according to its standards. The offer evaluation is 400 and it turns out that it is not better than the maximum evaluation existing in the memory ($ev_{\text{max}}$), which is of 500. Agent $p$ then sends back a normalised evaluation of $400/500 = 0.8$. When agent $q$ receives it, it realises that a competitor has provided a better service than his, thus being replaced as the preferred supplier (see Figure 6.7(b)).

Component interactions are similar as explained in previous cases (e.g. Section 6.3.2). Service evaluations and competition in DAEM are based on frequent local interactions. This is why agent $q$ re-sent its service offer even though it already knew it has the preferred supplier. Yet, if it had not done it, it would have never noticed any preference change. Finally, competition is encouraged by the normalised evaluation which gives an idea of how much improvement is needed for besting the new preferred supplier.
6.3 A practical execution example in the automotive business ecosystem

6.3.5 Rejecting suggested interactions

This time, when agent $p$ suggests software agent $r$ for interaction, software agent $p$ cannot engage the interaction due to discrepancies in the response time. Therefore, a relevant user in the company is requested for authorisation to carry on with the transaction (see Figure 6.8). Upon analysing the situation, the user points out they have had issues in the past with company $r$, even though they offer a good price for microcontrollers.

Therefore, the user in company $p$ decides to avoid further commercial transactions with company $r$. Consequently, software agent $p$ is notified of the decision and stops interacting with the software agent in company $r$. Furthermore, the decision is propagated to the DAEM layer as well, so that agent $p$ avoids giving high evaluations to agent $r$. This decision triggers preference changes in the ecosystem.

From this moment on, agent $p$ grants a low normalised evaluation (close to 0) to agent $r$ whenever they interact in the environment. Because, agent $r$ will no longer be the preferred agent for microcontrollers, its related software agent will not be suggested as potential partner to the agency layer in company $p$. As a result, agent $p$ looks for the second best microcontroller supplier it remembers. In this case it is agent $q$, so that when they interact again in the environment it will be granted a
normalised evaluation of 1.0 confirming that it is the preferred supplier (again) of agent \( p \).

These layer interactions show how the agency layer and the business layer are affected by DAEM decisions, and their reactions are then reflected in the ecosystem triggering a chain of preferences changes which are then captured again by the agents in the DAEM layer and the ecosystem adapts to the unexpected changes.

### 6.3.6 Authorising interaction

Going further with the example, agent \( p \) now encounters again agent \( s \), an agent whose microcontroller offers have not been good enough for agent \( p \)'s standards. However, this time agent \( s \) offers an improved service which granted a score of 550 when evaluated by agent \( p \). As a result, agent \( s \) receives a normalised evaluation of 1.0.

Similarly to previous cases, software agent \( p \) is notified of the suggested interaction and tries to engage a transaction with its counterpart software agent \( s \). But then again, a conflict in the interaction shows up when arranging the minimum shipment size, making software agent \( p \) to request authorisation to a relevant user in the business layer of company \( p \) (see Figure 6.9).
After an analysis and consultation with other users, it is decided to carry on with the transaction and software agent $p$ is notified on the decision and continues with the transaction with no further problems. Afterwards, software agent $p$ notifies the successful transaction to users in the company. Notice that in this case the DAEM layer is not notified on the issue because it has no significant impact on the ecosystem.

Finally, when agent $q$ is found again in the environment and interacts with agent $p$, it receives a normalised evaluation of $400/550 = 0.72$, knowing then that it has been replaced as the preferred supplier for microcontrollers.

6.3.7 A final analysis of the business ecosystem case

Typical approaches to supply chain formation (cf. a chain in DAEM) include SOAs, cf. [Fragidis et al., 2007]. Using an SOA in the same example could prove difficult to make the whole system adapt to changes in the environment. For instance, when the company $p$ is looking for a microcontroller, a Web service can look for it in a service directory. Once company $q$’s service is found, transactions could be made directly between Web services from both companies.

However, a problem may arise at this point: if a negotiation process is needed, services will not be able to perform it and act upon it because they are usually statically predefined [Ferronato, 2007], as opposing to software agent and MASs. This means that under the assumption of a static environment SOA works fine. Yet a business ecosystem is not a static environment, but rather a dynamic one where changes and innovations are always happening. Thus an SOA for business ecosystem support where unexpected changes occur render this technology not appropriate for adaptation.

Assuming that transactions between company $p$ and $q$ occur with no problems. Let us say then that company $s$ improves its service and is willing to have new partners. Because Web services do not offer the possibility to dynamically discover either new service offers or changes in the existing ones [Ferronato, 2007], company $p$ will find it difficult to know about company $s$. And thus wasting a possibly profitable opportunity. Thus, using SOAs for service discovery in business ecosystems simply avoids the supporting system to adapt to changes on resource improvements.

There are two ways, yet similar in essence, to cope with this situations using an SOA for business ecosystems: one is to continuously check the centralised service registry in order to detect when a new service is available; and the other one is to continuously try all service providers in order to detect innovations and determine who is better to interact with. A simulated centralised approach representing
these two solutions was tested and the results are presented in Section 5.6. Both solutions prove ineffective and unable to adapt to unexpected changes. This renders DAEM as a better solution for supporting business ecosystem than an SOA approach.

A business ecosystem supported by the DAEM layer in a practical setting is worth implementing because of the added value it would bring to the supported companies in terms of adaptation to unexpected changes. The reference model (see Figure 4.4), the DAEM layer (see Figure 6.1), and the macro and micro levels (see Figure 6.2 and 6.3 respectively) provide a starting design of a proper implementation. Based on the example presented in Section 6.3 and the experiments presented in Chapter 5, a DAEM implementation is a promising approach to put in practice for adaptation to unexpected changes in ecosystem domains. This supports the DAEM layer as a contribution of this thesis.

### 6.4 Execution context of a digital service ecosystem

This section presents a digital service ecosystem as an execution context in which DAEM could be applied to. A discussion on the practical utility of its implementation is presented in Section 6.5.3.

A digital service ecosystem is a virtual world where software components, Web services, agents, and other digital components interact and combine in order to fulfill a specific purpose or goal. The term service is used in this section and in Section 6.5 to refer to these digital components. Typically optimal compositions are sought in this domain to fulfill complex requests, cf. [Briscoe and De Wilde, 2006]. Again, the reference model shown in Figure 4.4 is used as a basis but now to support digital service ecosystems. The reference model considers three levels of abstraction: the DAEM level, the ecosystem domain supporting system and the physical world. These are translated to a system for digital service ecosystem support as follows.

Two conceptual layers are assumed to exist in a digital service ecosystem: a) a user layer where users send requests for a service. This layer is the “physical world” being supported by the underlying second layer; and b) a digital service layer where the digital components exist to fulfill service requests, either from the users or from other digital components. This layer is the equivalent to the ecosystem domain supporting system in the reference model.

Likewise for the business ecosystem case, the DAEM layer sits as a supporting layer to the digital service layer. In the DAEM layer the ecosystem elements are played out, updates from the upper layer are considered and incorporated here into the dynamism of the ecosystem and the adaptation process.
Figure 6.10 depicts the execution context of a digital service ecosystem in which DAEM supports adaptation to unexpected changes. Notice that the service descriptions come from the middle layer as opposing to the business ecosystem execution context. Likewise, suggested interactions go to the middle layer only. This is because in a digital service ecosystem typically the users are service consumers, i.e. most of the ecosystem dynamism occurs in the middle layer with no so strong connection to the top layer as in a business ecosystem. This is the main different between these two domains. Nevertheless, some approaches have tried to reduce the difference by using a digital service ecosystem to support business ecosystems, cf. [Ferronato, 2007].

The layers are explained in the following subsections. However, further details about the micro level of the digital service ecosystem execution context are left out of the scope of this thesis.

### 6.4.1 User layer

This user layer is where users request, query, evaluate, or monitor services through specific interfaces and applications. In this layer users may not know they are interacting with an agent or a Web service, they can be seen as a customer to the digital service ecosystem. Nonetheless, users can specify certain characteristics of the result they expect from the services such as quality or personalisation.
6.4.2 Digital service layer

The digital components existing in this layer offer services to the user layer and to other digital components within the digital service layer. The digital components perform actions to fulfill service requests and have the ability to aggregate to deliver added value services. In this execution context, the digital components send service description to the underlying layer to look for those components with which interacting is necessary to fulfil a service, either to aggregate an added value service or to help producing the result of one.

6.4.3 DAEM layer

Similarly to the business ecosystem case, the DAEM layer is where the digital service layer is projected mapping one digital component to one agent in the DAEM layer. Consequently, the digital components now have a DAEM agent each to help them find other components to fulfil their requests.

Service descriptions received from digital components are received here. Interactions are suggested to the digital service layer for fulfilling and composing services. These suggested interactions already consider unexpected changes in the ecosystem.

6.5 A Web service ecosystem: an example

This example illustrates how DAEM could support an ecosystem of Web services as an instance of a digital service ecosystem. Conventional approaches in this area use SOAs which fail to adapt to unexpected changes. Section 6.5.3 presents a discussion about this. Moreover, details of the interacting components of DAEM and those of the execution context are not mentioned again here since they were already covered in the previous execution context (see Section 6.3).

The example itself is focused around Ribbit¹, a suite of telecommunication services offered by British Telecom (BT) as a development platform for innovative telecommunication applications by diverse and independent solution providers. Such providers together with BT conform a digital service ecosystem of the sort targeted by DAEM.

Consider an added value service to a social networking site such as Facebook or LinkedIn, in which a user can send an SMS to all his contacts located in the proximity of a target address. The service is called SMSFriend and it is composed by the following services:

¹http://mdk.bt.com/
ProvideContacts. This service provides a list of contacts linked to a user. The service is provided by the target social networking site.

LocationOfContact. This service provides the location of a contact. Initially it is provided by the target networking site, which simply returns the address of the contact.

ListLocal. This service takes a list of contacts and returns a list of those whose address is within a 10 miles radius from a target address. This service is provided by a map or routing provider such as ViaMichelin or GoogleMaps.

SendMessage. This service sends a text message to a list of contacts; it is provided by a telecomm provider such as BT or Vodafone. This service is available from Web21c.

SMSFriend. This is the added value service bringing all other services together and handling interaction with the user at the user layer. This is owned by a subcontractor to the target social networking site.

Initially the different service providers are represented by an agent each within the DAEM layer, and have their own supporting components. Using DAEM mechanisms to form chains, an added value service is assembled with agents representing the services SMSFriend:FacebookContractor, ProvideContacts:Facebook, LocationOfContact:Facebook, ListLocal:GoogleMaps, and SendMessage:BT. The execution of such an ecosystem goes as follows.

6.5.1 A new service provider enters the ecosystem

At some point, one of the Web21c affiliates launches a new LocationOfContact service. The new service estimates the actual location of a mobile phone user by triangulating strength signals from the nearest signal transmitters. The affiliate, called Ycon², registers to the DAEM layer and sends its service description. Consequently, a new agent is created and placed in a random initial position in the environment. Let us call it agent $yc$.

Initially agent $yc$ moves randomly in the environment foraging for a consumer to its service. Inevitably agent $yc$ interacts with other agents in the environment, making them forage in different areas inducing their preferred partners to not find them when they come back to interact. This creates opportunistic encounters in the environment.

FacebookContractor’s agent $fbc$ is moving around in the environment and discovers the trail of agent $yc$ and decides to track it down because it could not find

²A random name.
Using DAEM within an Execution Context

Figure 6.11: A new service provider is found in the ecosystem.

Figure 6.12: Inter layer communication regarding the FacebookContractor for new available service.

Facebook’s agent \( f_{cbk} \). Eventually the two agents establish contact and interact as in Figure 6.11. It turns out the evaluation of the LocationOfContact service is better than the one provided by Facebook inevitably making agent \( f_{bc} \) to prefer agent \( yc \)’s service.

The agent \( f_{bc} \) informs its counterpart at the digital service layer, i.e. the actual service provider, who updates the composition of the SMSFriend service. The updated application now links all friends who are actually near a place, rather than those living there. Notice than any user at the user layer does not need to know about the internal service update. A better service is simply delivered and is now available for future service requests. Figure 6.12 depicts the inter layer interactions.

6.5.2 Request for a better service

Consider a user who wants a more generic SMSFriend service to use any provided list of contacts but to work on any available channel, not just mobile phone. The
request is taken into account by the FacebookContractor in the digital service layer and appropriate service descriptions are sent to the corresponding agent in the DAEM layer.

The agent $fbc$ forages for suitable services in the environment. It finds three producers, the agent $bt$ offering a SendMessage service which sends a text message to a mobile phone, the agent $msg$ offering a SendMessage service over Instant Messenger (i.e. chat), and agent $skp$ offering a SendMessage service over Skype (i.e. chat as well). Agent $bt$ is the only agent found offering the service over mobile phone, thus it is selected as a preferred producer. On the other hand, a comparison of services between agent $msg$ and agent $skp$ shows that the former offers a better service, thus it is selected as a preferred producer. Interactions with the corresponding digital components of these agents are suggested to the agent $fbc$’s counterpart in the digital service layer. Finally the composite service is presented to the user for consumption, see upper half of Figure 6.13.

At some point outside the whole system, the people at Skype notice that IMessenger is more popular than Skype and decide to add a new feature: sending text messages over mobile phones. This service is incorporated in a new version of their digital component and released in the digital service layer, which then sends its updated service description to the DAEM layer.
In the DAEM layer agents keep interacting even when they are not preferred as long as they find one another in the environment. Therefore, agent \textit{fbc} eventually finds out about the new service produced by agent \textit{skp}. Let us say that the service evaluation shows that this service is better than agent \textit{msg}'s and \textit{bt}'s. Thus agent \textit{skp} becomes preferred producer of agent \textit{fbc}. Consequently a new interaction is suggested to the upper layer where the service is simply provided to the user, see lower half of Figure 6.13. Notice that the switching of services occurs at the digital service layer, the DAEM layer simply adapted to the new circumstance and suggested a better service.

### 6.5.3 A final analysis of the Web service ecosystem example

This high-level example illustrates the adaptation facilities afforded by DAEM when entering the ecosystem and when a user demands better services. Whilst DAEM itself can be built to support contemporary software technologies such as Web services, the flexibility of behaviour demonstrated in the example above relies on the higher-level facilities of DAEM and not on the service registries and dynamic binding mechanisms which may be provided by Web services and other software implementations of the SOA, cf. [Ferronato, 2007].

Indeed in the above example using SOA, once the services are discovered and wired together in the composite service SMSFriend, the latter will not continue to discover new opportunities and negotiate details of interfaces. Also with no continuous, opportunistic search in the DAEM layer no autonomic substitution of SendMessage could be triggered but either remain unnoticed or in the best case cause re-wiring together of the composite SMSFriend by its author. The core difference in behaviour is the statically pre-defined nature of the current implementations of SOAs [Ferronato, 2007] compared to the dynamic ecosystem-inspired features of DAEM. This is the practical utility of DAEM as a supportive layer to this ecosystem domain. This supports the DAEM layer as part of an execution context, as a contribution of this thesis.

### 6.6 Discussion

There is an increasing effort in the research community to support approaches with MAS mediated by an environment as in the DAEM layer. Initial definition have been established to uphold the movement. By definition the DAEM layer constitutes a \textit{first-class abstraction environment} [Weyns et al., 2007] because it provides the setting and conditions for agent living and their interactions, cf. common environment [Omicini et al., 2004]. Moreover, the DAEM layer works as a resource context
6.6 Discussion

manager and overlay data structure after [Platon et al., 2007].

On the other hand, a DAEM execution context complies with a composite configuration [Valckenaers et al., 2007] because 1. the DAEM layer is similar to a simulation configuration due mainly to the virtual space the agents use to navigate in; and 2. the middle layer resembles an adaptive structured information system configuration because the software agents/components interact with other systems and users.

Both examples of DAEM-supported ecosystem domains focus on how DAEM suggestions can be incorporated in the physical world. Changes occurring in the DAEM ecosystem are notified to the higher layers for their attention. Likewise, decisions taken at the higher layer are brought down to the DAEM layer and incorporated in the evaluations agents make whilst foraging for their best interests. Such decisions trigger further preference changes which propagate in the ecosystem. In this way it is demonstrated the practical utility of the DAEM layer in a practical scenario, thus supporting the claim of the DAEM layer as a contribution of this thesis.

Other ecosystem domains can be supported by the DAEM layer. Using the reference model in Figure 4.4 a mapping needs to be made to determine what the agents and the resources are modelling. Then the DAEm layer is played out to see what it can come up with. A methodology for doing this mapping is out of the scope of this thesis. Here it is only shows how it would work if it were implemented.

Summary

This chapter introduced the DAEM layer architecture based on the model defined in Section 4.3. The architecture emphasises the environment as an independent entity from the agents and possesses the characteristic to mediate agent interactions. The internal components were presented as well as their algorithms detailing their functioning. Moreover, such an architecture was placed within two execution contexts for supporting business ecosystems and digital service ecosystems. Using such execution contexts, it is demonstrated how DAEM incorporates unexpected changes, user decisions and changes from the physical world. Finally, the DAEM layer is presented as a contribution to the thesis.
Chapter 7

Comparison With Other Ecosystem Approaches

This chapter reviews other approaches and compares them with DAEM. The approaches are grouped into models capturing ecosystem characteristics and/or target adaptation, and into architectures aiming at supporting systems with the same characteristics. That is, the approaches are compared with DAEM because it is a model and because of the supporting layer to an execution context, respectively. An analysis is also presented in which those approaches are compared with DAEM by the degree they cover adaptation to unexpected changes.

Section 7.1 describes the coverage degrees of adaptation used for the review of approaches. Section 7.2 reviews ecosystem models followed by a review of architectures in Section 7.3. A comparison is then presented in Section 7.4 before closing the chapter with a summary.

7.1 Degrees of adaptation

The approaches are analysed using four degrees of coverage of adaptation. These are used instead of the adaptation framework presented in Chapter 3 because the framework is focused on one single technology, i.e. MASs, whereas the approaches cover different technologies. The four degrees envisaged for adaptation coverage are explained below.

None. This degree of coverage is self-descriptive. It refers when an approach does not cover adaptation at all.

Limited. A low degree of coverage in which adaptation is predefined or simply assumed to occur.
Sufficient. This degree indicates that adaptation is achieved and might possess a notion of unexpectedness. However it is not adequate to adapt to unexpected changes.

Unexpected changes. A high degree of coverage in which adaptation to unexpected changes occurs.

7.2 Ecosystem models

7.2.1 ECHO

[Holland, 1995] presents an ecosystem model called ECHO consisting of agents living in a distributed environment containing resources which allow the proliferation of agent diversity. Agents interact and consume resources from each other. The type of interaction depends on the resource consumed. It is claimed that because of these interactions the whole population adapts itself until an equilibrium is reached even when changes occur, allowing the emergence of complex structures.

This approach targets adaptation by modelling an ecosystem of individuals, cf. cells. However, experiments reported by [Smith and Bedau, 2000] on an ECHO-based prototype show that the emergence of complex structures do not occur and it does not exhibit adaptation as claimed by [Holland, 1995]. The degree of adaptation achieved by this approach then is sufficient because even when the experiments disprove the approach, adaptation is modelled by reaching an equilibrium even in the presence of changes.

7.2.2 Evolved biological ecosystems

[Vandecasteele et al., 2004] explains that biological ecosystems can be built using evolutionary computation by codifying the presence or absence of diverse types of species with boolean values thus codifying a whole ecosystem in a single chromosome. Then to evolve a population of ecosystems using a standard genetic algorithm. The result is an optimised ecosystem indicating the species to form it, yet it does not show how the species interact.

This approach focuses on the creation of ecosystems rather than on adaptation. Thus it is considered to have a none degree of adaptation.

7.2.3 Algebra and visual notation for service interfaces

[Dumas et al., 2006] proposes a declarative approach to service interface adaptation for service ecosystems formation. The approach deals with the behavioural
perspective of service interfaces, i.e. controlling dependencies between message exchanges. The purpose of such an approach is to participate in different interactions portraying different behaviours for the same functionality. The approach introduces six operators: flow, scatter, gather, collapse, burst, and hide. These are used as functions where a behavioural interface is used as an input and another different behavioural interface is produced as the output. In the end, such an output should be equivalent to the required by an interacting service.

Although the approach deals with adaptation with an ecosystem inspiration, it focuses only on the interaction point between any two entities, not the ecosystem as a whole as in DAEM. Moreover the operators limit the sort of achievable adaptation to a reduced set of possibilities. Therefore this approach is regarded with a coverage degree of limited.

7.2.4 Ecological calculus

[Zhang and He, 2007] presents the ecological calculus framework aiming at synthesising concepts from universal logic, system theory, and ecology. The purpose is to create an approach to capture MAS interactions by representing the relevance of relationships between and among agents. This is captured by the concepts of longitudinal and latitudinal correlativity which help to reason to reach to valid conclusions using fuzzy concepts. It is claimed that this approach can be used to solve agent interaction problems such as negotiation and competition.

This theoretical work is inspired by ecosystems because it focuses on interactions among individuals, cf. agents. In spite of the ecosystem inspiration, this approach does not target adaptation. Therefore its adaptation coverage is none.

7.2.5 Population-based adaptive system

[Eiben et al., 2007] presents a taxonomy of adaptation composed by three learning mechanisms: individual, social, and evolutionary. Then it is suggested that a population-based adaptive system (PAS), comprising collections of agents performing actions and having behavioural rules, can obtain a population of individuals fitted (cf. optimised) to a particular situation by using the three learning mechanisms. This approach assumes that an agent already knows the means to find a better solution to a problem than the rest of the agent population. When such an agent faces that problem and obtains a better result, the new solution is simply shared to the rest of the population for social learning, thus adapting the whole population.

This approach also assumes that the behavioural rules the agents possess are different from agent to agent, resembling the CAS property of diversity. Evidently
this approach targets adaptation yet it does not consider unexpected problems (cf. changes). Therefore the degree of adaptation coverage is sufficient.

7.2.6 Coordination model to long-term transactions

[Razavi et al., 2007] introduces a coordination model to coordinate long-term transactions in digital business ecosystems based on SOAs. Transactions are represented by tree structures in which branch nodes are seen as coordinators or service composers and leaf nodes refer to basic services. Six composition types conform the model namely sequential, parallel, sequential alternative, parallel alternative, data-oriented and delegation. Moreover a formalism is presented to describe, manage, and reason about service dependencies. The model tries to minimise unexpected composition failures (cf. unexpected changes) by identifying potential anomalies in compositions. Furthermore it uses a rollback mechanism to isolate such failures and avoid them for the rest of the transaction.

Even though this approach considers unexpectedness in service compositions, adaptation is not regarded as a process to overcome unexpected composition failures, but rather to avoid them when reasoning about service dependencies. Yet this avoidance is transaction dependant, i.e. the same composition failure could be encountered in a different transaction. Therefore the degree of adaptation coverage remains limited.

7.3 Ecosystem architectures

7.3.1 DIET

[Marrow et al., 2003] presents an ecosystem inspired toolkit for emergent group formation using mobile agents (or infohabitants) to find an environment site among a distributed set with the most resources of their interest. Agents of different types are able to communicate locally with peers and migrate from one environment to another according to the resource availability. Collected environmental information along with the agent population is passed through an evolutionary process making the evolved population to migrate to the environment with the most available resources according to the agents’ preferences.

Regardless of the ecosystem inspiration this approach targets optimisation by balancing between resource availability and the most preferred environments, i.e. it does not target adaptation. Therefore the degree of adaptation coverage considered for this approach is none.
7.3.2 DBE

[Briscoe and De Wilde, 2006] presents an approach to business ecosystems in which evolutionary processes are used to find the optimal composition of suppliers for service requests. Individuals represent services and are considered the basic unit for evolutionary processing. A population of individuals are passed through a genetic algorithm to find the optimal composition to fulfil a compound service request. Individuals composing optimal solutions replicate and have a tendency to be considered in further service compositions. Moreover, different sets of individuals co-exist in inter-connected habitats (cf. nodes in a network), thus successful individuals tend to cluster in the habitats where they are most required for composition.

This approach is highly inspired by ecosystems, yet it does not target adaptation. Therefore its degree of adaptation coverage is none.

7.3.3 DAEDALUS

[Hettiarachchi et al., 2006] introduces the DAEDALUS framework used for migrating computationally evolved MASs from a simulated world to the real world. This approach considers unexpected changes as the environmental differences between the simulated environment and the real one, and as the result of actions performed by the agents in a real environment but not seen in the simulated one. When an agent finds an unexpected change it tries to overcome it. If it succeeds, that knowledge is propagated to the other agents, which using a fast learning process, learn collectively from the successful agent. The adaptation process occurs at the agent level not at the population (cf. system) level as opposing to DAEM.

This approach considers adaptation to changes by means of a learning process over a population of agents. Unexpected changes are considered as well, however it is assumed that agents possess the means to overcome these changes. Therefore the adaptation coverage is regarded as sufficient.

7.3.4 Open negotiation environment

[Muntaner Perich and De la Rosa Esteva, 2007] presents an approach aiming at studying adaptation and spontaneous composition of services by means of Dynamic Electronic Institutions (DEIs) formed by software agents. Such an approach makes an analogy between DEIs and business ecosystems and argues that the latter can be formed using DEIs. The authors focus on short-term associations also called temporary electronic institutions, and allow members of DEIs to align their norms and objectives “on the fly” to form partnerships.

Adaptation is targeted by this approach, however it is not clear whether it covers
unexpected changes. Thus it is considered with a sufficient degree of adaptation coverage.

### 7.3.5 Request-based virtual organisations

[Svirskas et al., 2008] presents an approach to build digital business ecosystems based on request-based virtual organisations. SMEs try to increase the competitiveness and their chance to react to market changes (cf. adaptation) by forming virtual organisations with complementary SMEs. These sort of organisations are built by intelligent negotiating agents who try to find the best match for their service requirements. Competition is claimed to be supported as a prompt reaction to the market, yet this is assumed to be supported only because virtual organisations are used.

Adaptation as such is not targeted by this approach, though reactions to market changes could be regarded as that. Nevertheless because this is only assumed to occur, the degree of adaptation coverage is limited.

### 7.3.6 Architecture for unanticipated environments

[Ding et al., 2009] proposes an architecture to support software adaptable to unanticipated environments. The approach consists of a first encapsulation of adaptation concerns in different architectural elements (cf. components) followed by a late binding of their relationships. This shall ensure an easy modification of the middleware to enable adaptation at runtime, yet those architectural elements have to be designed and made available in advance.

Clearly the approach “predefines” the adaptation scope a system could achieve in an “unanticipated” environment. Therefore its degree of adaptation coverage is limited.

### 7.3.7 Tuple based

[Villalba et al., 2010] proposes an architecture inspired by ecosystems portraying self-adaptation and self-organisation to support dynamic scenarios. The approach considers four types of ecosystem entities such as flora and consumers, and niches represented by tuple spaces which function as the interaction interfaces. The entities (agents) have needs and a “happiness” status they try to maximise by fulfilling their needs. Experiments show how the “happiness” levels reach a balanced state at the end of their simulations throughout a set of niches.

This approach is highly inspired by ecosystems and assumes that adaptation is achieved because of reaching a balanced state. However this does not demon-
Comparison With Other Ecosystem Approaches

Table 7.1: Comparison table of approaches to adaptation to unexpected changes.

<table>
<thead>
<tr>
<th>Ecosystem model</th>
<th>None</th>
<th>Limited</th>
<th>Sufficient</th>
<th>Unexp chg</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Holland, 1995)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(Vandecasteele et al., 2004)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Dumas et al., 2006)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(Zhang &amp; He, 2007)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Eiben et al., 2007)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(Razavi et al., 2007)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ecosystem architectures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Narrow et al., 2003)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Eriscoe &amp; De Wilde, 2006)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Hettiarachchi et al., 2006)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(Muntamer Perich &amp; De la Rosa Estove, 2007)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(Svirskas et al., 2008)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Ding et al., 2006)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(Villaíba et al., 2010)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DAEM</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 7.1: Comparison table of approaches to adaptation to unexpected changes.

This section compares DAEM against the approaches mentioned above which are selected based on their ecosystem inspiration and/or on their claim to support adaptation. This selection does not regard the application domain, but only the degree at which they target adaptation as explained in Section 7.1.

Table 7.1 presents a comparison of these approaches with DAEM. As can be appreciated, only DAEM covers adaptation to unexpected changes. This was demonstrated by the experiments presented in Chapter 5 and by putting the DAEM layer in execution contexts in two ecosystem domains. This emphasises the contributions of this thesis: DAEM as a model of adaptation to unexpected changes and a DAEM layer to support this sort of adaptation in ecosystem domains.

Summary

This chapter presented approaches inspired by ecosystems and/or target adaptation. A comparison then is presented in which only DAEM exhibit adaptation to unexpected changes.
Chapter 8

Conclusions

The research presented in this thesis aims to provide MASs with the capability to adapt themselves to unexpected changes occurring in complex, dynamic environments. The approach taken involves analysing CAS adaptation properties and applying them to MASs, thus providing a framework which allows the development of MASs which can cope with unexpected changes. The Design Science methodology [Hevner et al., 2004] underpins the research described in this thesis.

Chapter 2 presents examples of domains governed by dynamism and complexity in which unexpected changes demand for a proper adaptation approach for their support. The examples domains are business ecosystems, digital service ecosystems, manufacturing, transport, and city modelling. In the context of this thesis these domains are called ecosystem domains. Definitions of unexpected change and adaptation are also covered by this chapter.

Chapter 3 then introduces the definitions of an agent, an MAS, an adaptive agent and an adaptive MAS. The latter is used to derive dimensions for analysing other approaches to adaptation in the area of MASs. The result of such an analysis is a classification framework of adaptation consisting of five classes of adaptation in MASs namely automaton, control system, semi-isolated evolution, dynamic interactions, and ecosystem. The latter is selected as the promising approach to support an ecosystem domain. Thus, adaptation principles for adaptive MAS development and characteristics of natural ecosystems (an instance of CAS) are obtained as potential elements for building the approach.

Chapter 4 combines systematically the adaptation principles and the natural ecosystem characteristics under a common set of adaptation properties. The result of such a combination is a set of elements underpinning the approach of this research by deriving a formal model of a dynamic agent-based ecosystem called DAEM.

Chapter 5 takes the model as the basis for developing a running prototype. This
is then used to test the conditions under which DAEM is capable to converge to the expected configuration and then to adapt to unexpected changes in two occasions. The prototype is also compared against a simulated centralised approach suggesting that DAEM converges and adapts better than typical centralised mechanisms, e.g. as in SOAs. These results support DAEM as a contribution of this thesis.

Chapter 6 considers the model as the basis for deriving an architecture for building a DAEM layer within an execution context such as a business ecosystem and a digital service ecosystem. Then a couple of practical examples in those domains are explained: an automotive business ecosystem and a Web service ecosystem. The examples show the DAEM layer suitability for supporting ecosystem domains in practical scenarios. This supports the DAEM layer architecture as the second contribution of this thesis.

Chapter 7 presents other approaches to ecosystem modelling and architectures for adaptation. These are compared against DAEM rendering them not as capable as DAEM to adaptation to unexpected changes. This supports the novelty of the contributions of this thesis to the state-of-the-art.

8.1 Contributions

The main contributions of this thesis are aligned with the four main hypotheses stated at the beginning of this thesis. The contributions are described below together with their proven hypotheses. Table 8.1 presents a summary of the contributions.

A formal model of a dynamic agent-based ecosystem (DAEM). It provides the definitions of necessary elements and processes, such as adaptation, to consider when developing an MAS for supporting ecosystem domains. The tested hypotheses supporting this contribution are explained below.

- **H1. It is possible to create a functional adaptive MAS based on ecosystem adaptation properties.** DAEM definitions are based on characteristics of natural ecosystem modelling and principles of adaptive MASs development combined under the adaptation properties of CASs. Furthermore, a prototype was developed and tested, supporting this hypothesis.

- **H2. The resulting MAS adapts to unexpected changes.** A set of experiments were carried out in a hypothetical ecosystem to determine under which combination of parameter values the DAEM prototype is able to adapt to unexpected changes. The results demonstrate that DAEM con-
8.2 Future work

A number of suggested potential research projects can be carried out to extend the results of this thesis and investigate a number of interesting open issues worth

<table>
<thead>
<tr>
<th>Hypotheses</th>
<th>Tests</th>
<th>Contributions to Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1. It is possible to create a functional adaptive MAS based on ecosystem adaptation properties.</td>
<td>A prototype implementation.</td>
<td>A formal model of a Dynamic Agent-based Ecosystem (DAEM).</td>
</tr>
<tr>
<td>H2. The resulting MAS adapts to unexpected changes.</td>
<td>Experiments using unexpected changes</td>
<td></td>
</tr>
<tr>
<td>H3. The resulting MAS is more resilient to unexpected changes than a centralised approach.</td>
<td>Comparison of experiments using a simulated central directory.</td>
<td></td>
</tr>
<tr>
<td>H4. The resulting MAS supports adaptation to unexpected changes which other approaches cannot in practical scenarios.</td>
<td>Examples of practical scenarios and a comparison of approaches.</td>
<td>A DAEM layer architecture for supporting ecosystems domains.</td>
</tr>
</tbody>
</table>

Table 8.1: Contributions and validated hypotheses.

verges and stabilises to the expected solution even when facing unexpected changes. These results support this hypothesis.

- H3. *The resulting MAS is more resilient to unexpected changes than a centralised approach.* A simulated centralised approach was tested under the same hypothetical ecosystem showing that DAEM is more resilient to unexpected changes than the centralised approach. Thus, this hypothesis is supported as well.

**A DAEM layer architecture for ecosystem domains support.** It portrays DAEM as an instance of the model, its functionality and the algorithms of its components in order to support a context execution within an ecosystem domain. It is supported by the tested hypothesis explained below.

- H4. *The resulting MAS can support adaptation to unexpected changes which other approaches cannot in practical scenarios.* A couple of practical examples of ecosystem domains, an automotive business ecosystem and a Web service ecosystem, were used to show how DAEM incorporates ecosystem changes into the supported system and how decision taken are reflected as changes in the ecosystem. In addition, approaches targeting ecosystem modelling and architectures for adaptation to (unexpected) changes are reviewed and compared against DAEM. It is shown that they do not cope with adaptation to unexpected changes. This supports this hypothesis.
exploring. This list can be considered as a guideline for the interested researcher wishing to continue exploring this area.

- Incorporate propagation of resource values, so that a resource value depends on the accumulated resources contributing to it. This comes from the fact that sometimes the decision to choose between two producers depends on who actually provides them with resources. At least in the automotive industry, in some cases a company tells its potential suppliers whom they prefer for their providers.

- Test DAEM using more flexible configurations such as using different evaluation functions per agent, more than one resource per agent, etc. These will test whether different ecosystems can emerge and either compete or depend on one another.

- Analyse DAEM as an ecosystem model. For instance, analyse how its entropy changes over time. That is, using DAEM as a test bed to analyse other properties of self-organising systems.

- Analyse the patterns the agents create in the environment, for instance the dynamics of niche creation. Such an analysis will test other emergent behaviours not initially foreseen in DAEM, and test whether the patterns have an impact on adaptation and self-organisation.

- Fine tune DAEM behaviour using data from an existing ecosystem domain. This will test whether DAEM can be enhanced with characteristics particular of an ecosystem and and customised to work for it.

- Use data from an existing ecosystem domain to feed DAEM and test whether it can make estimations of future situations. That is, feeding DAEM with past information and decisions taken in order to test whether DAEM reaches the same outcome or a better one, thus suggesting that DAEM could be used for prediction of ecosystem domains.

- Test DAEM in a deployment environment, e.g. with a set of companies forming a business ecosystem. This requires tests of robustness and scalability, i.e. how many agents the environment can support without collapsing.

- Test DAEM suitability for other ecosystem domains. In the end, the application of DAEM depends on what to consider as a resource and as an agent, and on how to evaluate such a resource. For instance, in a traffic control scenario the agents can represent typical routes and the resources can be the different street segments whose value is the current load of cars. Thus routes
exchange resources whose value changes dynamically. Other problems can be treated in a similar way.

8.3 Final remarks

The majority of the reviewed approaches to adaptation in the area of MASs focus on the cognitive side of the problem attempting to solve the problem by converting agents into intelligent beings. Thus the straightforward assumption is that the intelligence of the overall system increases with the intelligence of individual agents and with the increase of the number of agents. Yet this has been shown to not hold for any type of changes.

The adaptation approaches belonging to the adaptation classes of automatic, control system and semi-isolated evolution in Chapter A, emphasise how they achieve intelligence and thus adaptation, specially by using learning techniques. However, they are limited to the stillness they assume in the environment. DAEM does not use any learning technique and the agents are reactive rather than with a high level deliberation process, they do not infer knowledge of the world (they even forget it). Yet they are able to adaptation to unexpected changes in a self-organising fashion.

The research areas of MASs and self-organisation are converging to one another (cf. [Di Marzo Serugendo et al., 2005]). Self-organisation is becoming an inspiration to intelligence (cf. [Parunak, 1997]). The contributions to knowledge of this thesis support this view and point forward in this direction.
Bibliography


Itzhak Benenson. Modeling population dynamics in the city: from a regional to


Suranga Hettiarachchi, William Spears, Derek Green, and Wesley Kerr. Distributed agent evolution with dynamic adaptation to local unexpected scenarios.


H. Van Dyke Parunak, Sven A. Brueckner, and John Sauter. Digital pheromones for coordination of unmanned vehicles. In Danny Weyns, H. Van Dyke Parunak,


Appendix A

Literature Review of Adaptation in Multi-Agent Systems

Examples of adaptation approaches and implementations in MAS have appeared almost since the beginning of MAS as a discipline. And they are appearing more frequently nowadays, but from fragmented streams of research aiming to enable adaptation of agent systems for different purposes.

Streams go from deterministic systems to flexible systems, from software to robotics, from evolutionary computation to complex systems, from organisation support to economic markets. Therefore, the examples presented in this chapter were selected due to their claim to be adaptive in some way to some situations, and not by their results quality nor the application domain. These are grouped according to the adaptation class they belong to, see Section 3.3 for more details.

Section A.1 presents the approaches under the automaton class. Then Section A.2 shows the approaches under the control systems class. Then the approaches under the semi-isolated evolution class are introduced in Section A.3. Then the approaches belonging to the complex interactions class are presented in Section A.4. Finally, the ecosystem class approaches are explained in Section A.5.

A.1 Automaton

A.1.1 A social reasoning for adaptation

A social reasoning for adaptation [Sichman and Demazeau, 1995] endows each agent with a knowledge base which is used to model the environment (including its changes) in a discrete way. Agents have different sets of actions they use to make plans and sometimes they have the same goals. But, when agents cannot reach them by their own, they may infer they need to cooperate in order to reach
A.1 Automaton

Agents re-plan and consider other agents’ actions and goals, i.e. one agent may infer that it needs help to achieve its goals and asks for help to other agents according to who it thinks may provide this help. Thus, when an agent with the same goal is asked for help, it decides to cooperate and does a re-planning for considering the other agent’s actions.

So, adaptation is enabled when a logical conclusion dictates a modification to an agent’s original plan. Additionally, agents in this approach are supposed to be suited for this environment and therefore they interact only with agents of the same type (but with different action set.)

A.1.2 The AMAS theory

The AMAS theory [Capera et al., 2003] is proposed for the design of complex adaptive systems. It is assumed that all agents must be cooperative. And thus, is suggested to design each agent with not only what it has to achieve, but with what it has not to do as well. In other words, agents must be endowed with the ability to detect non-cooperative situations. The latter can be any of the following: a) when an environment perception is not understood, b) when no agent activity is produced due to the perceived information, and c) when produced conclusions are not of any utility to other agents.

Therefore, the designer must provide the agents with the ability to distinguish between cooperative and non-cooperative situations, which in turn are considered the two (abstract) states agents can perceive from the environment. Additionally, the designer must find “exhaustively” all non-cooperative situations that can occur for each agent type, and for each of them, predefine the set of actions they can perform to switch back to a cooperative situation.

A.1.3 Adaptive heuristic for opponent classification (AdHoc)

The Adaptive Heuristic for Opponent Classification AdHoc [Rovatsos et al., 2003] allows agents to acquire knowledge from agent interaction experiences for further adaptation to adversaries’ behaviours. It generates agent classes whilst interacting with opponents agents. These classes are created explicitly in terms of a deterministic finite automaton whilst adaptation is enabled by anticipating other agents’ actions according to their current state. Experiments in the Iterated Prisoner’s Dilemma (IPD) game show the suitability of this approach for deterministic environments.
A.1.4 Reusable software components

Reusable software components [van Splunter et al., 2003] make agents to adapt their internal structure according to environment requirements. A software agent factory is used along with a generic agent model in order to explore different possible agent designs, so that producing agents according to environment needs. Thus, adaptation is determined by the combination of available components and to the environment conditions those components were originally developed for.

A.1.5 Memory-based adaptation

A memory-based adaptation [Lerman, 2004] allows robots, i.e. agents, to estimate environment states by means of local observations. A rolling memory window is used to store consecutive discrete past observations which in turn are used to make future decisions. Actions are predefined and explicitly represented as states in an automaton. Local observations trigger transitions and bring the automaton to other states, i.e. other actions. In this way, agents react to local observations in a deterministic way. Coordination and adaptation as well can be achieved when agents react to a peer having difficulties to perform a task.

A.1.6 Environment adaptation

An environment adaptation [Mertens et al., 2004] makes the environment to react according to changes provoked by agents and external events. The environment is divided into physical application environment (PAE) and logical application environment (LAE). The former deals with the execution environment (a lower layer which deals with the hardware) whilst the latter deals with the agents themselves. Constraint satisfaction problems (CSPs) are solved in this approach where the CSP is converted into a graph on top of the LAE. Each node represents a variable of the problem, and each edge flowing out from it represents a different value that can be assigned to the node. Nodes are distributed among several hosts, on top of the PAE, and agents navigate through them whilst assigning values to variables.

Changes are known in advance and are classified as: a) external events e.g. adding and removing nodes, b) changes requested by agents suggesting which nodes should belong to the same host because of assignation difficulties, and c) changes requested by the PAE. The LAE reacts to changes according to the change type: when a change is requested by an agent, the LAE modifies the layout as suggested by the agent. Or when the PAE reacts according to workload balance.
A.1.7 A generic component-based architecture

A generic component-based architecture [Amara-Hachmi and Fallah-Seghrouchni, 2005] allows to develop self-adaptive mobile agents. The architecture considers three main components for adapting agents whilst moving from one platform to another: context-description, context-awareness and reconfiguration. The former is part of the execution platform and the other two are part of the agent. The context-description is assumed to exist in each potential destination platform. This component contains information about the platform such as physical resources, MAS coordination protocols and user preferences.

Thus, when an agent is about to start its moving, the destination platform’s context-description component is read by the agent’s context-awareness component. The destination platform configuration is then analysed for compatibility. If there is a mismatch, a signal is sent to the reconfiguration component (inside the agent), which in turn determines the new software components to be added (available in a repository) to the agent when the latter arrives at the destination. Then, after the agent reallocation, the reconfiguration component removes all incompatible components from the agent whilst new ones are added. This form of adaptation relies on the availability of software component for predefined environment situations.

A.1.8 Protocol composition

Protocol composition [Desai et al., 2006] can be used to cope with dynamism in organisational processes. Protocols are seen as a collection of rules which govern interactions among business roles (performed by agents) and having specific goals. Thus, when protocols are combined, their goals are combined as well to achieve bigger goals. For protocol composition, axioms must be created for ontology matching, i.e. equivalent role definitions, data flows, implications for concept matching.

Three types of adaptation are considered: exceptions, business policy changes, and business model changes. The first one is handled by adding new interconnections among existing protocol participants. Business policy changes are handled by modifying existing interconnections among protocol participants. Finally, business model changes are handled by managing exceptions, business policy changes and creating new roles. In general, adaptation is accomplished by combining predefined interaction mechanisms. However, this is done at design time.
A.1.9 A social reasoning framework m²InFrA

A social reasoning framework, actually called m²InFrA [Rovatsos et al., 2006], is used to provide agents with adaptive strategies for argument-based negotiation. The actions agents can perform when sending proposals consist of 1. challenging by asking for reasons, 2. justifying proposals, 3. attacking justifications, and 4. co-inciding previous challenges, justifications and attacks.

Agent conversations are modelled as a sequence of turn-taking interactions between 2 agents, i.e. the modelling agent and the agent being modelled. Agent models are Agents being modelled can be classified due to their conversation patterns and eventually agent classes can be merged together according to similarities. Modelling agents adapt their behaviour by anticipating other agents’ actions whilst negotiating.

A.2 Control system

A.2.1 Computational ecosystems

Computational ecosystems [Glance et al., 1991] are used to analyse adaptation to fluctuations of time-periodic resources. A computational ecosystem is viewed as a type of distributed computation where independent agents compete for resources. Identical agents are considered for resource contending. These resources are selected on the basis of a payoff, which in turn depends on the resource’s performance such as time required for task completion, solution accuracy or required memory. The availability of resources in the environment is oscillatory resembling seasonal product prices. The whole system is simulated using the Huberman-Hogg theory which uses a set of mathematical functions for modelling the agents, their decisions, the received payoffs and the oscillation of resources.

A.2.2 Organisational adaptation simulations

Organisational adaptation simulations [Carley, 1998] are used to analyse strategy emergence. Agents represent organisational task-oriented individuals capable to perform classification tasks, information gathering, opinion communication, and are able to learn through experience. Agents form interaction networks, dependence networks and knowledge networks according to their (predefined) task nature. The CEO is modelled as a simulated annealer (i.e. a strategy learner) and is capable to tune the organisation performance by hiring, firing, redesigning networks, and reassigning agents to other tasks. The CEO adapts the way the whole organisation operates according to performance.
A.2 Control system

A.2.3 An adaptive organisational framework

An adaptive organisational framework [Fatima and Uma, 1998] permits agents to adapt their structure to optimise load balancing whilst accomplishing tasks such as monitoring, fault detection, diagnosis and treatment. Agents perform composition and decomposition special operations where two agents can merge into a single one, and one agent can split into two agents, respectively. Agent interactions are limited to perform only these two operations. By only performing these, agents adapt their structure in order to optimise load balancing and utilise resources efficiently of different sites they are performing in. In addition, task deadlines are met during this process.

A.2.4 An Artificial Immune System ARTIS

An Artificial Immune System ARTIS [Hofmeyr and Forrest, 2000] incorporates several properties from natural immune systems for supporting intrusion detection. The function of ARTIS is reduced to two main tasks: intruder detection and elimination. For the former task to be done, a distinction between nonself and self should be made, i.e. discern between harmful pathogens and everything else, respectively. For this purpose, negative detectors, i.e. lymphocytes, learn to tolerise self cells, so that they get activated when a nonself pathogen appears. Lymphocytes, self cells and pathogens are represented by bit strings. Thus, a negative selection algorithm creates random detectors and expose them to the environment, if they activate (match) any self cell bit string they are eliminated immediately. This phase is called toleration period.

After certain amount of time in this period, detectors become “mature” and their match threshold is increased, this way when they become activated they are not eliminated. Then, a memory-based training is carried out on a small set of nonself to detect patterns there and make future responses rapid and efficient when structurally related pathogens appear (although it was not explained how the learning process can be implemented in detail). Additionally, the system needs to be set by several tuning parameters, e.g. bit string length, activation threshold, toleration period. Adaptation is achieved when the system does not longer require to be trained with self cells, and so it adapts itself to new pathogens, i.e. it learns the structure of new encountered pathogens.

A.2.5 An economic model

An economic model [Schulenburg and Ross, 2000] enables agents to make investment decisions in a stock market environment. The market consists of several
elements: one non-intelligent agent that resembles the buy-and-hold strategy; onank agent for representing the strategy of leaving all the money earning interest;
the set of information about the market variables, i.e. prices, transaction volume, splits and dividends, price differences, moving averages, buy-and-hold standings and bank investment; three adaptive agents using a Learning Classifier System (LCS) each which receive different information sets about the current market situation; and finally, two assets to be traded with varying prices. Agent interactions are made through the environment indirectly, i.e. each time an agent affects the environment (market variables), another agent detects this alteration through the information acquired from the environment itself. Adaptation is enabled by tuning internal LCS parameters in order to optimise agent’s own benefits.

A.2.6 An adaptive organisational policy

An adaptive organisational policy [Fatima and Wooldridge, 2001] permits agents to allocate tasks and resources in several organisations. A task allocation protocol (TAP) is used within an organisation to form agent teams and distribute workload among them. This distribution is made according to current task throughput which is the function organisations try to optimise. When one organisation needs more resources (agents) to perform its assigned tasks, it uses a resource allocation protocol (RAP) to hire agents from other organisations. This quickens an adaptation process in each organisation where TAP rearranges agent coalitions whilst maximising task throughput.

A.2.7 Adaptive web site agent

Adaptive web site agent [Pazzani and Billsus, 2002] aids users to find additional information within a web site according to individual user’s interests and other users’ related interests and past actions. The agent serves as a guide through a web site and adapts its suggesting behaviour according to user’s actions rather than changing the structure and content of the web site itself. Several strategies for making recommendations are combined and adapted according to user’s preferences.

First, the agent learns through web logs how documents are related among them by analysing their similarity, references from one document to others, references from other documents to a single one, and the frequency a document is accessed along with others during the same session.

Second, the agent learns personalised profiles by analysing whether the user prefers to follow recommendations according to similarity, references to or made by other documents, and documents accessed along with others. Previous sessions are stored as cookies, so that the learning process does not do a re-analysis.
Third, the agent learns when to make a recommendation to the user according to explicit requests, focus on the web browser, and number of ignored recommendations. Finally, a neural network is used to determine the usefulness of each strategy per individual user. Adaptation is achieved by making approximation to the user’s preferences and behaviour.

A.2.8 Emergence of adaptive searching rules

Emergence of adaptive searching rules [Nepomnyashchikh and Podgornyj, 2003] can be achieved by the interactions between oscillators for agent orientation regulation. The purpose is to see whether behavioural rules can emerge from non-linear systems, i.e. the oscillators, whilst foraging. The agent architecture for movement control consists of three oscillators: two for direction control, i.e. left and right turns, which inhibit to each other, and the third one is used for activating the other two at periodic times.

Additionally, an odour sensor is used for inhibiting the direction control oscillators. This sensor captures differences in odour, i.e. an odour gradient, from the environment. Initially, the agent wanders (almost) randomly in the environment until it detects an odour gradient. This makes the odour sensor to signal a negative response to the oscillators making them to reduce their output. Then, the oscillators do not make the agent to turn, which makes it keep its current heading. The less difference in odour is detected, the more the agent resumes its wandering behaviour.

A.2.9 Monitoring interdependence graphs

By monitoring interdependence graphs [Guessoum et al., 2004b] structure adaptation of distributed systems are allowed to occur. The architecture is composed by an interdependence graph and three agent types: domain agents, monitor agents and host-monitor agents. The graph describes the system’s global behaviour where nodes represent specific-domain problem solvers and links between nodes indicate their relation importance (i.e. communication load). This relation importance can be used to identify whether the system depends on a small set of nodes (i.e. specific-domain problem solvers).

The domain agents represent activity executers where each of them is associated to a graph node. Each monitor agent is assigned to a specific domain agent for analysing domain agents’ message passing to each other. Then, they use that information to update links between nodes, i.e. the communication load between domain agents.
All monitor agents are allocated to a particular host which in turn is monitored by a host-monitor agent. When a link between two domain agents is modified (due to a specific task), the corresponding monitor agents notifies their host-monitor agent about it for global information accumulation.

After certain amount of time, host-monitor agents send global information to other host-monitor agents for further propagation to monitor agents. Then, each monitor agent uses this information to calculate its proportional participation in the global communication load and balance their links in the graph. Finally, monitor agents inform their associated domain agent about the changes and enforce them to act accordingly.

### A.2.10 Economic firm simulation

Economic firm simulation [Guessoum et al., 2004a] allows to study the interactions among economic firms and organisational forms. A firm is an autonomous entity that uses resources to provide goods and services in a competitive market. Firms are represented by agents that observe other firms in order to select a proper strategy for gaining more profit. Each agent uses a LCS to deal with dynamic variations of available goods and price fluctuation occurring in the environment. An organisational form is a collection of firms with similar behaviours and structure. They can vary their internal structure (member firms) and disappear according to variations in the environment. The interactions between an organisational form and its economic firms are mediated by monitor agents. The latter informs both parts about variation tendencies of firms’ resources and organisational forms’ population in order to act accordingly.

### A.2.11 A dynamic opponent modelling

A dynamic opponent modelling [Marín et al., 2005] for football simulation environments allows agents to use statistical information to predict opponents’ behaviours. Agents create opponent classes by means of statistical information about opponents’ movements, such as speed and displacement in several time intervals according to their relative position with the modelling agent. Classes are used to adjust agents’ behaviours to those of the opponents’ whilst playing a match. Adaptation is enabled by selecting the best action from a fixed set, according to the predicted opponents’ position.
A.2.12 Meta-rules in adaptive economic firms

Meta-rules in adaptive economic firms [Rejeb and Guessoum, 2005] are studied to deal with the exploration-exploitation dilemma. The environment consists of a market where firms (i.e. agents) can sell and buy goods whilst optimising resources and benefits. Firms are modelled as LCSs with meta-rules attached to them. These rules consider current performance and elapsed time to decide either to explore for new strategies or to exploit strategies already learnt.

A.2.13 Cooperative information-sharing

Cooperative information-sharing [Dutta et al., 2006] is used to deal with distributed resource allocation problems in network call routing. An extension of the post-task completion protocol (e-PTC) is proposed to ensure effective state diagnostics. Thus, e-PTC allows agents to share information not only when tasks are finished, but when fail as well. When a call forward request reaches its destination, or fails to do so, the agent there sends respectively either an “ack” or “drop” message all the way back to the call source. All agents involved in that path append their node state to that message, where the node state is the available bandwidth as a fraction of the maximum capacity (one call consumes one bandwidth unit).

Additionally, agents update an internal local model of the network using the received message with the involved nodes’ states, and assigning an accuracy factor to the involved nodes according to their relative distance. This way, agents create a probabilistic model with the estimation of end-to-end available bandwidth for further calls. Adaptation is achieved by making agents to have an as-most-as-possible accurate local model of the network. And therefore allowing them to decide different routes for new calls according to the estimated accuracy of the model.

A.2.14 Dialogue adaptation

Dialogue adaptation [Nguyen and Wobcke, 2006] allows a smart personal assistant (SPA) to tailor responses according to conversation context, and user’s device and preferences. A SPA is an application that allows users to have a spoken conversation with the system for manipulation of their e-mail and calendar. The SPA consists of a BDI agent acting as a dialogue manager, and a set of task assistant for handling domain-specific tasks such as e-mail checking. The manager agent holds a collection of plans for processing user requests. However, the dialogue control flow is not specified in advance, but derived using a plan selection mechanism.

Thus, the problem is to learn which plan best suits for a given context and execute it. A meta-level reasoning tool is utilised to generate possible plans. Then,
for all possible plans, the user is asked to select the most preferred one for that situation. Next, the selected plan, along with information of that particular situation, is used to feed a decision tree learner. The information of the situation consists of the device the user is utilising, the interaction mode (either GUI, text or speech), the task, and its priority. Thus, the SPA learns which plan to execute according to parameters of the current situation. In other words, it adapts itself by making an approximation to the working preferences of the user.

A.3 Semi-isolated evolution

A.3.1 Genetic learning

Genetic learning [Grefenstette, 1992] is used to adapt agent sets of strategies for multi-agent environments. A strategy is a set of rules where each rule has an associated value for utility estimation. Each strategy is codified as a chromosome and the whole strategy population is evolved using a derivated form of Genetic Algorithm (GA). In this case, six mutation operators are used: mutation and creep (make small changes at rule level), specialise (modifies firing rule parameters to closely match current conditions), generalise (modifies partially activated rules to cover current conditions), merge (combines rules that have the same right-hand side), and delete (suppresses rules with low activity level, low strength, and covered by rules with higher strength). The first five mutation operators produce new individuals (i.e. strategies) which compete with the parents in the next generation. This evolution phase is independent from the performance phase, i.e. when the evolution of strategies occur agents are as if isolated from the environment. In addition, each agent acquire information from the environment as discrete values and other agents are considered of the same type.

A.3.2 The evolution of MAS structure

The evolution of MAS structure [Vacher et al., 1998] concedes the creation of new agents using the idea of a GA. The three main GA operators (selection, crossover and mutation) were changed in order to be applied on agent behaviours and actions. Agents are selected according to current performance, then their behaviours are “crossed over” to produce new agents. Finally, a random mutation process is applied to the new generation of agents. In this way, the original MAS behaviour is adapted to create a better one throughout generations to solve optimisation problems. In the given example (job-shop problem with M machines and N jobs), two types of agents are needed: local agents for task-solving and global agents for task-solver allocation.
A.3 Semi-isolated evolution

A.3.3 The evolution of behavioural rule-sets

The evolution of behavioural rule-sets [Bassett and De Jong, 2000] permits homogeneous micro air vehicles (MAVs), i.e. agents, to cooperate in surveillance tasks. Agent's behaviour is depicted by a population of stimulus-response rule sets. Each individual in the population (i.e. an entire rule set) is represented by a chromosome as in a standard GA. Stimuli, like distance to other MAVs, are acquired by sensors and responses are actions such as speed and turn angle used to avoid collisions. Rule sets are evolved by a modified GA after each simulation in order to maximise coverage area in following runs.

A.3.4 Anthill framework

Anthill framework [Montresor et al., 2002] is used to support peer-to-peer applications by means of ant-like agents for computation load balancing. A set of tasks is considered distributed in many network nodes. A swarm of agents wander across the network looking for overloaded nodes. After certain amount of time, they start seeking for under loaded nodes. The information they collect are left in the nodes themselves for other agents to use, so interactions are made throughout the environment. Then, agents reallocate tasks among nodes they have visited according to differences in workload. Once tasks are done, results are sent back to the source node. New nodes in the network are considered, but no node interruption is taken into account whilst tasks are executing. A GA is used for ant parameter tuning at certain time intervals.

A.3.5 Ethogenetics

Ethogenetics [Landau and Picault, 2003] allows the evolution of agent behaviours by combining the benefits of both GAs and Genetic Programming (GP). An implementation of the approach called ATNoSFERES is presented and it consists of Augmented Transition Networks (ATNs) where edges link states and can be labelled with conditions and actions. ATNs are used for agent's behaviour representation and are created from bit strings. The general model consists of an initial population of agents having their own behaviour represented as a bit string. Then, for each agent a translator generates tokens from its bit string. These tokens are then interpreted as commands and data to build an ATN which in turn is used to guide the agent's behaviour. Then, agents are selected according to their performance in the environment and evolved using a classic GA.
A.3.6 The DIET toolkit experiments

The DIET toolkit [Marrow et al., 2003] is used in experiments of emergent group formation. They set a scenario where users have an interest category and they look for other users who share the same interest, and therefore the same information. Each user creates a DIET environment with several mobile agents which in turn know their user's interests. Agents navigate through other available environments whilst assessing each of them in order to determine the best one with more available resources. Then each agent moves to the preferred environment and spend a specific time there sharing user information with other agents and acquiring information from other users. Agents store all these data (preferred environment and agents inhabiting within it) in an internal chromosome. Next, agents return to their original environment for an evolution process, i.e., a GA is applied to all agents, so that a new population of agents is generated with biased information about preferred environments and agents living within.

Then the process starts again and continues until the agent population converges to set of most preferred environments. As a result, agent groups are formed whilst optimising resources across all environments. The problem with these experiments is that the ecosystem idea was addressed using a classical evolutionary computation approach, i.e., a GA, and because of it these experiments go outside the Ecosystem class and fall into the Semi-isolated Evolution group.

A.3.7 Biological ecosystems

Biological ecosystems [Vandecasteele et al., 2004] can be built using evolutionary computation. In essence, it is suggested to codify the presence or absence of diverse organism types within a single bit string, i.e. a 1 when a species is present and a 0 when it is not. Having this way a whole ecosystem codified in a single chromosome. So, the set of chromosomes represents a population of ecosystems. Then evolve all ecosystems using a standard GA. In the end, the intention is to get the set of species, or ecosystem, that optimises a specific function whilst paying minimum attention to the environment.

A.3.8 An advice-exchange mechanism

An advice-exchange mechanism [Nunes and Oliveira, 2005] in the pursuit domain allows learning agents to improve their performance by sharing information. Predators are considered either as GAs combined with neural networks or as Q-learning agents. The environment consists of a grid with several predators and a prey they have to catch. Predators request for and give episodic advice to peers in order
to learn from others’ experience. After each trial, own and others’ experience are used to produce better performance for following trials.

A.3.9 Evolution of strategies

Evolution of strategies [O’Riordan, 2005] for playing the IPD game in noisy environments allows agents to adapt their strategies throughout generations. All agents have the same pair of actions (i.e. cooperate or defect), but strategies are different from each other. A strategy consists of a sequence of actions an agent may play. Each agent has a set of strategies which are evolved offline using a standard GA, i.e. after playing IPD using the same set of strategies the GA is utilised to improve the set for next runs. Nothing in the environment is affected whilst strategies are evolving.

A.3.10 Bottleneck neural networks

Bottleneck neural networks [Paine and Tani, 2005] are used to train a robot’s behaviour (cf. agent’s behaviour) whilst navigating in a maze. The robot’s task consists of finding different goals from an initial position. For this purpose, the robot must obtain different skills such as collision-free manoeuvring whilst turning at corners, and navigation through the maze by sequencing turns when at corners as well. A continuous time recurrent neural network with bottleneck is used to control the wheels-motor outputs. The network receives its information through specialised neurons for sensory input and initial task processing. The synaptic connectivity weights, which are the network output producers, are adapted by means of a standard GA with a population consisting of 80 robots. The evolution of robots’ behaviours is made in a separate phase from operation.

A.4 Complex interactions

A.4.1 Emergent design system

Emergent design system [Voss, 2000] is the combination of Complex Adaptive Systems (CAS) and soft computing for adaptive construction of structures (e.g. buildings). Adaptive agents represent different types of basic building blocks, such as beams and columns, and their interactions allow the emergence of higher order building blocks like walls and rooms from the bottom to the upper levels of the structure. The interactions represent both information and connection mechanisms between every basic building block, so that when a new one is added to the structure it propagates its impact through all interactions (i.e. connections) already established
in the structure. This impact is analysed locally and it can trigger a re-organisation process at different levels. There is no mention of environment limitations, e.g. the space and form required for the final structure nor the limitation of basic building blocks.

A.4.2 Coordination artefacts

Coordination artefacts [Cannata et al., 2004] can be used to provide means of interactions among agents of different types for biological system simulations. Agents developed under any paradigm and using any programming language can coordinate themselves as long as they can interact to each other. For this purpose mediating artefacts can be implemented either as communication channels, shared data structures, blackboard, or scheduler. The latter two can be considered as coordination artefacts to help agents to synchronise their activities.

In particular, coordination artefacts can be used in simulations to model interaction patterns found in biological processes. TuCSoN is a coordination infrastructure for MAS and is considered to provide support for biological system simulations. It consists of programmable shared tuple spaces, i.e. tuple centres, agents use for interaction. In other words, by using tuple centres agent interactions occur by inserting, retrieving and reading chunks of information, i.e. the tuples. Because tuple centres guide agent interactions, the former can be adapted dynamically according to events happening in the environment in order to adapt the general behaviour of a set of agents or even the whole system.

A.5 Ecosystems

A.5.1 ECHO

ECHO [Holland, 1995] is an ecosystem model consisting of agents living in an environment of spatially distributed sites. Each site contains a different fountain of diverse resources which allow the proliferation of agent diversity and agent complex interactions. An agent consists of a chromosome of eleven sub-strings grouped into two sets, one for agent interactions (external tags) and the other one for resource processing control (internal conditions).

According to the resources agents process from the environment, agents can either combat, trade or mate. During these interactions agents consume resources from the agents they interact with. The interaction type depends upon the internal conditions and the resources agents consume due to the current interaction. Because of these interactions, the whole population adapts itself until an equilibrium is reached.
As a consequence, complex relationships arise such as symbiosis, food chain, and aggregated individuals. In the latter, complex structures emerge by having agents forming coalitions which in turn may form larger organisms. The emergence of complex structures along with the site distribution grant emergence of different species. If a change occurs in the environment, the population adapts itself again in order to find a new equilibrium. Moreover, it was not specified whether the environment is considered as either discrete or continuous. Yet experiments on ECHO reported by Smith and Bedau [2000] show that the emergence of complex structures do not occur and it does not exhibit adaptation.

A.5.2 The DIET toolkit

A toolkit for agent application development DIET [Marrow et al., 2001] (Decentralised Information Ecosystem Technologies) is based upon the idea of ecosystem dynamics in order to overcome common MAS issues such as adaptability. It consists of a three-layered architecture from which the lower one is of our interest: the core layer. It contains different families of agents (or infohabitants) capable to communicate locally with peers and migrate from one environment to another. The communication process is local between any two agents without regarding the agent type they belong to.

The environment contains resources (memory and CPU) agents have to share. One computer may hosts more than one environment and several networked computers create a distributed world. When an environment (or a computer) becomes unavailable agents re-allocate themselves according to available resources in the remaining environments. Local interactions and agent migrations allow the whole population to adapt itself to resource variations and the availability of environments. Nevertheless a representative example of an application using the DIET development toolkit was not presented.

Summary

This chapter presents the literature review on adaptive MASs. The different approaches are presented according to the adaptation class they belong. And only those approaches specifically claiming adaptation are reviewed.