UNIVERSITÉ DE LORRAINE

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Une architecture de contrôle de systèmes complexes basée sur la simulation multi-agent.

THÈSE

présentée et soutenue publiquement le

pour l'obtention du

Doctorat de l'Université de Lorraine

(spécialité informatique)

par

Tomás Navarrete Gutiérrez

Composition du jury

Président :	Sylvain Contassot-Vivier	
Rapporteurs :	René Mandiau	Professeur, Université de Valenciennes
	Michel Occello	Professeur, Université Grenoble 2
Examinateurs :	Abderrafiâa Koukam	Professeur, UT de Belfort-Montbéliard
	Sylvain Contassot-Vivier	Professeur, Université de Lorraine
	Laurent Ciarletta	Maître de Conférences, Université de Lorraine
Directeur de thèse :	Vincent Chevrier	Maître de Conférences, HDR, Université de Lorraine

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Agradezco al Consejo Nacional de Ciencia y Tecnología de méxico por la beca de estudios que financió mi doctorado.

Au terme de ce travail, je tiens à exprimer ma profonde gratitude ainsi que mes sincères remerciements à Vincent Chevrier, pour l'encadrement de ce travail, ses conseils, confiance, et patience. Mes remerciements s'adressent également à Laurent Ciarletta, qui s'est montré à l'écoute et s'est impliqué profondément dans mon travail de thèse.

Je tiens également remercier sincèrement Messieurs René Mandiau et Michel Occello pour avoir accepté de rapporter mes travaux de thèse et par la même occasion à tous les membres du jury : Monsieur Abderrafiâa Koukam et Monsieur Sylvain Contassot-Vivier.

Je souhaite remercier très fort les précieux collègues doctorants de l'équipe MAIA qui m'ont rendus ces années très riches. Nous avons intégré une très bonne ambiance entre les doctorants ce qui m'a permit de faire ma thèse dans un cadre de travail très agréable. Arnaud, Jano, Sylvain, Olivier Bourre, Olivier Buffet, Boris : sachez que avoir partagé tant d'heures (au resto, à la cafet, tuant des zombies ou des ogres) fut un énorme plaisir pour moi. Je remercie également Alain et Vincent, avec qui j'ai eu le plaisir de partager un bureau et avec qui j'ai pu à plusieurs reprises discuter de tout et de rien.

En fin je voudrais remercier les membres de ma famille pour leur compréhension et leur soutien durant ces années. Une pensée très particulière pour Milo, qui a été mon moteur principal dans la dernière ligne droite de la thèse.

Je dédie cette thèse
à Aurore.

Índice general

Introduction

Chapter 1
Complex Systems

1.1. Complex Systems Definition 5
1.2. Relevant Characteristics 7
1.3. Examples 10
1.4. Challenges in the study of complex systems 14
1.4.1. Modeling 14
1.4.2. Engineering 15
1.4.3. Control 15
1.5. Difficulties of control of complex systems 16
1.5.1. Local actions with global effects 16
1.5.2. Entities autonomy 16
1.5.3. Modeling 17
1.5.4. Preexisting systems 17
1.6. Governance and control of complex systems 17
1.7. Synthesis 18

Chapter 2
Related Work

2.1. Introduction 21
2.2. Control Theory 22
2.3. Equation Free approach 24
2.4. Modeling complex systems 26
2.4.1. Multi-agent paradigm 26
2.4.2. Agent-Based Models 27
2.5. Applications of multi-agent paradigm in the control of complex systems 31

2.5.1. Organic Computing 31
2.5.2. Control of Self-Organizing systems 33
2.5.3. Prosa 34
2.5.4. PolyAgent 36
2.5.5. Morphology 37
2.5.6. Emergent Engineering 39
2.5.7. Control of reactive multi-agent system with reinforcement learning tools . 40
2.6. Synthesis 40
2.7. Conclusions 41

Chapter 3
A control architecture

3.1. Introduction 43
3.2. Principles 44
3.3. Definition of the architecture 45
3.3.1. Architecture elements 45
3.3.2. Hypothesis of the architecture 46
3.3.3. Concise definition 47
3.3.4. Preliminary synthesis 47
3.4. Detailed view 48
3.4.1. Block Definition Diagram of the architecture 49
3.4.2. Internal Block Diagram of the architecture 51
3.4.3. Execution flow of the architecture 51
3.4.4. Complementary view of the elements of the control architecture 52
3.5. Assessment on the relevance of our proposition 54
3.5.1. Architecture elements and difficulties of the control of complex systems challenge 54
3.5.2. Equation free approach and multi-agent models in the architecture 55
3.5.3. Governance related aspects 56
3.6. Conclusion 57

Chapter 4
Proof of concept Implementation

4.1. Introduction 60
4.2. Peer-to-Peer networks as complex systems 60
4.3. Target system specification 61
4.3.1. Description 61

4.3.2. Target system state 61
4.4. Target system implementation 61
4.4.1. Internal behavior of the peers 62
4.4.2. Number of peers in the network 63
4.4.3. Network organization 63
4.4.4. Initial proportion of sharing peers 63
4.4.5. Update Scheme 63
4.4.6. Summary 63
4.5. Preliminary study on the behavior of the target system 64
4.5.1. Analytical point of view 64
4.5.2. Beyond the analytical model 66
4.5.3. Results and conclusions on the preliminary study 66
4.5.4. Retained Initial conditions 66
4.5.5. Focus on the behavior of the target system associated to retained initial conditions 68
4.6. Experimentation platform 69
4.7. Experimental Setup 70
4.7.1. Control Objective 70
4.7.2. Architecture implementation overview 70
4.7.3. Architecture implementation details 71
4.8. Experiments and Results 74
4.8.1. Description of experiments 75
4.8.2. Synthesis of results 76
4.8.3. Discussion on the results 76
4.9. Discussion 78
4.10. Conclusions 79

Chapter 5
Multi-agent simulation questions inside the architecture

5.1. Introduction 81
5.2. Initialization of models 82
5.2.1. Target systems implementation 82
5.2.2. Nominal Behavior 83
5.2.3. Experimental setup 86
5.2.4. Experiments and Results 88
5.2.5. Discussion 91
5.2.6. Conclusions 91

5.3. Model selection 92
5.3.1. Target system implementation 92
5.3.2. Experimental setup 92
5.3.3. Experiments and Results 93
5.3.4. Discussion 93
5.4. Conclusions 94

Chapter 6
Conclusion

Bibliography **97**

Annexe A
Résumé étendu : Une architecture de contrôle de systèmes complexes basée sur la simulation multi-agent.

A.1. Les systèmes complexes 112
A.1.1. Caractéristiques de Systèmes Complexes 112
A.1.2. Exemples de systèmes complexes 113
A.1.3. Les challenges dans l'étude des systèmes complexes 115
A.1.4. Gouvernance et contrôle de systèmes complexes 116
A.1.5. Difficultés du contrôle de systèmes complexes 116
A.2. Travaux connexes 117
A.2.1. Modélisation des Systèmes complexes 117
A.2.2. Le paradigme multi-agent et les systèmes complexes 120
A.2.3. Théorie du contrôle 120
A.2.4. Approche « equation-free » 121
A.2.5. Applications du paradigme multi-agent au contrôle de systèmes complexes 122
A.2.6. Synthèse 127
A.3. Proposition 128
A.3.1. Principes 128
A.3.2. Vue Générale 129
A.3.3. Flux d'exécution de l'architecture 130
A.3.4. Des principes à l'implémentation 130
A.4. Preuve de concept 133
A.4.1. Introduction 133
A.4.2. Les réseaux pair-à-pair comme systèmes complexes 133
A.4.3. Scénario expérimental 134

A.4.4. Implémentation de l'architecture 135
A.4.5. Expériences 136
A.4.6. Résultats 138
A.4.7. Discussion 139
A.4.8. Conclusions 140
A.5. Questions de la simulation multi-agent dans l'architecture 140
A.5.1. Scénario expérimental 140
A.5.2. Implémentation de l'architecture 141
A.5.3. Expériences 142
A.5.4. Résultats 142
A.5.5. Discussion 143
A.5.6. Conclusions 143
A.5.7. Selection des modéles 143
A.6. Conclusion 145

Introduction

Context

Within the last twenty years, we have observed that there are many systems, both natural and artificial, with a large number of participants that exhibit some characteristics at the group level that cannot be identified at the individual level. These are called "complex systems". The characteristics of complex systems are: a large number of autonomous entities, sensitivity to initial conditions, different organization levels, dynamic structures, emergent properties and different time and space scales within the system.

Examples of systems called complex usually cited in the literature are: the omnipresent internet, groups of insects, the economy, the human brain, electricity distribution networks, transport networks, etc. A new kind of science has emerged to deal with such systems. This new kind of science, known as complexity science is an interdisciplinary framework harnessed by the advances in different scientific fields such as chaos theory, sociology, nonlinear dynamics, biology, ecology, statistical mechanics, graph theory, thermodynamics, probability theory, numerical simulation and others (Bar-Yam, 1997).

One sign of the importance of this newborn science is the creation of research institutions dedicated to the study of complex systems. Some of these institutions, like the Santa Fe Institue and the NECSI in the United States, the ISCV in Chile, the ISCPIF and the IXXI in France, are not directly dependent on research programs of a specific university. Others, are departments within a university like The Department of Complexity Science and Engineering in the university of Tokyo in Japan or C3 in the UNAM in Mexico. Finally, some of them are "virtual" networks of researchers interested in complex systems, as the RNSC or the CSS and the CSIRO. Another sign of the importance of complexity science is the wide spectrum of contexts where complexity has been studied as a main concern (Nature insight, 2001; La Complexité, 2003). This spectrum ranges from nature, through human activities and up to artificial or man-made systems. Within the natural context (Camazine et al., 2001; Bak and Bak, 1996), there are physical processes (Nicolis and Prigogine, 1977), genetics (Kauffman, 1993), natural disasters (Ball, 2004), epidemics (Bolker and Grenfell, 1993), and ant foraging (Nicolis and Deneubourg, 1999). In the human activities context (Castellani and Hafferty, 2009) there are economy (Anderson et al., 1988) and management (Van Eijnatten, 2005). In the artificial or man-made context there are cellular automata (Wolfram, 1994; Langton, 1986) and communication networks (Kocarev and Vattay, 2005).

Yet, there is no formal, "one simple statement", commonly accepted definition of a complex system (Mitchell, 2009; Grobbelaar and Ulieru, 2007; Nicolis and Nicolis, 2007; Boccara, 2004).

Defining complexity or complex systems is by itself a challenge, as Horgan (1995) already noticed. Lloyd (2001) has found no less than forty-five different ways to measure (and thus define) complexity. Definitions in literature are as varied as the contexts within which they have been formulated. For Edmonds (1999) for example, complexity is

> "that property of a language expression which makes it difficult to formulate its overall behaviour, even when given almost complete information about its atomic components and their interrelations."

Given our interest in the multi-agent modeling within this thesis work, we could have been compelled to work with the definition given in the preface of (Boccara, 2004):

> "Although there is no universally accepted definition of a complex system, most researchers would describe as complex a system of connected agents that exhibits an emergent global behavior not imposed by a central controller, but resulting from the interactions between the agents. These agents may be insects, birds, people, or companies, and their number may range from a hundred to a million."

We agree with (Ladyman et al., 2012), that the problem of defining complexity science and even complex systems may benefit from the work of the philosophy of science. This is an objective out of the scope of this thesis. We shall focus in this work not on the definition of a complex system but rather on the characteristics of complex systems. In this way, we expect our work to be applicable to systems sharing characteristics with those of complex systems may they be natural or artificial, man-made or man-centered. Our work takes place within the context of complex systems to build up from preexisting work and not be forced entirely to create the context. We shall focus on one particular challenge that arises when systems exhibit characteristics inherent to complex systems: "control".

Control challenge

We have observed that there are complex systems all around us but also, we have also noticed that given some of their characteristics, their behavior is not always the one sought or desired. Undesired phenomena within them, have been observed : energy outages (Newman et al., 2011; Pourbeik et al., 2006), internet outages (Dainotti et al., 2011; Smith, 2011; Paxson, 2006), recurring financial crises (Fender et al., 2012; Vivier-Lirimont, 2008; Llaudes et al., 2010; Artus and Lecointe, 1991), transport networks traffic bottlenecks (Lee et al., 2011; Xu et al., 2011; Kerner and Klenov, 2009).

In this thesis work, we shall focus on the challenge of driving a complex system to exhibit a certain behavior. Control theory assumes that given a model, an optimal way to control a target system exists. This optimization is based on using analytical models. However, we consider that such models are rarely useful in the context of complex systems. It has been recognized that given the characteristics of complex systems the concept of governance is better suited (Chavalarias et al., 2009) than control. Multi-agent modeling and simulation have been identified as suitable tools for complex systems (Phan and Amblard, 2007a). Based on these elements we will tackle the challenge of control of complex systems.

We propose an architecture to control a complex system based on an equation-free approach with multi-agent model simulation. In our proposition, we will integrate different existing elements such as the equation-free approach, the multi-agent paradigm and the basic concepts of control theory. The objective of this thesis work is to evaluate the implications of integrating all these elements in one coherent architecture. The evaluation takes place at two levels: find out if the proposition is feasible and know how to compare the answers given to the questions implied in using the multi-agent paradigm in our context.

Thesis plan

Chapter 1. Presents our working definition of complex systems and explains characteristics of complex systems relevant to our work. In this chapter we describe the current challenges of the study of complex systems such as engineering and design, modeling and control. We elaborate on the specific difficulties of the control challenge.

Chapter 2. Different approaches existing in the literature to deal with the control of complex systems are presented.

Chapter 3. Our proposition on a control architecture to govern a complex system based on an equation-free approach with multi-agent model simulation is detailed in this chapter. We state the different hypothesis under which our architecture is conceived and the principles used to design it. The architecture is initially presented from a coarse point of view to establish the links between the difficulties of the control challenge previously defined and the elements of the architecture that specifically deal with them. Then, we present a more detailed description that allows to identify different decisions that must be made when implementing our architecture. We pay special attention to the questions implied by using the multi-agent paradigm in our proposition, from an abstract point of view.

Chapter 4. A first implementation of our architecture made within the context of the free-riding phenomenon in peer-to-peer file sharing networks is presented in this chapter. We define an experimental environment under which we drive a complex system to exhibit a desired behavior through the usage of our architecture.

Chapter 5. We show within the same experimental context as the previous chapter, different answers given to questions of validation, calibration and translation of multi-agent models in our architecture.

Chapter 6. We present a synthesis of our findings and identify different future directions and development of our work.

Chapter 1

Complex Systems

Contents

1.1. Complex Systems Definition 5
1.2. Relevant Characteristics 7
1.3. Examples 10
1.4. Challenges in the study of complex systems 14
1.4.1. Modeling 14
1.4.2. Engineering 15
1.4.3. Control 15
1.5. Difficulties of control of complex systems 16
1.5.1. Local actions with global effects 16
1.5.2. Entities autonomy 16
1.5.3. Modeling 17
1.5.4. Preexisting systems 17
1.6. Governance and control of complex systems 17
1.7. Synthesis 18

1.1. Complex Systems Definition

In this thesis work we have decided to work with the definitions of complexity subjects (complex systems and their challenges) resulting of a collaborative effort, namely the "French Complex Systems Roadmap" of the French National Network for Complex Systems [1] (Chavalarias et al., 2009). However, this is not the only collective effort: (Rouse, 2007) for example, is the work of a group of thought leaders in the United States. One important difference between the two approaches, despite being collaborative efforts, is in the applications they keep as goals of their research plans. In the work of (Rouse, 2007) health-care, infrastructure, environment, security, and competitiveness are the key objectives of their research goals. Whereas in (Chavalarias et al., 2009) the objective of the collaborative work is to establish directions in research regarding a wide landscape of scientific disciplines (and at the same time, one science of complexity). Both collective works agree in aspects such as the importance of modeling and design of complex

[1] As can be seen in the website of the complex systems registry (CSS, 2012a), the French roadmap is also used as the basis for a European roadmap within the context of a European community research program.

systems as well as the phenomena of interest to study such as time and space inter-dependencies and different time and space scales in complex systems. We have selected (Chavalarias et al., 2009) as our reference framework because it directly addresses the challenge of control of complex systems.

Our working definition of a complex system is thus as follows.

Definition 1 *A "complex system" is in general any system comprised of a great number of heterogeneous entities, among which local interactions create multiple levels of collective structure and organization. (Chavalarias et al., 2009)*

This definition of a complex system focuses on three "core concepts", all of them at different levels of the system. We define the different description levels of a complex system as follows.

The core concepts included in the definition are:

- The characteristics of the entities belonging to the system: great number and heterogeneous.
- The mechanisms responsible for the global behavior of the system: interactions among entities.
- The structure of the global behavior of the system: multiple levels of organization and structure.

The core concepts and the different description levels defined above are related in the following ways. The first concept regards the microscopic or local level. It tells us what to look for when looking at the entities of a complex system. The second concept regards both the macroscopic or global level and the microscopic or local level. It tells us where to look for when looking at mechanisms producing the behavior of a complex system. At the global level, we have the behavior of the complex system. The mechanisms responsible for the global behavior are at the local level because they are rooted in the interactions of the entities. The third concept is completely at the global level. It tells us what to look for when looking at the structure and organization of the system.

We have also chosen this definition because it is the result of the joint work of a large group of researchers of major French research institutions (of different scientific fields), and not only authors of a book or members of a research institute or university department. Since this definition is the result of a collective effort, it has the advantage of being domain independent. From this point of view, we consider it useful for different scientific fields, like sociology, informatics or physics because it is stated at a relatively abstract level. It does not say for example that the constituent parts of a complex system are molecules, robots, customers of a cell-phone company or anything specific to a particular scientific field. Also, there is no specification of how many are "a great number" of entities, or how is the heterogeneity of the entities to be measured. This level of abstraction is necessary to study systems that share some characteristics but that do not belong to the same scientific field.

Although there is not one single definition of complex systems, many authors agree on common characteristics shared by complex systems. A quick look at the literature on complex systems shows that the characteristics of complex systems are not limited to the three core concepts from our definition, see for example Sitte (2009). We shall focus on some the characteristics of complex systems relevant to the difficulty of their study (Amaral and Ottino, 2004).

1.2. Relevant Characteristics

The following is a set of characteristics that we consider relevant to the difficulties of studying complex systems. The set of characteristics is not explicitly given within the cited framework.

Emergence

Its definition is usually directly linked to Aristotelian principle of "the whole is more than the sum of the parts". This means that the behavior exhibited by the whole system cannot be solely inferred from the behavior of its constituent elements. Emergence is one of the most controversial characteristics of complex systems (Fromm, 2004). The center of the controversy is the almost "magical" sudden appearance of a property of a system. The controversy sometimes takes the form of defining if emergence is only in the eye of the observer. Despite that such considerations are close to the study objects of philosophy, we agree with (Brückner, 2000) who points out that a general theory of this concept has not yet been found.

In our context, emergence is the implicit idea behind the second core concept of our definition of a complex system. An emergent property of a complex system is one that is observed at a level different from the local (global or any intermediary one) but that is not directly given at the local level.

Definition 2 *A phenomenon is emergent if and only if we have:*

- *a system of entities in interaction whose expression of the states and dynamics is made in an ontology or theory D;*
- *the production of a phenomenon, which could be a process, a stable state, or an invariant, which is necessarily global regarding the system of entities;*
- *the interpretation of this global phenomenon either by an observer or by the entities themselves via an inscription mechanism in another ontology or theory D'.*

(Müller, 2003)

Another concept related to emergence present in the French roadmap of complex systems (Chavalarias et al., 2009), but not explicitly given in the definition of complex system is *immergence*. Emergence can be seen as the influence of local interactions on the global outcome, but it is not the only influence originating at one level and having consequences at another level. There also is influence from the global outcome of a complex system executed at local level. This influence is known as *immergence*. Immergence is the process by which the global behavior of the system influences the way interactions are held among entities in a feedback loop.

The feedback loop means that the output of a system will be the input of the same system or process. In the complex systems context this means that the interactions among entities will eventually take as input the global behavior of the system, which at its turn is the output of the interactions among entities.

Different levels of description

At least two different levels to describe a complex system can be considered: the local or microscopic and the global or macroscopic. The description level of a complex system is said to be local or microscopic when it exclusively employs characteristics inherent to the entities of the

system. The description level of a complex system is said to be global or macroscopic when it employs characteristics of groups of entities of the system.

At the local level, we can define each of the elements of the system, as well as the way they interact with other elements. At the global level, we can characterize the system as a whole.

Additionally, if the elements of the system can be organized in hierarchies or groups, other intermediate description levels are possible.

Scale

Complex systems are made of entities that can be described at their local level. The entities interact with each other, at their own (local) time and space scale. However, when the system is observed at a "macroscopic" or global level, the view changes. The time and space scales at which the whole system evolves (that is, at global level) are orders of magnitude away from the local scales.

Sensitivity to initial conditions

Very small differences in the initial conditions of a complex system produce significantly different outputs at quantitative and qualitative level. A usual example of a system highly sensitive to initial is the chaotic Lorenz system. It is made of the ordinary differential equations:

$$\begin{aligned} \dot{x}(t) &= \sigma(y - x) \\ \dot{y}(t) &= x(\rho - z) - y \\ \dot{z}(t) &= xy - \beta z \end{aligned}$$

When $\rho = 28, \sigma = 10, \beta = \frac{8}{3}$ the system has chaotic solutions. The evolution of the x variable with different initial conditions is illustrated in figures 1.1 and 1.2.

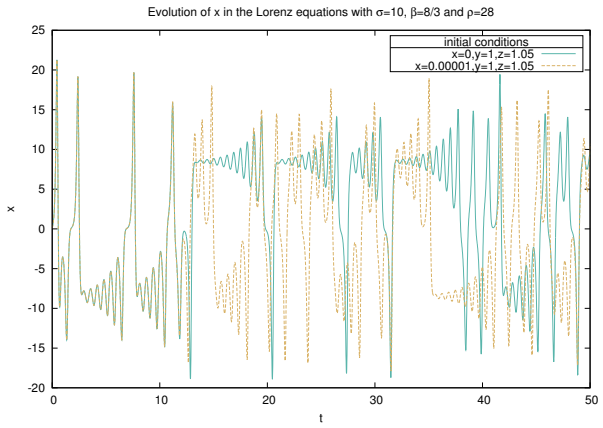
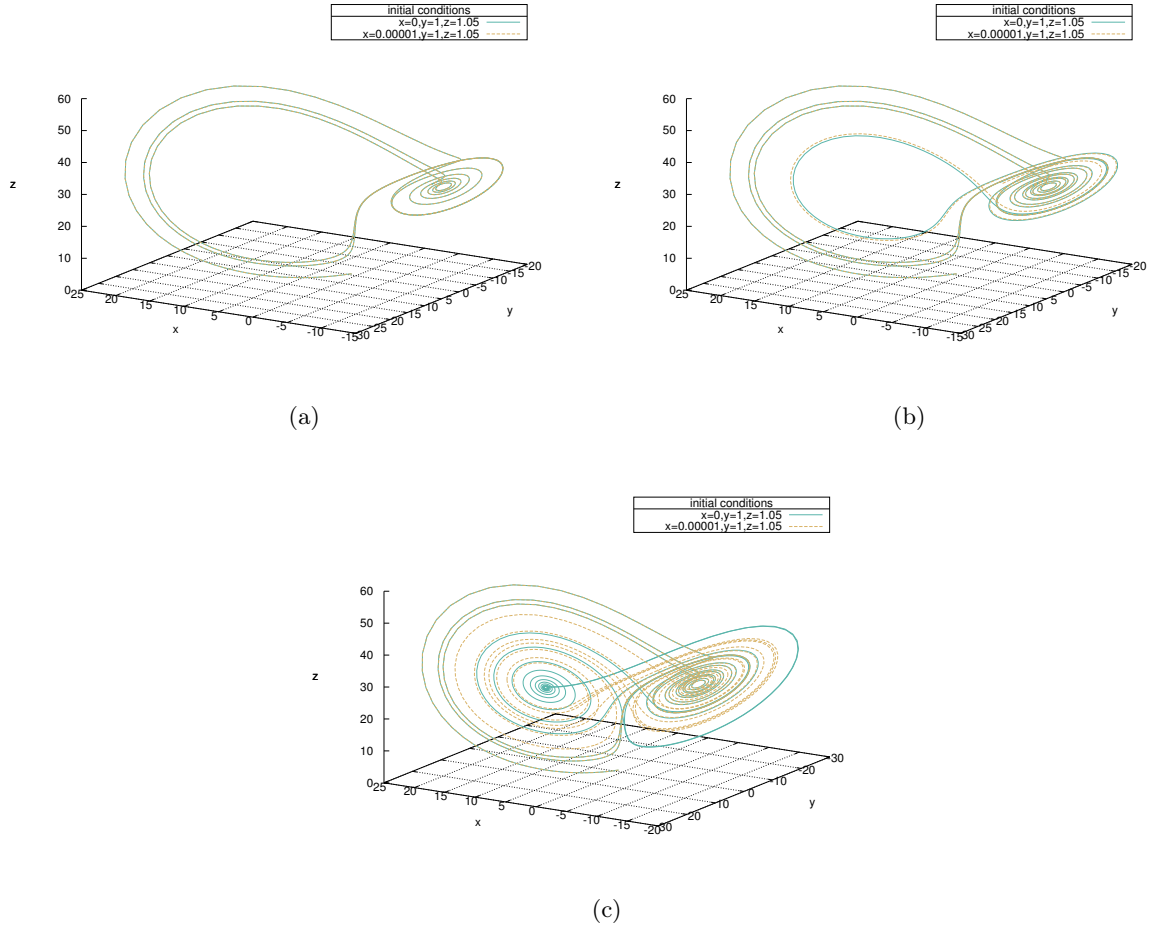


Figure 1.1: The evolution of x in the Lorenz equations, with initial conditions differing only in the x variable by 1×10^{-05}.

Nonlinear dynamics

The principle of superposition is the basis to differentiate between linear and nonlinear dynamics of a system. The principle states that, in a linear system, the net response at a given



(a) (b)

(c)

Figure 1.2: The evolution of Lorenz system of equations with two initial conditions. In 1.2a the system follows the same trajectory, after some time (400 more iterations of the system) we see in 1.2b that the trajectories begin to take different directions and, in 1.2c, we see that after 2400 iterations the trajectories are completely different.

place and time caused by two or more stimuli is the sum of the responses which would have been caused by each stimulus individually. In other words, the sum of the parts equals the whole. A nonlinear system is one whose output is not directly proportional to its input. In other words, the dynamics cannot be expressed as a sum of the behaviors of its parts.

In complex systems the sensitivity to initial conditions plus the nonlinear dynamics means that a small perturbation may cause a big effect, or even no effect at all. In linear systems, effect is always directly proportional to cause.

Dynamic structures

The structures in a complex system are the product of interactions between entities of the system. Like the interactions, the structures in the complex system evolve over time.

Structure may refer to the way interactions take place, or the way the entities are organized.

The object of study of this work are complex systems. Earlier we have presented our working definition of complex systems as well as some characteristics relevant to the study of complex systems. However the characteristics are not necessarily part of the original context of our definition. Furthermore, the set of characteristics we presented is voluntarily not exhaustive. We take a look now at the relationship between the core concepts of our complex system definition and the relevant characteristics we are interested in as well as other characteristics found in the literature.

Definition core ideas and characteristics

Our work is based on a definition of a complex system which is given in terms of three core concepts. Some characteristics of complex systems can be directly linked to such core concepts, like the different levels of description and emergence. However, other characteristics like dynamic structures and organizations at the different time and space scales are only indirectly linked to the three core concepts. The importance of the characteristics we focused on, regarding the control challenge, is what guided us in selecting the list of characteristics. This importance shall be highlighted through this chapter. In figure 1.3 we illustrate the previous links between the three core concepts of the definition of a complex system with the characteristics of complex systems.

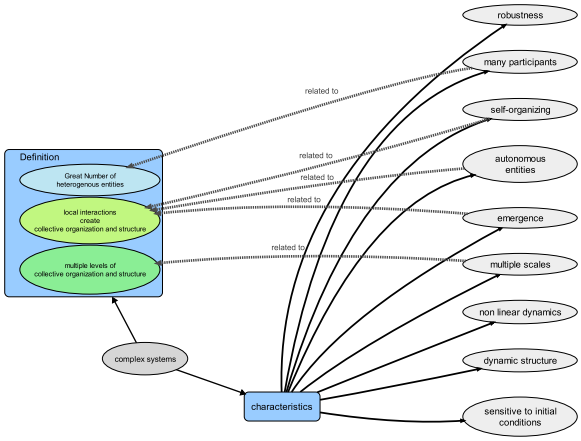


Figure 1.3: Relationships between core concepts of complex systems definition used in this thesis and the characteristics of complex systems

So far we have given characteristics of complex systems, from an abstract point of view. In the following section, we provide examples of complex systems in different contexts to identify what the abstract definitions refer to, in concrete cases. Two of the examples are present in nature and the other two are artificial, or man-made.

1.3. Examples

Natural

Ant foraging

Social insects like ants, bees or termites are known for accomplishing collective tasks such as nest building, defense and foraging. If we consider an ant colony as a complex system, the entities

composing it are the ants. Ants in a colony organize to find food and bring it to the colony. At the local level, ants are rather simple organisms and at least their communication means are limited. However, at the global level, an ant colony is capable of collectively producing emergent phenomena. An example of an emergent behavior in ant colonies is the foraging phenomenon (Deneubourg et al., 1990). Experiments have allowed to observe how by the usage of pheromones, ants communicate the position of a food source. The initial experimental protocol consists in establishing two different "bridges" from the nest to a food source. The results of the experiments show that eventually, the ant colony collectively chooses only one of the two paths to the food source.

The Brain

One way to conceive the brain as a complex system is by considering that its entities are the neurons. There is a large number of neurons (and other kinds of cells) in a human brain. At the local level of description, neurons are the basic elements of the brain. Additionally, they can be subdivided into compartments, synapses, channels, molecules and so forth, with relevant physiological and dynamical properties on all levels (Olbrich et al., 2011a). At the global level, the brain is capable of producing emergent behavior. Different time scales exist in the brain spanning several orders of magnitude: synaptic processes may occur within sub-milliseconds to milliseconds whereas cognitive processes occur in fractions of seconds to perhaps hours, and memory phenomena last for seconds to years. It can be assumed that processes inside the brain are not isolated acts but instead they are embedded into a network of intricate mutual dependencies across time (Olbrich et al., 2011a). However, it is still not understood what neural signals mean or how they give rise to global cognitive behaviors. And all this despite the vast knowledge on the structure of neurons and their interactions (at the chemical level) with other neurons. Most neural dynamics models are basically nonlinear because of voltage-dependent ion channels, firing thresholds, nonlinearities in synaptic transmission and other mechanisms. Some models such as rate-based ones, describe neural behavior in terms of average rates of action only. Within these models, the firing rate of a neuron is given by a sigmoidal function of the membrane potential. Successful usage of these models depends on overcoming a serious obstacle. Because of the large number of neurons in the brain, neural dynamics happens in a very high-dimensional state space, while the tools of nonlinear dynamics are easily applicable only to low-dimensional or moderately high-dimensional systems (Olbrich et al., 2011b).

Artificial

Internet

Internet is one of the biggest man-made techno-social system. "Modern techno-social systems consist of large-scale physical infrastructures (such as transportation systems and power distribution grids) embedded in a dense web of communication and computing infrastructures whose dynamics and evolution are defined and driven by human behavior" (Vespignani, 2009).

Internet is made of a very large number of heterogeneous entities: computers, servers, mobile phones, internet things (home appliances, cars) ... (Willinger et al., 2002; Park, 2005). As a complex system, we could consider that at the local level we have every device which is capable of communicating with any other device in the internet. However, a more accurate description would involve several levels of organization among the different entities: routers, autonomous systems, local area networks, wide area networks, tier "x" level networks, etc.

Internet as a complex system has different space levels. Internet allows communication from one place on the world, to virtually everywhere else (provided internet connection exists). But, also, communication is possible between entities connected to the network that are just next to each other. Communication can take place at hundreds of thousands of kilometers of distance, or at mere some meters. The current speeds of communication in internet allows to communicate almost instantaneously. Posting a comment on a website can take milliseconds or seconds before it is publicly available all over the world, however the time necessary before other entities acknowledge this novelty can take minutes or even days.

Cellular Automata

A cellular automaton is a collection of "colored" cells on a grid of specified shape that evolves through a number of discrete time steps according to a set of rules based on the states of neighboring cells. The rules are then applied iteratively for as many time steps as desired (Weisstein, 2012).

If we consider that a cell is an entity, we can associate as follows the characteristics of a cellular automaton to those of complex systems. The set of rules can be seen as the interactions between entities. Cells change their color based on the interactions with other neighboring cells. No specific cell dictates how all the other cells will behave, which means that the entities are autonomous. The rules are given in terms of the entities, and thus the global behavior of the whole automaton cannot be deduced directly from the entities. A rule for a cell can be stated as: "for current time step, if all neighbors have color black, at next time step, the current cell will have color black". Elemental cellular automata are "lines" of cells, where each cell only has two neighbors. The boundary conditions of the line are "toric", meaning that the first cell has as neighbors the second cell and the last cell. A Wolfram rule specifies the next color in a cell, based on its color and the color of its immediate neighbors. Given the boundary conditions of the line of cells, each cell only has two neighbors. A neighborhood is made of a cell and its two neighbors. There are thus $2^3 = 8$ possible configurations for a cell. We can consider each of the possible states as a three bit word, where a bit in 1 means the cell is full (colored in blue for example) and a 0 bit means that the cell is empty (or colored in white). State 010 corresponds to the following neighborhood of cells: □■□.

The outcomes of the rule are encoded in binary. For example, for rule 30 the outcomes are 00011110. An outcome of 1 means that at the next time step the cell will be colored, and an outcome of 0 means that at the next time step the cell will be white. Each binary digit of the outcome corresponds to a neighborhood configuration. We can label the bits in the rule from right to left (from least significant to most significant) as outcome $0, 1, 2, 3, 4, 5, 6, 7$ and then associate each label with a neighborhood. The decimal representation of the neighborhood configuration corresponds to a label of outcome. For example, with rule $30 = 00011110$, neighborhood □■□ with binary representation 010 and decimal representation 2 will be associated to outcome with label 2 which is 1 (meaning a colored cell). The whole list of neighborhood configurations and outcomes for rule 30 are:

Neighborhood configuration	7	6	5	4	3	2	1	0
	■■■	■■□	■□■	■□□	□■■	□■□	□□■	□□□
	↓	↓	↓	↓	↓	↓	↓	↓
Next color of cell	□	□	□	■	■	■	■	□
Outcome	0	0	0	1	1	1	1	0
Outcome label	7	6	5	4	3	2	1	0

Figure 1.4 illustrates the evolution of cellular automaton with Wolfram rule 30 after 15 time steps.

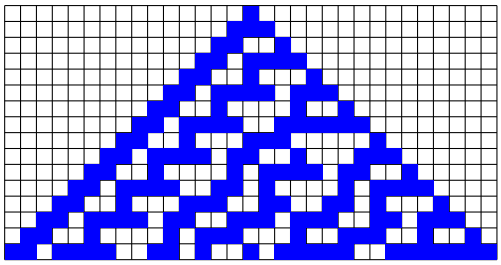
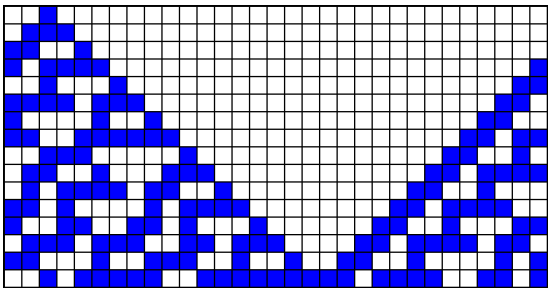
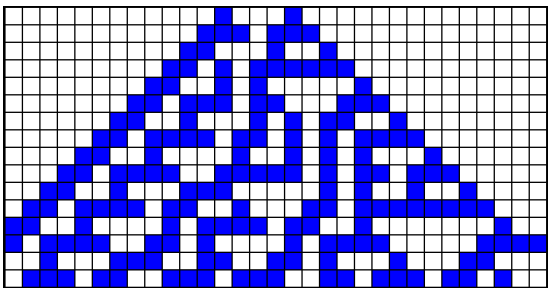


Figure 1.4: Evolution after 15 steps of Wolfram rule 30 with a single colored cell (Wolfram, 2002, page 55). The initial state is at the top, the final state is at the bottom.

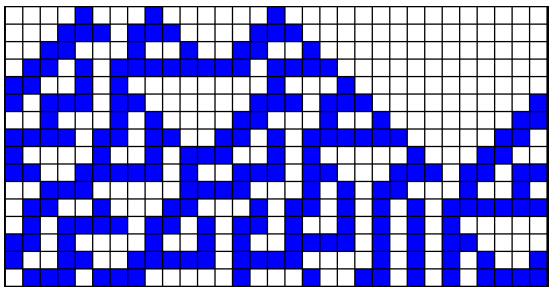
In cellular automata we observe the sensitivity to initial conditions by simply changing the amount of initially colored cells in the same automaton or by changing the spatial distribution of the initially colored cells. Figure 1.5 illustrates this with the cellular automaton of Wolfram rule 30.



(a) Same number of initially colored cells as figure 1.4, but different spatial distribution.



(b) Only two colored cells in initial condition.



(c) Same number of initially colored cells as figure 1.5b but with different spatial distribution.

Figure 1.5: Illustration of the sensitivity to initial conditions of Wolfram rule 30. The initial state is at the top, the final state is at the bottom.

There is no clear way to identify the time and space scales of a cellular automaton. Under a computer simulation, it takes from just a few time steps up to hundreds to observe patterns that can be interesting at the global level.

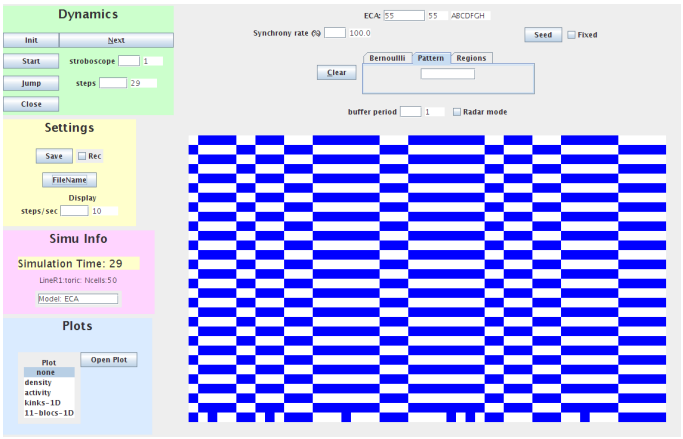


Figure 1.6: The elemental cellular automaton with Wolfram rule 55 after 29 time steps. The initial state is at the bottom, the final at the top. Obtained with FiatLux (Fates, 2012)

Interactions usually consist of observing the color of neighboring cells and then change its own color. If a cellular automaton is simulated with a computer we can assume that the interaction takes very little calculation time. A generation in a cellular automaton is when all cells in the automaton have changed their color.

(Hanson, 2009) portrays the presence of emergence in cellular automata as follows:

> In cellular automata, the system's state consists of an N-dimensional array of discrete cells that take on discrete values and the dynamics is given by a discrete time update rule. The "phenomena" that emerge in CA therefore necessarily consist of spatio-temporal patterns and/or statistical regularities in the cell values. [...] Possibly the simplest type of emergent phenomenon in CA is synchronization, which is the growth of spatial regions in which all cells have the same value. A synchronized region remains synchronized over time (except possibly at its borders) and it may either temporally invariant (i.e., the cell values to not change in time) or periodic (the cells all cycle together through the same temporal sequence of values).

If we simulate the elemental cellular automaton with Wolfram rule 55, we can see an example of synchronization, and thus emergence as previously stated. This is illustrated in figure 1.6, where synchronized regions with temporal period 2 emerge from random initial conditions.

1.4. Challenges in the study of complex systems

Based on our reference framework (Chavalarias et al., 2009), we present below some of the current challenges of the study of complex systems.

1.4.1. Modeling

A model is any physical, mathematical, or logical representation of a system, entity, phenomenon or process. In the scientific context, a model serves a specific purpose: a model exists as long as we can answer to certain questions regarding the modeled object by observation and manipulation of the model (Minsky, 1965). Models are the scientific tools used when it is either impossible or impractical to create experimental conditions where outcomes can be directly measured.

There are three elements to the modeling challenge of complex systems: the nonlinear nature of their dynamics, the different levels of description of complex systems and the availability of big amounts of information observed from complex systems.

Analytical models of complex systems have a very limited usefulness. That is, although it is possible to create an analytical model of a complex system, it will most often not be solvable either because of the nonlinearity or because of the high number of variables.

Complex systems have multiple levels of description, and models should take into consideration these levels. This is not feasible with analytical models because they only describe systems at the global aggregated level dynamics.

Nowadays, technological advance allows to have abundant information at very fine levels of complex systems. It is a challenge to find how to use this information to create models that can accurately represent the system.

1.4.2. Engineering

Human-made complex systems are designed to serve a purpose, to exhibit certain behavior. In other words, they are meant to work in a certain specific way. Classical engineering reaches its limits when dealing with complex systems (Gershenson, 2007).

A first difficulty of the engineering complex systems has to do with conciliating two seemingly contradicting aspects. One might want to build a complex system whose global behavior should exclusively emerge from local interactions, without directly inscribing in the entities what the global behavior should be. The question here is to find out what local interactions will produce the global output. That is, the global behavior is identified (we know what we want it to be) but the local interactions leading to it are not. Put in other words, the question is obtaining the desired output, despite the fact that the way the system arrived to that output that can not be known, is too complex or not computationally reproducible (Buchli and Santini, 2005). Another aspect of this difficulty is whether it is an acceptable assumption for all application domains to not know the mechanisms that lead a system to exhibit a certain behavior or produce a certain output.

Another difficulty is in building complex systems on top of other complex systems. Technological development has allowed to build systems that we consider as complex today, like internet. More and more systems are built on top of already deployed complex systems.

Finally, when engineering any kind of system, engineers are confronted to designing control mechanisms to make sure the system behaves as desired and as we shall see in the following section, control of complex systems is already a challenge by itself.

1.4.3. Control

To control an object means to influence its behavior so as to achieve a desired goal (Sontag, 1998). All along this thesis we shall call a "target system" a system that we wish to control. Traditional control theory assumes that a model of the target system is available and that it is analytically (or numerically) possible to use it. Control implies having a model and, as stated before, modeling complex systems is already a challenge by itself. If we assume, for example, that one way to "modify" the global behavior of a complex system is by influencing at the local level (because local interactions produce the global outcome), we need a model including those two different description levels in order to evaluate the effects of control actions as well as the evolution of the target system.

Moreover, given that complex systems are made of autonomous entities with different levels of description, the question of how to observe the system becomes a supplementary complication to achieve control.

1.5. Difficulties of control of complex systems

Certain characteristics of complex systems make their study particularly challenging (Amaral and Ottino, 2004; Helbing, 2007; Rouse, 2003; Bar-Yam, 2003). The control of complex systems is one of the current challenges of complex systems, as we stated in 1.4.3. We present in this section the characteristics of complex systems that we consider to be related to the challenge of control of complex systems. They are: the global effects of local actions, the autonomy of the entities of a complex system, modeling and, finally, dealing with preexisting systems.

1.5.1. Local actions with global effects

One of the three core ideas of our definition of complex system is that the structures and organization arises from the interactions between the system elements. These interactions happen at the local level, because they are held between the system entities. However, as emergent phenomena arise in complex systems, we can see that the repercussions of local actions (possibly control actions) will be seen at a global level. To clearly identify the nature of the influence is complicated because of thec sensitivity to initial conditions, and because of the nonlinear dynamics of complex systems. Thus any control action should be carefully designed with this in mind. However, due to the difficulty of modeling complex systems by using models considering at the same time the local and global level of description of the system, this is not a trivial task.

The difficulty in the context of complex systems control is that if we assume that control actions shall be applied at a given level (local for example), the repercussions will be observed at another level (global for example).

1.5.2. Entities autonomy

In some complex systems, it is not possible to tamper with the entities autonomy. Because of legal, ethical and technical reasons, it is impossible to directly modify the behavior of entities. For instance, it is not possible to consider a control action in a target system where changes to the internal working of the entities would mean to substantially alter them. We think for example of mechanical or electronic entities. And even if we could overcome the technical reasons, we would still need to have the "blueprints" or a good model of the entity, in order to efficiently modify its inner workings.

On the other hand, we can consider that it is eventually possible to change the inner workings of "inanimate" entities in a target system.

Additionally, autonomy of participants means that as time goes by, because of immergence and emergence, and also because no central entity in the system dictates how other entities should behave, the individual behavior may change, leading to changes at all levels. This is also a problem, if the new behavior is not considered in the "model" used in the control mechanisms of the system.

The difficulty posed by the participants autonomy is that it is not always possible to consider direct modification of the inner behavior of the entities and that the autonomy may lead the system to evolve to not previously contemplated conditions.

1.5.3. Modeling

We have acknowledged that complex systems include different description levels, at least two: local and global. Also we said that they exhibit nonlinear dynamics and usually are sensitive to initial conditions. Analytical models taking into consideration these two aspects, rapidly grow too difficult to be solved. Mathematical models rely on the identification of the key system components, often representing them in a discrete manner. This limits mathematical models because emergence present in complex systems arises as a consequence of local interactions, and cannot be previously identified as a key system component. Mathematical models are analogues, but cannot provide significant insight into the continuous internal process of a complex system (Polack et al., 2008). Moreover, they only take into consideration the global point of view, usually not explaining the reasons locally leading to the global behavior of the modeled system (Edmonds and Bryson, 2004).

The difficulty is to create a model that takes into account the multiple levels of a complex system. In the control context, a model of a complex system must take into account multiple levels because control actions taken at a level will have an influence at other levels. On the one hand, if we take control actions at the local level, by emergence, the global behavior of the system will be affected. On the other hand, if we take control actions at the global level, by immergence, the local level interactions will be affected, closing a feedback loop.

1.5.4. Preexisting systems

Systems may exhibit an undesired behavior because their internal control mechanisms have become useless. This can occur when their control mechanisms were not built to consider the conditions that provoked the change in the behavior. These conditions include: open system with new entities or environmental changes, the evolution of the behavior of the autonomous entities in the system, and the addition of whole new systems running on top of the preexisting system.

If the internal control mechanisms are no longer useful, they must be modified. This modification may imply for some systems, stopping the target system. Take for example the internet. It is technically possible to modify the behavior of a router in an autonomous system, but it would definitely be disastrous to modify all the routers at the same time (not to speak to the technical limits given by the fact that not all routers in a network belong to the same stake holders).

The difficulty of preexisting systems in the context of control of complex systems is that when the control mechanisms built with the system are no longer useful, it is not always possible, on the one hand, to stop the system to change the control mechanisms and on the other hand, it is not always possible to change the entities inner behavior.

Control theory assumes that given a model, an optimal way to control a target system exists. It has been recognized that given the characteristics of complex systems, the concept of governance is better suited (Chavalarias et al., 2009) than control. In the following section we take a closer look at the notion of governance and how it applies to complex systems.

1.6. Governance and control of complex systems

The term “governance” originated in the political science and sociology domains.

> 'governance' is now often used to indicate a new mode of governing that is distinct from the *hierarchical control model,* a more co-operative mode where state and non-state actors participate in mixed public/private networks. (Mayntz, 2003)

Within the social sciences context, governance theory seeks to improve the social order in a **new** way, by considering societies as systems with hierarchical governments to steer them.

In our reference framework, governance is preferred over control because some of the assumptions of control theory do not hold within the complex systems context. Complex systems have several different scales, they usually have multiple dimensions and they imply the presence of many heterogeneous points of view. One first assumption of control theory that does not hold within the context of complex systems is that it is not a trivial task to obtain an analytical model that can deal with ease with all these aspects.

A second assumption is optimality. Within the complex systems context it should be considered possible to drive the behavior of the system but not always in the optimal conditions (not always maximizing a function). We cannot obtain an optimal value as a reference if we cannot have a model that produces it. In more mathematical terms, if we cannot have a model yielding a function to maximize, we cannot have optimal control.

1.7. Synthesis

The challenge of control of complex systems, as we stated it in 1.5 is directly related to some characteristics of complex systems.

- The local interactions produce the global outcomes of the system
- They are made of decentralized systems made of autonomous entities
- They are not easily (or at least usefully) modeled by analytical models
- Preexisting complex systems may not be legally, or technically stopped or tampered with in order to control them

We consider that the control of complex systems is defined by overcoming a series of difficulties given by the characteristics of complex systems.

The modeling challenge is of capital importance by itself and also because it is directly related to the engineering challenge. In the control challenge, modeling is one difficulty but the local actions with global effects difficulty is closely related to the difficulty of modeling.

From the intuitive definition of control of section 1.4.3, we deduce that it is necessary to have a model to identify the behavior that we wish to observe or to avoid. One major difficulty that any control mechanisms of a complex system faces is modeling the evolution of the system behavior. Overcoming this difficulty means to characterize the evolution of the behavior of the target system, taking into consideration:

- The different levels of a complex system (local and global for example).
- The emergence of global outcome from local interactions.

A second difficulty that control mechanisms for complex systems face are preexisting systems. On one hand, overcoming this difficulty means to identify ways to control a system where the autonomy of the entities must be respected (because of legal, ethical or technical reasons). On

the other hand, it means to identify effective control actions in contrast to those already built in the target system that may be failing.

A third difficulty is to identify control actions that take into consideration emergence. Overcoming this difficulty means to identify a model where control actions are included. Modeling the behavior is not trivial and modeling the behavior when control actions are applied, is not trivial either.

New ways to find solutions to the control problem that do not intend to be optimal must be found. Specially, a better suited term to denote the "lead of a complex system to a desired state" that does not include optimality, is necessary. The term "governance" allows to relax the notion of optimality implicit in control.

Additionally, it is an hypothesis also taken as basis for work in research projects such as the French thematic network on the governance of complex systems (rncsgouv2012, 2012). Our selection of governance over control is beyond the semantic context. We adhere to the ideas behind the concept of governance: find a new way to control a complex system, without assuming that optimal control is possible.

Although we shall continue to refer to the "control" challenge, our work takes place under the same hypothesis of governance: *optimal control for a complex system is not possible*. The main reason behind this hypothesis is that optimal control implies the presence of an analytical model that allows to obtain a function to be maximized (to obtain an optimal value). And as we have already stated, this kind of models are hardly useful, because they do not take into consideration the different levels of organization of complex systems.

In the following chapter, we will take a look at works that have been confronted to the difficulties of the control challenge.

Chapter 2

Related Work

Contents

2.1. Introduction 21
2.2. Control Theory 22
2.3. Equation Free approach 24
2.4. Modeling complex systems 26
2.4.1. Multi-agent paradigm 26
2.4.2. Agent-Based Models 27
2.5. Applications of multi-agent paradigm in the control of complex systems 31
2.5.1. Organic Computing 31
2.5.2. Control of Self-Organizing systems 33
2.5.3. Prosa 34
2.5.4. PolyAgent 36
2.5.5. Morphology 37
2.5.6. Emergent Engineering 39
2.5.7. Control of reactive multi-agent system with reinforcement learning tools 40
2.6. Synthesis 40
2.7. Conclusions 41

2.1. Introduction

We present different works that have been confronted to the control of complex systems. That is why we present at first place the framework of control theory. From this framework we shall focus on the basic elements of a control loop and on the place of models in it. Secondly, we present the basic ideas of the "equation-free" approach. We shall focus on how it allows to analyze a system at macroscopic level, without explicit solution to macroscopic equations. Given that the context of modeling and simulation is at the heart of control and our proposition, we thirdly present, the concept of modeling within the context of our work, the multi-agent paradigm as well as some elements pertinent to the modeling and simulation framework. Finally, we present different applications of the multi-agent paradigm in the context of the control of complex systems.

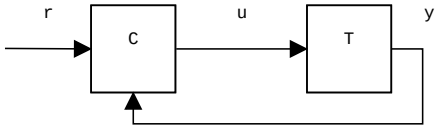


Figure 2.1: Closed feedback loop. The output u of system C is used as the input for system T while the output y of system T is used as the input of system C.

2.2. Control Theory

Definition

The purpose of control theory is to determine which control actions will make a system reach or maintain a certain state (Sontag, 1998; Ogata, 2001). Within control theory, dynamical systems are described in terms of ordinary differential equations (ODE) like:

$$\begin{cases} \dot{x}(t) = f(x(t), \alpha(t)) \quad (t > 0) \\ x(0) = x^0 \end{cases}$$

where x is a function representing the evolution of the state of the system and f depends also on some "control" function α. In control theory, a feedback controller measures the effect of inputs on the process outputs and the results yielded by the controller are used again as input to the process, closing the loop (Astrom and Murray, 2008).

Feedback in control

Feedback exists when two entities mutually influence each other. In the context of control theory and control engineering, feedback is defined as being a loop where the output of one system is the output of another one. Figure 2.1 shows a graphical representation of a feedback loop. If the link between the output of T and the input of C disappears, we say that the loop is open, otherwise the loop is said to be closed.

In control engineering, feedback loops are exploited to gain control of a system. The basic setup of a control feedback loop is as follows: One of the systems in the loop is the system to be controlled (the target system T) and the other one, the controlling system (C). The input of the controller system C is r, a reference value indicating the desired output of system T. The output of system C is u which in turn, is the input of system T. The output of system T is y. The difference between the observed output of the target system y and the reference value r is called the error e.

The goal of control theory is to find a control law that will be used to establish the outputs of the controller system, so that the output of the target system is as close as possible to the reference (which in turn is the input of the controller system). The control law of a controller system is its core. Since most of the tools used by control theory to find such laws are issued from operations research, ODE and game theory, it is natural to find that such laws have an analytical form.

The simplest way to implement a control law in a feedback loop is by using an "on-off" approach. It consists in applying the maximal control action possible (u_{max}) whenever the difference e between the reference value r and the measured output of the target system y is positive, and a minimal control action when the difference e is negative. The control law of an on-off controller only considers as input the current output of the target system. When there

is no difference, no control action is applied. A limit of this approach is that every time there is a difference ($e \neq 0$) one control action at one of the limits of the spectrum is applied. This controller is known to yield the target system oscillating and not very useful when the target system requires a certain input value in order to maintain the output at a particular level.

$$u = \begin{cases} u_{max} & e > 0 \\ u_{min} & e < 0 \end{cases}$$

PID controllers overcome the limit of on-off controllers by taking into consideration a bigger set of information when creating the control law. PID stands for Proportional Integral and Derivative controller. In the family of PID controllers we first have the Proportional controllers. The control laws of these controllers define a range $e_{min} < e_{max}$ called the proportional band. When the error e is within the domain, the control action will be proportionally adjusted to error. Otherwise, the control action will be maximal for a maximum error and minimal for a minimum error.

$$u = \begin{cases} u_{max} & e \geq e_{max} \\ k_p e & e_{min} < e < e_{max} \\ u_{min} & e \leq e_{min} \end{cases}$$

Despite this new control law with a proportional approach is advantageous over the one of the on-off approach, it is known to have a limit: e should have a nonzero value in order to keep the target system at a steady desired value of y.

Proportional Integrative controllers overcome this limit by making the control action proportional to the integral of the error over the lapse of time $u(t) = \int_0^t k_i e(\tau) d\tau$. In other words, the history of error e is considered.

A final refinement consists in letting the control law consider a prediction of the error T_d time units ahead. This new element, the prediction, could be simply obtained for example by using a derivative over the error.

The final control action will be thus given in a PID controller as the sum of three terms: the one issue of the proportional control law, the one of the integrative control law and finally the one of the derivative control law.

A control law of the PID family can be described as the sum of three elements.

1. one that is meant to adjust the control action to be proportional to the error
2. one that is meant to adjust the control action by taking into consideration the past history of the error
3. one that is meant to adjust the control action by taking into consideration a prediction of the error

Elements two and three are called respectively integrative and derivative because they have an exact match to such operations in calculus. That is, the second element will have the form of an integral and the third one of a derivation. These elements are mathematical models of the error behavior. When developing a control feedback loop within the context of control theory, we seek to find the mathematical models that best fit the system. When we say best, this could mean either that the dynamics of the system are accurate or that the desired results are obtained within some ranges (optimized).

Definition 3 *Feedback control loop: measure (regularly) the state of the system, and based on these measurements, some controllable parameter(s) is adjusted to drive the system to a certain state.*

Modeling

The basic working hypothesis of control theory is that there is a model of the target system that identifies the evolution of the target system in terms of its inputs and outputs. Based on this knowledge, the modifications necessary to be executed to the inputs to produce a certain output or behavior can be directly obtained.

Models used in classical control theory, such as ordinary differential equations, are analytical and thus directly represent the global functioning of the system. In this kind of representation, the phenomena leading to this global functioning are rarely made explicit in terms of local actions.

One first problem regarding the usage of control theory for complex systems is that, it considers analytical modeling of the system possible. Even if this is possible, the resulting model needs to account for the nonlinear dynamics of complex systems. This means, that the resulting model will be hardly solvable.

Synthesis

Models are necessary in control theory to foresee the effects of control actions and to estimate the future state of a target system. This work is supported on the hypothesis that to model complex systems, it is necessary to take into consideration the multiple levels of description present in them. From this point of view we consider traditional analytical models to be of little use in this case, because they only take into consideration one level. Moreover, analytical models are usually of little usefulness because they are not always solvable.

The equation-free approach directly deals with the case when explicit solutions to global analytical models are not available but local models are available. In the following section we present the approach and summarize the key ideas that we consider relevant to our work.

2.3. Equation Free approach

Equation-free refers to a paradigm for multiscale computation and computer-aided analysis (Samaey, 2010). It is designed to be used in problems when the evolution of a system is observed at a global or coarse level while accurate models are only given at a local or finer scale level of description. The paradigm proposes to bypass the derivation of explicit macroscopic evolution equations when these exist but are not solvable because they are not in a closed form. The central idea is to avoid the explicit definition of coarse equations by using short bursts of appropriately initialized fine-scale simulation (Kevrekidis et al., 2003, 2004; Siettos et al., 2006; Siettos, 2011). This is not directly possible because of computational cost and because finer scale models are not always easy to be analyzed. Performing coarse-scale computational tasks with fine-scale models is often unfeasible: direct simulation over the full spatiotemporal domain of interest can be computationally prohibitive. Additional modeling tasks, such as numerical bifurcation analysis, are often impossible to perform on the fine-scale model directly: a coarse steady state may not imply a steady state for the fine-scale system.

The solution to overcome this limits of computational approaches given in the equation-free paradigm is as follows. Short bursts of fine scale simulation (short computational experiments) are designed, executed, and their results processed and fed back to the process, in integrated protocols aimed at performing the particular coarse-grained task (the detection of a macroscopic instability, for example). Two models, a fine-scale (f) and a coarse scale (F), are assumed to exist, each with an associated time-stepper. The fine-scale model is given by: $\partial_t u = f(u)$. The time-stepper for the fine-scale model is given by: $u(t + \mathrm{d}t) = s(u(t), \mathrm{d}t)$. The coarse model is

given by $\partial_t U = F(U)$. The coarse model time-stepper is given by: $U(t + \delta t) = S(U, \delta t)$. The fine scale model involves fine-scale variables: $u(t)$. The coarse model is given in terms of coarse variables which are assumed to exist, but are unavailable in closed form: $U(t)$.

The key building block of the approach is the coarse time-stepper. It can be seen as an algorithm that will use the output of a fine-scale simulator (given certain initial conditions) to set boundaries to the coarse model and thus be able to express it in a closed form. Formally speaking, the coarse time-stepper can be described as follows. Given an initial condition for the coarse variables $U(t^*)$ at t^* the coarse time-stepper involves:

- Lifting. Create fine-scale initial conditions $u(t^*)$, consistent with $U(t^*)$
- Simulation. Use the fine-scale simulator to compute the fine-scale state $u(t)$ at $t \in [t^*, t^* + \delta t]$
- Obtain the coarse state $U(t^* + \delta t)$ from the fine-scale state $u(t)$.

Once a closed form of the coarse model is found, the approach suggests to use numerical methods to obtain solutions to the equations describing it. The suggested methods include: coarse projective integration (Gear et al., 2002) and patch dynamics (Samaey et al., 2009).

Examples of the equation-free approach include its usage in chemotaxis (Erban et al., 2006) and data clustering (Samaey et al., 2008). In (Samaey et al., 2008), data clustering is accomplished by a multi-agent system. Agents move data items by picking them up and dropping them next to similar data items. The items are picked up and dropped with respect to some probability which is determined by the presence of similar data items in the local neighborhood. The clusters are therefore formed in an emergent way. The objective is to understand the performance of such a system. Instead of exhaustively exploring (experimentally) the spatiotemporal domain, they use the equation-free approach. They conclude that applying the approach is not a trivial task, and propose an iterative bottom-up technique to identify the variables to use in the coarse level.

The important ideas that we retain from this approach are:

- It allows to obtain information about the state of a system given in a global description level when only a model given in the local description level exists.
- This is done by simulating the model given at local description level but this simulation may take too much calculation time or resources.
- To overcome this difficulty, a coarse time-stepper decides when to use the simulation of the local level of description model and specially how to translate the results that are given by at a local description level to the global description level.

This work is supported on the hypothesis that the multi-agent paradigm in particularly useful in the specific context of modeling complex systems. In the following section we elaborate on this hypothesis and present the basic concepts of the multi-agent paradigm. We shall first present the general concepts of the multi-agent paradigm and then the specific concepts related to multi-agent models.

Multi-agent models are by nature experimental: they have to be simulated in order to provide answer to the questions they were built for. Therefore, further in the following section, we shall present the general framework of modeling and simulation of (Zeigler et al., 2000), where our work is placed.

2.4. Modeling complex systems

The multi-agent paradigm has been identified as being suitable for modeling complex systems within specific domains like sociology (Conte et al., 1997; Bonabeau, 2002; Tsvetovat and Carley, 2004; Amblard and Phan, 2006; Gilbert, 2008) biology (Grimm and Railsback, 2005; Thomas et al., 2003, 2007) and urban studies (Galland et al., 2009; Gaud et al., 2008). They have also been identified as suited to engineering complex tecno-social systems including for example: wireless communications, (Jamont and Occello, 2009; Jamont et al., 2010), traffic (Mandiau et al., 2008; Ksontini et al., 2012). Jennings and Bussmann share the point of view of (Polack et al., 2008, 2009) in which they all argue that the agent-oriented mindset provides suitable abstractions useful in controlling, engineering as well as in simulating and modeling complex systems out of a domain specific point of view. The abstractions are the following (Demazeau, 1995).

- **Agents.** The basic abstraction in the paradigm, are the equivalent of autonomous entities in a complex system.
- **Agent organization.** Hierarchical or heterarchical organization in complex systems.
- **Interactions.** Interactions between entities in the complex system
- **Environment.** The environment where the entities of the complex system evolve.

2.4.1. Multi-agent paradigm

The multi-agent paradigm is characterized by the decentralized execution of autonomous entities. The autonomous entities are called agents. We consider that an agent can be defined as follows (Ferber, 1995).

Definition 4 *An agent is an autonomous entity capable of acting to achieve its goals. An agent is a social entity that interacts with other agents or with its environment to achieve its goals when necessary. An agent can represent a physical or a virtual entity.*

A multi-agent system is comprised of the following elements: agents, an environment and the interactions (agent with agent and agent with environment) (Ferber, 1995). An agent holds the following characteristics.

Definition 5 *Characteristics of an agent (Ferber, 1995).*

- *capable of acting in an environment*
- *capable of communicating with other agents*
- *motivated by a set of tendencies*
- *has its own resources*
- *capable of perceiving its environment*
- *has only one partial representation of its environment*
- *has capabilities and offers services*
- *its behavior tends to satisfy its objectives taking into consideration its own resources and capabilities, based on its perceptions, representations and communications.*

Within the multi-agent paradigm, the global dynamics of a system, at the macroscopic level is not given in advance (opposite to the analytical models). The global dynamics is the outcome of the interaction of each agent's behavior, at the microscopic level. The advantages of this approach are that: it can represent and simulate open systems; it can take into consideration, from the moment of creating the model, the dynamic and heterogeneous characteristics of the individual behaviors; and finally, it can analyze the importance of a local behavior on the global functioning of the system (Parunak et al., 1998).

The paradigm may be used to study complex systems by engineering complex systems or by modeling preexisting systems. The tools to engineer multi-agent systems include simulation platforms (Netlogo, Repast, MadKit, CORMAS, JADE), interaction protocols (Contract Net interaction protocol, KQML, FIPA-ACL) and programming languages (JACK) among others. The main tool used to study preexisting systems are agent-based models.

2.4.2. Agent-Based Models

In general terms, a model is a simplified representation of a phenomenon. Within this thesis work we are going to focus on models considered from the point of view of the modeling and simulation framework of (Zeigler et al., 2000).

In the framework, the basic entities are: source system, model, simulator and experimental frame. Additionally, the framework defines the relationships between the entities. The source system is the preexisting system that we are interested in modeling. It is the source of observable data. The data obtained by observation or experimentation with the source system is called the behavior database. The experimental frame is a specification of the conditions under which the system is observed or experimented with. A simulator is the entity in charge of obeying the instructions of a model and generating a behavior. Figure 2.2 illustrates the modeling and simulation framework.

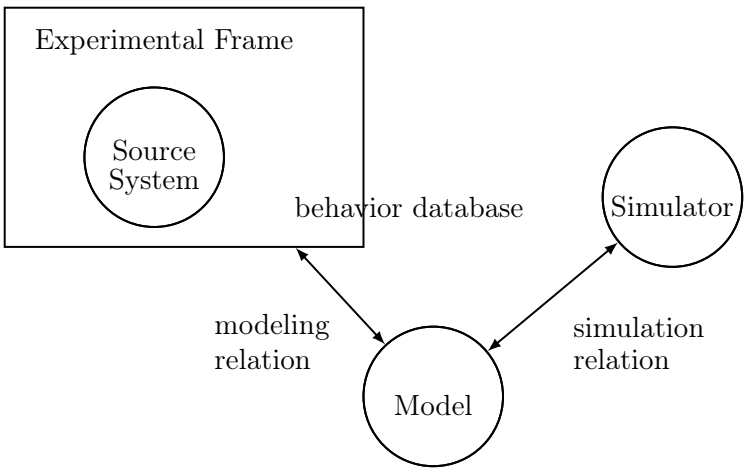


Figure 2.2: Modeling and Simulation framework entities and their relationships. Taken from (Zeigler et al., 2000)

The basic modeling relation is validity. Validity refers to the relation between model, system and experimental frame. It is the degree to which a model faithfully represents its system counterpart. The simulation relationship refers to degree of correctness at which a simulator simulates a model if it is guaranteed to faithfully generate the model's output trajectory given

Level	Specification name	What we know at this level
0	Observation Frame	How to simulate the system with inputs: what variables to measure and how to observe them over a time base;
1	I/O behavior	Time-indexed data collected from a source system; consist of input/output pairs.
2	I/O function	Knowledge of initial state; given an initial state, every input stimulus produces a unique output.
3	State transition	How states are affected by inputs; given a state and an input what is the state after the input stimulus is over; what output event is generated by a state.
4	Coupled components	Components and how they are coupled together. The components can be specified at lower levels or can even be structure system themselves - leading to hierarchical structure.

Table 2.1: The System Specification Hierarchy, adapted from (Zeigler et al., 2000, chapters 1 and 5)

its initial state and its input trajectory.

From this point of view, one way to classify models is by looking at the degree of knowledge gained from the simulated system. Zeigler et al. in (Zeigler et al., 2000) specify a set of levels at which models can be constructed, in terms of the knowledge of the system. This set of levels called "system specification hierarchy" is presented in table 2.1. The hierarchy starts at level 0, when only observations of inputs and outputs are available. The highest level, allows to describe the interactions of each of the components of the model as well as the inputs and outputs.

A multi-agent model is a multi-agent system, as previously defined in section 2.4.1. A multi-agent system is built to serve a specific purpose. This means that when building a multi-agent system, we are usually interested in making it behave in a certain way, given some initial conditions. The purpose of a multi-agent model is to answer questions regarding the modeled system. In contrast to "building" a multi-agent system, a multi-agent model allows to explain the mechanisms that produce the global behavior rather than just making it appear. In both cases, the global behavior is emergent.

When a multi-agent system is used as a model, it naturally shares characteristics with simulation models, as defined in the simulation and modeling framework.

Definition 6 *A simulation model is a set of instructions, rules, equations or constraints for generating input and output behavior. (Zeigler et al., 2000)*

It is common practice to define a multi-agent model by defining the following elements.

Definition 7 *Elements of a multi-agent model.*

1. *The behavior of each agent in the model.*
2. *The interactions of each agent in the model (with other agents and the environment).*
3. *The environment in which agents are situated.*

Regarding the way inner agents behaviors are designed, three basic architectures exist: reactive agents, cognitive agents and hybrid agents (Phan and Amblard, 2007b). **Reactive Agents** have an inner behavior given by simple sensor-actuator loops. They have no representation of the other agents in the system or the environment. Tasks define the actions to be executed by the agents. Actions are driven by the association of perceptions to tasks. **Cognitive Agents** actions are driven by reasoning, based on knowledge of other agents and environments. These architectures are also known as Belief Desire Intention (or BDI). **Hybrid Agents** include reactive and cognitive architectures in the inner behavior plus a module that controls which behavior takes place at each time.

The way interactions are specified in a multi-agent model is concerned with two main issues: specifying which agents interact with which other agents and the dynamics of the interactions. Agents typically interact with a subset of the other agents in the system, usually called the agent's neighbors. How agents are connected to each other to is generally termed an agent-based model's topology. Typical topologies include a spatial grid or network of nodes and links, where the agents are represented by the nodes, and the links the relationships. In static networks, links are pre-specified and do not change. For dynamic networks, links, and possibly nodes, are determined endogenously according to the mechanisms programmed in the model. In cellular automata, static topologies like lattices or grids are used to define the neighbors of each cell. In other multi-agent models, dynamic topologies are used, and they are specified using graph theory. Communication among agents enables interactions. Two ways to implement communication between agents are found in the literature: direct communication and stigmergic communication. The principle behind stigmergic communication is that a trace is left in the environment by one agent. Then other agents (or the same agent) will be able to sense the trace and use it.

One important aspect related to interactions among agents in a multi-agent model is the definition of the moment at which each agent is active. This is usually called the scheduling scheme of the model. If the simulation is to be executed with discrete time steps, the scheduling specifies which agent is going to be executed at each time-step. If the model includes time-based specification of agents, the simulation can be done following a continuous time flow.

The environment may simply be used to provide information on the spatial location of an agent relative to other agents or it may provide a rich set of information regarding the environment. Also, the environment may be used to support the communication among agents (stigmergy).

Simulation

For the purpose of this thesis work, we adopt the following notation to describe an an agent-based model. A model M, with parameters $Par_M = \{p_1, p_2, \ldots, p_n\}$, executed over time period T. The set λ_M contains the initial values given to each of the parameters of a model. To differentiate between an abstract model and its parameters from a concrete model with specific values given to the parameters we propose to call the latter a model "instance". An instance of M is given by $M^1 = \langle Par_M, \lambda_M^1 \rangle$. A state of the model is given by the specific values of each of the parameters. We call the result of the simulation of model instance M^1, r_{M^1}. We call the final state of the model at time T the result s^f. We call s^{init} the initial state of the model at time 0. Table 2.2 summarizes the notation.

The agent-based approach to modeling is computational: it requires simulation to obtain the global behavior of the system, given some initial conditions. The software simulator is in charge of running each agent in the model, during a given amount of time.

A very important aspect of a simulation is the order under which each of the agents is

Name	Meaning
M	A model.
Par_M	The set of parameters $\{p_1, p_2, \ldots, p_n\}$ of the model. This set includes parameters characterizing the agents, the environment and the interactions.
M^n	An instance of model M with initial values λ_{M^1}
λ_{M^1}	The initial values given to the parameters of model instance M^1
T	The time duration definition of the model.
r_{M^1}	The result of the simulation of model instance M^1.
S_M	The set of states of model M during simulation $\{s_M^{init}, s_M^1, \ldots, s_M^f\}$.

Table 2.2: Notation used to define an agent-based simulation.

scheduled to be executed in the simulator. This order is usually known as the update scheme, the execution order or the scheduling type. We shall call "synchronous" the update scheme when all the agents in the model are scheduled to be activated at the same time. We shall call "asynchronous" the update scheme when only a subset of the agents are activated at the same time. Without proper specification of an update scheme, models with identical agent behavior definitions may lead to different, even contradictory results (Fatès and Chevrier, 2010; Navarrete Gutiérrez et al., 2011).

The classical usage of an agent-based model to answer questions regarding a macroscopic variable y after a period of time T given initial conditions of the variable y_{init} is to randomly initialize the model n times and later execute a statistical analysis of the results (for the interesting variable). The random initialization accounts for the fact that the macroscopic variable may correspond to different microscopic configurations in the model. In this course of action, the model has an associated update scheme that is executed by the simulator. The initialization part makes sure that a specific microscopic configuration of the model corresponds to the desired macroscopic state of variable y. The simulation time is defined in the model itself and can be discrete, continuous or event-based. More importantly, the classical course of actions simulates the whole range of time between the time of the initial conditions t_i until T. This can be described by the algorithm 1.

Algorithm 1: Prototypical course of action of agent-based simulation to obtain a macroscopic variable after T time, given y_i initial conditions.

input : Initial conditions y_i
input : Number of model instances to execute n
input : Duration of each simulation T

forall the n *instances* **do**
  $\lambda_M^n \leftarrow$ Random Initial Parameters (y_i);
  Initialize Model(M^n, λ_M^n);
  Simulate Model (M^n, T);
  $r[n] \leftarrow$ Extract Macroscopic Value of Variable(y, $s_{M^n}^f$)
end
$y_{T+i} \leftarrow$ StatisticalAggregation(r);

Validation and Calibration

Validating a model in general terms and in the broad context of validation and simulation is generally defined as "the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" (Law, 2009) Within the framework of (Zeigler et al., 2000), there are three levels of validity for a model: replicative, predictive and structural. Replicative validity means that for all possible experiments within the experimental frame, the behavior of the model and system agree withing acceptable tolerance. Predictive validity requires replicative validity but also the ability to predict as yet unseen system behavior. Structural validity requires replicative validity but also mimics in step-by-step, component-by-component the way in which the system does its transitions.

Assessing the validity of a model is important because only a valid model, will yield answers that can be taken as answers for questions directed to the modeled system.

Most models have free parameters whose proper values are unknown. **Calibrating** a model means to find the values for a set of parameters of the model that makes it exhibit a certain (usually observed from the modeled system) behavior. If a model has replicative validity, it can reproduce the observed behavior of the modeled system but this is no guarantee that it can produce credible predictions (no predictive validity). The importance of the calibration process is that it helps to improve the validity of a model.

2.5. Applications of multi-agent paradigm in the control of complex systems

In section 1.5 we settled our point of view regarding the challenge of controlling complex systems. We argue that the difficulty of the challenge is related to certain characteristics of complex systems. Namely the modeling of complex systems, the relationship between local actions and global results, the autonomy of the elements of complex systems and preexisting systems.

We now give a present a selection of work related to the control of complex systems that applies the multi-agent paradigm.

2.5.1. Organic Computing

Definition

Organic Computing (called OC in the reminder of this chapter) is a German research initiative whose object of study are technical complex systems (ocwebsite, 2012; Schmeck, 2005; Würtz, 2008). OC intends to provide a full framework to study and design technical complex systems. Within the framework, an architecture to drive the behavior of such systems is defined. The main idea behind their proposal is that technical systems share characteristics such as emergence and self-organization with living (organic) systems. Living systems are considered to deal with the control problem through self-* properties. Therefore, other (technical in this case) systems may also benefit (particularly when dealing with control) from having self-* properties as well as from learning.

Organic computing has the vision of addressing the challenges of complex distributed technical systems by making them life-like. It intends to achieve this by endowing such systems with self-* (self-organization, self-optimization and self-configuration) capabilities as well as with adaptation (Mnif et al., 2007).

They assume that to achieve this, they have to give the system adequate degrees of freedom from the design. An emergent behavior, which can be positive or negative, may result as a consequence of this design decision. Therefore, a regulatory mechanism is necessary to, enable adequate reactions to control the (sometimes completely unexpected) emerging global behavior while at the same time fosters self-organization. The regulatory mechanism proposed within the OC framework is a generic "observer / controller" architecture (Richter et al., 2006).

Generic Observer / Controller Architecture of OC

The architecture was initially sketched in (Müller-Schloer, 2004). A full description of it can be found in (Richter et al., 2006). The architecture is a general pattern constituting a higher-level control loop. In the architecture, the production (complex technical) system is called the "System under Observation and Control" or "SuOC". Two other elements compose the architecture: an observer and a controller (deemed O/C pair). Within the architecture, the SuOC, the observer and the controller are one whole system called "Organic system". In the OC approach, the target system and the control system (observer + controller) make one organic entity, independent but symbiotic (the SuOC will not break down if observer+controller breaks down).

The generic architecture is proposed as having one observer, one controller and one production system but different arrangements or mixtures of O/C pairs are foreseen. These would produce different kinds of architectures: with different levels of decentralization and hierarchization, as described in (Cakar et al., 2008; Tomforde et al., 2009).

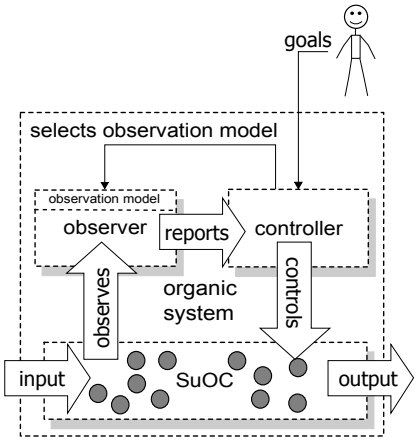


Figure 2.3: Observer / Controller generic architecture.

Observer

The observer measures the collective behavior (micro and macro level properties) of the SuOC through sensors and reports aggregated system parameters characterizing the global state and the dynamics of the system to the controller. The situation parameters vector, is reported to the controller who takes appropriate actions to influence the SuOC. The situation parameters include an evaluation of the future of the system, created inside the observer module. The observation behavior is variable.

Controller

The controller evaluates situation parameters with respect to the goal defined by the user and influences the system through actuators. It is the controller's task to guide the self-organization process between the elements, but to interfere only when necessary. The controller may affect the SuOC by influencing:

- the local decision rules of the participants of the system
- the system structure (communication between the participants, number of participants)
- the environment (assuming that it will indirectly influence the system by changing the data observed by the participants through their local sensors)

Examples. Work has been carried out within the OC initiative in different application domains such as traffic control (Cakar et al., 2008; Prothmann et al., 2008), networking (Tomforde et al., 2009, 2010) and ecology (Mnif et al., 2007).

2.5.2. Control of Self-Organizing systems

We are interested in the control of self-organizing systems. The specific approach that we are interested in is inscribed within the context of ubiquitous computing. Before going into detail of the approach of self-organization that we are interested in, we shall briefly present the ubiquitous computing context.

Ubiquitous Computing (or also pervasive computing) is used to describe Information and Communication Technology systems that enable information and tasks to be made available everywhere, and to support intuitive human usage, appearing invisible to the user. The vision of the world from the ubiquitous computing perspective is that of one where people are surrounded by computing devices and infrastructure supporting us in everything we do (Weiser, 1991; Satyanarayanan, 2001; Ciarletta, 2011).

Ubiquitous computing systems are characterized by the following two main properties.

- They are situated in human-centered personalized environments, interacting less obtrusively with humans.
- They are part of, and used in, physical environments, sensing more of the physical environment.

Systems created under the ubiquitous computing context are interesting because they share characteristics with complex systems. Complex systems are however a more general classification of the world, since ubiquitous computing systems are limited to the inclusion of computers in the environment, in contrast to complex systems theories that can be applied to systems made of human beings (without any contact with computers) or bacteria, or piles of sand.

It is commonly accepted within the ubiquitous computing context that one of the key design challenges is to find out if a system designed by specifying local interactions only can be controlled, constrained or coherent at higher level.

We shall focus on the self-organizing engineering point of view of ubiquitous computing, since it is related to the multi-agent paradigm. The self-organization engineering community is also interested in driving a system made of autonomous entities to exhibit a desired behavior.

Definition

> [Self-organization is a] process where a system changes its internal organization to adapt to changes in its goals and the environment without *explicit external control* (Di Marzo Serugendo et al., 2005).

(Brun et al., 2009) argue that control feed-back loops are useful to engineer self-organizing (a form of complex systems or at least systems sharing properties with complex systems) software systems. They insist on the need for explicitly specifying the feed-back control loops present in the design and engineering of systems. Also, they propose to widely research natural systems to find new types of control loops.

The point of view of engineering self-organizing systems is that, the autonomy of the elements should lead to a coherent global behavior. However, solely defining a system with autonomous elements cannot give guarantees about the global behavior of the system.

In (Edmonds, 2005), Edmonds argues that at some point engineering limits the self-organizing of a system and suggest to use experimental methods to obtain guarantees about the global outcome of self-organizing systems.

Despite the usage of experimental methods to guarantee the coherent behavior of the self-organizing systems, additional experimentation is needed in order to cope with the "open" systems.

Examples. Empirical approaches have demonstrated the feasibility of self-organizing systems (Wolf et al., 2005). This empirical approach promises to give guarantees of the overall global behavior of a system built with self-organizing autonomous entities.

Particularly speaking, they are interested in building (through engineering) systems that exhibit a desired behavior as an emergent phenomenon. However, given the nonlinear nature of emergence in such systems, they cannot guarantee *a priori* the behavior of the system. That is why they propose to use experimental methods in the engineering process.

2.5.3. Prosa

Product, Resource, Order Staff Architecture or PROSA is a reference architecture to build holonic based manufacturing execution systems. Holonic manufacturing systems (HMS) is "an international industrially driven project addressing systematization and standardization, research, pre-competitive development, deployment and support of architectures and technologies for open, distributed, intelligent autonomous and co-operating systems on a global basis" (Van Leeuwen and Norrie, 1997; Gruver et al., 2003). HMS is based on the work of Koestler on the modeling of biological and social systems as systems consisting of self-contained elements and capable of functioning as autonomous entities in a cooperative environment. Such an entity is called a "holon". The term was introduced in (Koestler, 1989) and refers to an entity that is at the same time part of a whole, and a whole by itself. A group of holons is called a holarchy. A holarchy is a hierarchy of self-regulating holons which behave in three different ways. The first way is as autonomous wholes in supra-ordination to their parts. The second way is as dependent parts in sub-ordination to their parts. The last way is in co-ordination with their local environment.

Within the HMS context, a holon is an autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing or validating information and physical objects. The holon consists of an information processing part and often a physical processing part. A holon can be part of another holon. Also, a holon is considered to be autonomous: capable of creating and controlling the execution of its own plans and strategies (and to maintain

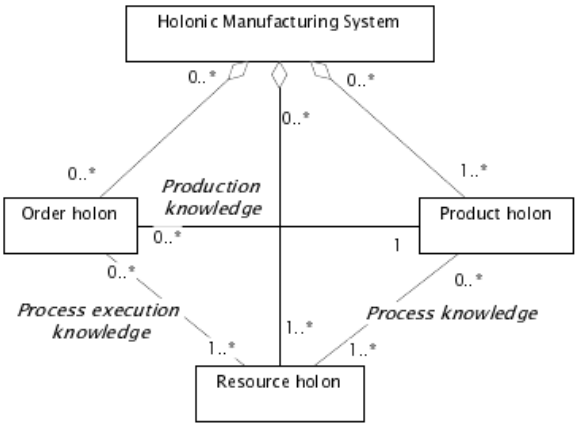


Figure 2.4: Holons from PROSA and their relations. Taken from (Van Brussel et al., 1998)

its own functions). Cooperation in the HMS context is the process that leads a set of holons to mutually develop acceptable plans and executes them. Self-organization in an HMS is the ability of holons to collect and arrange themselves to achieve a production objective. In HMS, a holarchy defines the basic rules for cooperation of the holons, thereby limiting their autonomy.

PROSA was proposed in (Van Brussel et al., 1998) and consists of the following holons:

Product. The physical products being manufactured and human and computing support to initiate and monitor the activity to produce the product. It is responsible for recipes or process plans.

Resource. The physical processes or transportation resources, its control systems and any necessary human based operations. It is responsible for resource handling.

Order. The requirements of a particular order including information such as product quantities, due dates, costs and priorities. It is responsible for the internal logistics.

Staff. Optional support element that provides coordination between holons and ensures that global goals are achieved. These allow for the use of centralized algorithms and for the incorporation of legacy systems.

The PROSA reference architecture uses the specialization and aggregation concepts borrowed from the object oriented programming paradigm to describe the organization of the holons. PROSA uses the Unified Modeling Language (a standardized[2] general-purpose modeling language specially used for drawing object-oriented diagrams[3]) to model the typical relationships that could exist between holons of a generic manufacturing system. The interactions among the basic holons of PROSA are depicted in figure 2.4.

Basic holons negotiate among them to obtain a satisfying solution for every holon. Holons are considered for the negotiation, as a global whole group, or at different group levels, as necessary. The holons can be aggregated or specialized, in regard to the object oriented programming paradigm. This organization of holons is used to create a taxonomy of agents for a specific fabrication application domain.

There are three steps in the methodology: identify holons and their responsibilities for the specific system or application, design the holons, implement them and finally run the system.

[2]ISO/IEC 19505-1 & 19505-2

[3]The language is formally maintained by the OMG group `http://www.omg.org`

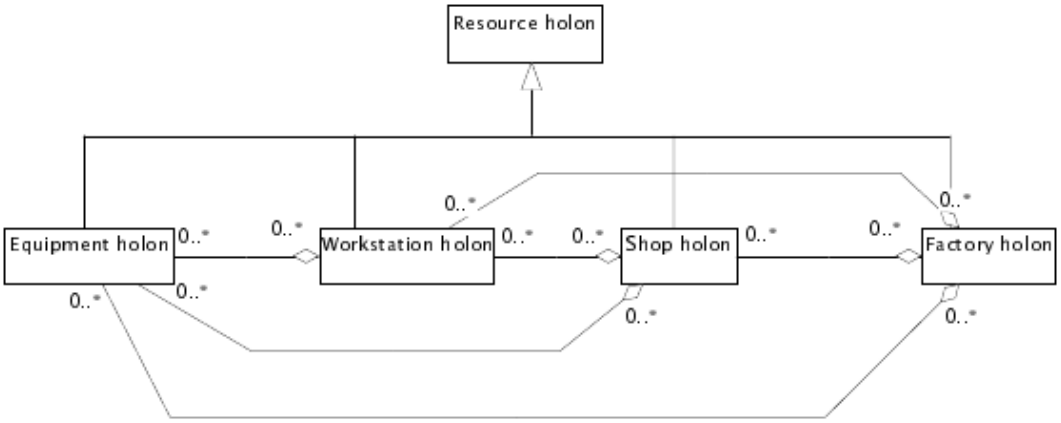


Figure 2.5: Aggregation of specialized resource holons. Taken from (Van Brussel et al., 1998)

Within PROSA, aggregation of a large number of entities are considered to produce a complex system behavior which is difficult to understand and to predict. The solution they propose to solve the problem is to structure the entities in a hierarchy. A typical holarchy in PROSA is shown in figure 2.5

Examples. PROSA is a generic reference architecture. Implementations of the architecture include: application of a PROSA-based system, developed under the MASCADA project, in a car body painting plant (Brückner et al., 1998), a control architecture for an automated guided vehicle system capable of being robust in the presence of disturbances (Liu et al., 2000), production control system of semiconductor wafer fabrication facilities entitled FABMAS (Mönch et al., 2003), railroad control (De Swert et al., 2006), photographic products out of large rolls of photographic foil (Saint-Germain et al., 2003) and an open job-shop plant for weaving machines components (Zamfirescu et al., 2003).

2.5.4. PolyAgent

In agent-based modeling, the fundamental entity is the agent, which represents one discrete entity from the system being modeled. An agent executes only one trajectory per run (simulation) hence alternative trajectories accessible to the entity in the evolution of a realistic system are not explored.

The solution proposed by the polyagent approach is to introduce a new modeling construct called “polyagent”. Each domain entity is represented by various agents in the polyagent approach. In contrast to traditional agent-based modeling, where one agent represents one single domain entity. A “polyagent” is the modeling construct introduced in the agent-based modeling work of Parunak and Brueckner. It is an approach to systems modeling designed to address some limitations of agent-based modeling (Parunak et al., 2007; Brueckner et al., 2009). It consists of two basic elements: the polyagent construct and stigmergic interactions.

A single persistent avatar manages the correspondence between the domain and the polyagent, and a swarm of transient ghosts that explore alternative behaviors of the domain entity. The avatar corresponds to the agent representing an entity in a conventional multi-agent model of the domain. Each ghost interacts with the ghosts of other avatars through digital pheromones, exploring many alternative trajectories in a single run that can proceed faster than real time for many reasonable domains. Figure 2.6 depicts in UML the kind of relationships between a polyagent, an avatar and a ghost. It persists as long as its entity is active, and maintains state

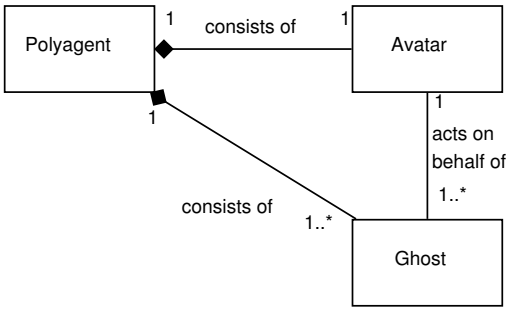


Figure 2.6: A polyagent represents an domain entity with an avatar and several ghosts.

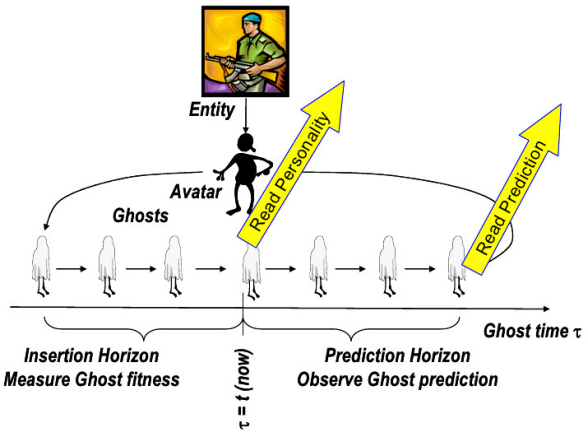


Figure 2.7: In the polyagent approach each avatar generates a stream of ghosts that sample the personality space of its entity. They evolve against the entity's recent observed behavior, and the fittest ghosts run into the future to generate predictions. Taken from (Parunak and Brueckner, 2006a)

information reflecting its entity's state. Its computational mechanisms may range from simple coordination to sophisticated reasoning.

Examples. It was first applied in a factory scheduling problem (Brueckner, 2000).

In (Parunak and Brueckner, 2006b) the polyagent approach is used to evolve a model of a group of soldiers and extrapolate its behavior into the future. Path planning for robotic vehicles. In (Parunak et al., 2005) the polyagent represent aircraft whose route needs to be computed.

2.5.5. Morphology

Morphology is a method to control a multi-agent system, based on the global shape of the system (Campagne et al., 2005). In this work, the target system is defined in terms of goals and sensors. Each system goal has a specific form which is a specific value for each sensor linked to a goal. These values are included in a particular field and have to be in this field to respect the goal. This part is called the Natural Behavior Restriction (NBR). A goal is made of: a list of sensors, a planning linked to the list of sensors and a field for each sensor to know if a sensor has a good value during the processing.

The aim is to direct the system behavior (communication between agents) towards the form or shape linked to the goal (in the case of a simple goal). If the goal is more complex, there is a planning for all subgoals processing and a specific form per subgoal. The basic idea is to project

the state of the agent organization in an abstract geometrical space, from various measurements made at the agent level. The projection is called morphology. The hypothesis underlying the approach is that the shapes representing the state of the system are correlated to the behavior of the system. A geometrical form is a specific algorithm adapted to a particular behavior to succeed a simple or complex goal.

The morphology approach proposes to build control mechanisms of a system through three "organizations".

- The *aspectual* organization, senses the external environment.
- The *morphological* organization describes the state of the aspectual environment in geometrical space.
- The *analysis* organization controls the aspectual organization using description(s) done by the morphological organization.

The interactions of the three different organizations and the environment of the system are presented in the schema of figure 2.8. The system designer puts the general trends that the target system should follow in the analysis organization. The control loop starts with the aspectual organization which evolves as the inputs from the sensors evolve. Then, the morphological organization obtains an aspectual landscape from the aspectual organization and translates the landscape to a geometrical shape. Based on the geometrical shape of the current state, the analysis organization simulates various ways to obtain a desired geometrical shape and applies the necessary control actions to the aspectual organization to obtain a particular shape. The aspectual organization interacts with the actuators of the system to make it exhibit the behavior correlated to a shape.

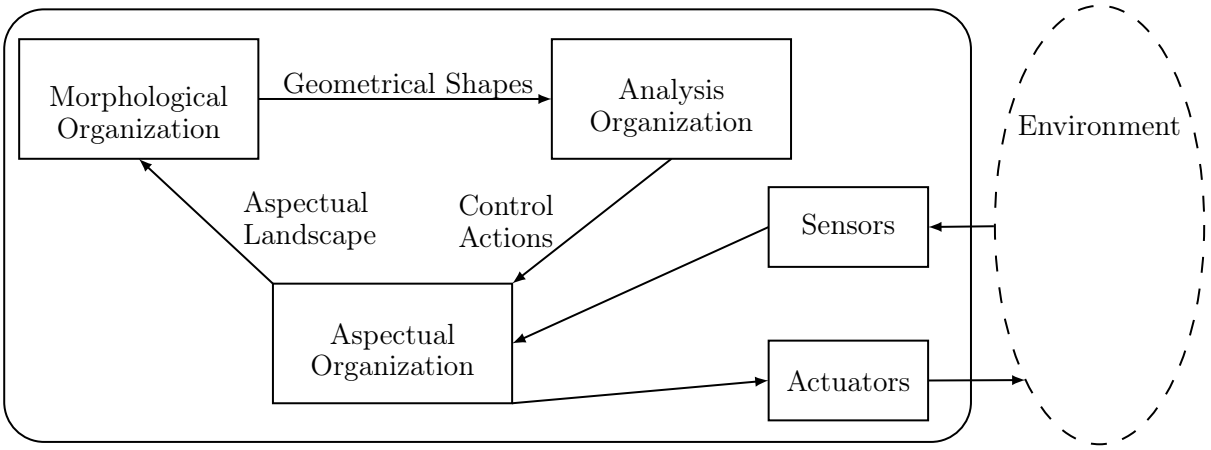


Figure 2.8: In the morphology architecture of control, the built system has as inputs the data provided by the sensors and as outputs the actions executed by the actuators.

Example. In (Camus and Cardon, 2006) the authors present a method based on a multi-agent system to move towards a generic algorithm in order to control robot in real time. The model used to control the multi-agent system is based on the work of (Campagne, 2005). The software which manages the robot is a multi-agent system. It receives all sensors values in input to make a scene representation, interpret data, build an action plan and send effectors values to

the output. All agents receive all values from all sensors. All agents in the system are treated in parallel. If a value matches an agent role, this agent is activated to exchange data with other agent which has a link with it. The activation rate of a role gives the value of the associated sensor in the input. When several agents are simultaneously activated, a geometrical form appears. Then, the system is controlled based on the morphology of the appearing geometrical form. Experiments are based on the Sony Aibo ERS7 robot, where the behavior of the robot is observed according to its ontology and goals.

2.5.6. Emergent Engineering

Emergent engineering is a methodological framework for deploying large-scale networked systems. It is illustrated with self-organized security (SOS) scenarios (Doursat and Ulieru, 2008; Ulieru and Doursat, 2011). It involves an abstract model of programmable network self-construction in which nodes execute the same code, yet differentiate according to position. It relies on defining the basic entities and the mechanisms by which these entities are able to create reliable architectural components. Control is broken down and distributed locally at every entity of the system.

At the microscopic level is the genotype. It is the rules of behavior of the agents. At the macroscopic level is the phenotype. It is the resulting overall behavior of the collective. The approach consists of indirectly designing the genotype (or letting it evolve towards an appropriate phenotype), rather than directly building the phenotype.

The principles behind the emergent engineering approach are:

- Bottom-up design of the entities. Build systems by specifying generic building blocks and interactions that should lead to a coherent and desired emergent behavior.

  Instead of instructing each entity about what to do at each moment, micro-controllers are distributed among the entities. Each entity has rules of two possible kinds:

  1. Positive feedback that amplifies small local fluctuations, giving the system a tendency to create new macroscopic structures.
  2. Negative feedback that dampens or corrects the agent's response, and tunes its behavior more finely.

- Co-evolving the system with the environmental dynamics. This can be done by specifying how the genotype may vary and how the phenotype may be selected. After reaching structural maturation on a short deployment time scale, the system should switch the bulk of its activity from executing the developmental part of its genotype (dynamic architecting, which positions the actors within the network so that they can best perform their activity in coalitions or teams) to executing the functional part of its genotype (adaptive control obtained by executing their roles within the teams to realize the most effective action plans).

Example. The emergent engineering approach has been applied to build an SOS network, that develops in an emergent way (the control mechanisms are generic and instantiated in every entity of the network). All nodes in the network carry the same program (their genotype). The basic idea is that a generic network is programmed but gradually every agent and entity in the network differentiates according to (limited) positional awareness.

2.5.7. Control of reactive multi-agent system with reinforcement learning tools

The approach is intended to control a reactive multi-agent system (Klein et al., 2005, 2008; Klein, 2009) by using reinforcement learning techniques to learn the global behavior of the target system and choosing the control actions. In it, the different reachable behaviors of the system are considered as global states. Control is defined in this approach as: making the system show a target global behavior thanks to actions taken correctly chosen and performed at the right time. The type of the considered possible control actions can be local or global, depending on the capabilities of the controller.

The approach consists in the following phases. First, characterize the global behavior and its automatic measurement. Second, choose the best control actions. Third, determine a control policy that indicates the actions to perform. The decision of which control action to apply is taken on-line (while the target system is in execution). Also, the decision is taken dynamically: a decision is dependent on the state of the target system.

Examples. The feasibility of the approach is presented within a Pedestrian Model in (Klein et al., 2008). In it, reinforcement learning techniques are used to learn the behavior of the system and select the best control actions. In the example, the target system is a simulator of a reactive multi-agent system made of pedestrians walking in a circular corridor. In the system, the agents are leaded by a sum of forces.

2.6. Synthesis

We keep a series of characteristics from the different approaches presented in this chapter that we now group by the way the target system state is estimated, the way the local and global levels are taken into consideration and the characteristics of the control mechanisms.

Target system state estimation. We have identified different ways to estimate the future target system state. For (Klein, 2009), it is a matter of reinforcement learning. With the PROSA and PolyAgent approaches, it is a matter of exploring the different possible futures while the target system is on execution. Exploring different possible futures, allows to take directly into consideration the sensibility to initial conditions as well as the non linear nature of complex systems dynamics.

Local and global levels. Some of the approaches we presented deal with the modeling difficulty through the multi-agent paradigm. An exception being the approach of (Klein, 2009), where the global state of the system is modeled with a state machine. Using multi-agent models allows to take into consideration at the same time, the local and the global description levels of the target system.

Control mechanisms. We have identified two kinds of control mechanisms in regard to the moment they were built.

- Endogenous control mechanisms are built at the same time as the target system. This kind of mechanisms hold the hypothesis that it is possible to modify the internal behavior of the entities in the target system. As a consequence of this, in those systems where because of technical, legal or ethical reasons, it is not possible to change the behavior of the entities, this kind of mechanisms are not well suited. Hence, this kind of mechanisms are better suited to artificial complex systems.

- Exogenous control mechanisms are built outside and not at the same time as the target system. This kinds of mechanisms hold the hypothesis that direct modification of the internal behavior of the target system entities is barely possible. As a consequence of this, we consider that this kind of mechanisms are better suited to be applied in natural as well as artificial complex systems.

2.7. Conclusions

In this thesis work, we consider the problem of driving a complex system to a certain state, that is, on controlling it. We saw that feed-back control loops are the basic building blocks within control theory. We also saw that within these control loops a means to estimate the future state of the target system is necessary. The traditional tool used to estimate the future state of the target system are analytical models. However, such tools ought to have solutions in order to be useful.

We have already set our stand point on the difficulties that models of complex systems have to overcome within the control challenge: they must take into consideration *a)* the different levels of a complex system (local and global for example) and *b)* the emergence of global outcome from local interactions.

In this chapter, we presented the equation-free approach to model systems at a global level when only local level models are available. We can consider that the resulting models created with an the equation-free approach will include the local level of description of a complex system and the global as well because it includes two models with different description levels. The fine-scale model corresponds to the local description level and the coarse model corresponds to the global level.

Also, we presented the multi-agent modeling paradigm to model complex systems. This paradigm explicitly allows to take into consideration the different levels of a system and allows to explicitly observe the effects of local interactions at the global level.

The first conclusion that we draw from this chapter is that building a feed-back control loop, with an equation-free approach to estimate the future state of a target system allows, at least in theory, to overcome the limits of the analytical models. Moreover, we conclude that by using the multi-agent modeling paradigm within the control loop, we can overcome the difficulties we have identified, of modeling complex systems: multi-agent models explicitly take into consideration the global and local level and allow to explicitly represent emergence.

A second conclusion that we obtain from this chapter concerns the different applications of the multi-agent paradigm to control complex systems. What we saw in the presented example applications is that they tackle the problem from the “design” or “engineering” point of view and as such, they are hardly applicable without major modifications to preexisting systems.

Chapter 3

A control architecture

Contents

3.1. Introduction 43
3.2. Principles 44
3.3. Definition of the architecture 45
3.3.1. Architecture elements 45
3.3.2. Hypothesis of the architecture 46
3.3.3. Concise definition 47
3.3.4. Preliminary synthesis 47
3.4. Detailed view 48
3.4.1. Block Definition Diagram of the architecture 49
3.4.2. Internal Block Diagram of the architecture 51
3.4.3. Execution flow of the architecture 51
3.4.4. Complementary view of the elements of the control architecture 52
3.5. Assessment on the relevance of our proposition 54
3.5.1. Architecture elements and difficulties of the control of complex systems challenge 54
3.5.2. Equation free approach and multi-agent models in the architecture . . . 55
3.5.3. Governance related aspects 56
3.6. Conclusion 57

3.1. Introduction

The objective of this thesis work is to evaluate the pertinence of multi-agent based simulation in the context of control of complex systems. As a requirement for the evaluation, we consider that it is necessary to have a coherent system that addresses the control challenge, including multi-agent based simulation. In this chapter, we propose such a coherent system with the form of a control architecture.

Regarding the different difficulties of the control challenge stated in 1.5, and based on the conclusions of the previous chapter, we shall build a solution to tackle the challenge, as follows.

Modeling the global effects of local actions. We shall focus on the equation-free approach to model complex systems. Particularly, we shall focus on using multi-agent microscopic models. The multi-agent approach shall allow us to directly take into consideration the local and global dynamics of the target system.

Preexisting systems. We shall focus on exploring exogenous control mechanisms. This shall allow us to implement control mechanisms from the outside of a target system.

Autonomy of entities. The hypothesis held by the implementation of exogenous control mechanisms ("it is barely possible to change the internal behavior of an entity") shall allows us to respect the autonomy of the entities of a target system.

These conclusions are the foundations that shall guide our proposition of an architecture integrating the previous elements.

We shall first describe the principles supporting the architecture in 3.2. Then, in 3.3, we will firstly expose the constituent elements of the architecture. Secondly, we will present the hypothesis underlying our proposition. Thirdly, we will propose a concise definition of the architecture. Finally in the same section, we will provide a preliminary synthesis of the questions specific to multi-agent based simulation inside the architecture. Afterward, for the purpose of exhaustiveness, we shall present a detailed description of all the elements of the architecture in 3.4. Since this chapter provides the conceptual framework (the architecture) that will enable the evaluation we are interested in, we shall end by giving an assessment of our proposition in section 3.5. At the end of the chapter, in section 3.6 we shall design the next steps in the path of our evaluation of pertinence.

3.2. Principles

The principles of the control architecture are three: explicit feedback loop, exogenous implementation, and equation-free models modeling of the target system and control actions.

Feedback loop

When we introduced the feedback control loop concept in section 2.2 we defined it as the influence of one system over another. From the feedback control loop we borrow the following concepts:

- We have two systems, C and T. System C is our architecture and system T is the target system to be controlled.
- System C will influence system T in order to make the output y of T be as close as possible to a reference value. The output of system T is used as input for system C which in turn will produce an output that will become the input of system T.

A first schematic representation of the feedback loop is presented in figure 3.1.

Exogenous implementation

The exogenous principle of the architecture is understood in two senses.

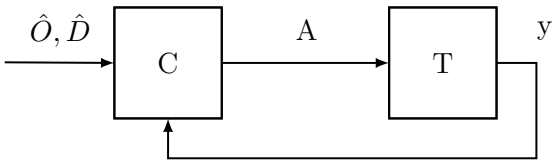


Figure 3.1: Feedback control loop including all the inputs and outputs of the architecture.

On one sense, it is exogenous because system C is meant to be an independent system. That is, if it stops working, it will not prevent system T from working. On another sense, it is exogenous because it is built under the hypothesis that the preexisting system T cannot be stopped to add the architecture as a control mechanism.

Equation-free modeling

The models used to determine the state of the target system as well as the effects of control actions are simulable models given at a local description level, in particular, multi-agent based models.

In section 2.3, we summarized the most important idea from the approach: the need for a coarse time-stepper that controls when and how simulations of the local description level model are done. The functions of the coarse time stepper of the equation-free approach are implemented each time multi-agent model simulation is required.

3.3. Definition of the architecture

The above mentioned principles give a guideline on how to build our architecture. We shall first identify the basic elements that conform the architecture, to present later on this same section (see 3.3.3) a concise definition.

3.3.1. Architecture elements

We have two different separate systems: our architecture C and a target system T. Each system has its own parameters and dynamics.

The architecture is made of the following elements.

Control objective. The objective of the architecture is to guide the target system to exhibit a desired state. It is deemed $\hat{D}$.

Future State estimation. The estimation of the future state of the target system is obtained through multi-agent model simulation. Because of the nature of multi-agent simulation, we shall have multiple models giving as result multiple future state estimations. We retake the notation for multi-agent model simulation summarized in table 2.2.

A model is deemed M, the set of parameters of the model is deemed $Par_M = \{p_1, p_2, \ldots, p_n\}$, the set of initial values given to the parameters of the model is deemed λ_M. An instance of the model is deemed M^1 and the result of the simulation of an instance M^1 is deemed r_{M^1}.

Observation of the target system. This is necessary to be able to identify if the control objective is attained or not. Or, put in other words, it is a way to identify the state of

the target system. Also, it is necessary for the initialization, calibration and validation of multi-agent models. The set of observations is deemed $\hat{O}$.

Simulation of control actions. The possible effects of control actions applied to the target system are estimated with multi-agent models.

Application of control actions. We have a concrete control actions to be applied to the target system. The set of control actions is deemed $\hat{A}$. Because the outcomes of multi-agent model simulation are multiple, we shall decide at some point which outcome to use as the basis to decide which control action to apply. It is therefore necessary to have a criteria to compare them.

3.3.2. Hypothesis of the architecture

The architecture we propose is based on a set of hypothesis concerning the availability of some elements. To be able to implement a concrete instance of the control architecture we are proposing, the following requirements must be met.

Available models

The control architecture is based on the usage of simulation of multi-agent models. These models must be valid and already available to be used as an off the shelf element to be included in the architecture.

We assume that simulation duration is small enough to use the results in a relevant context. This means that the results should be available before the observed state changes.

Available control actions

The means to influence the behavior of system T are control actions. We assume that such control actions exist and that they have been defined at the local description level. Moreover, we consider that the control actions have been proven to have an influence on the behavior of system T.

Preexisting target system

We assume that target system T is already deployed and in execution. The need for an exogenous control mechanisms such as our architecture comes from the fact that it has been observed that the behavior of system T is not being controlled by its endogenous control mechanisms. Moreover, a means to observe and apply (already defined) control actions must be available to the control architecture to be instantiated.

Control objective

We assume that the control objective is already defined. Given that our architecture focuses on the behavior of system T, the control objective shall be defined in the same description level of the observable behavior of system T. Furthermore, given that the future state of system T will be estimated using multi-agent models, a means to compare the results of the simulations with the control objective must also be available.

3.3.3. Concise definition

Now that we have described the constituent elements of the architectures, we are going to center our attention on the way these elements are linked with each other. Our proposition takes the above mentioned principles and elements, to form a control architecture that has the following form.

> The architecture we propose starts by estimating the state of the system, then compares the estimated future state to the desired one to see if any actions are required to meet the guarantees of keeping the system at a certain state. Then the control action to apply is calculated and finally applied. Inside the control loop we just described, the multi-agent models are used to *predict the evolution of the system* and to *test the possible effects of control actions*, instead of equations.

From this concise description of the architecture we can already identify the relationships between the difficulties of the control problem of complex systems and the principles of our proposition.

- Local actions produce global effects. The principle of using multi-agent simulation in the architecture allows to estimate the future state of the target system explicitly taking into consideration both levels.
- Modeling. Using multi-agent models within an equation-free approach allows to tackle the challenge without analytic models.
- Preexisting systems. The exogenous nature of the architecture implies that :
  - the architecture runs in parallel to the target system
  - it doesn't change the internal behavior of the elements of the target system

  As such, the architecture is suitable to be implemented to control preexisting systems.

3.3.4. Preliminary synthesis

We have just described the constituent elements of the architecture and the form of the architecture. Moreover we have identified the relationships between the elements and principles of the architecture with the difficulties of the control challenge. There is one element at the core of our proposition that deserves special attention: multi-agent model simulation.

Multi-agent model simulation poses specific questions related to the implementation of the relationship between the model and the target system. This questions can be grouped as follows.

Validity relationship. This relationship requires of a way to identify if the results of the simulations are valid. The results of simulations are used to identify the (current future) state of the target system (with and without control actions). The execution flow of the architecture depends on the accuracy of the simulation results.

Calibration. Models have as parameters that are calibrated to produce the answers to the questions they are asked. This requires a way to find out how to calibrate the inputs of the models.

Translation of elements from the model to elements of target system. By nature, models are simplifications. However, in our context we require to have concrete control actions that make reference to concrete elements (entities or environmental) of the target system. It is required to have a means to translate the results of simulations to concrete elements in the target system.

We shall further elaborate on the matter (on section 3.5), once we have settled the specifics of our architecture.

3.4. Detailed view

We use the SysML language to present an initial black-box coarse view of the architecture. SysML allows to describe a system from various points of view, ranging from abstract to detailed. We have chosen to use SysML to define the architecture because it allows to do it from an abstract point of view in a non-ambiguous way. An additional reason to have chosen a formal description language is that our architecture can be more easily diffused and hence, better reproduced.

SysML. The Systems Modeling Language (SysML) is a general-purpose modeling language particularly suited for dynamical systems engineering applications. It is a standard published and managed by the OMG (SysML Specification, 2012). SysML is defined as an extension of a subset of the Unified Modeling Language (UML). SysML specifies nine diagram types. Out of those nine diagrams, seven belong to the specification of UML 2. SysML adds requirement and parametric diagrams. SysML diagrams contain diagram elements (mostly nodes connected by paths) that represent model elements in the SysML model, such as activities, blocks, and associations. A taxonomy of such diagrams, is presented in figure 3.2.

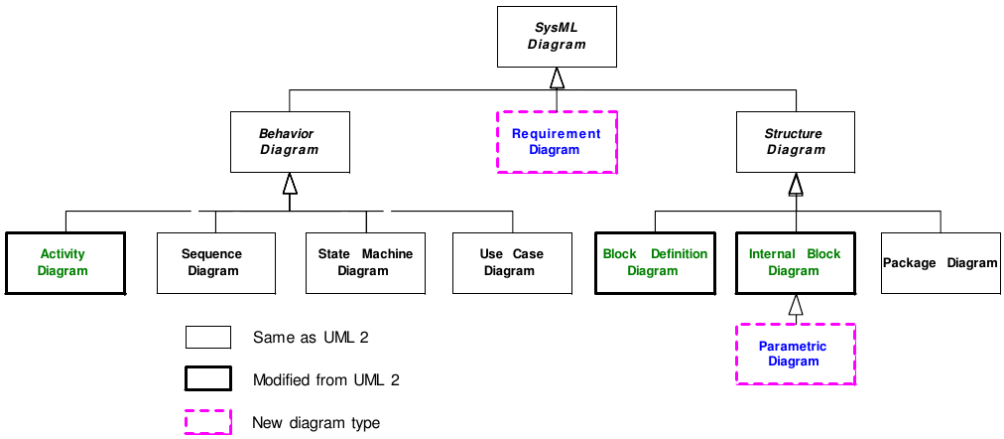


Figure 3.2: SysML Diagram Taxonomy, taken from (SysML Specification, 2012). The generalization and specification notation of UML is used.

The diagrams of a SysML model describe three different aspects of a system: the behavioral, structural and requirements aspects.

- Structure diagrams
  - The Block Definition Diagram (BDD), replacing the UML 2 class diagram
  - The Internal Block Diagram (IBD), replacing the UML 2 composite structure diagram

  - The Parametric Diagram, a SysML extension to analyze critical system parameters
  - The Package Diagram remains unchanged
- Behavior diagrams
  - The activity diagram has been slightly modified in SysML
  - The sequence, state chart, and use case diagrams remain unchanged
- The requirements diagrams is a SysML extension

In SysML, a system model is made of blocks. Blocks are modular units of system description. Each block defines a collection of features to describe a system or other element of interest. These may include both structural and behavioral features, such as properties and operations, to represent the state of the system and behavior that the system may exhibit.

- The Block Definition Diagram provides a black box representation of a system block (for example the main block), alongside the hierarchy of its composite blocks. Block definition diagrams represent blocks in terms of properties and operations. A property can represent a role or usage in the context of its enclosing block. A part defines a local usage of its defining block within the specific context to which the part belongs. A classifier behavior is an operation of a block that is considered to be the one that is executed whenever a block is instantiated.
- The Internal Block Diagram in SysML captures the internal structure of a block in terms of properties and connectors between properties. The Internal Block Diagram or IBD provides the white box or internal view of a system block, and is usually instantiated from the BDD to represent the global assembly of all blocks within the main system block.
- Composite blocks from the BDD are instantiated on the IBD as parts. These parts are assembled through connectors, linking them directly or via their ports (standard ports with exposed interfaces and/or flow ports).

In comparison with UML 2, the SysML IBD redefines the composite structure diagram by supporting blocks and flow ports. Ports are a special class of property used to specify allowable types of interactions between blocks. Blocks may also specify operations or other features that describe the behavior of a system.

There are more types of diagrams in the SysML language, but we do not use them all in the definition of our architecture. Our architecture is intended as generic pattern. The diagrams we shall use in the following subsections, are sufficient to present the architecture at the degree of abstraction we are interested in.

3.4.1. Block Definition Diagram of the architecture

In terms of SysML, the architecture at a broad level is represented by a block called “Architecture”. It is composed of blocks charged of the different functions of the architecture. These blocks are:

1. Observe Target System. It is charged to provide the architecture with information observed from the target system.

2. Estimate Future State. It is charged to execute the multi-agent simulation of models used to estimate the target system.

3. Simulate Control Actions. It is charged to execute the multi-agent simulation of models of the effects of possible control actions.

4. Apply Control Actions.

The diagram in figure 3.3 represents the hierarchy of elements in the architecture.

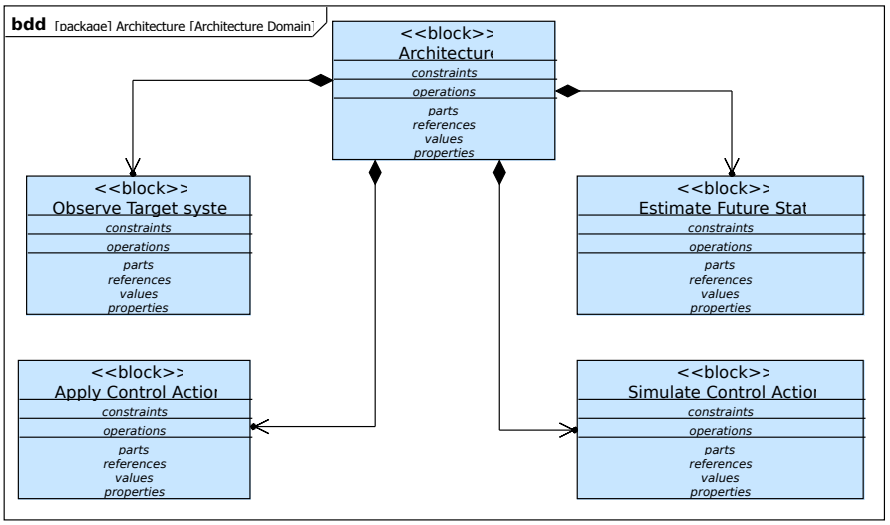


Figure 3.3: BDD diagram of the architecture.

The inputs of C are the different elements that must be fed to the architecture for it to work. The outputs of C are the different elements produced by the architecture.

Inputs of the architecture

We shall denote $\hat{O} = \{o_1, o_2, \ldots, o_n\}$ the observations made from the target system T. Figure 3.1 is in fact too simple to describe all the processes present in a feedback control loop. One missing element for example is the observation of the system. In general, systems conceived with a control theory approach expect to have some noise in the measurement of the outputs because of actuators or random noise. In our context, complex systems, it is more evident that we will not be able to have full access to y (either because it is impossible or because of the same "noise"). So, although 3.1 depicts y as being the input for system C, in fact the input is the observation of T.

The reference from the control feedback loop will be deemed $\hat{D}$ in the architecture. There can be, associated to the control objective, a time horizon. Associating a time horizon to a control objective means that the control objective should be met for the associated time horizon. The subscripts in the different elements of $\hat{D}$ indicate the expected behavior at a certain time horizon. These time horizons could be thought of short term (t_a), mid term (t_b) and long term (t_c).

Outputs of the architecture

The output of our architecture are control actions to be applied, denoted by $A = \{a_1, a_2, \ldots, a_n\}$. Table 3.1 presents a summary of the inputs and outputs of the architecture elements.

Description	Notation	Input for		Output for	
		C	T	C	T
Observations	$\hat{O} = \{o_1, o_2, \ldots, o_n\}$	✓			✓
Control Actions	$A = \{a_1, a_2, \ldots, a_n\}$		✓	✓	
Control Objectives	$\hat{\mathrm{D}} = \{d_{t_a}, d_{t_b}, d_{t_c}\}$	✓			

Table 3.1: Different inputs and outputs of the architecture.

3.4.2. Internal Block Diagram of the architecture

In this diagram we specify the interactions between the different blocks of the architecture. An association between two blocks defines an interaction. If the association is made with the use of ports, the port type specifies the type of element flowing from one block to the other. Figure 3.4 illustrates the different interactions among the architecture parts. The classifier behavior of the architecture is called the "Main Control Loop" and it interacts with all the other parts. The different items flowing from one block to another are specified in the activity diagram of figure 3.5.

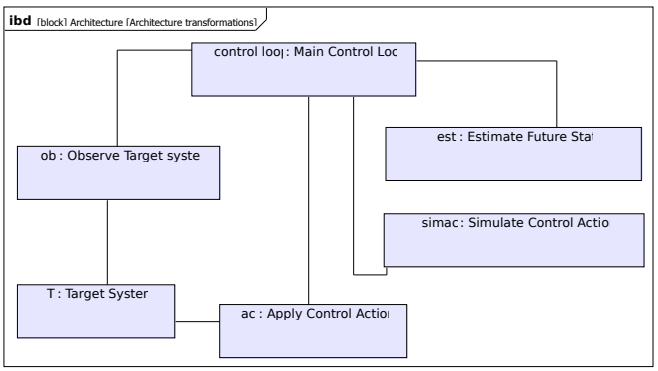


Figure 3.4: The main control loop is the classifier behavior of the architecture. Each line depicts an interaction between parts.

Now that we have defined the functions of the architecture as blocks and the different inputs and outputs of the architecture as well as the interactions between the blocks, let us consider the order in which the operations of each block take place.

3.4.3. Execution flow of the architecture

The classifier behavior of the architecture (the "Main Control Loop"), is where the execution flow should be implemented. This part of the architecture initiates and stops as necessary the execution of the functions of the parts involved in the execution flow. The execution flow can be summarized as follows.

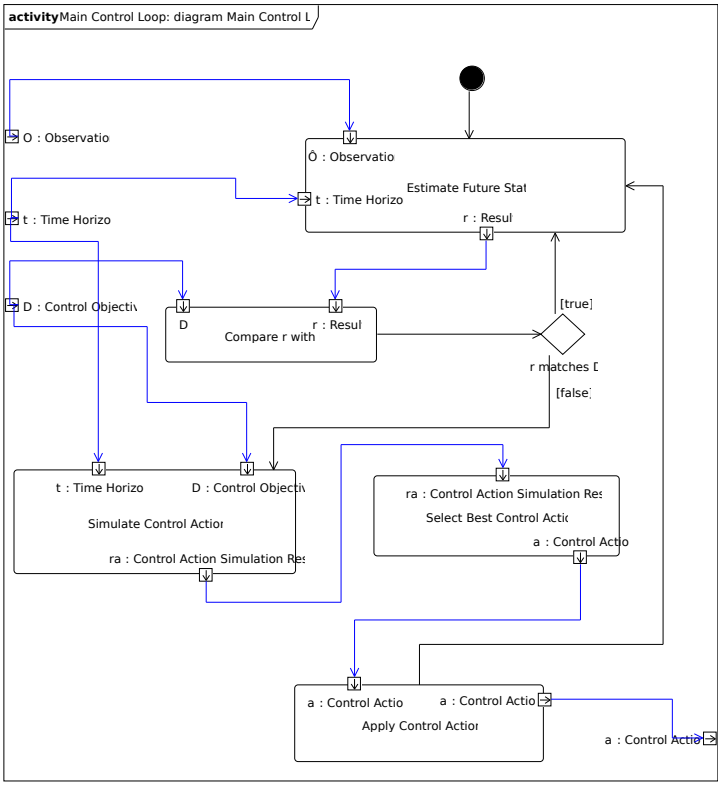


Figure 3.5: Item flow of the blocks in the architecture.

The first activity executed by the architecture is to estimate the future state of the target system up to a time horizon corresponding to the control objective. Then, the estimated future state is compared to the control objective to see if any actions are required. If control actions are required, they are determined by the *simulate control actions* block and then applied by the *apply control actions* block. If no control actions are necessary, the main control loop restarts. This execution flow is illustrated in figure 3.5.

3.4.4. Complementary view of the elements of the control architecture

Future State Estimation

The *estimate future state* block is responsible for carrying out this function. Since this block requires as input the observations from the target system, the *observe target system* block should be instantiated in advance. The main control loop is in charge of passing the observations $\hat{O}$ as well as the time horizon t to the *estimate future state* block. The block simulates the evolution of T up to time horizon t. The result of the simulation r is passed back to the main control loop.

Decide if control actions are necessary

The main control loop is responsible for comparing the results of the simulations with the control objective. If the results of the simulation indicate that for time horizon t the state of the system differs from the one indicated by the control objective D, the main control loop

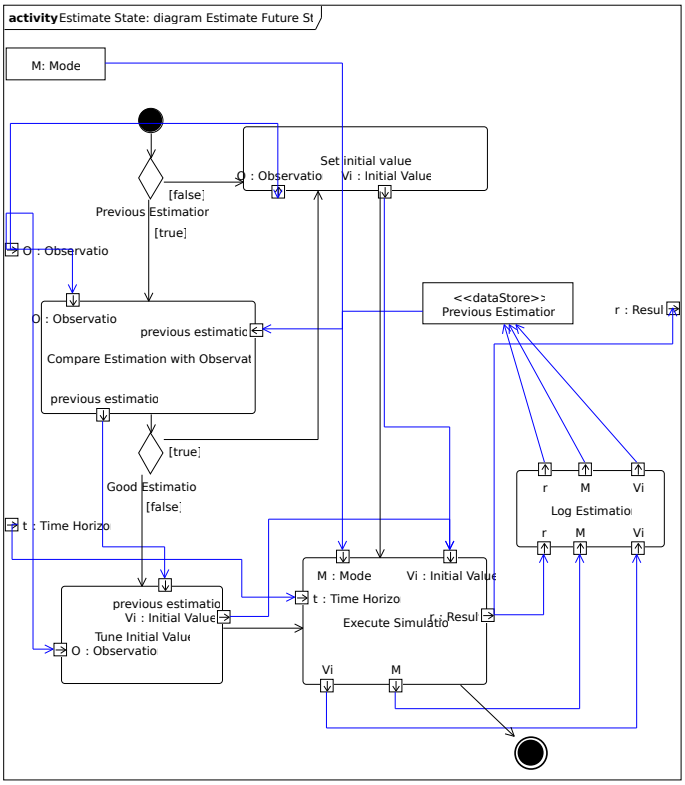


Figure 3.6: Activity Diagram of the Estimate Future State classifier behavior.

instantiates a *simulate control actions* block. If, on the contrary, the r does not differ from D, the main control loop will go back to estimate the future system state.

Simulate control actions

The *simulate control actions* block will receive from the main control loop the control objective D and the time horizon t and the observations $\hat{O}$ of T. It will determine the possible outcome of applying control actions given in $\hat{A}$ to the system. The result of the simulation ra will be passed to the main control loop.

Select control action

The results ra passed from the *simulate control actions* block will be compared by the main control loop to determine which control actions to apply. The set of chosen control actions are passed to the *apply control actions* block. Afterwards the main control loop will go back to estimate the system state.

Apply control actions

The control actions a will be adapted to the target system context and applied. At this point the main control loop will go back to estimate the system state.

3.5. Assessment on the relevance of our proposition

Now that we have made a detailed description of all the constituent elements of the architecture and a functional decomposition, we shall turn our attention to the pertinence of our proposition.

We evaluate the pertinence of our proposition from three points of view. The first point of view concerns the pertinence of the principles of our architecture in regard to the difficulties of the control challenge. This is discussed in 3.5.1.

The second point of view concerns the pertinence of using multi-agent model simulation in a feedback control loop, within an equation-free context. This is discussed in 3.5.2.

The third point of view concerns the relevance of the specific questions raised by the usage of multi-agent model simulation in a governance context. This is discussed in 3.5.3.

3.5.1. Architecture elements and difficulties of the control of complex systems challenge

We defined the difficulties of controlling complex systems in section 1.5. These difficulties are: modeling complex systems, local actions with global effects, the autonomy of participants and preexisting systems.

Based on the definition of the architecture, we shall now identify how our architecture takes into consideration the characteristics of complex systems in the control challenge.

Modeling

In our control architecture we are faced to modeling the complex system T in the *estimate future state* and *simulate control actions* blocks. The principle of the architecture is to use an equation-free approach to model T. Within this approach, simulation is used to calculate the evolution of T. We specifically consider multi-agent models.

Local actions with global effects

One of the characteristics of complex systems having a great influence on the control challenge is emergence. In particular, the *simulate control actions* and *estimate future state* blocks are concerned with emergence. The principle of the architecture is to use multi-agent models. The *simulate control actions* block has to specifically apply control actions defined at the local description level in order to achieve a control objective D that may be defined in either local or global description level. We consider that assuming control actions defined a the local description level is plausible given the emergence present in complex systems and also because complex systems are decentralized.

Preexisting systems

When dealing with complex systems that are already deployed, we believe to be a safe hypothesis to consider that it is not possible to completely stop them in order to apply control actions. Moreover, given that complex systems are made of autonomous entities, we assume that the internal behavior of the entities cannot be modified (or else they would no longer be autonomous). In particular, the *apply control actions* block of the architecture is concerned with this hypothesis. The architecture is a system running in parallel to system T, as an exogenous system. The main execution flow of the architecture is conceived as a feedback loop, although

the outputs of C that become the inputs of T are not indispensable for the proper execution of T. They are only the means to influence the behavior of T but they can be absent at any moment. In this way, if the architecture is stopped it will not stop the execution of the system T (no matter if the control objective has been reached or not).

3.5.2. Equation free approach and multi-agent models in the architecture

As explained in section 2.3 the equation-free approach requires models suitable of being simulated. Yet, not just any model suitable of being simulated can be used within the context of our architecture. The models we consider as appropriate for the architecture should have two characteristics. They should be ready to be executed and they should be built with a degree of knowledge of the modeled system that allows explicit description of the interactions of all components of the model.

The main idea behind the equation-free approach is the use of microscopic models to perform macroscopic analysis of a model. The approach considers the usage of sound mathematical techniques to speed-up the microscopic simulation. In the context of agent-based simulation, we can synthesize this idea with algorithm 2.

Algorithm 2: Course of action of the equation-free approach with multi-agent simulation to obtain a macroscopic variable after T time (y_{T+i}), given y_i initial conditions.

input : Initial conditions y_i at time t_i
input : Number of model instances to execute n
input : Time of desired prediction T

$InitialConditions \leftarrow y_i$;
$t \leftarrow t_i$;
while $t < T$ **do**
  forall the n *instances* **do**
    $\lambda_M^n \leftarrow$ Random Initial Parameters ($InitialConditions$);
    Initialize Model(M^n, λ_M^n);
    Simulate Model (M^n, t);
    $r[n] \leftarrow$ Extract Macroscopic Value of Variable(y, $s_{M^n}^f$)
  end
  $t, InitialConditions \leftarrow$ Analysis(r);
end
$y_{T+i} \leftarrow$ StatisticalAggregation(r);

If we compare the course of action previously described with the one of algorithm 1 we see that in both cases we need to know the time horizon of interest T. The difference is that in the classical course of action, every time step between t_i and T are simulated, whereas in the equation-free approach this is avoided. The "Analysis" function (driven by sound mathematical techniques such as coarse projective integration or patch dynamics (Gear et al., 2002; Kevrekidis et al., 2003)) extrapolates the results of the simulation. These extrapolations become new initial conditions and they are used to reinitialize the simulation at a time step comprised between t_i and T with the obtained initial conditions.

The objective of the architecture we propose is to show the specifics of using an equation-free approach within a control feed-back loop. The architecture is meant to take advantage of existing models of the target system. From this point of view, we consider that the model should

be available and ready to be executed. That is, we expect the models to be used to be fully specified and already validated. Fully specified means that all relevant parameters should be defined as well as their value ranges. Moreover, the degree of validity of the model outputs, in regard to the inputs, should also be already defined.

In regard to the system specification hierarchy presented in 2.4.2, the models required for our architecture must be at the "Coupled component" level. At this level, not only do we know how to generate the behavior of the simulated system, but we have sufficient knowledge of the simulated system as to be able to establish links between the simulated objects and the ones in the simulated system. In our architecture we use simulation of multi-agent models to determine which control actions to apply.

The control actions to be applied will be described in terms of the simulation. There are two possibilities of how to apply the control actions resulting from the simulations: *immediately* and *after translation*. *Immediately* means that, if the model (where the control actions come from) was constructed with elements that can be directly identified in the target system. *After translation* involves different phases. First, identify the characteristics of the elements involved in the control actions. Second, identify the elements in the target system sharing these characteristics. Third, redefine the control actions in terms of the elements of the target system, and fourth, apply them. The application of control actions after translation is particularly important when models are initialized with values taken at the macroscopic level.

3.5.3. Governance related aspects

The main objective of the architecture is to drive the system to a certain state. The way to know if the objective has been reached is by comparing the estimations made regarding the evolution of the state (with or without control actions) with the control objective. This comparison is an operation carried out by the classifier behavior of the architecture.

So far we have described the series of operations performed in the architecture to achieve control of a target system under the assumption that control is definitely achieved. Let us now turn our attention to the situation when the comparison between the estimated behavior (with or without control actions) indicates that the estimations significantly differ from the observations. If we have bad estimations it means that the results of the simulations are not correct.

- One important assumption of our proposition is that models are valid at the coupling component level. On the other hand, in the context of complex systems, we assume that the structures of the target systems as well as the behavior of the entities is dynamic. As a consequence of this, the validity of the models may as well evolve.

- Additionally, models have inherent simplifications and under the context of complex systems, the hypothesis that support the simplifications may also evolve over time.

- Our architecture is exogenous to the target system. It is reasonable to consider that we can observe the different interactions between the entities. However, it might be more difficult to observe aspects related to the coupling of the entities with the environment. The coupling refers to the update scheme and the perception horizon of the agents in a multi-agent model.

- Again, from the complex systems point of view, we can consider that control actions defined at the implementation moment of the architecture were valid, but that in time they became less efficient.

We consider that to deal with all these aspects it is necessary to explore multiple possible outcomes of simulations (for state estimation and control action effects estimation). Different simulations might yield different results for different coupling aspects or for different structures of the target system. This is a choice to be made when implementing the architecture.

3.6. Conclusion

The aim of this thesis work is to evaluate the pertinence of using multi-agent models in the context of control of complex systems. To fulfill the aim of this thesis, we intend to execute the evaluation firstly by proposing a coherent system that directly tackles the difficulties of control of complex systems. In this chapter, we defined such a coherent system: our control architecture.

The architecture pattern is built upon elements that are already implemented in different approaches, at different degrees. The elements are the future evaluation of the state system through multi-agent simulation, the usage of the equation-free paradigm to guide simulations, and an exogenous conception of the architecture. The definition of the architecture constitutes the foundations of our work.

We presented the principles of our proposition: a feed-back loop, an equation-free approach with multi-agent simulation and exogenous. More importantly, we clearly identified how the principles of our architecture address directly the different difficulties of the control of complex systems challenge. From this point of view, we consider that the architecture we propose is the coherent system that is necessary to evaluate the pertinence of multi-agent model simulation within the control of complex systems context.

The next step to fulfill the aim of this thesis work, is to study the implications of integrating multi-agent model simulation in an equation-free approach, in the context of control of complex systems. To study these implications, we shall now concentrate on a concrete implementation of the architecture in chapter 4.

The third step shall be to evaluate the influence of the governance hypothesis in the implementation of our proposition. Let us recall that although we stick to the term control within this thesis work, we place our work under the basic hypothesis of governance: it is impossible to attain optimal control of a complex system.

Introducing multi-agent model simulation in our proposition implies that issues related to the multi-agent modeling approach shall arise. Namely, issues concerning how to establish the relationship between the model and the target system: validity of models, calibration and translation of entities from the model to those of the target system. Further, in chapter 5 we shall take a closer look at the different choices that have to be made when implementing the architecture, related to the use of multi-agent model simulation in a context of non optimal control.

Chapter 4

Proof of concept Implementation

Contents

4.1. Introduction 60
4.2. Peer-to-Peer networks as complex systems 60
4.3. Target system specification 61
4.3.1. Description 61
4.3.2. Target system state 61
4.4. Target system implementation 61
4.4.1. Internal behavior of the peers 62
4.4.2. Number of peers in the network 63
4.4.3. Network organization 63
4.4.4. Initial proportion of sharing peers 63
4.4.5. Update Scheme 63
4.4.6. Summary 63
4.5. Preliminary study on the behavior of the target system 64
4.5.1. Analytical point of view 64
4.5.2. Beyond the analytical model 66
4.5.3. Results and conclusions on the preliminary study 66
4.5.4. Retained Initial conditions 66
4.5.5. Focus on the behavior of the target system associated to retained initial conditions 68
4.6. Experimentation platform 69
4.7. Experimental Setup 70
4.7.1. Control Objective 70
4.7.2. Architecture implementation overview 70
4.7.3. Architecture implementation details 71
4.8. Experiments and Results 74
4.8.1. Description of experiments 75
4.8.2. Synthesis of results 76
4.8.3. Discussion on the results 76
4.9. Discussion 78
4.10. Conclusions 79

4.1. Introduction

The purpose of this chapter is to present an example implementation of the architecture to assess the feasibility of our proposition and to illustrate how to instantiate the architecture in a concrete case. In particular we want to know to what extent the architecture controls a given complex system and what concrete elements are necessary to instantiate the architecture.

To demonstrate the feasibility of our proposition we are going to instantiate it to control a concrete target system. The target system shall be a simulated peer-to-peer network and we shall focus on a collective phenomenon present in peer-to-peer networks called free-riding. The control will be related to the contribution level of the network, that is, the proportion of peers that share in the network.

We are going to set an experimental protocol where the target system shall exhibit a global behavior that we wish to avoid: a majority of peers that do not share. In this same experimental protocol we are going to instantiate each block of the architecture and configure it to fulfill the control objective of keeping the system in the desire state (majority of sharing peers).

To demonstrate the feasibility of our architecture, we shall define an experimental setup including a target system and the architecture. Before detailing the experimental setup we shall first describe our target system. In section 4.2 we present the application domain of the target system and the phenomenon that we are interested in.

In sections 4.3 and 4.4 we define the target system used in the experimental setup.

A preliminary study necessary to establish the initial conditions of the target system necessary for the objectives of this chapter is presented in section 4.5.

The experimentation platform used to develop the work presented in this chapter is presented in section 4.6.

The architecture implementation is defined further as part of the experimental setup in section 4.7. At the end of the experiments, we shall compare the evolution of the contribution level of the system with and without our architecture to verify if control was achieved.

The results and specifics on the experiments are presented in section 4.8.

We shall discuss the obtained results in regard to the objectives of our work and in regard to the specific issues of the application domain in section 4.9.

Finally, in section 4.10 we make an assessment of the work accomplished in this implementation in regard to the objectives of this thesis work.

4.2. Peer-to-Peer networks as complex systems

A peer-to-peer computer network is one in which each computer in the network can act as a client or server in the network, allowing shared access to various resources such as files, computing time, and sensors without a central server. Peer-to-peer file sharing networks like BitTorrent (Cohen, 2003) , Kademlia (Maymounkov and Mazières, 2002) or eDonkey (Heckmann et al., 2004) are techno-social systems that have some characteristics shared with complex systems (Shahabi and Banaei-Kashani, 2007), namely those that we are interested in: they are composed of autonomous participants, they may have millions of participants and because of their self-organizing and open nature they exhibit nonlinear behaviors and it is not simple to gather information from them (Aidouni et al., 2009; Stutzbach and Rejaie, 2005).

We are particularly interested in a collective phenomenon present in this kind of networks: free-riding.

In peer-to-peer file sharing networks each member of the network or "peer" is supposed to contribute to the system by sharing its files in return for being able to download files from other peers (Androutsellis-Theotokis and Spinellis, 2004). Free-riding in a peer-to-peer file-sharing network means to be able to download files from other peers without sharing any. Empirical reports such as: (Saroiu et al., 2002; Makosiej et al., 2004; Yang et al., 2005; Zghaibeh and Harmantzis, 2008) show that free-riding is wide spread on file-sharing networks. Extensive presence of free-riders in a network can degrade its performance up to the point of rendering it useless, that is why it is important to control their presence (Krishnan et al., 2004).

Free-riding has been investigated for centuries in the form of social dilemmas such as *The tragedy of commons* (Glance and Huberman, 1994). As such, the problem is a prototypical example of a family of problems with fixed point dynamics and emergence. Thus, studying it, can help to shed light into a whole family of problems.

However, the current experimental work will be on the assessment of our proposal.

4.3. Target system specification

We are interested in the free-riding phenomenon in peer-to-peer networks. Instead of using an already deployed peer-to-peer network, we are going to work with a simulator.

We use a simulator since it is not easy to directly work with an already deployed file-sharing network because of technical and even legal reasons. Also, using a simulator allows us to measure every parameter of the system and hence, better assess the performance of the architecture. Another advantage is the possibility to deal with uncertainty and be able to reproduce the experiments, impossible otherwise within a real peer-to-peer network.

Our target system is a peer-to-peer network simulated using the simulator "PeerSim" (Jelasity et al., 2003), developed within the European projects "Bison" and "DELIS".

4.3.1. Description

The target system is characterized by a set of peers connected in a network of a certain topology. Peers have a "free-market" behavior: they share if they have an interest in doing so. The interest is given in terms of the proportion of the neighbors of a peer that share and an internal variable. The behavior is detailed in 4.4.1 The set of peers is detailed in 4.4.2 and the topology is detailed in 4.4.3.

4.3.2. Target system state

The global variable that we are interested in is the "contribution level". It is the proportion of peers in the network that share. We shall call this variable x. When the contribution level is greater or equal than 0.50, this means that there is a majority of peers in the network that share. Otherwise, we have a majority of free-riders in the network.

4.4. Target system implementation

The parameters of the target system can be implemented in different ways. In the following subsections we present how we have implemented each of the components we previously specified.

4.4.1. Internal behavior of the peers

The behavior dictates when a peer$_i$ shares or not in the network. It is configured by:

- A boolean sharing state: Sharing$_i$
- An internal generosity: θ_i
- A decision function.
- An internal depth d_i of neighbors used to estimate the global contribution level.

Generosity. In the experiments we have arbitrarily used a uniform random distribution of the generosity. Using a uniform distribution allows to compare the results of the experiments with the analytical model from which the peer behavior is inspired (Feldman et al., 2006).

Decision function. All peers in the target system have a behavior inspired from (Feldman et al., 2006): a peer i has an inner generosity θ_i and will share if its inner generosity is higher or equal to the cost of sharing in the network. The cost of sharing for peer i is defined as the reciprocal of the proportion of its neighbors that share. We shall refer to this behavior as the "free market" behavior. This decision is represented in algorithm 3.

Before making the decision of sharing or not, a peer calculates the "local" contribution level: the proportion of the neighbors of a peer that share. Algorithm 4 describes this.

Algorithm 3: Peer decision function

```
if generosity ≥ cost then
    share ← True;
else
    share ← false;
end
```

Algorithm 4: Topological model

```
forall the timeStep do
    forall the agent do
        x ← calculateLocalContribution(depth d);
        cost ← 1/x;
        agent.decide();
    end
end
```

Local contribution (with depth d). d is the depth of the neighbors each peer uses to estimate the global contribution level. The peers in the target system are restrained to calculate the cost of sharing based only in information available at a local level. In real peer-to-peer networks, full access to the information of the network is not always available.

When a peer estimates the contribution level by taking into consideration only its direct neighbors, we say that it performed an estimation of depth $d = 1$. An estimation of depth

$d = 2$ means that, besides its direct neighbors, a peer will take into consideration the direct neighbors of its direct neighbors, and so on for other values of d.

Other models of the behavior of peers in a file sharing network exist such as:(Vassilakis and Vassalos, 2007; Krishnan et al., 2004; Jian and MacKie-Mason, 2008). The one chosen for this work is interesting because of its ease to be instantiated as a multi-agent model.

4.4.2. Number of peers in the network

N is set up at the beginning of the simulation and will remain constant over time. Although peer-to-peer networks are open systems with high degree of churn, the amount of peers is kept constant to simplify the further analysis of the results.

4.4.3. Network organization

The network is created following the Watts and Strogatz algorithm (Watts and Strogatz, 1998). This algorithm allows to produce different families of graphs by only changing the value of parameter p: structured ($p = 0$) graphs, small-world graphs ($p = 0.25$), semi-structured graphs ($p = 0.50$), semi-random graphs ($p = 0.75$) and random graphs $p = 1$. This feature of the algorithm is particularly interesting because it will allow us to test the architecture with different families of graphs with little effort regarding the coding. An additional parameter necessary to create the graph is k, the amount of neighbors of each node at the beginning of the algorithm.

The algorithm starts with the graph of a circular, one dimension lattice made of N nodes. Each i node has links to its k closest neighbors: to the left $i-1, i-2, \ldots, i-\frac{k}{2}$ and to the right $i+1, i+2, \ldots, i+\frac{k}{2}$. The last node has as neighbors to the right nodes $0, 1, \ldots, \frac{k}{2}$, which makes the lattice circular. Then, each link to the right of each node is rewired with probability p to another randomly selected node not belonging to the initial set of neighbors of the current node. Figure 4.1 shows three examples of graphs obtained with the Watts and Strogatz algorithm.

4.4.4. Initial proportion of sharing peers

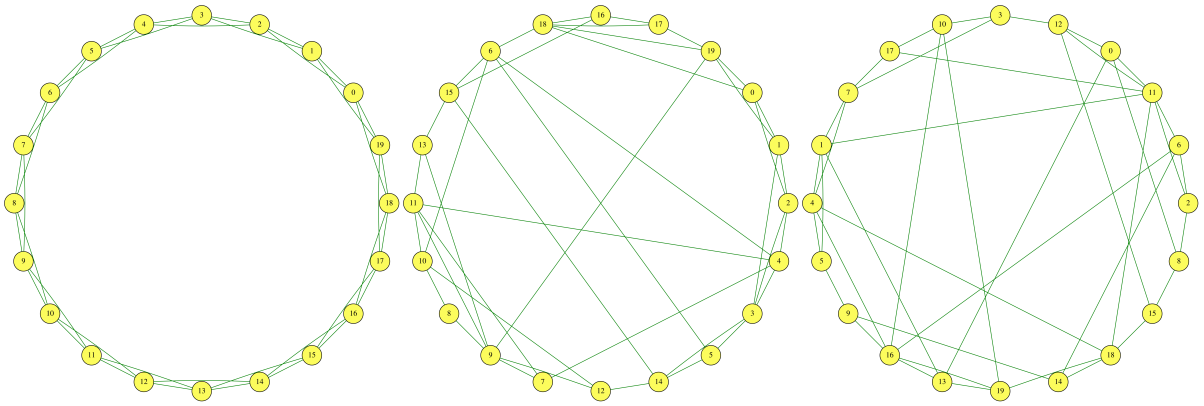
The initial sharing state of the peers is randomly set at the beginning of the simulation. Peers who will have a sharing state of "*true*" are drawn from a uniform random distribution. The proportion is deemed x_{init}.

4.4.5. Update Scheme

All the peers in the simulator are scheduled to take their decision and change their sharing state accordingly synchronously. This means that they are all scheduled to be active at each time step and that they change their decision once every peer has taken its decision. Although the simulator allows to establish other update schemes, for the purpose of simplicity, through this thesis work we considered only the synchronous one.

4.4.6. Summary

In order to facilitate further reading of this chapter, we present a summary of the different concepts of the target system and the associated notation in table 4.1.



(a) (b) (c)

Figure 4.1: Example graphs generated with the algorithm of (Watts and Strogatz, 1998) with $N = 20$ initial nodes and $k = 4$ neighbors per node. (a) $p = 0$ (circular lattice), (b) $p = 0.25$ (small-world graph), (c) $p = 1$ (random graph)

Notation	Meaning
N	Number of peers in the network.
"free-market"	Behavior of the peers.
θ	Generosity of a peer.
d	Depth of neighborhood used to estimate local contribution level.
k	Average number of neighbors per peer.
p	Probability of having a neighbor out of the k original ones.
x	Contribution level
x_{init}	Initial contribution level

Table 4.1: Target system lexical summary.

4.5. Preliminary study on the behavior of the target system

Our objective is to implement our architecture to guide the target system to a specific behavior. To fulfill our objective, we shall established an experimental setup consisting of:

- A target configured to exhibit an undesired behavior
- The control architecture with the objective to guide the target system to a desired behavior.

Since we are interested in implementing the architecture in this particular context, we need to find out what are the initial conditions that eventually lead the system to a majority of free-riding peers. We have conducted a preliminary study for this purpose. The results of this study shall be used to implement the target system, further in the experimental setup.

4.5.1. Analytical point of view

The free-market behavior inspired from (Feldman et al., 2006) allows to rapidly define an analytical model describing the dynamics of the contribution level of the system, in terms of the distribution of the generosity of the peers and the initial contribution level.

Assuming (1) a particular distribution of generosity among the users of the peer-to-peer network (the distribution of θ is uniform between 0 and a maximum value θ_{max}), (2) a synchronous decision of agents, and (3) a complete perception of the network; it is possible to obtain an exact solution to the corresponding analytical model [4]. The basis to obtain the analytical model is to consider that the peers that will share are those that have a generosity greater or equal than the current cost of sharing in the network. Given that the cost of sharing is the reciprocal of the contribution level, and that the generosity is modeled with a probability distribution, the contribution level will be given by: $x = Prob(\theta \geq \frac{1}{x})$.

This analytical solution yields a fixed point dynamics with two attractors: x_1 and 0. The final contribution level of the system x_f with an initial contribution level x_{init} will be given by:

$$x_f = \begin{cases} x_1 & \text{if } x_{init} \geq x_2, \\ 0 & \text{if } x_{init} < x_2. \end{cases}$$

So, if we give to θ_{max} a value of 10, the attractors of the system are $x_1 = .8872$ and $x_2 = 0.1127$. This can be interpreted as: "if the initial contribution level x_{init} is higher or equal than 0.1127, the final contribution level x_f will be of 0.8872 and zero otherwise". The values of x_1 and x_2 for other values of θ_{max} are as follows:

θ_{max}	10	100	1000	1,2,3	4	5	6	7	8	9
x_1	0.88	0.98	0.99	Undef.	0.50	0.72	0.78	0.82	0.85	0.87
x_2	0.11	0.01	0.001	Undef.	0.50	0.27	0.21	0.17	0.14	0.12

Figure 4.2 shows the expected behavior of the contribution level as given by the analytical model of (Feldman et al., 2006) with $\theta_{max} = 10$ and different values of x_{init}. The iterative evaluation has the following form: $x_{iteration+1} = Prob(\theta \geq \frac{1}{x_{iteration}})$, where $x_0 = x_{init}$.

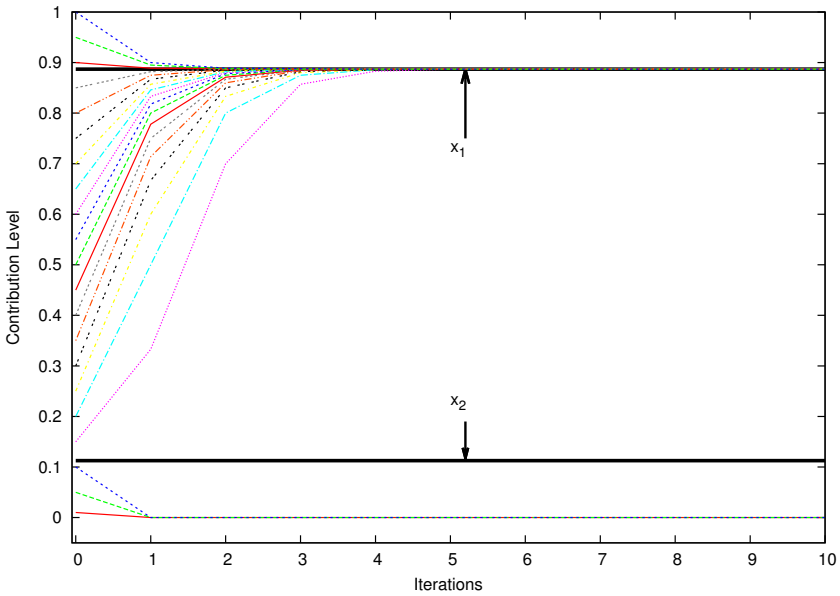


Figure 4.2: Contribution level evolution as given by an iterative evaluation of the analytical model of (Feldman et al., 2006) with $\theta_{max} = 10$ and different values of x_{init}.

[4] The interested reader can consult (Feldman et al., 2006) for all the details of the solution.

4.5.2. Beyond the analytical model

Although the analytical point of view defines the dynamics of the contribution level, it does not include all the parameters of the target system we shall implement for the experimental setup, namely the network topology in which the peers are organized. Moreover, it is not a multi-agent based model.

We conducted a study (Navarrete Gutiérrez et al., 2011, 2010) where we varied the different parameters of the target system including a network topology.

We used different configurations of the network changing the d and p parameters. The explored networks were configured using values for $d = \{1, 2, 3\}$ and for $p = \{0, 0.25, 0.50, 0.75, 1\}$. For each combination of the two parameters, we used a range of different initial contribution level x_{init} from 0 to 1 in steps of 0.01. In this study, we used a value of $N = 1000$ peers in the network. We used a uniform random distribution of the generosity of peers in $U(0, \theta_{max} = 10)$ so that we can easily compare our results to the analytical model. The value of θ_{max} is arbitrary and it intuitively corresponds to saying that a peer with the maximum generosity will share if at least 10% of its neighbors share. In total, the study was executed on $d \times p \times x_{init} = 3 \times 5 \times 100 = 1500$ different configurations. For each configuration we simulated the system 1000 times, over 200 time steps. Figure 4.3 presents the results of the simulations.

4.5.3. Results and conclusions on the preliminary study

If we compare the resulting dynamics of the contribution level of these experiments, with the expected behavior of the analytical model of (Feldman et al., 2006) shown in figure 4.2 we can see on one hand, that our "topological model" reproduces, under certain configurations the same qualitative and quantitative dynamics. On the other hand, we can also see that in some configurations, our model has the same qualitative dynamics, but quantitatively speaking it differs.

4.5.4. Retained Initial conditions

As a conclusion of the previous study, we decided to use a configuration of the target system with $d = 1$ and an initial contribution level of $x_{init} = 0.018$ to test our architecture.

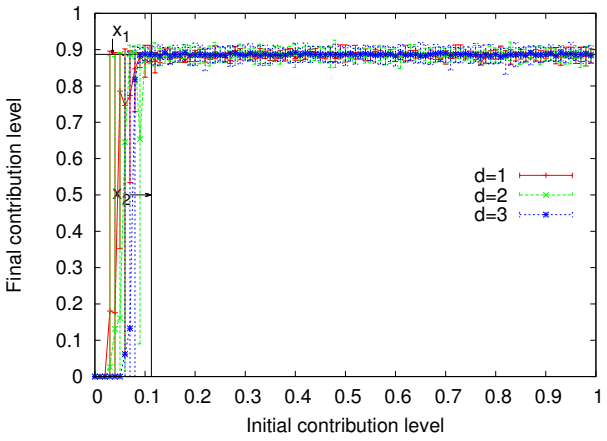
Regarding d, we consider that it is this configuration that presents the most interest because it is the one that indicates that peers have only access to "local" (in the "closest" sense) information to estimate cost of sharing in the whole network.

Regarding x_{init}, we consider that it is this configuration that assures that we have (for most of the families) an evolution of the contribution level that will eventually lead to a system where all peers free-ride.

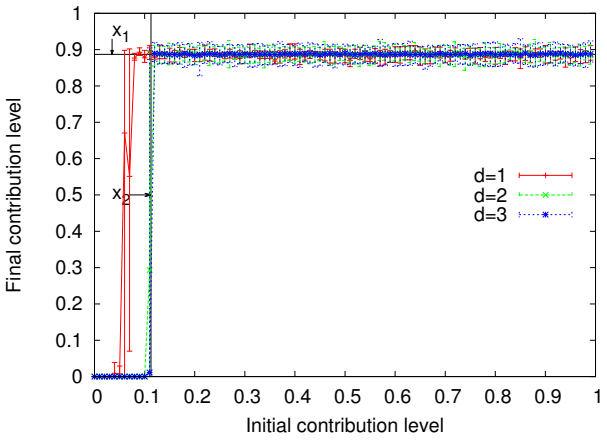
Also, we decided to keep the different values of p used in the study, to test our architecture, to cover a wide family of networks.

Regarding the distribution of the generosity among peers, we kept a uniform distribution between $(0, \theta_{max})$ to be able to compare our results with the analytical model and also because the original analytical model has no known solutions for other distributions. Finally, during the study we conducted more experiments with different values of N, but they were all essentially similar. The values of N (1000 and 10000) allow to keep the simulation time short.

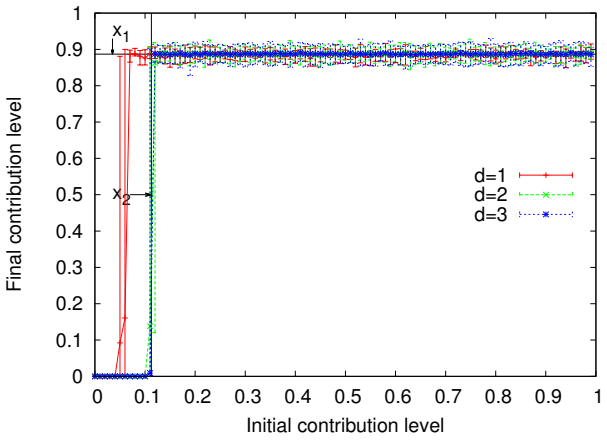
Table 4.2 summarizes the initial conditions of the target system and whether or not these values varied along the experiments.



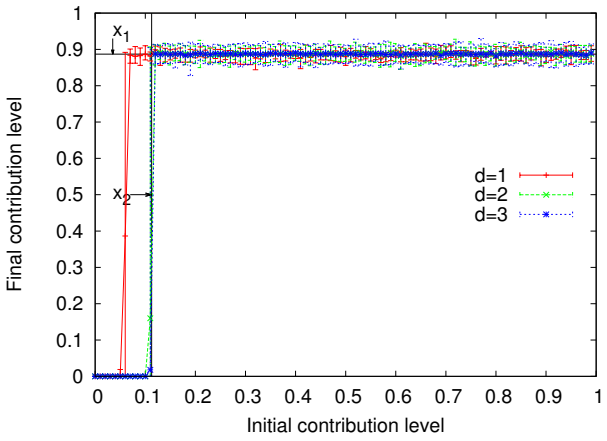
(a) $p = 0$



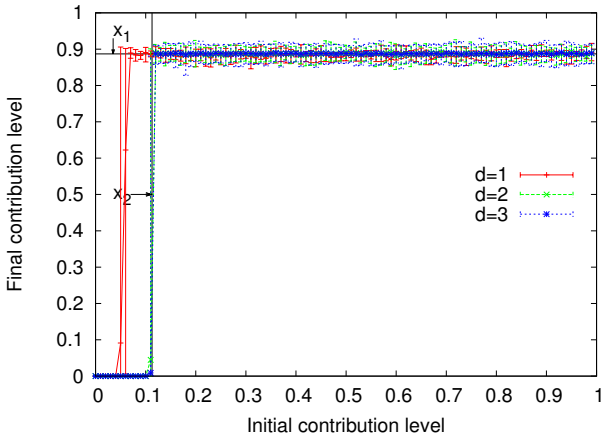
(b) $p = 0.25$



(c) $p = 0.50$



(d) $p = 0.75$



(e) $p = 1$

Figure 4.3: Summary of results of preliminary study on the target. Each figure shows the global dynamics of the network in terms of the "contribution level". Each image depicts the average contribution level found after 200 time steps and after conducting 1000 repetitions.

Parameter	Initial Condition	
	Changes in the experiments	
	Yes	No
N	$1000, 10000$	
$\theta \sim U(0, \theta_{max})$		$\theta_{max} = 10$
p	0, 0.25, 0.50, 0.75, 1	
d		1
k		10
x_{init}		0.018

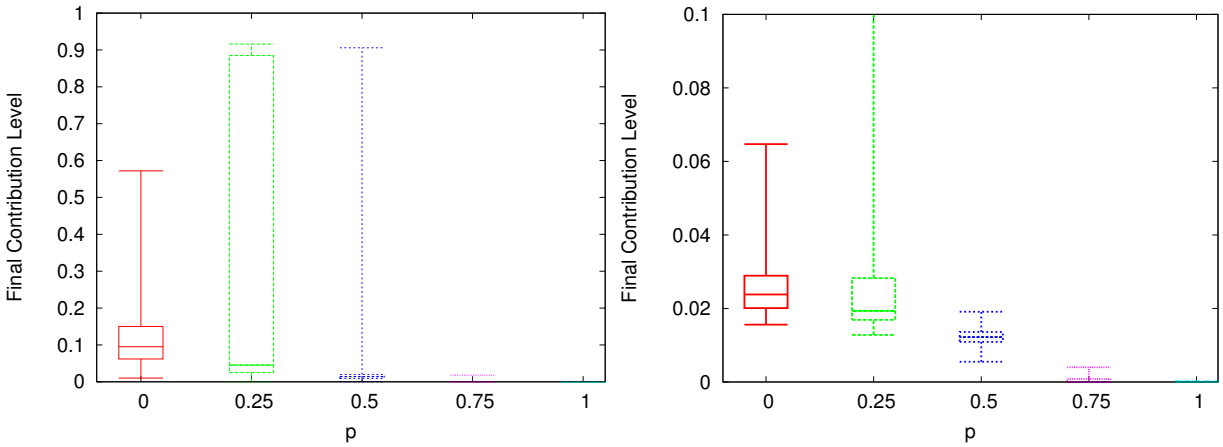
Table 4.2: Initial conditions of the target system used in the experimental setup.

4.5.5. Focus on the behavior of the target system associated to retained initial conditions

We found that in almost all the configurations and given the initial conditions, the final contribution level will be lower than 0.50. The details are as follows.

In the structured and random networks ($p = 0$ and $p = 1$), the observed nominal behavior of the system is to systematically reach a final contribution level value lower than 0.50, except for one single case out of 1000 where $p = 0$ and $N = 1000$. In the $N = 1000$ scenario for the semi-structured or semi-random network $p = 0.50$ the final contribution level will be lower than 0.50 in 95% of the cases.

In the case of the target systems configured as "Small World" networks ($p = 0.25$) for the scenario where $N = 1000$ the final contribution level will be about 53% of the times lower than 0.50. In the other scenario ($N = 10000$) for the same network kind, we see that about 80% of the times the final contribution level will be lower than 0.50. Tables 4.3 and 4.4,and figure 4.4 summarize the nominal behavior of the target system with the retained initial conditions.



(a) $N = 1000$ (b) $N = 10000$

Figure 4.4: Minimum, first quartile, median, 3rd quartile and maximum value of the contribution level at the final step of the nominal behavior of the target system observed in the experiments.

	Percentage of cases where $x_f \geq 0.50$				
	$p = 0$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 1$
N=1000	0.1%	46.8%	4.1%	0	0
N=10000	0	19.5%	0	0	0

Table 4.3: Summary of the nominal behavior of the target system. Percentage of experiments (out of 1000) where the final contribution level reached a value higher or equal to 50% for each of the network families ($p = 0, 0.25, 0.50, 0.75, 1$).

	Mean $x_f \pm$ standard deviation				
	$p = 0$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 1$
N=1000	0.11 ± 0.06	0.42 ± 0.42	0.04 ± 0.17	0.00 ± 0.00	0 ± 0
N=10000	0.02 ± 0.00	0.18 ± 0.34	0.01 ± 0.00	0.00 ± 0.00	0 ± 0

Table 4.4: Summary of the mean contribution level of the nominal behavior of the target system.

4.6. Experimentation platform

As part of our work we developed an experimentation platform because:

- Our proposition includes the equation-free approach and the multi-agent approach. These two approaches are highly experimental.
- We need to be able to reproduce our experiments in a controlled environment.

The platform is made of three main software components: the target system simulator PeerSim, the architecture implementation and a set of scripts to automatize the execution of different scenarios and collect statistical information.

PeerSim is built with the Java programming language and has its own set of libraries. These libraries are sufficient to instantiate the target system we use in the experiments. The libraries provide sufficient elements to implement the peers and their behavior, as well as the topology in which they are organized.

The architecture as well as the multi-agent models were programmed using Java. The topology in the models was built using the "jung"[5] graph libraries. Statistical operations like random number generation or random number distribution sampling were implemented with the colt[6] scientific and technical computing library.

The scripts necessary to automatically execute multiple repetitions of an experiment and collect the results were written in the unix command shell "bash". The data collected from the executions was analyzed with the "R" statistical software (R Core Team, 2012).

A bash script integrating the three elements was in charge of executing the repetitions of an experiment. The tasks that the script was charged to were:

- Change configuration files of PeerSim and the architecture, to match those of the particular scenario being run.
- Change the random seeds used to initialize different elements of the simulator and the architecture. Random seeds were generated and stored in advance, for reproducibility.

[5] `http://jung.sourceforge.net/`
[6] `http://acs.lbl.gov/software/colt/`

- group the results in different files and ran other scripts written in "R" to generate the statistical characterization of the results, once a set of repetitions was finished.

4.7. Experimental Setup

We settled in the introduction of this chapter the need for an experimental setup to illustrate the implementation of the architecture in a concrete case. The setup consists of a target system and the architecture.

Regarding the target system, section 4.3 detailed the specifications of the target system, as well as the initial conditions that we shall use. The initial conditions are summarized in table 4.2.

Regarding the architecture, we shall now specify how it was implemented. Following the scheme used in chapter 3 to define the architecture, we shall first present an overview of the elements of the architecture and then detail the specific inputs and outputs as well as the inner workings, of each block of the architecture.

4.7.1. Control Objective

In qualitative terms, the control objective is to avoid a majority of free-riders in the target system. Quantitatively speaking, the control objective set for the experiments is defined as follows: keep the final contribution level of the target system x_f above 0.50. The control objective in the notation of the architecture is $D = x_f \geq 0.50$

4.7.2. Architecture implementation overview

The architecture blocks are instantiated as follows.

0. **Main block** The main execution flow starts with the *Observe target system* block, then passes control to the *Estimate future state* block. If the results of the latter mean that the future state of the system differs from the control objective, then the execution flow passes to the *Simulate control actions* and *Apply control actions* blocks. If no control actions are necessary, the flow restarts. After the control actions are applied, the execution flow restarts.

1. **Observe Target System.** Obtain a *random* sub-graph G_{est} by sampling q nodes, from T.

2. **Estimate Future State.** Initialize one instance of multi-agent model $M_{topological}$ with G_{est} to predict the future state of the contribution level $r_{M_{topological}}$ for one time horizon, $t_a = 1$. Compare $r_{M_{topological}}$ with the control objective D to decide if any control actions are necessary.

3. **Simulate control actions.** Simulate the application of the available control actions using one instance of $M_{topological}$(the same created in block *Estimate Future State*). Choose the control action a_n that would give the highest contribution level.

4. **Apply Control actions.** *trick n selected peers to make them share*

4.7.3. Architecture implementation details

We present here the specific details of the implementation of each of the architecture blocks. For each of the blocks, we will specify first, what are the inputs and outputs, and after, the internal functioning of the block.

0. Main block

A special condition regarding the availability of control actions was implemented in the main block. This condition accounts for the fact that we assume that we can trick a maximum number of peers in the network and once we have tricked the maximum number of tricks possible, there are no more control actions to test.

The main block of the architecture is represented in algorithm 5.

Algorithm 5: Main block execution flow

input : Random Peer

while *Architecture in execution* **do**
  $G_{est}(P, E_{est}) \leftarrow$ Observation of Target System (RandomPeer);
  $r_{M_{topological}} \leftarrow$ Estimate Future State $(M_{topological}, G_{est}, t_a)$;
  $x_f \leftarrow$ Mean$(r_{M_{topological}})$;
  if $x_f < D_{t_a}$ *and control actions are available* **then**
    $r_{M_{topological-withactions}} \leftarrow$ Simulate Control Actions $(M_{topological-withactions}, G, t_a)$;
    $a \leftarrow$ Select Best Control Action $(r_{M_{topological-withactions}})$;
    Apply Control Actions (a);
    Update Available Control Actions ();
  end
end

1. Observation of target system

- **Input:**
  - Random node to sample from target system.
  - Number of nodes to sample q.
- **Output:**
  - $G_{est}(P, E_{est})$ Graph with the set of sampled peers P and edges E_{est} .
  - θ_n The internal generosity of each peer .
  - $Sharing_n$ The sharing state of each peer .

A peer is randomly selected from the target system. The status (sharing or not), its generosity (θ_n) and the list of neighbors of the peer are recorded. The neighbors of the peer are put in a queue of peers to sample. The next peer from the queue is sampled and so on, until q peers have been sampled. An example of this creation process when we have a sample size of $q = 2$ is illustrated in figure 4.5. As we can see in the figure, the final number of sample peers is not necessarily equivalent to the number of sample nodes. We shall note n the final number of peers sampled from the network. In the presented example, although $q = 2$, the final number of peers in the sampled graph n is 7.

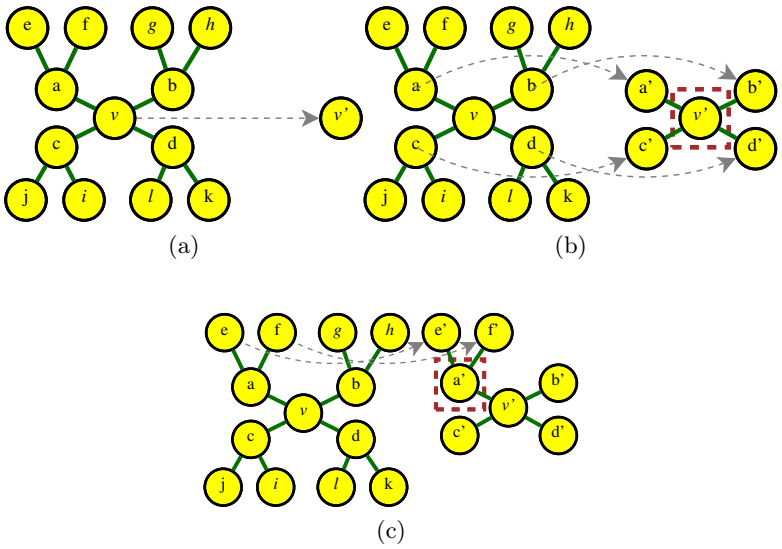


Figure 4.5: *(a)* We first select a random node v from the original graph and add it to our model. *(b)* We add all the neighbors of v in the model. *(c)* We randomly select a neighbor (a') of v in the original graph and add all of its neighbors to the model.

The sharing state of the peers is necessary to estimate the state of the system. The internal generosity and the neighborhood of the sampled nodes are used to initialize the multi-agent model used to predict the evolution of the system and the model used to test the control actions. At the end, the information gathering process yields the following data:

- $P = \{p_1, p_2, \ldots, p_n\}$. The set of peers from the target system sampled.
- $\theta_n = \{\theta_{p_1}, \theta_{p_2}, \ldots, \theta_{p_n}\}$. The generosity of each of the sampled peers.
- $Sharing_n = \{s_{p_1}, s_{p_2}, \ldots, s_{p_n}\}$. The sharing state of each of the sampled peers.
- $G_{est} = (P, E_{est})$. A graph describing the links between the sampled peers.

2. Estimation of future state

- **Input:**
  - $G_{est}(P, E_{est})$ Graph with the set of sampled peers P and edges E_{est}.
  - θ_n The internal generosity of each peer.
  - $Sharing_n$ The sharing state of each peer.
- **Output:**
  - $r_{M_{topological}}$

We have used a multi-agent implementation of the peer behavior previously described. We consider the model to be valid given our previous study on the subject (Navarrete Gutiérrez et al., 2011, 2010). It is suitable to be simulated and the behavior of the agents is exactly the same behavior as that of the entities in the target system.

Multi-agent model $M_{topological}$

In the model, agents represent the peers of the target system. The agents in the model are organized in a network. The model is built based on the information obtained from the sampling process. Each of the n agents in the model represents one of the peers observed during the sampling process. All agents are scheduled to take their decisions at each time step and commit their state change once that all the agents have made their decision. Agents have the same decision function as the peers in the target system, cf. algorithm 3. The whole model is given by: $M_{topological} = \langle Par_{M_{topological}}, \lambda_{M_{topological}} \rangle$. Model $M_{topological}$ has the following set of parameters $Par_{M_{topological}}$.

- **Behavior.** All agents have the free-market behavior defined in algorithm 4.
- **The number of agents in the model.** n
- **The set of the agents in the model.** $\hat{A} = \{a_1, a_2, \ldots, a_n\}$
- **The characteristics of the agents**

  The generosity of each agent $\theta_M = \{\theta_{a_1}, \theta_{a_2}, \ldots, \theta_{a_n}, \}$

  The sharing state $Sharing = \{sharing_{a_1}, sharing_{a_2}, \ldots, sharing_{a_n}, \}$

  The depth of neighborhood used to calculate the contribution level for each agent $\hat{d} = \{d_{a_1}, d_{a_2}, \ldots, d_{a_n}\}$.
- **A graph defining which agents interact with which agents.** $G_M = (A, E_M)$ A graph where each node is an agent from $\hat{A}$ and the edges E represent the links between agents.

Model Initialization

The model $M_{topological}$ is directly initialized with the information received as input for the block as follows.

- The number of agents in the model is the same as the number of peers sampled n.
- Each agent a_n in the model will have the characteristics of its counterpart peer p_n:

  $\theta_{a_n} = \theta_{p_n}$

  $Sharing_{a_n} = Sharing_{p_n}$

  Depth of neighborhood d is assumed to be known for the target system such that $\forall n \; d_{a_n} = d$
- The edges E_M of the graph G_M are identical to the edges of the graph G_{est}, E_{est}.

3. Simulate control actions

- **Input:**
  - $G_{est}(P, E_{est})$ Graph with the set of sampled peers P and edges E_{est}.
  - θ_n The internal generosity of each peer.
  - $Sharing_n$ The sharing state of each peer.

- **Output:**
  - a_n a set of peer ids to trick.

The control action that we will test is to trick a peer to make it share. In fact this action means to modify the perception of the environment of the peer to be controlled, as to make it believe that all its neighbors are sharing. The effects of control actions are simulated using $M_{topological}$, also initialized with the same information and in the same manner as in the *Estimate Future State* block.

We have settled a maximum number of peers to control (cf. table 4.6). The different control actions tested consist of different amounts of peers to be controlled. For example, if at any given moment, no peer has been tricked to share, the first control action to be tested is to trick one peer, the second one to trick two peers, and so on up to the maximum number. Once a control action involving m tricked peers has been executed, the maximum number of peers to be tricked will decrease by m.

In the experiments, we will test two different ways to select the peers to trick: random selection and based on an centrality metric relative to the portion of the network sampled. The intuitive hypothesis that lead us to use importance metrics is that peers with a better centrality metric in the network have more influence in the network and are thus better candidates. The importance metrics used are HITS (Kleinberg, 1999) and betweenness (Steen, 2010).

The criteria used to decide which control action is the best one is the gain in contribution level after one time step.

4. Apply control actions

- **Input:**
  - ids of peers to trick

The resulting control action to be applied is defined by tricking a specific peer from the target system. Applying this control action only requires to identify the peer in the target system and change its perception of the network to make him share. Therefore, the block directly applies the control action.

4.8. Experiments and Results

Up to now, we have defined a target system, and its parameters. We executed a preliminary study on the target system to identify the values of the parameters that make the system evolve to a specific state that we are interested in (majority of peers free-riding). We have also specified the internal functioning as well as the different inputs and outputs of the architecture. Finally, we have also defined an experimental setup: execute the control architecture plugged to control the target system configured to evolve to an undesired state. The objective of the experiments is to avoid the target system from reaching the undesired state.

The different inputs and outputs of the architecture are summarized in table 4.5.

We present first a description of the set of experiments executed, then we provide a synthetic view of the results and finally present a discussion of the results.

Architecture Parameter	Notation	Values within the experiments
Information	$O = \{o_1, o_2, o_3, o_4\}$	$o_1 = \{P$ Peers from the network of $T\}$ $o_2 = \{$Edges linking the peers of $o_1\}$ $o_3 = \{\theta_n$ Generosity of each peer of $o_1\}$ $o_4 = \{Sharing_n$ Sharing state for each peer of $o_1\}$
Models	$\{M_{topological}\}$	Topological Model (See 4.7.2)
Parameters	$\{Par_{M_{topological}}\}$	$p_{1M_{topological}} = \{$Agents have free-market behavior$\}$ $p_{2M_{topological}} = \{n$ Agents in the model$\}$ $p_{3M_{topological}} = \{$Graph made of $V = o_1$ and $E = o_2\}$ $p_{4M_{topological}} = \{$Depth $d = 1\}$ $p_{5M_{topological}} = \{$Update Scheme: Synchronous$\}$
Results	$R = \{r_{M_{topological}}\}$	$r_{M_{topological}} = \{$future contribution level x_f for $t_a\}$
Time horizons	t_a	$t_a = 1$ time step ahead
Control Actions	$\hat{A} = \{a_1, a_2, \ldots, a_{10}\}$	$a_n = \{$Trick n nodes to believe all its neighbors share$\}$
Control Objectives	$D = \{d_{t_a}\}$	$d_{t_a} = x_f \geq 50\%$

Table 4.5: Different inputs and outputs of the architecture for the experiments.

4.8.1. Description of experiments

We have conducted a series of experiments divided into two scenarios. In the first one, the target system is configured to have $N = 1000$ peers and in the second one $N = 10000$.

For all scenarios, the target system is configured with the parameters indicated in table 4.6. We generated 1000 instances of the target system for each family of networks ($p = 0, 0.25, 0.50, 0.75, 1$), for each scenario ($N = 1000, 10000$). Each importance metric to select the peers to trick (random, betweenness and HITS) was tested on every generated instance of the target system.

In our experiments we have measured the final contribution level after 200 time steps of execution of the target system. The control architecture is plugged into the target system from the first time step. For each experiment, we measured the number of times when the final

Parameter	Changes in the experiments	
	Yes	No
q		6
maximum number of peers to trick		10
strategy to choose peers to trick	random, betweenness, hits	

Table 4.6: Values of the parameters of the architecture used in the experiments.

	$x_f \geq 0.50$				
	$p = 0$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 1$
N=1000					
random	0.8%(+0.7%)	95.5%(+48.7%)	68.8%(+64.7%)	13%(+1.3%)	0
betweenness	0.8%(+0.7%)	96.2%(+49.4%)	65.3%(+61.2%)	0.4%(+0.4%)	0
hits	0.9%(+0.9%)	91.7%(+44.9%)	64.1%(+60.0%)	0.4%(+0.4%)	0
N=10000					
random	0	57.8%(+38.3%)	0	0	0
betweenness	0	57.0%(+37.5%)	0	0	0
hits	0	42.9%(+23.4%)	0	0	0

Table 4.7: Summary of results. Percentage of experiments (out of 1000) where the final contribution level reached a higher or equal to 0.50 for each of the network families ($p = 0, 0.25, 0.50, 0.75, 1$). Each cell of the table shows the improvement percentage with regard to the nominal behavior of the target system (c.f. table 4.3). In color ■ the cases considered as "control attained".

contribution level x_f reached a value higher or equal than 0.50.

4.8.2. Synthesis of results

The full results are summarized in tables 4.7 and 4.8. We present next a synthetic view of the results, grouping them by network family and by metric.

By network family

Network families $p = 0$, $p = 0.75$ and $p = 1$. The architecture didn't control the target system.

Small World network family ($p = 0.25$) and semi-structured network family ($p = 0.50$). For both scenarios $N = 1000$ and $N = 10000$, the architecture managed to control the target system for the small world network family. In the case of the semi-structured network family, the architecture was successful only in the $N = 1000$ scenario.

By metric

The architecture controlled the target system three times. In two of those cases ($N = 10000$ with $p = 0.25$ and $N = 1000$ with $p = 0.50$) a random selection of the peers to control was slightly better than using the importance metrics.

4.8.3. Discussion on the results

We discuss here the the results obtained from the experimental setup from two points of view. First, we shall concentrate on the meaning of the results in terms of the efficiency of control and then on the specifics of the control actions considered.

	Mean x_f and standard deviation									
	$p = 0$		$p = 0.25$		$p = 0.50$		$p = 0.75$		$p = 1$	
	$\bar{x}_f$	σ	$\bar{x}_f$	σ	$\bar{x}_f$	σ	$\bar{x}_f$	σ	$\bar{x}_f$	σ
N=1000										
nominal	0.11	0.06	0.42	0.42	0.04	0.17	0.00	0.00	0.0	0.0
random	0.20	0.09	0.84	0.17	0.61	0.39	0.02	0.09	0.01	0.00
betweenness	0.20	0.09	0.85	0.16	0.58	0.40	0.01	0.05	0.01	0.00
hits	0.20	0.09	0.81	0.23	0.57	0.41	0.01	0.04	0.01	0.00
N=10000										
nominal	0.02	0.00	0.18	0.34	0.01	0.00	0.00	0.00	0.00	0.00
random	0.03	0.00	0.52	0.42	0.01	0.00	0.00	0.00	0.00	0.00
betweenness	0.03	0.00	0.51	0.42	0.01	0.00	0.00	0.00	0.00	0.00
hits	0.03	0.00	0.39	0.42	0.01	0.00	0.00	0.00	0.00	0.00

Table 4.8: Summary of experiment results. In each column the mean final value and the standard deviation of experiments (all, including cases where control was not necessary). In color ■ the cases considered as "control attained" and in color ■ the nominal behavior.

Efficiency of results

The control action tested in the experiments can be interpreted as adding a maximum of 1% of sharing peers to the network for the scenario with $N = 1000$ and 0.1% for the scenario with $N = 10000$. To be able to state that the final contribution level is a result of applying our control actions, we also executed experiments to investigate the nominal behavior of the system when $x_{init} = 0.028$ for $N = 1000$ and $x_{init} = 0.019$ for $N = 10000$ (1% and 0.1% more sharing peers, accordingly) as initial condition, without the architecture plugged into the target system. The results of these experiments show that it is the case that the system exhibits systematic drop to zero of the contribution level when $x_{init} = 0.028$ for $N = 1000$ and $x_{init} = 0.019$ when $N = 10000$. This confirms that the behavior exhibited by the target system in the experiments is a result of the influence executed by our control architecture.

In the cases where we can say that the architecture controlled the network free-riding level it managed to do it with few resources. Less than 10% of nodes where sampled to estimate the state of the network in the $N = 1000$ scenario and less than 1% in the $N = 10000$ one. In both scenarios the amount of nodes that were "tricked" to achieve control was 10, which represent 1% of the nodes of the $N = 1000$ scenario and 0.1% for the $N = 10000$ scenario.

The best performance in the experiments was achieved with the small world network family. This is good news given that there is evidence showing that certain peer-to-peer networks have a Small-World structure (Watts, 1999; Jovanović et al., 2001; Buchanan, 2002; Stutzbach et al., 2008; Park et al., 2007; Bassett and Bullmore, 2006; Wang et al., 2009).

If we compare the performance of the different importance metrics used to select the peers to trick, we see that our hypothesis to this regard is not confirmed. That is, randomly selecting the nodes instead of calculating an importance metric gives sometimes better results. Thus the influence of the nodes chosen to be controlled or tricked is accordingly too little to produce a change of state in the network.

The control actions

The control action considered in this implementation is intended to modify the perception of the peers, and not to directly make the choice of sharing or not for them. The specific details of how to control an individual peer to make it share are not presented here. In a real peer-to-peer network, the control action we considered could be implemented through technical means, such as client or protocol control, or through other means such as incitation (Anceaume et al., 2005; Ma et al., 2006; Karakaya et al., 2009). Yet, as such, this control action should be included in the software-clients of the network, which implies a modification of the behavior of the peers.

The objective of the implementation is to allow a first assessment of the feasibility of the architecture rather than to display the most realistic simulation of a peer-to-peer file-sharing-network. The specifics of how to implement the considered control action, in a real network are out of scope of our work.

4.9. Discussion

The behavior of the agents in the multi-agent model used in the implementation exactly corresponds to the behavior of the peers in the target system. Ultimately defining the validity of this particular behavioral model (or any other) of the users behind the software clients connected to a peer-to-peer network is out of the scope of our work. To the best of our knowledge, there are no behavioral models widely accepted in the peer-to-peer community or considered as the most valid.

The chosen scheduling policy (complete synchronicity) of the peers in the simulated target system and the agents in the model is indeed a simplifying hypothesis of any open system. However, our work is initially focused on evaluating the feasibility of the architecture. Further work should allow to explore other scheduling policies.

The equation-free approach includes numerical methods to accelerate the simulation of the microscopic models. In this implementation of the architecture we did not implement any particular method to accelerate the simulation because we did not observe the need to do so during the previous study we conducted and during the building and testing of the experimental platform. We observed in fact that this was not necessary because the simulation time of the models we used was not disproportionate to our simulation resources (in time and memory). The computers used to execute the different experiments have Intel Core Quad CPU processors with a speed of 2.40 GHz and four gigabytes of RAM. Also, this is an hypothesis that we make in the implementation, the cost of simulating the system is not high enough to require the usage of the numerical methods included in the equation-free approach.

The results of the experimental setup of this chapter, show explicitly that there are cases where the architecture fails to attain control of the target system. We assume that this is strongly related to the quality of the answers given by the multi-agent model inside the control loop of the architecture. Our work is focused on evaluating the pertinence of using multi-agent models in a control of complex systems context. Assessing the validity of the model is out of scope. Moreover, providing a model with level four of the modelling and simulation hierarchy of system specification, is a prerequisite to be able to implement the architecture. Put aside the assessment of the validity (at level four of the modelling and simulation hierarchy) of the model, issues directly related to the use of multi-agent model simulation in our architecture show up.

4.10. Conclusions

The objective of this chapter was to demonstrate the feasibility of our proposition through the implementation of an experimental setup. The conclusions that we obtain from the implementation concern two points of view: the concrete application domain and the fulfillment of the objectives of the chapter.

On the application domain

We have conducted a set of experiments within the example implementation demonstrating that our architecture is capable of influencing a target system to show a certain behavior. Specifically, we controlled the contribution level of a peer-to-peer network by avoiding it to show a majority of free-riders.

The results show that the architecture is capable of controlling the target complex system based on very little information gathered from the target system (small sampling size) and with little intervention (small number of tricked nodes).

On the control architecture

The objectives of the chapter were two:

- Illustrate how to implement the architecture.
- Evaluate the feasibility of our proposition through a concrete implementation.

The first objective is satisfied through the implementation itself. We showed that once the hypothesis to implement the architecture (defined in 3.3) are met, it is possible to implement the architecture. Specifically, the implementation had the form of an incremental and refinement development: we proposed first an overview of the blocks, and then specified the inputs, outputs and internal workings.

The second objective can be broken down in two levels: one corresponding to the demonstration of the feasibility and another corresponding how to establish the relationship between the model and the target system come up.

Again, the demonstration of the feasibility is satisfied by the implementation itself.

It is the level corresponding to the relationship between the multi-agent model and the target system that deserves the most attention because *a)* it is at the heart of the architecture and *b)* the objective of this thesis is to evaluate the usage of multi-agent models in a control of complex systems context.

Given that the architecture is meant to control a system, we propose to identify the issues in terms of the performance of the control obtained (or not) by the architecture.

The following is a list of issues that may be at the origin of the performance of the architecture.

Information gathered from the target system might not be good enough. The question here is to know if information of better quality from the target system means better state estimations or model predictions and thus better control performance. Off-line information sources, for instance, could be combined with the information gathered from the target system. This is a problem related to the validity of the model.

Control actions are not the right ones. The question here is to know if under the assumption of perfect state estimations and behavior prediction, the available control actions are being adequately applied.

The metrics used to select the nodes to control have about the same impact in the performance of the architecture than a random selection of the nodes. What is the impact of having more information on the performance of the metrics used ? This is a problem related to the translation of control actions from the model to the target system.

Calibration of model. The model used to estimate the state of the target system and the possible effects of control actions are directly initialized with the information obtained from the observation process. Would treating the information previously yield better results ?

This proof of concept implementation allowed to demonstrate the feasibility of our proposition. We managed to drive a system to a certain behavior with our architecture. However, we saw also that this worked well for some parameter ranges of the target system. The next step to take is to address the questions directly related to the multi-agent paradigm within the architecture to see if the given answers can help to improve the performance and to explicitly address the questions.

Based on the possibilities offered by the architecture, we shall explore in the following chapter the answers given to the questions of validation, calibration and translation of the models.

Chapter 5

Multi-agent simulation questions inside the architecture

Contents

5.1. Introduction 81
5.2. Initialization of models 82
5.2.1. Target systems implementation 82
5.2.2. Nominal Behavior 83
5.2.3. Experimental setup 86
5.2.4. Experiments and Results 88
5.2.5. Discussion 91
5.2.6. Conclusions 91
5.3. Model selection 92
5.3.1. Target system implementation 92
5.3.2. Experimental setup 92
5.3.3. Experiments and Results 93
5.3.4. Discussion 93
5.4. Conclusions 94

5.1. Introduction

This chapter is concerned with the specific questions related to multi-agent model simulation in the context of control of complex systems. In chapter 4 we demonstrated the feasibility of our approach through the illustration of the implementation of our architecture in a concrete case. The purpose of this chapter is to identify what is the influence of the specific issues of multi-agent model simulation in the performance of the architecture.

We are going to identify the questions in the implementation of the architecture, by illustrating the different choices that have to be made when executing the simulation of multi-agent models in the different blocks of the architecture.

We are, once again, going to develop our work in the application context of peer-to-peer networks and the phenomenon of free-riding. We reuse some of the elements of the previous experimental setup: the target system shall be the same (a simulated peer-to-peer network), whereas some of the blocks of the architecture shall be implemented differently.

First, in section 5.2, we are going to center our attention in the specific issue of calibration of a model. Calibrating a model means to find the values for a set of parameters of the model that makes it exhibit a certain behavior. It is an important aspect inherent to modeling because, it allows to assess the validity of the responses given by the model. Specifically, we will concentrate on how to modify the ways to initialize the models so that they yield valid answers.

We shall setup a series of experiments with different initialization methods, namely: direct and indirect initialization. At the end of the experiments we are interested to see if estimating the initial values of the models instead of directly using the observations yields better results.

Second, in section 5.3 we are going to concentrate on the implications of three different issues related to multi-agent model simulation: having multiple models producing multiple future state predictions, changing the time horizons of the simulations and the amount of information gathered from the target system. We shall setup a series of experiments where we are going to use multiple models in the architecture and we are going to vary the amount of information gathered from the target system and the time horizons. At the end of the experiments we are interested to see if: more information gathered from the system yields better results, having longer time horizons has better results and what are the implications of using multiple models.

The importance of this series of experiments is to allow to study what are the implications of changing the different elements we propose: initialization method, model selection, observation, control actions, time horizons.

5.2. Initialization of models

One of the questions inherent to the usage of models, is the calibration of the models. The importance of calibration of a model is that it helps to improve the validity of its responses. Here we illustrate how to use different calibration principles in the context of a peer-to-peer network.

More precisely, as we use multi-agent models, we need to translate observations to the microscopic point of view from partial macroscopic observations.

We are going to consider two possible ways to initialize a model $M_{topological}$, that we shall call direct and indirect. We are interested here in measuring the accuracy of the predictions made by the models, therefore, in this particular series of experiments we shall not implement control actions.

5.2.1. Target systems implementation

We continue to use the simulator of a peer-to-peer network in this series of experiments. This time, we are going to use two target systems, called A and B. The specification of the target systems is the same as that of the previous chapter. Here we present the different values given to their parameters.

Target System A

- $N = 1000$
- Generosity of users follows a random uniform distribution in $(0, \theta_{max} = 10)$.
- Neighborhood depth of 1.
- Graph linking the agents generated with the Watts and Strogatz algorithm with parameters $k = 10$ and $p = 0.25$.

- Initial conditions

  Contribution level: 1% users of the network are randomly selected and their state is set to "sharing". The state of the rest of the users is set to "not sharing"

  Generosity: for all users, taken from a random uniform distribution in $(0, \theta_{max} = 10)$

Target System B

- $N = 5000$
- Generosity uniformly distributed in $(0, \theta_{max} = 6)$
- Neighborhood depth of 1
- Graph linking the agents generated with the Watts and Strogatz algorithm with parameters $k = 10$ and $p = 0.15$
- Initial conditions

  Contribution level: 20% users of the network are randomly selected and their state is set to "sharing". The state of the rest of the users is set to "not sharing"

  Generosity: for all users, taken from a random uniform distribution in $(0, \theta_{max} = 6)$

		Target System A	Target System B
Parameter	N	1000	5000
	θ	$U(0, \theta_{max})$	$U(0, \theta_{max})$
	d	1	1
	G	W&S (p, k, N)	W&S (p, k, N)
Initial conditions	θ_{max}	10	6
	G	$p = 0.25$,$k = 10$,$N = 1000$	$p = 0.15$,$k = 10$,$N = 5000$
	x_{init}	0.01	0.20

Table 5.1: Resume of parameters and initial conditions for the target systems.

5.2.2. Nominal Behavior

Target system A. It exhibits two different behaviors given the same initial conditions. In 71% of cases, we have a network with a majority of free-riders, and in 29% of cases, we have a majority of sharing peers.

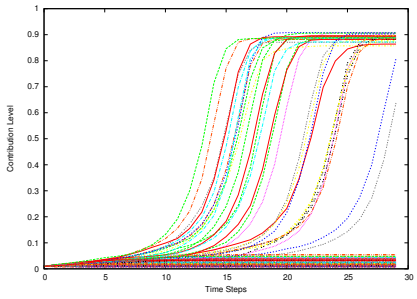
In 71% cases, the contribution level after 30 time steps is comprised between 0.006 and 0.055 with a mean of 0.021 and a standard deviation of 0.011. In the remaining cases, the contribution level is between 0.639 and 0.908 with a mean of 0.875 and a standard deviation of 0.049. Altogether, the final contribution level was between 0.006 and 0.90 with a mean of 0.26941 and a standard deviation of 0.388790.

Target system B. This target system exhibits only one type of behavior: in all cases, there is a majority of free-riders.

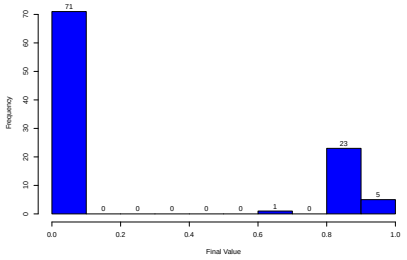
The contribution level after 30 time steps is between 0.119 and 0.157 with a mean value of 0.140 and a standard deviation of 0.007.

We decided to analyze the behavior after 30 time steps because from the preliminary study presented in 4.5, we knew that the contribution level was stable for most of the target system specifications.

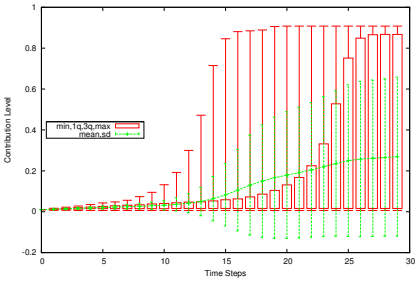
Figure 5.1 shows the nominal behavior of target systems A and B.



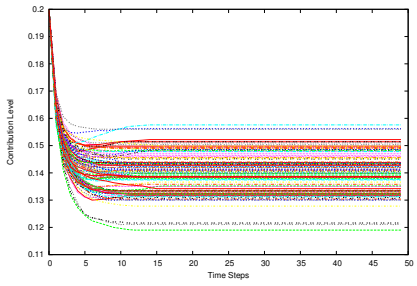
(a) Target System A: all dynamics



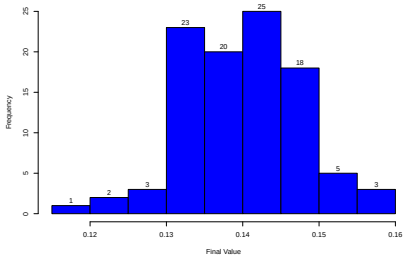
(b) Target System A. x_f at time step $t = 30$



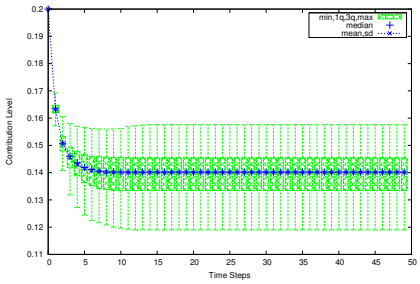
(c) Target System A. Minimum, first quartile, median, third quartile, maximum, mean and standard deviation



(d) Target System B: all dynamics



(e) Target System B. x_f at time step $t = 30$



(f) Target System B. Minimum, first quartile, median, third quartile, maximum, mean and standard deviation

Figure 5.1: Nominal behavior of target systems A (a, b, c) and B (d, e, f). For each time step, the contribution level of the target system.

5.2.3. Experimental setup

In this series of experiments, we are going to execute the architecture and the target system and we shall compare the different results of the simulations of the future state with the real state of the target system. The real state is the one reported by the target system (in this case, the simulator PeerSim).

The comparison takes place between the prediction of the contribution level given by the model and the real contribution level. In this implementation of the architecture we are going to test different amounts of information gathered from the target system used to initialize the models. Also we are going to test different initialization methods for the models.

We shall first present an overview of the changes that were made to the architecture and then the specific details of how this changes were implemented.

Architecture implementation overview

Since in this new series of experiments we are interested in evaluating the performance of the architecture in terms of the different manners to initialize the models, we present only the changes made to the *estimation of future state* block. All the other blocks are implemented in the same manner as in the implementation of chapter 4 (see 4.7.3 for the details).

0. **Main block.** Same execution flow, except the control action simulation block was not implemented.

1. **Observe target system.** Obtain a *random* sub-graph G_{est} by sampling q nodes, from T.

2. **Estimation of future state.** The execution flow of the block is the same as in the previous chapter, what changes is the particular way to initialize the model used to estimate the future state the size of the sample used in the *Observe target system* block (q). The model used is also the one of the previous chapter, that we resume in the following.

   - **Multi-agent model $M_{topological}$.** We continue to use the topological model defined in 4.7.2. A topological model $M_{topological}$ has the following set of parameters $Par_{M_{topological}}$.
     - **Behavior.** All agents have the free-market behavior defined in algorithm 4.
     - **The number of agents in the model.** n
     - **The set of the agents in the model.** $\hat{A} = \{a_1, a_2, \ldots, a_n\}$
     - **The characteristics of the agents**
       The generosity of each agent $\theta_M = \{\theta_{a_1}, \theta_{a_2}, \ldots, \theta_{a_n}, \}$
       The sharing state $Sharing = \{sharing_{a_1}, sharing_{a_2}, \ldots, sharing_{a_n}, \}$
       The depth of neighborhood used to calculate the contribution level for each agent $\hat{d} = \{d_{a_1}, d_{a_2}, \ldots, d_{a_n}\}$.
     - **A graph defining which agents interact with which agents.** $G_M = (A, E_M)$ A graph where each node is an agent from $\hat{A}$ and the edges E represent the links between agents.
     - **A synchronous update scheme.**

   To simulate the model, it is necessary to specify the values of the parameters, namely: $\hat{A}$, θ_M, $Sharing$, $\hat{d}$, G_M and the update scheme. Also, a duration of the simulation t must be supplied.

3. **Simulate control actions.** Not implemented.

4. **Apply control actions.** Not implemented.

Architecture implementation details

We detail here how the different initialization methods we used in this set of experiments.

- **Direct model initialization.** This is the kind of initialization already used in the implementation shown in chapter 4. To directly initialize the values for the parameters of the model, we first gather information from the target system as previously described in 4.7.2. Afterwards, the topological model M is initialized as follows:

  - The number of agents in the model is the same as the number of peers sampled n.
  - Each agent a_n in the model will have the characteristics of its counterpart peer p_n:

    $\theta_{a_n} = \theta_{p_n}$

    $Sharing_{a_n} = Sharing_{p_n}$

    Characteristic d is assumed to be known for the target system (d), such that $\forall n \; d_{a_n} = d$
  - The edges E_M of the graph G_M are identical to the edges of the graph G_{est}, E_{est}.

- **Indirect model initialization.** After the information gathering process takes place, the topological model M is initialized as follows.

  - The number of participants in the network is assumed to be known. The value will be the size of the target system N.
  - It is assumed to be known that the generosity in the target system follows a uniform law, in $(0, \theta_{max})$, with θ_{max} being an integer. The value of θ_{max} is estimated based on the sample: the maximum value of the sample, rounded to the next integer value.
  - It is assumed that the structure of the target system graph was generated with the Watts and Strogatz algorithm. The graph of the model is created using the same algorithm. The parameters of the algorithm are initialized as follows.
    - Parameter p of the algorithm. It is assumed to be known from the target system.
    - Parameter k of the algorithm. It is estimated based on the observation process. It is set as the mean number of neighbors per node in the sampled graph.
    - Parameter n of the algorithm. Set as equal to N (previously defined)

$\lvert(\overline{x_f} - \overline{Prediction})/\overline{x_f}\rvert$				
	Target System A		Target System B	
Sample Size	Direct	Indirect	Direct	Indirect
$q = 1$	1.28 ± 3.81	1.24 ± 3.50	1.59 ± 1.48	1.81 ± 7.69
$q = 10$	1.27 ± 3.18	0.92 ± 1.89	1.26 ± 0.91	1.87 ± 5.24
$q = 20$	1.12 ± 2.60	0.82 ± 1.44	0.88 ± 0.60	1.68 ± 3.60
$q = 100$	0.58 ± 0.53	0.47 ± 0.53	0.35 ± 0.25	1.21 ± 1.52

Table 5.2: Average proportional prediction error at final time step.

5.2.4. Experiments and Results

The experiments were conducted varying parameters q in $1, 10, 20, 100$ for each target system (A and B) and each initialization method.

For each configuration we repeated the experiment 100 times with different random seeds. The experiments were conducted in the following order.

1. Obtain a sub-graph of the target system.

2. At the first time step of the execution, create an instance of the model and initialize it using the obtained sub-graph.

   - For the direct instances experiments the information gathered is directly used to initialized the model.
   - For the instances initialized with macroscopic values, first a maximum θ_{max} is determined from the sample, and then a model with an estimated network size is initialized with values of generosity and initial sharing state derived from the observations.

3. For the rest of the time steps: repeat the information gathering process and update the sharing state of each agent in the model, accordingly to the observations.

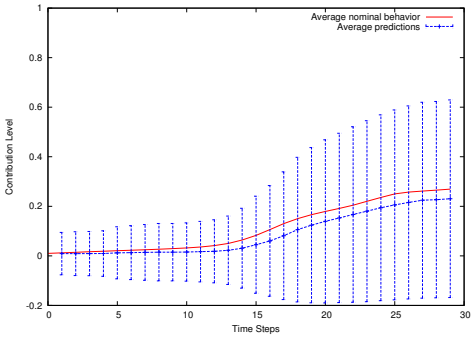
Repeat this 10 times for the same target system.

We do this for each value of sample size q, for each target systems A and B and for each initialization method.

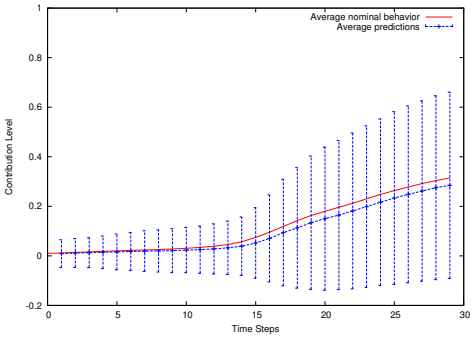
At the end, we have (2 target systems A and B)x(100 times)x(4 sample sizes)x(10 instances)x(2 initialization methods) = 16000 experiments. The whole execution of the experiments lasted about 40 hours.

The results of this experiments show that the predictions obtained through simulation have a high average proportional error but that they are successful in predicting the average dynamics of the target system.

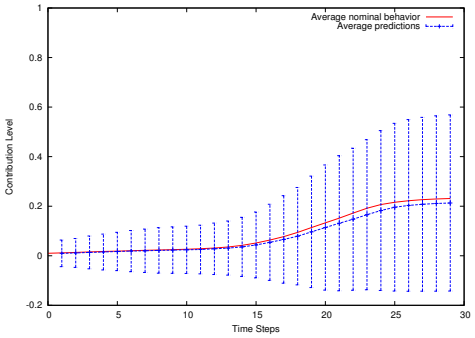
The results of the experiments are summarized in table 5.2. It shows the average proportional error between the predicted values at the final time step, and the average value of the contribution level. Figures 5.2 and 5.3 show the average predicted contribution level as obtained by the simulations compared against the real average behavior of the target systems.



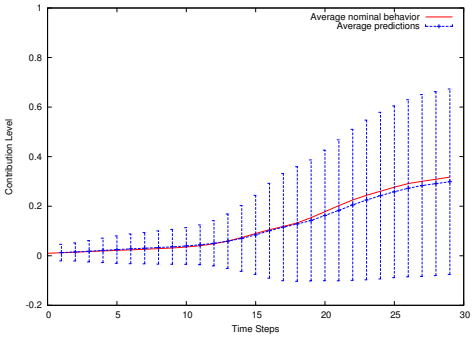
(a) q=1 direct initialization



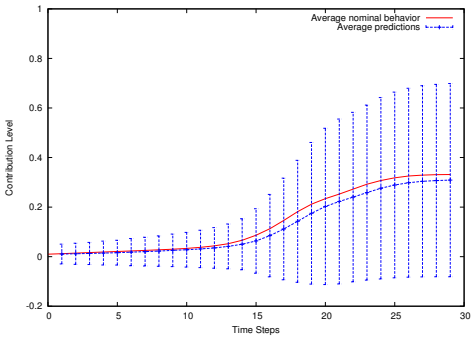
(b) q=1 indirect initialization



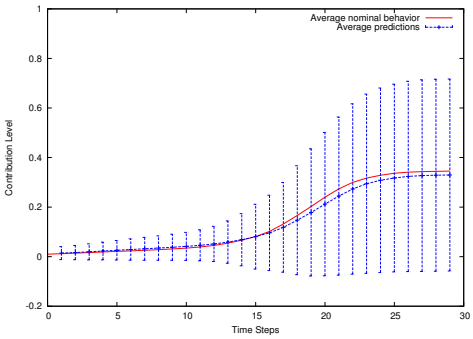
(c) q=10 direct initialization



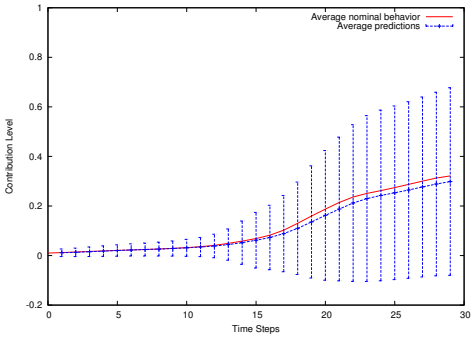
(d) q=10 indirect initialization



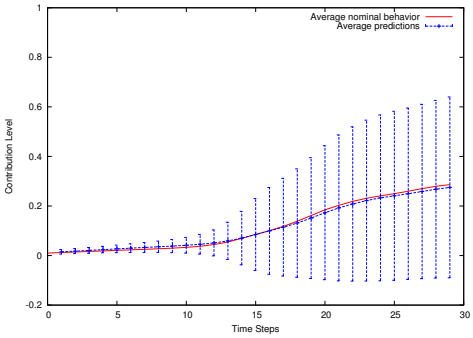
(e) q=20 direct initialization



(f) q=20 indirect initialization

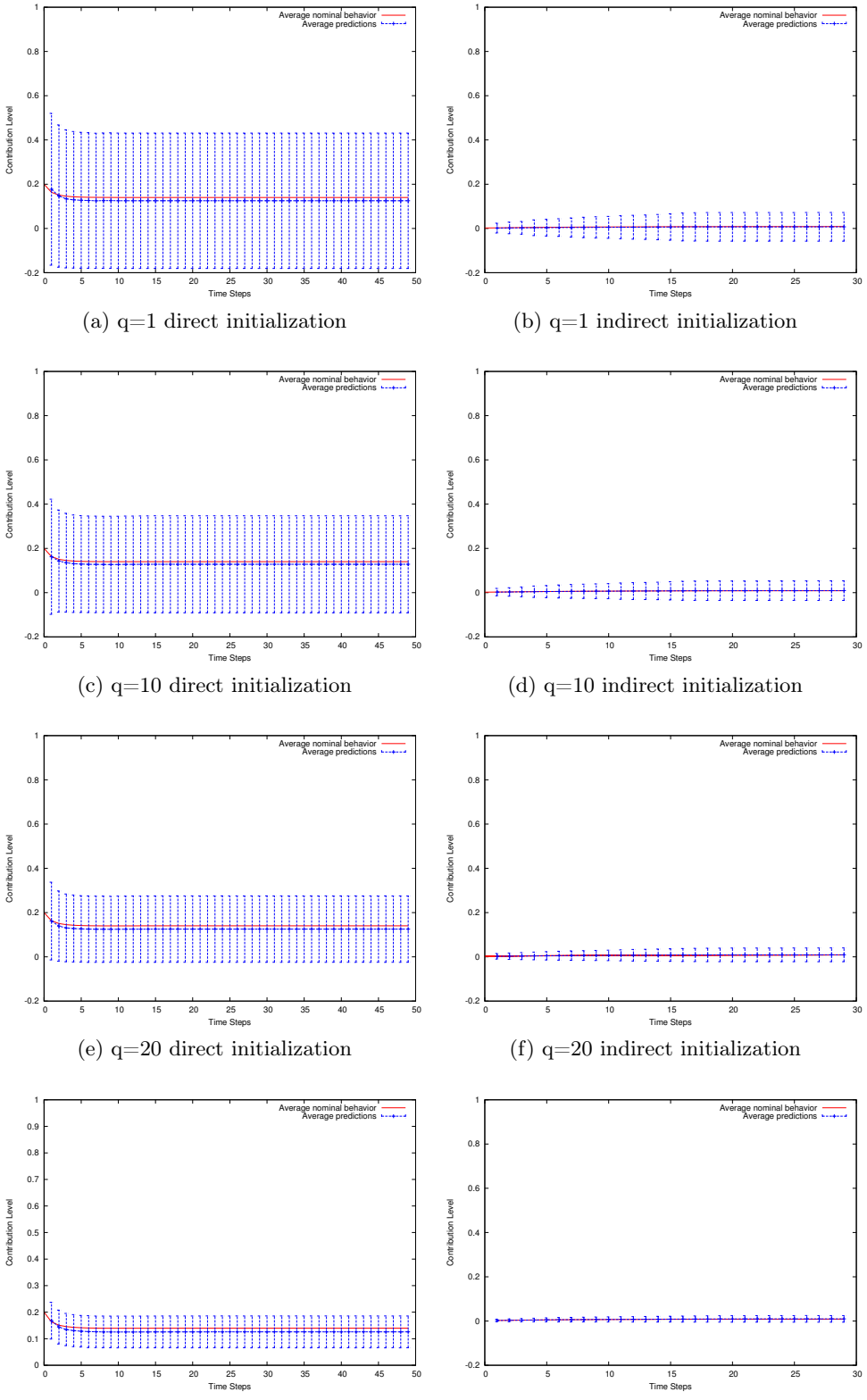


(g) q=100 direct initialization



(h) q=100 indirect initialization

Figure 5.2: Quality of the predictions for Target systems A with direct and indirect initialization. Average nominal behavior and average state estimations plus standard deviation.



(a) q=1 direct initialization

(b) q=1 indirect initialization

(c) q=10 direct initialization

(d) q=10 indirect initialization

(e) q=20 direct initialization

(f) q=20 indirect initialization

(g) q=100 direct initialization

(h) q=100 indirect initialization

Figure 5.3: Quality of the predictions for Target system B with direct and indirect initialization. Average nominal behavior and average state estimations plus standard deviation.

5.2.5. Discussion

In the previous chapter implementation, we directly used the information obtained as the result of the *Observe Target System* block directly to initialize a model. This meant, that the size of the network (in terms of peers) was different from the size of the model (in terms of agents) but that each agent in the model had a direct corresponding peer in the target system.

If we look at table 5.2 the indirect initialization was not significantly more effective than the direct initialization. On the other hand, if we concentrate on the quality of the estimations made with the indirect initialization the case is the other: for the target system B, the indirect initialization is more precise (smaller standard deviation) than the direct initialization.

Also looking at table 5.2, we can deduce that the predictions have a high proportional error. It must be noticed here, that this may be an effect of the different qualitative behaviors of target system A. Target system A has a mean final contribution level of 0.26 but if we look at the histogram of figure 5.1b, we can see that the instances of the system never actually reach that value. In resume, we can consider that seeing such a big proportional error, at least for target system A, in global, is an effect of evaluating the mean error of a mean behavior.

If we take a look at the average predicted behavior compared against the average real behavior of the target systems, depicted in figures 5.2 and 5.3, we see that the predicted dynamics are not contradiction with the observed mean nominal behavior. This allows to consider that although the mean proportional error seems to be high, we have valid average predictions of the dynamics.

5.2.6. Conclusions

We have compared the outcome of multi-agent models initialized in two different ways. One is by directly using the information gathered from the target system. The other is by deducing a macroscopic level value of the parameters of the models and then initialize the models with the values obtained in this way.

If we had access to the global full behavior of the system, for example in an off-line situation, we could compare the outcome to the real value. We analyzed the outcome of our simulations in this way. Results show that the bigger the sample the smaller the variance in the predictions.

However, we can consider that full access to the global behavior is impossible because of legal matters, because it is technically impossible to save all the required information, or because it is simply not available, etc.

If multi-agent models are to be used in the architecture as we propose, they could be used as off-line indications of how to initialize the model. The results of these experiments show that for the different sets of experiments, a threshold of the size of the sample might be determined off-line and then used on-line when possible.

If we had implemented the architecture including control actions, using indirect initialization, we would have faced an additional decision regarding the application of the control actions.

Let us suppose that the model selected used in the simulation of control actions is the same one used in the estimation of future state (as was the case in the experiments of the previous chapter). Let us also consider that the initialization of the model in the *simulation of control actions* block is also "indirect".

Control actions selected to be applied will still have the form "trick the peers corresponding to agents $\{a_x, a_y, \ldots, a_z\}$" but since these agents do not have a *direct* corresponding peer in the target system, a special translation mechanism should be created.

We can imagine, for instance, that this translation could be done by identifying the characteristics of agents $\{a_x, a_y, \ldots, a_z\}$ and finding the peers in the target system with matching

characteristics. Possible characteristics could be: the internal generosity, the evolution of the sharing state, or its importance in the network, relative to the graph (such as HITS or betweenness, as we did in chapter 4). A possible way to avoid the implementation of a translation mechanism for control actions, would be to use indirect initialization for the *future state estimation* block but keeping a direct initialization for the models in the control action simulation block.

This would only change the moment when the translation should be made. In this case we would still need translation, but from the results of the future state estimation to the values used to initialize the models in the *control action simulation* block.

5.3. Model selection

In multi-agent model simulation, the usual course of action is to execute multiple simulations for identical initial conditions, and then to statistically analyze the results. In the architecture, multi-agent simulations are used in the *Estimate Future state* and *Simulate control actions* blocks. Up to now, we have implemented this blocks with one model running at each time. Let us focus now on using several models at the same time. Having multiple models implies that multiple future states will be predicted, and thus a criteria to select the model and the result to use is necessary. In this series of experiments we illustrate one such criteria.

5.3.1. Target system implementation

We shall keep target system A previously described for this series of experiments because we saw that the different dynamics it presents are more interesting for the control problem.

5.3.2. Experimental setup

Objective

In this series of experiments we will again implement the architecture with a control objective $D = x_f \geq 0.50$.

Architecture implementation

For simplicity, we shall focus on using multiple instances of $M_{topological}$ at the same time for the *Estimate Future State* block. What will change from previous implementations is that we shall directly initialize ten instances of the topological model at the same time at the first time step. The specification of the blocks of the architecture of this set of experiments is the same as the one of the previous implementation (see 5.2.3), thus we directly present the details of the implementation.

1. **Observe Target system.** We varied in this implementation the size of the sample used to observe the target system. We used $q = \{1, 5, 20, 50, 100\}$.

2. **Estimate Future State Block.** We decided to implement the criteria to select the result to be passed to the main block of the architecture ($r_{M_{topological}}$) in terms of the error between the predictions of each model and observations. Specifically, we compare at each time step the previous prediction of the contribution level with the observed contribution level at the current time step. When we say observed contribution level here, we mean the proportion

$x_f \geq 0.50$						
t_a	*max.* peers to trick	q				
		1	5	20	50	100
1	10	99%	95%	94%	89%	64%
15	10	85%	86%	84%	75%	100%
15	30	N.A.	100%	100%	100%	97%

Table 5.3: Control results for target system A with direct initialization. Percentage of experiments where the final contribution level was higher or equal to 0.50 for each of the sample sizes ($q = 1, 5, 20, 50, 100$). *N.A.:* Not tested because with a sample size of $q = 1$, we assumed that we could not trick more peers than sampled.

of peers sharing in the sampled sub-graph G_{est}. The importance of the criteria is that the model with the least error is the one to be used in the *simulate control actions* block.

A variation in regard to the previous implementations of the architecture, concerning the time horizons used in the model simulation was executed in this block.

3. **Simulate control actions.** Same control actions as those used in the previous chapter: trick n peers to make them believe that all its neighbors share thus driving it to share. The same criteria to select the control actions is used: the control action giving the highest increase in the contribution level is selected. The change in regard to the previous implementation of this block concerns the number of model instances used inside the block. The model used in this block is the same as the one of the *estimate future state* block. In the *estimate future state* block we had multiple instances of the model but one is selected as being the most accurate. The model instance selected as the most accurate in the *estimate future state* block is the one used in this block.

5.3.3. Experiments and Results

The experiments were conducted varying parameters of time horizon t_a, the maximum number of nodes to trick *maxspins* and the sample size q. The different time horizons used were $t_a = \{1, 15\}$. At every experiment we used 10 instances of the $M_{topological}$. The number of nodes to trick was 10 for values $t_a = \{1, 15\}$ and 30 for $t_a = \{15\}$. One configuration of this implementation consisted of: $|t_a| \times 10$ instances of $M_{topological} \times |\text{max. peers to trick}| \times |q|$. In sum there were $3 \times 5 = 15$ different configurations of the architecture tested for target system A. Each configuration was repeated 100 times.

The results are summarized in table 5.3. From the results, we can see that implementing multiple models in the *estimate future state* yields better results, compared to those of the previous chapter. In the implementation with only one model of the previous chapter, for network family $p = 0.25$ we managed to augment, in average, 47% the cases where $x_f \geq 0.50$ (see table 4.7). In this implementation with ten models, we see an average augmentation of almost 70% (target system A stabilizes at $x_f < 0.50$ in 29% of cases, see 5.2.2).

5.3.4. Discussion

The architecture is based on multi-agent model simulation. This means, that given the experimental nature of multi-agent simulation, at some point the architecture will have several model instances running at the same time. Whether we have multiple instances of a single

multi-agent model or different multi-agent models, a decision concerning the selection of the model should be made. In this series of experiments we exemplify the decision by implementing as criteria the quality of the predictions of the model. In this particular implementation the variable to control is global, and thus, we selected a criteria that compares the global outcome of the simulations. Nonetheless, we can imagine criteria related to local characteristics of the system. For example, the amount of correct sharing state (of each peer) predictions of each model could be used as the criteria. This would imply of course, that a correspondence between the agents and the peers in the network exists (whether direct or indirect).

Now, the previous criteria are discrete in the sense that we are interested in selecting one model *and* its results. But we can imagine also that instead of limiting to one result, we could be interested in using all the results by aggregating them. In the case of a global variable like the one that we are interested in this implementation, we can imagine aggregating in a mean value the results of all the instances we simulated, or taking a median value from the observed results. But, once again, we could imagine that variable that we are interested in controlling is local. In this case, we would also have to define a way to compare the local characteristics in the results of each model to the other models.

5.4. Conclusions

We saw through the different experimental setups presented in this chapter, that the modular design of the architecture, allows to incrementally improve a specific implementation.

More importantly, we saw that a trade-off should be found each time a decision must be made. For example, when a simulation block is instantiated with macroscopic values, instead of directly, the quality of the predictions may be improved. However, this would force to define a translation mechanism to make correspondences between the agents in the model and the entities in the target system.

Another trade-off should be made when using multiple models at the same time in the simulation blocks. Although this ensures that the sensitivity to initial conditions is taken into consideration, having multiple models forces to define selection criteria to choose the model and result to be effectively used to compare with the control objective for example.

We cannot give general conclusions to answer the specific multi-agent questions inside the architecture addressed in this chapter because it is dependent on the application. However, applying the architecture in other experimental setups can lead to gain knowledge on how to address these questions from a general point of view and the modular nature of our architecture offers the coherent framework to experimentally investigate them.

Chapter 6

Conclusion

In this thesis work, we focused on the challenge of driving a complex system to exhibit a certain behavior. More precisely, we studied the pertinence of using multi-agent model simulation in the context of control of complex systems.

We proposed a coherent architecture that integrates multi-agent simulation in a control loop using the equation-free approach. This architecture has the form of a generic pattern where several multi-agent simulations can be used to estimate the future state of the system to control, as well as, the effects of local control actions.

Our work is focused on the characteristics of complex systems rather than on the definition of a complex system. From this point of view, we expect our work to be applicable to systems sharing characteristics with those of complex systems may they be natural or artificial, man-made or man-centered.

This architecture was implemented in an experimental platform and applied to the free-riding problem in peer-to-peer file sharing networks. This implementation illustrated how each component of the architecture can be instantiated. We have demonstrated that the architecture can attain control of a system under conditions where the expected behavior cannot be reached. The problem we used, even if it was simplified, is a prototypical example of a family of problems with fixed points dynamics and emergence. The problem prototype can be related to the tragedy of the commons or to boolean networks, for example.

Introducing multi-agent simulation in our proposition implies that issues concerning how to establish the relationship between the model and the target system come up. Namely: the validity and calibration of models, and the translation of entities from the target system to the elements of the model. We showed that the modular design of the architecture is convenient to investigate such issues and to assess the decision made. This modular design also enables to incrementally improve a specific implementation.

The generic and modular design of the architecture enabled us to explore in the short and mid term aspects that we did not investigate:

- The application example we used can be further investigated along different dimensions to better assess the architecture: open target system, heterogeneous behaviors of peers, noise on the observations, etc. We can also envisage studying other application domains.
- In control theory, the family of PID controllers is the most widely used. In the implementations of the architecture, we had an "on-off" controller. We consider worthy to investigate how to include the notions of proportionality and error correction of PID controllers into our architecture.

- Another aspect concerns the type of control actions. Instead of tampering with the entities already present in the target system, exploit the open nature of the system and add lure entities as control action. The importance of such control actions is related to the implications they have on the implementation of the architecture. Adding lure entities means that decisions shall be taken regarding how to insert them in the target system (where in the network, for example). Thus, a way to identify the correspondence of the lures in the simulation to their place in the target system must exist.

- The different operating regimes of the target system are another aspect. Peer-to-peer networks, are known to have different operating regimes in terms of day time (and time-zone). Such a characterization of the system could lead to an implementation where one model would be used for the slow regime and one for the fast regime. The kind of regimes are already a hint on how to select the model to use: for fast regimes of the target system (implying fast state changes) one could be compelled to use models that keep the pace with the target system (give fast but inaccurate results) and for slow regimes, one could use models that are not as fast as the target system (giving very accurate results but taking significant time to do it).

- In the implementations, we always used the information coming from the target system to (directly or indirectly) initialize the models. However, we can imagine that another source of information for the initialization of the models could be the existing execution traces of the system. The analysis of the traces, could be used for example, to calibrate the model, or to identify different execution regimes of the system.

- Also we consider worthy to study different ways to make evolve the models used in the architecture, such as learning techniques, or genetic algorithms for example.

As long term perspectives, the experience gained from investigating the above mentioned aspects, can be used to elaborate a systematic approach to apply our architecture on a given target system.

In our work, we have always used an instance of the architecture to control a target system. We can imagine, nonetheless, to use multiple instances and make them cooperate. This brings new questions such as, how to coordinate the instances, how to solve the possible mutual influences of the instances, etc.

Bibliography

Adar, E. and Huberman, B. A. (2000). Free riding on gnutella. *First Monday*, 5(10). [online].

Aidouni, F., Latapy, M., and Magnien, C. (2009). Ten weeks in the life of an eDonkey server. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–5. IEEE.

Amaral, L. and Ottino, J. (2004). Complex networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 38(2):147–162. 10.1140/epjb/e2004-00110-5.

Amblard, F. and Phan, D. (2006). *Modélisation et simulation multi-agents: Applications pour les Sciences de l'Homme et de la Société*. Hermès science publications.

Anceaume, E., Gradinariu, M., and Ravoaja, A. (2005). Incentives for P2P Fair Resource Sharing. In *Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing*, pages 253–260. IEEE Computer Society Washington, DC, USA.

Anderson, P. W., Arrow, K., and Pines, D. (1988). *The Economy as an Evolving Complex System*. Redwood City, Addison-Wesley Co.

Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371.

Artus, P. and Lecointe, F. (1991). Crise financiére et crise de l'endettement privé aux Etats-Unis. *Revue française d'économie*, 6(1):37–85.

Astrom, K. and Murray, R. (2008). *Feedback systems: an introduction for scientists and engineers*. Princeton Univ Pr.

Bak, P. and Bak, P. (1996). *How nature works: the science of self-organized criticality*, volume 212. Copernicus New York.

Ball, P. (2004). *Critical mass: How one thing leads to another*. Farrar Straus & Giroux.

Bar-Yam, Y. (1997). *Dynamics of Complex Systems*. Studies in Nonlinearity. Westview Press.

Bar-Yam, Y. (2003). When systems engineering fails-toward complex systems engineering. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 2, pages 2021–2028. IEEE.

Bassett, D. and Bullmore, E. (2006). Small-world brain networks. *The neuroscientist*, 12(6):512–523.

Boccara, N. (2004). *Modeling complex systems*. Springer Verlag.

Bolker, B. and Grenfell, B. (1993). Chaos and biological complexity in measles dynamics. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 251(1330):75–81.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7280.

Brückner, S. (2000). *Return from the Ant*. PhD thesis, Humboldt-Univ.

Brückner, S., Wyns, J., Peeters, P., and Kollingbaum, M. (1998). Designing agents for manufacturing control. In *Proceedings of the 2nd AI & Manufacturing Research Planning Workshop*, pages 40–46.

Brueckner, S. A. (2000). *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. PhD thesis, Departement of Computer Science.

Brueckner, S. A., Belding, T. C., Bisson, R., Downs, E., and Parunak, H. V. D. (2009). Swarming Polyagents Executing Hierarchical Task Networks. In *SASO*, pages 51–60. IEEE Computer Society.

Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Müller, H., Pezzè, M., and Shaw, M. (2009). Engineering self-adaptive systems through feedback loops. In Cheng, B. H., Lemos, R., Giese, H., Inverardi, P., and Magee, J., editors, *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 48–70. Springer.

Buchanan, M. (2002). *Small World: Uncovering Nature's Hidden Networks*. Weidenfeld & Nicolson.

Buchli, J. and Santini, C. (2005). Complexity engineering, harnessing emergent phenomena as opportunities for engineering. *Reports of the Santa Fe Institute's Complex Systems Summer School*.

C3 (2012). Centro de Ciencias de la Complejidad - UNAM. `http://c3.fisica.unam.mx/`.

Cakar, E., Hähner, J., and Müller-Schloer, C. (2008). Investigation of generic observer/controller architectures in a traffic scenario. *INFORMATIK*, pages 733–738.

Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulez, G., and Bonabeau, E. (2001). *Self-organization in biological systems*. Princeton University Press.

Campagne, J.-C. (2005). *Systèmes multi-agents et morphologie*. PhD thesis, Université Pierre et Marie Curie.

Campagne, J.-C., Cardon, A., Collomb, E., and Nishida, T. (2005). Massive Multi-agent systems control. In Hinchey, M., Rash, J., Truszkowski, W., and Rouff, C., editors, *Formal Approaches to Agent-Based Systems*, volume 3228 of *Lecture Notes in Computer Science*, pages 275–280. Springer Berlin / Heidelberg. 10.1007/978-3-540-30960-4_20.

Camus, M. and Cardon, A. (2006). Dynamic programming for robot control in real-time: towards a morphology programming. In *The 2006 International Conference on Artificial Intelligence*.

Castellani, B. and Hafferty, F. (2009). *Sociology and complexity science: a new field of inquiry*. Springer Verlag.

Chavalarias, D., Bourgine, P., Perrier, E., Amblard, F., Arlabosse, F., Auger, P., Baillon, J.-B., Barreteau, O., Baudot, P., Bouchaud, E., Ben Amor, S., Berry, H., Bertelle, C., Berthod, M., Beslon, G., Biroli, G., Bonamy, D., Bourcier, D., Brodu, N., Bui, M., Burnod, Y., Chapron, B., Christophe, C., Clément, B., Coatrieux, J.-L., Cointet, J.-P., Dagrain, V., Dauchot, K., Dauchot, O., Daviaud, F., De Monte, S., Deffuant, G., Degond, P., Delahaye, J.-P., Doursat, R., D'Ovidio, F., Dubois, A. M., Dubruelle, B., Dutreix, M., Faivre, R., Farge, E., Flandrin, P., Franceschelli, S., Gaucherel, C., Gaudin, J.-P., Ghil, M., Giavitto, J.-L., Ginelli, F., Ginot, V., Houllier, F., Hubert, B., Jensen, P., Jullien, L., Kapoula, Z., Krob, D., Ladieu, F., Lang, G., Lavelle, C., Le Bivic, A., Leca, J.-P., Lecerf, C., Legrain, P., L'Hôte, D., Loireau, M., Mangin, J.-F., Monga, O., Morvan, M., Muller, J.-P., Negrutiu, I., Peyreiras, N., Pumain, D., Radulescu, O., Sallantin, J., Sanchis, E., Schertzer, D., Schoenauer, M., Sebag, M., Simonet, E., Six, A., Tarissan, F., and Vincent, P. (2009). French Roadmap for complex Systems 2008-2009. `http://hal.archives-ouvertes.fr/hal-00392486`.

Ciarletta, L. (2011). Co-simulation and Multi-models for Pervasive Computing as a Complex System. In *HCI (16)*, pages 197–206.

Cohen, B. (2003). Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72.

Conte, R., Hegselmann, R., and Terna, P. (1997). *Simulating social phenomena*. Springer.

CSIRO (2012). Centre for Complex Systems Science. `http://www.csiro.au/Organisation-Structure/Divisions/Marine--Atmospheric-Research/Complex-systems-science-2.aspx`.

CSS (2012a). Complex Systems Registry. `http://main.csregistry.org`.

CSS (2012b). Complex Systems Society. `http://www.cssociety.eu/`.

Dainotti, A., Squarcella, C., Aben, E., Claffy, K. C., Chiesa, M., Russo, M., and Pescapé, A. (2011). Analysis of country-wide internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pages 1–18, New York, NY, USA. ACM.

De Swert, K., Valckenaers, P., Saint German, B., Verstraete, P., Hadeli, and Van Brussel, H. (2006). Coordination and control for railroad networks inspired by manufacturing control. In *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, pages 201–206.

Demazeau, Y. (1995). From Interactions To Collective Behaviour In Agent-Based Systems. In *In: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo*, pages 117–132.

Deneubourg, J.-L., Aron, S., and Goss, S. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–169.

Di Marzo Serugendo, G., Gleizes, M.-P., and Karageorgos, A. (2005). Self-organization in multi-agent systems. *Knowledge Engineering Review*, 20:165–189.

Doursat, R. and Ulieru, M. (2008). Emergent engineering for the management of complex situations. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, page 14. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Edmonds, B. (1999). What is Complexity? - the philosophy of complexity per se with application to some examples in evolution. In Heylighen, F., Bollen, J., and Riegle, A., editors, *The evolution of complexity*, volume 8 of *Einstein meets Magritte*. Kluwer, Dordrecht.

Edmonds, B. (2005). Using the Experimental Method to Produce Reliable Self-Organised Systems. In Brueckner, S. A., Di Marzo Serugendo, G., Karageorgos, A., and Nagpal, R., editors, *Engineering Self-Organising Systems*, volume 3464 of *Lecture Notes in Computer Science*, pages 319–330. Springer Berlin / Heidelberg. 10.1007/11494676_6.

Edmonds, B. and Bryson, J. (2004). The Insufficiency of Formal Design Methods The Necessity of an Experimental Approach-for the Understanding and Control of Complex MAS. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 938–945. IEEE Computer Society.

Erban, R., Kevrekidis, I. G., and Othmer, H. G. (2006). An equation-free computational approach for extracting population-level behavior from individual-based models of biological dispersal. *Physica D: Nonlinear Phenomena*, 215(1):1–24.

Fates, N. (2012). FiatLux Simulator. `http://fiatlux.loria.fr/`.

Fatès, N. and Chevrier, V. (2010). How important are updating schemes in multi-agent systems? An illustration on a multi-turmite model. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 533–540. International Foundation for Autonomous Agents and Multiagent Systems.

Feldman, M., Papadimitriou, C. H., Chuang, J., and Stoica, I. (2006). Free-riding and whitewashing in peer-to-peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1010–1019.

Fender, I., Hayo, B., and Neuenkirch, M. (2012). Daily pricing of emerging market sovereign CDS before and during the global financial crisis. *Journal of Banking and Finance*.

Ferber, J. (1995). *Les systèmes multi-agents: vers une intelligence collective*. InterEditions Paris.

Fromm, J. (2004). *The emergence of complexity*. Kassel university press.

Galland, S., Gaud, N., Demange, J., and Koukam, A. (2009). Environment Model for Multiagent-Based Simulation of 3D Urban Systems. In *the 7th European Workshop on Multi-Agent Systems (EUMAS09)*.

Gaud, N., Galland, S., Gechter, F., Hilaire, V., and Koukam, A. (2008). Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, 16(10):1659–1676.

Ge, Z., Figueiredo, D., Jaiswal, S., Kurose, J., and Towsley, D. (2003). Modeling peer-peer file sharing systems. In *IEEE INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*.

Gear, C., Kevrekidis, I., and Theodoropoulos, C. (2002). Coarse integration/bifurcation analysis via microscopic simulators: micro-Galerkin methods. *Computers & chemical engineering*, 26(7-8):941–963.

Gershenson, C. (2007). *Design and control of self-organizing systems*. PhD thesis, Vrije Universiteit Brussels.

Gilbert, N. (2008). *Agent-based models* . Quantitative applications in the social sciences. Sage Publications.

Glance, N. and Huberman, B. (1994). The dynamics of social dilemmas. *Scientific American*, 270(3):76–81.

Grimm, V. and Railsback, S. (2005). *Individual-based modeling and ecology*. Princeton Univ Pr.

Grobbelaar, S. and Ulieru, M. (2007). Complex networks as control paradigm for complex systems. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 4069–4074.

Gruver, W., Kotak, D., van Leeuwen, E., and Norrie, D. (2003). Holonic Manufacturing Systems: Phase II. In Marík, V., McFarlane, D., and Valckenaers, P., editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744 of *Lecture Notes in Computer Science*, pages 1083–1083. Springer Berlin / Heidelberg. 10.1007/978-3-540-45185-3_1.

Hanson, J. E. (2009). Cellular Automata, Emergent Phenomena in. In Meyers, R. A., editor, *Encyclopedia of Complexity and Systems Science*, pages 768–778. Springer New York. 10.1007/978-0-387-30440-3_51.

Hardin, G. (1968). The Tragedy of the Commons. *Science*, 162(3859):1243–1248.

Hardin, R. (Fall 2008). The Free Rider Problem. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab Center for the Study of Language and Information Stanford University Stanford, CA 94305-4115.

Heckmann, O., Bock, A., Mauthe, A., and Steinmetz, R. (2004). The eDonkey file-sharing network. In *GI Jahrestagung (2)*, volume 51 of *LNI*, pages 224–228. GI.

Helbing, D. (2007). *Managing Complexity: Insights, Concepts, Applications*. Springer Publishing Company, Incorporated, 1st edition.

Horgan, J. (1995). From complexity to perplexity. *Scientific American*, 272(6):104–109.

ISCPIF (2012). Institut des Systèmes Complexes Paris Île-de-France. `http://www.iscpif.fr`.

ISCV (2012). Instituto de Sistemas Complejos de Valparaíso. `http://www.sistemascomplejos.cl`.

IXXI (2012). Rhône Alpes complex systems institue. `http://www.ixxi.fr`.

Jamont, J. and Occello, M. (2009). A multiagent tool to simulate hybrid real/virtual embedded agent societies. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 501–504. IEEE.

Jamont, J.-P., Occello, M., and Lagrèze, A. (2010). A multiagent approach to manage communication in wireless instrumentation systems. *Measurement*, 43(4):489–503.

Jelasity, M., Montresor, A., and Babaoglu, O. (2003). A modular paradigm for building self-organizing peer-to-peer applications. In *Proceedings of the International Workshop on Engineering Self-Organising Applications*.

Jennings, N. R. and Bussmann, S. (2003). Agent-based control systems. *IEEE Control Systems Magazine*, 23(3):61–73.

Jian, L. and MacKie-Mason, J. (2008). Why share in peer-to-peer networks? In *Proceedings of the 10th international conference on Electronic commerce*. ACM New York, NY, USA.

Jovanović, M., Annexstein, F., and Berman, K. (2001). Modeling peer-to-peer network topologies through small-world models and power laws. In *IX Telecommunications Forum, TELFOR*.

Karakaya, M., Korpeoglu, I., and Ulusoy, O. (2009). Free Riding in Peer-to-Peer Networks. *IEEE Internet Computing*, 13(2):92–98.

Kauffman, S. (1993). *The origins of order*. Oxford University Press.

Kerner, B. S. and Klenov, S. L. (2009). Phase transitions in traffic flow on multilane roads. *PHYSICAL REVIEW E*, 80(5, Part 2).

Kevrekidis, I., Gear, C., and Hummer, G. (2004). Equation-free: The computer-aided analysis of complex multiscale systems. *AIChE Journal*, 50(7):1346–1355.

Kevrekidis, I., Gear, C., Hyman, J., Kevrekidis, P., Runborg, O., and Theodoropoulos, C. (2003). Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Comm. Math. Sci*, 1(4):715–762.

Klein, F. (2009). *Contrôle d'un Système Multi-Agents Réactif par Modélisation et Apprentissage de sa Dynamique Globale*. These, Université Nancy II.

Klein, F., Bourjot, C., and Chevrier, V. (2005). Dynamic design of experiment with MAS to approximate the behavior of complex systems. In *Multi-Agents for modeling Complex Systems (MA4CS'05) Satellite Workshop of the European Conference on Complex Systems (ECCS'05)*, Paris/France.

Klein, F., Bourjot, C., and Chevrier, V. (2008). Controlling the Global Behaviour of a Reactive MAS : Reinforcement Learning Tools. In *9th Annual International Workshop "Engineering Societies in the Agents World" - ESAW 08*, Saint-Etienne, France.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632.

Kocarev, L. and Vattay, G. (2005). *Complex dynamics in communication networks*. Springer-Verlag New York Inc.

Koestler, A. (1989). *The Ghost in the Machine*. An Arkana book. Penguin Group (Canada).

Krishnan, R., Smith, M., Tang, Z., and Telang, R. (2004). The impact of free-riding on peer-to-peer networks. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, page 10.

Ksontini, F., Espié, S., Guessoum, Z., and Mandiau, R. (2012). Traffic Behavioral Simulation in Urban and Suburban–Representation of the Drivers' Environment. *Advances on Practical Applications of Agents and Multi-Agent Systems*, pages 115–125.

La Complexité (2003). La Complexité.

Ladyman, J., Lambert, J., and Weisner, K. (2012). What is a complex system? *European Journal of Philosophy of Science*.

Langton, C. G. (1986). Studying artificial life with cellular automata. *Physica D*, 22D(1-3):120–49.

Law, A. M. (2009). How to Build Valid and Credible Simulation Models. In Dunkin, A., Ingalls, R. G., Yücesan, E., Rossetti, M. D., Hill, R., and Johansson, B., editors, *Winter Simulation Conference*, pages 24–33. WSC.

Lee, W.-H., Tseng, S.-S., Shieh, J.-L., and Chen, H.-H. (2011). Discovering Traffic Bottlenecks in an Urban Network by Spatiotemporal Data Mining on Location-Based Services. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 12(4):1047–1056.

Liu, S., Gruver, W., Kotak, D., and Bardi, S. (2000). Holonic manufacturing system for distributed control of automated guided vehicles. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 3, pages 1727–1732 vol.3.

Llaudes, R., Salman, F., and Chivakul, M. (2010). The Impact of the Great Recession on Emerging Markets. `http://ssrn.com/abstract=1750726`.

Lloyd, S. (2001). Measures of complexity: a nonexhaustive list. *Control Systems, IEEE*, 21(4):7–8.

Ma, R. T. B., Lee, S. C. M., Lui, J. C. S., and Yau, D. K. Y. (2006). Incentive and service differentiation in P2P networks: a game theoretic approach. *IEEE/ACM Trans. Netw.*, 14(5):978–991.

Makosiej, P., Sakaryan, G., and Unger, H. (2004). Measurement study of shared content and user request structure in peer-to-peer Gnutella network. *Design, Analysis and Simulation of Distributed Systems*, pages 115–124.

Mandiau, R., Champion, A., Auberlet, J., Espié, S., and Kolski, C. (2008). Behaviour based on decision matrices for a coordination between agents in a urban traffic simulation. *Applied Intelligence*, 28(2):121–138.

Maymounkov, P. and Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 53–65, London, UK. Springer-Verlag.

Mayntz, R. (2003). New challenges to governance theory. In Bang, H. P., editor, *Governance as social and political communication*, pages 27–40. Manchester Univ Pr.

Michel, F., Ferber, J., and Drogoul, A. (2009). Multi-Agent Systems and Simulation: A survey from the Agent Community's Perspective. In Uhrmacher, A. M. and Weyns, D., editors, *Multi-Agent Systems Simulation and Applications*, Computational Analysis, Synthesis, Design of Dynamic Models, chapter 1, pages 3–51. CRC Press.

Minsky, M. (1965). Matter, Mind and models. In *Proceedings of International Federation of Information Processing Congress*, volume 1, pages 45–49.

Mitchell, M. (2009). *Complexity: A guided tour*. Oxford University Press, USA.

Mnif, M., Richter, U., Branke, J., Schmeck, H., and Müller-Schloer, C. (2007). Measurement and Control of Self-organised Behaviour in Robot Swarms. In Lukowicz, P., Thiele, L., and Tröster, G., editors, *Architecture of Computing Systems - ARCS 2007*, volume 4415 of *Lecture Notes in Computer Science*, pages 209–223. Springer Berlin / Heidelberg. 10.1007/978-3-540-71270-1_16.

Mönch, L., Stehli, M., and Zimmermann, J. (2003). FABMAS: An Agent-Based System for Production Control of Semiconductor Manufacturing Processes. In Marík, V., McFarlane, D., and Valckenaers, P., editors, *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744 of *Lecture Notes in Computer Science*, pages 1085–1085. Springer Berlin / Heidelberg.

Müller, J.-P. (2003). Emergence of Collective Behaviour and Problem Solving. In Omicini, A., Petta, P., and Pitt, J., editors, *ESAW*, volume 3071 of *Lecture Notes in Computer Science*, pages 1–21. Springer.

Müller-Schloer, C. (2004). Organic computing: on the feasibility of controlled emergence. In Orailoglu, A., Chou, P. H., Eles, P., and Jantsch, A., editors, *CODES+ISSS*, pages 2–5. ACM.

Nature insight (2001). Complex Systems.

Navarrete Gutiérrez, T., Siebert, J., Ciarletta, L., and Chevrier, V. (2010). Impact des dimensions spatiale et temporelle dans la modélisation d'un phénomène collectif de type free-riding. In *18èmes Journées Francophones des Systèmes Multi-Agents - JFSMA'10*, pages 119–130, Mahdia Tunisie.

Navarrete Gutiérrez, T., Siebert, J., Ciarletta, L., and Chevrier, V. (2011). Impact des dimensions spatiale et temporelle dans la modélisation d'un phénomène collectif de type øg free-riding". *Revue d'Intelligence Artificielle, Editions Hermès*, 25(5):625–651.

NECSI (2012). New England Complex Systems Institute. `http://www.necsi.edu`.

Newman, D., Carreras, B., Lynch, V., and Dobson, I. (2011). Exploring Complex Systems Aspects of Blackout Risk and Mitigation. *Reliability, IEEE Transactions on*, 60(1):134–143.

Nicolis, G. and Nicolis, C. (2007). *Foundations of complex systems: nonlinear dynamics, statistical physics, information and prediction*. World Scientific Publishing Company.

Nicolis, G. and Prigogine, I. (1977). *Self-organization in nonequilibrium systems: from dissipative structures to order through fluctuations*. John Wiley & Sons New York.

Nicolis, S. and Deneubourg, J. (1999). Emerging patterns and food recruitment in ants: an analytical study. *Journal of Theoretical Biology*, 198(4):575–592.

ocwebsite (2012). Organic Computing Initiative website. `http://www.organic-computing.org`.

Ogata, K. (2001). *Modern control engineering*. Prentice Hall PTR.

Olbrich, E., Achermann, P., and Wennekers, T. (2011a). The sleeping brain as a complex system. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1952):3697–3707.

Olbrich, E., Claussen, J. C., and Achermann, P. (2011b). The multiple time scales of sleep dynamics as a challenge for modelling the sleeping brain. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1952):3884–3901.

Park, J., Celma, O., Koppenberger, M., Cano, P., and Buldu, J. M. (2007). The Social Network of Contemporary Popular Musicians. *Int. J. of Bifurcation and Chaos*, 17:2281–2288.

Park, K. (2005). The Internet as a complex system. *The Internet as a large-scale complex system*.

Parunak, H. V. D. and Brueckner, S. A. (2006a). Concurrent Modeling of Alternative Worlds with Polyagents. In Antunes, L. and Takadama, K., editors, *MABS*, volume 4442 of *Lecture Notes in Computer Science*, pages 128–141. Springer.

Parunak, H. V. D. and Brueckner, S. A. (2006b). Extrapolation of the Opponent's Past Behaviors. In Kott, A. W. and McEneaney, W. M., editors, *Adversarial reasoning : computational approaches to reading the opponent's mind*, chapter 1.3, pages 49–76. Chapman and Hall/CRC.

Parunak, H. V. D., Brueckner, S. A., and Sauter, J. (2005). Digital Pheromones for Coordination of Unmanned Vehicles. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems*, volume 3374 of *Lecture Notes in Computer Science*, pages 246–263. Springer Berlin / Heidelberg.

Parunak, H. V. D., Brueckner, S. A., Weyns, D., Holvoet, T., Verstraete, P., and Valckenaers, P. (2007). E Pluribus Unum: Polyagent and Delegate MAS Architectures. In Antunes, L., Paolucci, M., and Norling, E., editors, *MABS*, volume 5003 of *Lecture Notes in Computer Science*, pages 36–51. Springer.

Parunak, H. V. D., Savit, R., and Riolo, R. (1998). Agent-Based Modeling vs Equation-Based Modeling: A Case Study and Users' Guide. *Lecture Notes in Computer Science*, 1534:10–25.

Paxson, V. (2006). End-to-end routing behavior in the Internet. *COMPUTER COMMUNICATION REVIEW*, 36(5):43–56.

Phan, D. and Amblard, F. (2007a). *Agent-based Modelling and Simulation in the Social and Human Sciences*. Bardwell Press.

Phan, D. and Amblard, F., editors (2007b). *Agent-based modelling and simulation in the social and human sciences*. GEMAS Studies in Social Analysis. Bardwell.

Polack, F., Hoverd, T., Sampson, A., Stepney, S., and Timmis, J. (2008). Complex systems models: engineering simulations. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 482–489. MIT Press, Cambridge, MA.

Polack, F. A., Andrews, P. S., and Sampson, A. T. (2009). The engineering of concurrent simulations of complex systems. In *2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 217–224. IEEE Press.

Pourbeik, P., Kundur, P., and Taylor, C. (2006). The anatomy of a power grid blackout - Root causes and dynamics of recent major blackouts. *Power and Energy Magazine, IEEE*, 4(5):22–29.

Prothmann, H., Rochner, F., Tomforde, S., Branke, J., Müller-Schloer, C., and Schmeck, H. (2008). Organic Control of Traffic Lights. In Rong, C., Jaatun, M., Sandnes, F., Yang, L., and Ma, J., editors, *Autonomic and Trusted Computing*, volume 5060 of *Lecture Notes in Computer Science*, pages 219–233. Springer Berlin / Heidelberg. 10.1007/978-3-540-69295-9_19.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Ramaswamy, L. and Liu, L. (2003). Free riding: A new challenge to peer-to-peer file sharing systems. In *Proceedings of the Hawaii International Conference on Systems Science*, page 220.

Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., and Schmeck, H. (2006). Towards a generic observer/controller architecture for Organic Computing. In Hochberger, C. and Liskowsky, R., editors, *GI Jahrestagung (1)*, volume 93 of *LNI*, pages 112–119. GI.

rncsgouv2012 (2012). Approche Enactive pour la Gouvernance de Systèmes Socio-Techniques. `http://membres-liglab.imag.fr/bahttp://membres-liglab.imag.fr/badeig/siteRNSC/index.php`deig/siteRNSC/index.php.

RNSC (2012). Réseau National Des Systèmes Complexes. `http://rnsc.fr`.

Rouse, W. (2003). Engineering complex systems: implications for research in systems engineering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 33(2):154–156.

Rouse, W. (2007). Complex engineered, organizational and natural systems. *Systems Engineering*, 10(3):260–271.

Saint-Germain, B., Valckenaers, P., Van Brussel, H., Hadeli, O., Zamfirescu, C., and Verstraete, P. (2003). Multi-agent manufacturing control: an industrial case study. In *Proceedings of the 7th IFAC workshop on intelligent manufacturing systems, Budapest, Hungary*, pages 227–32.

Samaey, G., Holvoet, T., and Wolf, T. D. (2008). Using Equation-Free Macroscopic Analysis for Studying Self-Organising Emergent Solutions. In Brueckner, S. A., Robertson, P., and Bellur, U., editors, *SASO*, pages 425–434. IEEE Computer Society.

Samaey, G., Roberts, A., and Kevrekidis, I. (2009). Equation-free computation: an overview of patch dynamics. *Multiscale methods: bridging the scales in science and engineering*, page 216.

Samaey, Y. K. G. (2010). Equation-free modeling. *Scholarpedia*, 5(9):4847.

Santa Fe Institue (2012). Santa Fe Institue. `http://www.santafe.edu`.

Saroiu, S., Gummadi, P., Gribble, S., et al. (2002). A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking*, volume 2002, page 152.

Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17.

Schmeck, H. (2005). Organic computing-a new vision for distributed embedded systems. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 201–203. IEEE.

Shahabi, C. and Banaei-Kashani, F. (2007). Modelling P2P data networks under complex system theory. *Int. J. Comput. Sci. Eng.*, 3(2):103–111.

Siettos, C. (2011). Equation-Free multiscale computational analysis of individual-based epidemic dynamics on networks. *Applied Mathematics and Computation*, 218(2):324–336.

Siettos, C., Rico-Martinez, R., and Kevrekidis, I. (2006). A systems-based approach to multiscale computation: Equation-free detection of coarse-grained bifurcations. *Computers and Chemical Engineering*, 30(10-12):1632–1642.

Sitte, R. (2009). About the predictability and complexity of complex systems. In Aziz-Alaoui, M. and Bertelle, C., editors, *From System Complexity to Emergent Properties*, volume 44 of *Understanding Complex Systems*, pages 23–48. Springer Berlin / Heidelberg. 10.1007/978-3-642-02199-2_3.

Smith, R. (2011). The dynamics of internet traffic: self-similarity, self-organization, and complex phenomena. *Advances in Complex Systems*, 14(6):905.

Sontag, E. (1998). *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Verlag.

Steen, M. (2010). *Graph Theory and Complex Networks*. Createspace.

Stutzbach, D. and Rejaie, R. (2005). Capturing accurate snapshots of the Gnutella network. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2825–2830 vol. 4.

Stutzbach, D., Rejaie, R., and Sen, S. (2008). Characterizing unstructured overlay topologies in modern P2P file-sharing systems. *IEEE/ACM Transactions on Networking*, 16(2):267–280.

SysML Specification (2012). OMG System Modeling Language. `http://www.omg.org/spec/SysML/`.

The Department of Complexity Science and Engineering (2012). Tokyo university. `http://www.k.u-tokyo.ac.jp/complex/index-e.html`.

Thomas, V., Bourjot, C., and Chevrier, V. (2003). Du collectif pour la satisfaction individuelle : un modèle d'auto-organisation. In *Journees Francophones de Systemes Multi Agents'2003*, pages 261–265, Hammamet, Tunisie. none.

Thomas, V., Bourjot, C., Chevrier, V., and Desor, D. (2007). Démarche incrémentale pour l'évaluation d'un modèle multi-agent en éthologie. In *ARCo'07 : Cognition – Complexité – Collectif*, pages 177–188, Nancy France.

Tomforde, S., Steffen, M., Hähner, J., and Müller-Schloer, C. (2009). Towards an Organic Network Control System. In Nieto, J. G., Reif, W., Wang, G., and Indulska, J., editors, *Autonomic and Trusted Computing*, volume 5586 of *Lecture Notes in Computer Science*, pages 2–16. Springer Berlin / Heidelberg. 10.1007/978-3-642-02704-8_2.

Tomforde, S., Zgeras, I., Hähner, J., and Müller-Schloer, C. (2010). Adaptive Control of Sensor Networks. In Xie, B., Branke, J., Sadjadi, S., Zhang, D., and Zhou, X., editors, *Autonomic and Trusted Computing*, volume 6407 of *Lecture Notes in Computer Science*, pages 77–91. Springer Berlin / Heidelberg.

Tsvetovat, M. and Carley, K. M. (2004). Modeling Complex Socio-technical Systems using Multi-Agent Simulation Methods. *KI*, 18(2):23–28.

Ulieru, M. and Doursat, R. (2011). Emergent engineering: a radical paradigm shift. *International Journal of Autonomous and Adaptive Communications Systems*, 4(1):39–60.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry*, 37(3):255–274.

Van Eijnatten, F. (2005). Methodological aspects of chaos and complexity in organisation and management. In *Workshop of the Netherlands Organization for Methodological Research in the Social Sciences (NOSMO), Socio-Cybernetica Working Group, Radboud University, Nijmegen, November*, volume 25.

Van Leeuwen, E. and Norrie, D. (1997). Holons and holarchies. *Manufacturing Engineer*, 76(2):86–8.

Vassilakis, D. and Vassalos, V. (2007). Modelling real p2p networks: The effect of altruism. In *Seventh IEEE International Conference on Peer-to-Peer Computing, 2007. P2P 2007*, pages 19–26.

Vespignani, A. (2009). Predicting the behavior of techno-social systems. *Science*, 325(5939):425.

Vivier-Lirimont, S. (2008). Une analyse de la dimension réseau des fragilités bancaires et financières. *Revue française d'économie*, 22(3):61–95.

Wang, L., Wang, Z., Yang, C., Zhang, L., and Ye, Q. (2009). Linux kernels as complex networks: A novel method to study evolution. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 41–50. IEEE.

Watts, D. (1999). *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press.

Watts, D. and Strogatz, S. (1998). Collective Dynamics of 'small-world' networks. *Nature*, 393:440–442.

Weiser, M. (1991). The computer of the twenty-first century. *Scientific American*, 265(3):94–104.

Weisstein, E. W. (2012). Cellular Automaton. `http://mathworld.wolfram.com/CellularAutomaton.html`.

Willinger, W., Govindan, R., Jamin, S., Paxson, V., and Shenker, S. (2002). Scaling phenomena in the Internet: Critically examining criticality. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 1):2573–2580.

Wolf, T. D., Holvoet, T., and Samaey, G. (2005). Engineering Self-Organising Emergent Systems with Simulation-based Scientific Analysis. In *Proceedings of the Fourth International Workshop on Engineering Self-Organising Applications*, pages 146–160. Universiteit Utrecht.

Wolfram, S. (1994). *Cellular Automata and Complexity: Collected Papers*. Addison-Wesley.

Wolfram, S. (2002). *A new kind of science*. Wolfram Media Inc.

Würtz, R. P. (2008). *Organic Computing*, volume 21 of *Understanding Complex Systems*. Springer Berlin / Heidelberg.

Xu, R., Huang, L., Yin, S., Yao, D., Zhang, H., and Peng, L. (2011). An Analysis on Traffic Flow Characteristics and Lane Changing Behaviors in Beijing Urban Expressway Bottlenecks. In *2011 14TH IEEE International Conference on Intelligent Transportation Systems-ITSC*, pages 822–827. IEEE.

Yang, M., Zhang, Z., Li, X., and Dai, Y. (2005). An empirical study of free-riding behavior in the maze p2p file-sharing system. *Lecture notes in computer science*, 3640:182.

Zamfirescu, C., Valckenaers, P., Van Brussel, H., and Germain, B. (2003). A case study for modular plant control. *Holonic and Multi-Agent Systems for Manufacturing*, pages 1090–1091.

Zeigler, B., Praehofer, H., and Kim, T. (2000). *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. Academic Press.

Zghaibeh, M. and Harmantzis, F. (2008). Revisiting free riding and the Tit-for-Tat in BitTorrent: A measurement study. *Peer-to-Peer Networking and Applications*, 1(2):162–173.

Annexe A

Résumé étendu : Une architecture de contrôle de systèmes complexes basée sur la simulation multi-agent.

Introduction

Au cours des vingt dernières années, on a pu observer qu'il existe beaucoup de systèmes aussi bien naturels qu'artificiels avec un grand nombre de participants qui présentent des caractéristiques au niveau du groupe qui ne peuvent pas être identifiées au niveau individuel. On qualifie ces systèmes de « complexes ».

Les caractéristiques des systèmes complexes sont : un grande nombre d'entités autonomes, une sensibilité aux conditions initiales, différents niveaux d'organisation, des structures dynamiques, l'émergence de propriétés, et des dynamiques faisant intervenir différentes échelles de temps et d'espace.

Des exemples de systèmes comportant ces caractéristiques sont : les réseaux de distribution d'électricité, Internet, le système financier mondial, les réseaux de transport. On a observé aussi dans ces systèmes des comportements non désirés : coupures de courant électrique (Newman et al., 2011; Pourbeik et al., 2006), pannes du routage sur internet (Dainotti et al., 2011; Smith, 2011; Paxson, 2006), crises financières récurrentes (Fender et al., 2012; Vivier-Lirimont, 2008; Llaudes et al., 2010; Artus and Lecointe, 1991), et aussi embouteillages dans les transports (Lee et al., 2011; Xu et al., 2011; Kerner and Klenov, 2009).

Nous proposons un mécanisme de contrôle pour les systèmes complexes, sous la forme d'une architecture de contrôle conçue comme étant « exogène » au système cible avec une approche « equation-free ». Cette architecture utilise la simulation de modèles multi-agents pour évaluer l'impact global d'actions de contrôle locales avant d'appliquer la plus pertinente.

L'approche multi-agents est particulièrement adaptée pour la modélisation de systèmes dont les propriétés globales ne découlent pas directement des propriétés des composants du système (Phan and Amblard, 2007a), comme cela est le cas dans les réseaux pair à pair, par exemple. Elle permet d'étudier les conséquences d'un ou plusieurs comportements individuels sur la globalité du système.

Cette thèse évalue la pertinence d'utiliser un modèle multi-agent du système cible pour construire un mécanisme de contrôle.

A.1. Les systèmes complexes

Dans le contexte des sciences de la complexité, il existe plusieurs définitions d'un système complexe. Nous avons choisi une définition issue d'un travail collectif plutôt qu'une définition isolée directement liée à un chercheur ou à un domaine.

Définition 1 *Un « système complexe » est en général un système composé d'un grand nombre d'entités hétérogènes, parmi lesquels des interactions locales créent plusieurs niveaux de structures collectives et de l'organisation. (Chavalarias et al., 2009)*

Bien qu'il n'y ai pas de définition consensuelle et simple, il y a plus d'accord sur les caractéristiques de ces systèmes. Un regard sur la littérature existante concernant les systèmes complexes nous montrera que les caractéristiques ne se limitent pas aux trois concepts que nous retrouvons dans la définition donnée, comme souligne par exemple Sitte (2009). Nous allons donc nous concentrer sur les caractéristiques des systèmes complexes pertinentes aux difficultés de l'étude de systèmes complexes (Amaral and Ottino, 2004).

A.1.1. Caractéristiques de Systèmes Complexes

Emergence

Définition 2 *Une propriété émergente est le résultat (à l'échelle globale) des interactions entre les entités (au niveau local).*

Définition 3 *L'Immergence est le processus par lequel le comportement global du système influe sur les interactions entre les entités, dans une boucle de rétroaction.*

La boucle de rétroaction indique que la sortie d'un système sera l'entrée du même système ou processus. Dans le contexte des systèmes complexes, cela signifie que les interactions entre les entités prendront éventuellement en entrée le comportement global du système, qui, à son tour est la sortie (résultat) des interactions entre les entités.

Différents niveaux de description

Au moins deux niveaux différents pour décrire un système complexe peuvent être considérés : l'échelle locale ou microscopique et l'échelle macroscopique ou globale.

Le niveau de description d'un système complexe est dit local ou microscopique quand il emploie exclusivement les caractéristiques inhérentes aux entités du système.

Le niveau de description d'un système complexe est dit global ou macroscopique quand il emploie les caractéristiques des groupes d'entités du système.

Au niveau local, nous pouvons définir chacun des éléments du système, ainsi que la façon dont ils interagissent avec d'autres éléments. Au niveau global, nous pouvons caractériser le système comme un ensemble.

Échelle

Les systèmes complexes sont constitués d'éléments qui peuvent être décrits à leur niveau local. Ces éléments interagissent les uns avec les autres, à leur propre échelle de temps et d'espace. Toutefois, lorsque le système est observée à un niveau « global » ou macroscopique le point de vue change. Le temps et les échelles spatiales au cours desquels les entités du système évoluent peuvent être différents en plusieurs ordres de grandeur vis-a-vis des échelles microscopiques.

Sensibilité aux conditions initiales

Les systèmes complexes sont sensibles aux conditions initiales. De très petites différences dans celles-ci peuvent produire des résultats significativement différents au niveau quantitatif et qualitatif.

Structures dynamiques

Les structures dans un système complexe sont le produit d'interactions entre les entités du système. Tout comme les interactions, les structures dans le système complexe changent au fil du temps.

Pour éclaircir les caractéristiques des systèmes complexes précédemment énoncés, nous proposons de les identifier dans quatre exemples concrets de systèmes complexes.

A.1.2. Exemples de systèmes complexes

La recherche de nourriture chez les fourmis

Les insectes sociaux comme les fourmis, les abeilles ou les termites sont connus pour accomplir des tâches collectives telles que la construction du nid, la défense contre des prédateurs et la recherche de nourriture. Si l'on considère une colonie de fourmis comme un système complexe, les entités qui la composent sont les fourmis. Dans une colonie, les fourmis s'organisent pour trouver de la nourriture et l'apporter à la colonie. Au niveau local, les fourmis sont des organismes relativement simples avec des moyens de communication limités. Cependant, au niveau global, une colonie de fourmis est capable de produire collectivement des phénomènes émergents. Un exemple d'un comportement émergent dans les colonies de fourmis est le phénomène de recherche de nourriture (Deneubourg et al., 1990). Des expériences ont permis d'observer comment, par l'utilisation de phéromones, les fourmis communiquent la position d'une source de nourriture.

Le cerveau

Une des façons de concevoir le cerveau comme un système complexe est de considérer que ses entités sont les neurones. Il existe un grand nombre de neurones (et d'autres types de cellules) dans un cerveau humain. Au niveau local, les neurones sont les éléments fondamentaux du cerveau. En outre, ils peuvent être subdivisés en compartiments, synapses, canaux, molécules et ainsi de suite, avec les propriétés physiologiques et dynamiques à tous les niveaux (Olbrich et al., 2011a). Au niveau global, le cerveau est capable de produire un comportement émergent. Ils existent différentes échelles de temps dans le cerveau, couvrant plusieurs ordres de grandeur : les processus synaptiques peuvent se produire dans un écart de temps en millisecondes alors que les processus cognitifs se produisent dans un écart de temps pouvant aller de secondes jusqu'à des heures. De façon similaire, les phénomènes de mémoire peuvent durer entre quelques secondes et plusieurs années. On peut supposer que les processus à l'intérieur du cerveau ne sont pas des actes isolés, mais plutôt qu'ils sont intégrés dans un réseau de dépendances mutuelles complexes à travers le temps (Olbrich et al., 2011a). Cependant, il n'est pas encore compris ce que signifie les signaux neuronaux ou comment ils donnent lieu à des comportements cognitifs globaux. Et tout cela en dépit de la vaste connaissance sur la structure des neurones et leurs interactions (au niveau chimique) avec d'autres neurones.

Internet

Internet est l'un des plus grands systèmes fabriqués par l'homme, il est fait d'un très grand nombre d'entités hétérogènes : ordinateurs, serveurs, téléphones mobiles, dispositifs divers (appareils électroménagers, voitures) (Willinger et al., 2002; Park, 2005). En tant que système complexe, on pourrait considérer que, au niveau local, nous avons tous les appareils qui sont capables de communiquer avec tout autre appareil dans l'Internet. Toutefois, une description plus précise devrait inclure plusieurs niveaux d'organisation entre les différentes entités : routeurs, les systèmes autonomes, des réseaux locaux, réseaux étendus, etc.

Internet a des échelles spatiales de niveaux différents. Il permet la communication d'un endroit de la planète, à presque n'importe quel autre endroit (si une connexion à internet est disponible dans les deux endroits). Mais la communication est également possible entre des entités connectées au réseau qui sont l'une à côté de l'autre. La communication peut avoir lieu à des centaines de milliers de kilomètres de distance, ou à quelques mètres à peine. Les vitesses actuelles de communication dans Internet permettent de communiquer presque instantanément. Poster un commentaire sur un site internet peut prendre des secondes avant qu'il ne soit accessible au public partout dans le monde, mais le temps nécessaire avant que d'autres entités (personnes, moteurs de recherche, caches de fournisseurs d'accès à internet par exemple) reconnaissent cette nouveauté peut prendre quelques minutes, voire plusieurs jours.

Les automates cellulaires

Un automate cellulaire est une collection de cellules « colorées » sur une grille de forme déterminée, qui évolue à travers un certain nombre de règles en temps discret selon un ensemble de règles fondées sur les états des cellules voisines. Les règles sont ensuite appliquées itérativement autant de fois que désiré (Weisstein, 2012).

Si l'on considère qu'une cellule est une entité, on peut associer comme suit les caractéristiques d'un automate cellulaire à celles des systèmes complexes. L'ensemble des règles peut être considéré comme les interactions entre les entités. Les cellules changent de couleur en fonction des interactions avec d'autres cellules voisines. Aucune cellule spécifique ne dicte pas la façon dont toutes les autres cellules vont se comporter, ce qui signifie que les entités sont autonomes. Les règles sont données en fonction des cellules, et donc le comportement global de l'automate ne peut pas être déduit directement des cellules. Il n'y a pas de façon claire d'identifier les échelles de temps et l'espace d'un automate cellulaire. Les interactions entre cellules peuvent dans un pas de temps peuvent être très simples et prendre très peu de temps de simulation pour être exécutés. Sous une simulation sur ordinateur, il peut prendre très peu de pas temps ou bien des centaines pour arriver à observer les tendances qui peuvent être intéressantes au niveau global.

(Hanson, 2009) représente la présence de l'émergence dans les automates cellulaires comme suit.

> Dans les automates cellulaires, l'état du système se compose d'un tableau à N-dimensions de cellules qui prennent des valeurs discrètes et la dynamique est donnée par une règle mise à jour en temps discret. Les phénomènes qui émergent dans le AC se composent donc nécessairement de modèles spatio-temporels et / ou des régularités statistiques dans les valeurs des cellules. [...] La synchronisation est peut-être le type le plus simple de phénomène émergent dans le AC. La synchronisation est la croissance des régions spatiales dans lesquelles toutes les cellules ont la même valeur. Une région synchronisée reste synchronisée avec le temps (sauf peut-être à ses frontières) et elle peut l'être temporellement invariant (les valeurs des cellules

ne changent pas dans le temps) ou périodique (les cellules suivent toutes une même séquence temporelle des valeurs).

A.1.3. Les challenges dans l'étude des systèmes complexes

Les sciences de la complexité sont utiles pour caractériser ou identifier les systèmes complexes. Elles sont également utiles pour créer de nouvelles approches pour étudier les systèmes complexes. Dans l'étude de systèmes complexes, plusieurs challenges sont recensés : la modélisation, l'ingénierie et le contrôle de systèmes complexes (Chavalarias et al., 2009).

Modélisation

Il y a trois éléments au défi de modélisation de systèmes complexes : la nature non-linéaire de leur dynamique, les différents niveaux de description des systèmes complexes et la disponibilité de grandes quantités de données observées à partir de systèmes complexes. Étant donné la nature non-linéaire de la dynamique des systèmes complexes, les modèles analytiques ne sont pas toujours utilisables. Bien qu'il ne soit pas impossible de créer un modèle analytique, il sera le plus souvent non solvable soit en raison de la non-linéarité soit en raison du nombre élevé de variables. Les systèmes complexes ont plusieurs niveaux de description, et les modèles devraient prendre en considération ces niveaux. Ce n'est pas possible avec des modèles analytiques, car ils ne décrivent que la dynamique au niveau global. De nos jours l'avance technologique permet de disposer d'informations abondantes à des niveaux très fines de des systèmes complexes. Trouver comment utiliser cette information pour créer des modèles qui peuvent représenter avec précision le système est un défi.

Ingénierie

Les systèmes complexes fabriqués par l'homme, sont conçus pour servir un but et avoir un certain comportement. En d'autres termes, ils sont conçus pour fonctionner d'une manière spécifique. Le développement technologique a permis de construire des systèmes que nous considérons comme complexe d'aujourd'hui, comme Internet. De plus en plus les systèmes sont construits au-dessus des systèmes complexes déjà déployés. L'ingénierie classique atteint ses limites lorsqu'il s'agit de systèmes complexes. Le challenge dans l'ingénierie de systèmes complexes est l'obtention d'un fonctionnement désirée, malgré le fait que la façon d'arriver à cet fonctionnement ne puisse pas être connue, car trop complexe ou non reproductible. Un autre aspect du défi est de savoir si cela est une hypothèse acceptable pour tous les domaines d'application.

Contrôle

Contrôler un objet est influencer son comportement de manière à atteindre un objectif souhaité. Tout au long de cette thèse, nous nommerons un « système cible » un système que nous voulons contrôler. La théorie du contrôle traditionnel suppose que le modèle du système cible est disponible et qu'il est analytiquement (ou numériquement) possible de l'utiliser. Contrôler implique d'avoir un modèle et comme nous l'avons dit avant, la modélisation des systèmes complexes est déjà un défi en soi. En outre, étant donné que les systèmes complexes sont composés d'entités autonomes avec différents niveaux de description, la question de savoir comment observer le système devient une complication supplémentaire pour obtenir un contrôle. Autrement dit, la seule façon de « modifier » le comportement global d'un système complexe est en exécutant une influence sur le système au niveau local parce que les interactions locales produisent le

résultat global. Ce travail de thèse porte sur le contrôle de systèmes complexes. La théorie du contrôle suppose que, compte tenu d'un modèle, il existe une manière optimale pour contrôler un système cible.

Il a été reconnu qu'à cause des caractéristiques des systèmes complexes, le concept de gouvernance est mieux adapté que celui de contrôle. Dans la section suivante, nous abordons cette notion.

A.1.4. Gouvernance et contrôle de systèmes complexes

Une première hypothèse de la théorie du contrôle qui ne tient pas dans le cadre de systèmes complexes, est la possibilité de modéliser les systèmes sous forme analytique. Une deuxième hypothèse est l'optimalité. Dans le contexte des systèmes complexes il devrait être considéré comme possible de guider le comportement du système, mais pas toujours dans des conditions optimales (pas toujours en maximisant une fonction). Nous ne pouvons pas obtenir une valeur optimale de référence si nous ne pouvons pas avoir un modèle qui la produit. En termes plus mathématiques, si nous ne pouvons pas avoir un modèle qui donne une fonction à maximiser, nous ne pouvons pas avoir un contrôle optimal. Ces hypothèses précédentes nous conduisent à considérer comme une possibilité l'exploration de multiples cours d'action pour contrôler un système complexe, au lieu de considérer exclusivement l'approche « maximiser un fonction » de la théorie du contrôle.

De nouvelles façons de trouver des solutions au problème de contrôle qui n'ont pas l'intention d'être optimales doivent être trouvées. Un terme mieux adapté pour désigner l'action de « guider un système complexe à une position ou état désiré » qui n'implique pas l'optimalité est nécessaire. En disant cela, nous considérons que le contrôle optimal d'un système complexe est hors porté du cadre de cette thèse. La raison principale derrière cette hypothèse est que le contrôle optimal implique la présence d'un modèle analytique qui permet d'obtenir une fonction à maximiser (pour obtenir une valeur optimale). Et comme nous l'avons déjà dit, ces modèles ne sont pas toujours utilisables, car ils ne prennent pas en considération les différents niveaux d'organisation de systèmes complexes.

A.1.5. Difficultés du contrôle de systèmes complexes

Comme nous l'avons souligné, le contrôle de système complexes est un challenge actif dans l'étude de ce type de systèmes. Nous présentons un ensemble d'éléments que nous considérons comme ayant une importance particulière.

Actions locales avec des effets globaux Les interactions dans un système complexe se produisent au niveau local, parce qu'elles ont lieu entre les entités du système. Cependant, à cause de l'émergence présente dans les systèmes complexes, nous pouvons voir que les répercussions des actions locales seront perçues au niveau global. Ainsi, toute action de contrôle doit être soigneusement conçue dans cet esprit. Toutefois, en raison de la difficulté de modélisation des systèmes complexes ceci n'est pas une tâche triviale.

Autonomie des entités Pour des raisons juridiques, éthiques et techniques, il est impossible de modifier directement le comportement des entités de certains types de systèmes. Les actions de contrôle qui pourraient être appliquées au système ne devraient pas avoir l'intention de modifier directement le comportement interne des entités du système. Si des mesures de contrôle appliquées modifient le comportement interne des entités, l'autonomie des entités disparaît.

Modélisation Les modèles mathématiques reposent sur l'identification des composants clé du système, souvent représentés d'une manière discrète. Ceci limite les modèles mathématiques parce que l'émergence dans les systèmes complexes apparaît en conséquence des interactions locales, et ne peut pas être identifiée à l'avance comme composante clé du système.

Systèmes préexistants La modification de l'autonomie pourrait signifier l'interruption de l'exécution du système cible, et ceci n'est pas toujours désiré ou même possible. De plus, un système complexe ouvert peut exhiber un comportement différent de celui désiré. Cela peut arriver par exemple, dans des systèmes ouverts, sensibles aux changements dans l'environnement, comme conséquence de l'évolution du comportement des entités. Ceci semble particulièrement possible si les entités du système ont l'autonomie suffisante pour ce faire. Dans ce cas il semble naturelle que des mécanismes de contrôle fonctionnant avec un modèle du comportement des entités soient rendu inutiles ou inefficaces si le comportement change.

A.2. Travaux connexes

Introduction

Nous présentons différents travaux reliés au contrôle de systèmes complexes. D'abord nous nous concentrons sur la modélisation de systèmes complexes. L'objectif de cette thèse est d'évaluer la pertinence de la modélisation et simulation multi-agent pour le contrôle de systèmes complexes. C'est pour cela que, nous présentons le paradigme multi-agent ainsi que des éléments pertinents du cadre de la modélisation et la simulation en général. Ensuite nous nous concentrons sur l'évaluation de l'état du système. Nous présentons le cadre de la théorie du contrôle et l'approche « equation-free ».

Enfin, nous présentons différents travaux appliquant le paradigme multi-agent dans le contexte du contrôle systèmes complexes.

A.2.1. Modélisation des Systèmes complexes

Le Paradigme Multi-Agent

Le cadre de pensée ou paradigme multi-agent se caractérise par l'exécution et répartie et parallèle d'entités autonomes. Ces entités autonomes sont appelles « agents ».

Définition 4 *Un agent est une entité autonome capable d'agir pour atteindre ses objectifs. Un agent est une entité sociale qu'interagit avec d'autres agents ou avec l'environnement pour atteindre ses objectifs. Un agent peut représenter une entité physique ou virtuel.*

Un système multi-agent est composé des éléments suivants : les agents, l'environment et les interactions (agent-agent ainsi que agent-environnement) (Ferber, 1995). Dans le paradigme, la dynamique globale d'un système, au niveau macroscopique n'est pas donnée à l'avance. La dynamique globale est le résultat des interactions des comportements de chaque agent, au niveau microscopique. De ce fait, un des avantages particulièrement important du paradigme, est qu'il permet d'analyser l'influence d'un comportement local par rapport au comportement global du système (Parunak et al., 1998).

Le paradigme peut être utilisé dans le cadre de l'ingénierie de systèmes ou bien dans le cadre de la modélisation des systèmes pré-existants. Les outils pour concevoir des systèmes multi-agents comprennent, entre autres les plateformes de simulation multi-agent (Netlogo, Repast, MadKit,

CORMAS, JADE) les protocoles d'interaction (Contract Net interaction protocol, KQML, FIPA-ACL), et les langages de programmation (JACK). Le principal outil utilisé pour étudier les systèmes préexistants sont les modèles basés agents.

Modèles basés agent

De façon générale, un modèle est une représentation simplifiée d'un phénomène. Dans ce travail de thèse, nous adoptons le point de vue du cadre « modélisation et simulation » de (Zeigler et al., 2000).

Dans ce cadre les entités principales sont : le système source, le modèle, le simulateur et le cadre expérimental. Le cadre définit aussi les différents rapports entre ces entités. Le système source est le système pré-existant modélisé. Il est la source des informations observées. Les informations obtenues par observation ou expérimentation forment la base de données de comportements. Le cadre expérimental est la spécification des conditions sous lesquelles le système est observé. Le simulateur est l'entité chargée de mettre en œuvre les instructions d'un modèle et de générer un comportement. Ces éléments sont illustrés dans la figure A.1. (Michel et al.,

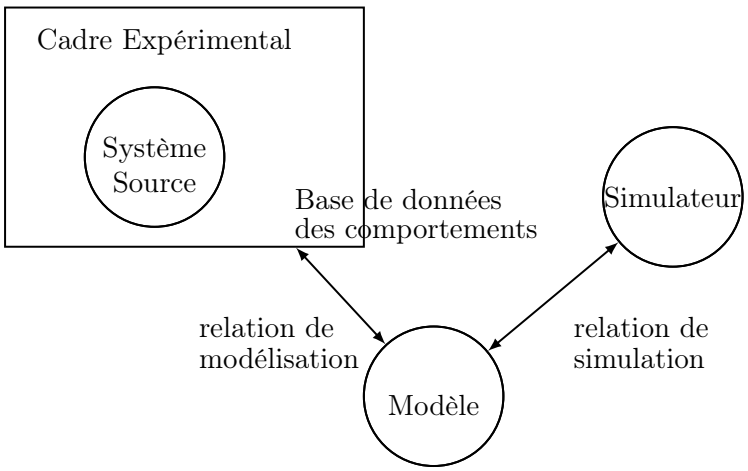


FIGURE A.1 – Entités et relations du cadre « Modélisation et Simulation ». Reprise de (Zeigler et al., 2000)

2009) Les relations décrites dans le cadre de modélisation et simulation de Zeigler sont décrites dans le paradigme multi-agent comme le couplage entre les différents éléments d'un modèle.

Du point de vue du cadre de Zeigler, les modèles peuvent être classifiés en fonction du niveau d'information acquis à partir du système simulé. Dans le cadre modélisation et simulation, Zeigler et al. proposent une hiérarchie de spécification de systèmes pour classifier les modèles. Cette hiérarchie est reproduite dans le tableau A.1. Quand un système multi-agent est utilisé comme un modèle, il partage naturellement les caractéristiques des modèles de simulation, tel que défini dans le cadre simulation et modélisation de Zeigler.

Définition 5 *Un modèle de simulation est un ensemble d'instructions, de règles, d'équations ou de contraintes pour générer le comportement d'entrée et de sortie. (Zeigler et al., 2000)*

Il est d'usage de définir un modèle multi-agent à partir des éléments suivants :

Niveau	Nom	Acquises à ce niveau
0	Cadre d'observation	Comment simuler le système avec des entrées : quelles variables mesurer et comment les observer dans un laps de temps.
1	Comportement E/S	Données récoltées à partir du système source et étiquetées en fonction du temps ; sous la forme de couples entrée / sortie
2	Fonction E/S	État initial connu ; pour un état initial donnée, chaque stimulus d'entrée produira une sortie unique.
3	Transition d'état	La manière dont les états sont influencés par les entrés ; pour un état donné et une entré donnée, quel est l'état une fois le stimulus terminé ; quel événement de sortie est produit par un état.
4	Composants couplés	Composants et comment ils sont couplés ensemble.

TABLE A.1 – Hiérarchie de spécification de systèmes, adapté de (Zeigler et al., 2000, chapitres 1 et 5)

Définition 6 *Éléments d'un modèle multi-agent.*

1. *Le comportement de chaque agent.*
2. *Les interactions de chaque agent (avec d'autres agents et avec l'environnement).*
3. *L'environnement dans lequel les agents sont situés.*

Simulation

L'approche multi-agent à la modélisation est computationelle : il faut simuler le modèle pour obtenir le comportement global du système à partir de certaines conditions initiales. Le simulateur est en charge de l'exécution de chaque agent dans le modèle, au cours d'une période de temps donnée. L'ordre dans quel chacun des agents est prévu pour être exécuté dans le simulateur est un aspect très important d'une simulation multi-agent. Cet ordre est généralement connu sous le nom de régime de mise à jour, ou type de planification. Nous appellerons un régime de mise à jour « synchrone » lors que tous les agents dans le modèle sont programmés pour être activés en même temps. Nous appellerons un régime de mise à jour « asynchrone » lors que seulement un sous-ensemble des agents sont activés en même temps. Sans spécification correcte d'un régime de mise à jour, des modèles avec des définitions de comportement d'agents identiques, peuvent produire des résultats différents, même contradictoires (Fatès and Chevrier, 2010; Navarrete Gutiérrez et al., 2011).

Validation et Calibration

Définition 7 *Valider un modèle multi-agent consiste à identifier les conditions sous lesquelles il produit le même comportement que le système modélisé.*

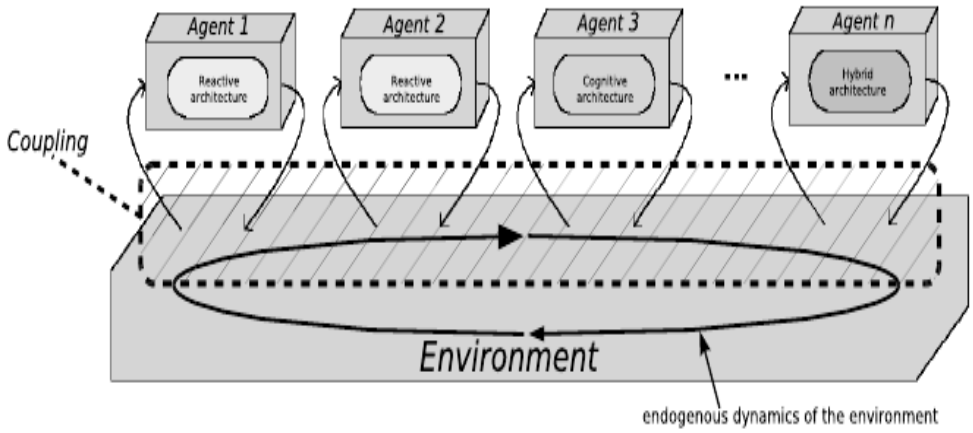


FIGURE A.2 – Représentation schématique du couplage entre les différents éléments d'une simulation multi-agent. Reprise de (Michel et al., 2009).

Définition 8 *Calibrer un modèle multi-agent consiste à trouver les valeurs pour un ensemble de paramètres du modèle qui permettent au modèle d'avoir un comportement spécifique (habituellement observé à partir du système modélisé).*

A.2.2. Le paradigme multi-agent et les systèmes complexes

Le paradigme de multi-agent a été identifié comme étant approprié pour modéliser des systèmes complexes dans des domaines spécifiques comme la sociologie (Conte et al., 1997; Bonabeau, 2002; Tsvetovat and Carley, 2004; Amblard and Phan, 2006; Gilbert, 2008) et la biologie (Grimm and Railsback, 2005; Thomas et al., 2003, 2007). L'approche a aussi été identifiées comme appropriée à l'ingénierie de systèmes complexes (Jennings and Bussmann, 2003). Jennings and Bussmann partagent le point de vue de (Polack et al., 2008, 2009) dans lequel ils soutiennent tous que le paradigme multi-agent fournit des abstractions appropriées utiles dans le contrôle, l'ingénierie aussi bien que dans la simulation et la modélisation de systèmes complexes d'un point de vue non spécifique au domaine d'application. Les abstractions sont les suivantes.

- **Agent**. L'abstraction de base dans le paradigme, est l'équivalent d'entités autonomes dans un système de contrôle.
- **Organisation des agents** Organisation hiérarchique ou heterarchique dans systèmes complexes.
- **Environement.** L'environment ou les entités du système complexe evolouent.
- **Interactions** Interactions entre les entités des systèmes complexes.

A.2.3. Théorie du contrôle

L'objectif de la théorie du contrôle est de déterminer les actions de contrôle qui font qu'un système arrive ou reste dans un état spécifique (Sontag, 1998; Ogata, 2001). Dans la théorie du contrôle, les systèmes dynamiques sont décrits à l'aide d'équations différentielles ordinaires sous la forme : $\begin{cases} \dot{x}(t) = f(x(t), \alpha(t)) \quad (t > 0) \\ x(0) = x^0 \end{cases}$

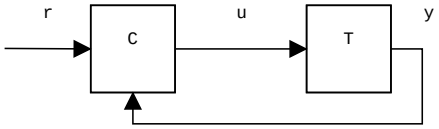


FIGURE A.3 – Boucle de rétro-action fermée. La sortie u du système C est utilisée comme entrée par le système T pendant que la sortie y du système T est utilisé comme entrée du système C.

Dans ces équations, x est une fonction qui représente l'évolution de l'état du système et f dépend d'une fonction de contrôle α. Dans la théorie du contrôle, un contrôleur de rétro-action mesure l'effet des entrées du système sur les sorties et les résultats obtenus par le contrôleur sont utilisés à nouveau comme entrées pour le système, formant ainsi une boucle (Astrom and Murray, 2008).

Définition 9 *Dans un boucle de contrôle de rétro-action l'état du système est mesuré (de façon régulière) Ensuite, des paramètres de contrôle sont ajustés pour conduire le système à un état spécifique en se basant sur ces mesures.*

Rétro-action dans le contrôle

La rétro-action est présente quand deux entités exercent une influence mutuelle l'une sur l'autre. Dans la théorie du contrôle, la rétro-action est une boucle où les sorties d'un système sont les entrées d'un autre système. La figure A.3

L'objectif de la théorie du contrôle est de trouver une loi de contrôle qui sera utilisée pour établir les entrés du contrôleur de telle sorte que les sorties du système cible soient aussi proches autant que possible de la référence (qui à son tour est l'entrée du contrôleur). La loi de contrôle est le cœur du contrôleur. De par l'origine analytique de la théorie du contrôle, les outils appliqués pour obtenir et définir les lois de contrôle sont issus des approches comme la recherche opérationnelle, l'optimisation et la théorie de jeux.

Modélisation

Les modèles utilisés dans la théorie du contrôle, tels que les équations différentielles, sont analytiques. Selon la forme des équations, celles-ci peuvent être résolues mathématiquement (par exemple en cas de convergence des valeurs vers un point fixe), ou leur solution peut être calculée de manière approchée avec le calcul numérique. Cependant, une résolution n'est pas toujours possible ou alors elle fait appel à des hypothèses très simplificatrices.

A.2.4. Approche « equation-free »

L'approche equation-free est un paradigme pour le calcul multi-échelle et l'analyse assisté par ordinateur (Samaey, 2010). Il a été construit pour être utilisé quand l'évolution d'un système est observée à partir d'un niveau global (ou grossier) et qu'en même temps on dispose des modèles précis du système qui sont spécifiés au niveau local (ou fin). L'idée principale de l'approche est d'éviter la définition explicite d'équations au niveau global en les remplaçant par des courtes rafales de simulations de niveau local (Kevrekidis et al., 2003, 2004; Siettos et al., 2006; Siettos, 2011).

Nous retenons de cette approche les idées suivantes :

- Générer des informations sur l'état d'un système au niveau de description global est possible quand on dispose d'un modèle qui est donné dans un niveau de description local.
- Cela est fait à partir de la simulation du modèle au niveau local, mais cette simulation peut être très couteuse en ressources (temps de calcul notamment).
- Pour réduire le coût, un planificateur macroscopique (appelé « coarse time stepper » en anglais) décide quand utiliser la simulation et surtout comment traduire les résultats qui sont donnés en un niveau de description local au niveau de description global.

A.2.5. Applications du paradigme multi-agent au contrôle de systèmes complexes

Organic Computing

L'approche dite « Organic Computing » (appellée OC dans le reste du chapitre) est une initiative de recherche allemande portant sur l'étude de systèmes complexes technologiques (oc-website, 2012; Schmeck, 2005; Würtz, 2008). Cette approche propose un cadre d'étude complet pour étudier et créer des systèmes complexes technologiques. Le cadre définit une architecture pour guider le comportement de systèmes. L'approche OC est construite à partir de l'hypothèse suivante : les systèmes complexes technologiques ont des caractéristiques telles que l'émergence et l'auto-organisation, qui sont présentes aussi dans les systèmes vivants (organiques). Le contrôle de systèmes vivants est fait à partir de propriétés auto-*. Donc, la création et contrôle de systèmes complexes technologiques peuvent être faits en donnant à ces systèmes des propriétés auto-*, de l'apprentissage, de la robustesse et de l'adaptation (Mnif et al., 2007).

Pour ce faire, le système doit avoir des degrés de liberté adéquates. Un comportement émergent, désiré ou non désiré, peut être une conséquence de la conception. Par conséquent, un mécanisme de régulation est nécessaire pour permettre des réactions adéquates ainsi que pour contrôler le comportement global émergent (parfois inattendu) tout en favorisant l'auto-organisation. Le mécanisme de régulation proposé dans le cadre d'OC est une architecture générique « observateur / contrôleur » (Richter et al., 2006).

Architecture générique observateur / contrôleur

L'architecture est détaillée par ses créateurs dans (Richter et al., 2006; Müller-Schloer, 2004). Elle est définie comme un pattern abstrait, sous la forme d'une boucle de contrôle, comportant les éléments suivants :

Le système à construire. Appellé « SuOC » (pour System under Observation and Control, en anglais).

Un observateur. Chargé de récolter des informations du SuOC. Les informations récoltées sont transmises au contrôleur qui prendra les actions nécessaires pour influencer le SuOC. Le comportement de l'observateur est variable.

Contrôleur. Chargé de guider l'auto-organisation entre les éléments du système en interférant uniquement si nécessaire. Le contrôleur peut influencer le SuOC en intervenant sur :

- les règles de décision des participants du système
- la structure du système (communication entre les participants, nombre de participants)

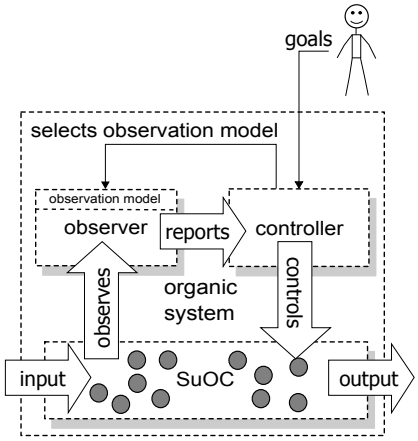


FIGURE A.4 – Architecture générique Observateur / Contrôleur.

- l'environnent (en considérant que le contrôleur pourra exercer une influence en changeant les données que les participants observent avec leurs capteurs)

La figure A.4 montre les différentes interactions des éléments de l'architecture de l'approche OC.

Contrôle dans l'informatique ubiquitaire

Il est communément admis dans le contexte de l'informatique ubiquitaire que l'un des défis clés de la conception est de savoir si un système conçu en spécifiant uniquement les interactions locales peut être contrôlé, ou s'il peut avoir un comportement cohérent au niveau global.

On va s'intéresser au point de vue « auto-organisation » inscrit dans le contexte de l'informatique ubiquitaire, car il est centré sur le paradigme multi-agent. La communauté d'ingénierie par auto-organisation est également intéressée pour conduire un système composé d'entités autonomes à présenter un comportement souhaité.

> [L'auto-organisation est un] processus par lequel un système change son organisation interne pour s'adapter aux changements de ses objectifs et de l'environnement, sans *contrôle externe explicite* (Di Marzo Serugendo et al., 2005).

(Brun et al., 2009; Edmonds, 2005) proposent d'utiliser des boucles de contrôle explicites, comme élément central dans l'ingénierie de logiciels auto-organisants. Le point de vue de l'ingénierie de logiciels considère les logiciels auto-organisants comme des systèmes qui ont des caractéristiques similaires à celles de systèmes complexes. Pour l'ingénierie de logiciels auto-organisants, l'autonomie des éléments d'un logiciel auto-organisant doit conduire à un comportement cohérent au niveau global. Pour donner une certitude sur la cohérence du comportement au niveau global, l'approche propose d'utiliser des méthodes expérimentales afin de caractériser le comportement global d'un système.

Nous retenons de l'approche auto-organisation qu'elle s'intéresse à la construction (par ingénierie) des systèmes qui présentent un comportement désiré comme un phénomène émergent. Toutefois, étant donnée la nature non linéaire de l'émergence dans de tels systèmes, le comportement du système n'est *a priori* pas garanti. C'est pourquoi l'approche auto-organisation propose d'utiliser des méthodes expérimentales dans le processus d'ingénierie.

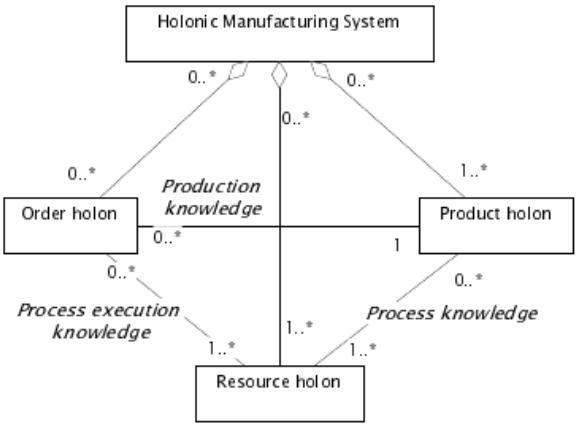


FIGURE A.5 – Holons de PROSA et ses relations. Reprise de (Van Brussel et al., 1998)

Prosa

PROSA est une architecture de référence pour la construction de systèmes de fabrication. L'architecture est basée sur le concept de « holon ». Le concept a été créé par (Koestler, 1989). L'idée est que chaque chose est non seulement un tout, mais fait également partie d'un plus grand tout, définissant ainsi un « hol-on ». Un groupe d'holons est appellé une holarchie.

HMS (pour Holonic manufacturing systems en anglais) est un projet industriel international concentré sur la standardisation, la recherche, le développement, l'implémentation et le support d'architectures et technologies pour des systèmes de fabrication répartis, intelligents et autonomes (Van Leeuwen and Norrie, 1997; Gruver et al., 2003).

PROSA est une architecture proposée dans le cadre de HMS. Dans la définition de PROSA (Van Brussel et al., 1998) nous retrouvons les holons suivants :

Produit. Le produit tangible qui est fabriqué. Il est responsable des différentes recettes et plans de fabrication.

Ressource. Les processus physiques ou ressources de transportation ainsi que ses systèmes de contrôle et ses interactions avec les opérateurs humains.

Commande. Les requis d'une commande, incluant des informations comme les quantités de produit, dates de livraison, coûts et priorités.

Staff. Élément de support optionnel qui coordonne les interactions entre les holons et qui garantit que les objectifs au niveau global soient atteints.

La figure A.5 illustre les interactions typiques des holons basiques de PROSA. Dans PROSA, l'agrégation d'un grande nombre d'entités est sensée produire un comportement complexe qui est difficile d'être analysé ou prédit. La solution proposée dans PROSA est d'organiser les entités en une hiérarchie. Une holarchie typique de PROSA est illustrée dans la figure A.6.

PolyAgent

Dans la simulation multi-agent traditionnelle, l'entité fondamentale est l'agent. Ce dernier représente une entité discrète du système modélisé et suivra une seule trajectoire par simulation. De ce fait, les différentes trajectoires accessibles à l'entité dans l'évolution du système modélisé ne sont pas explorées.

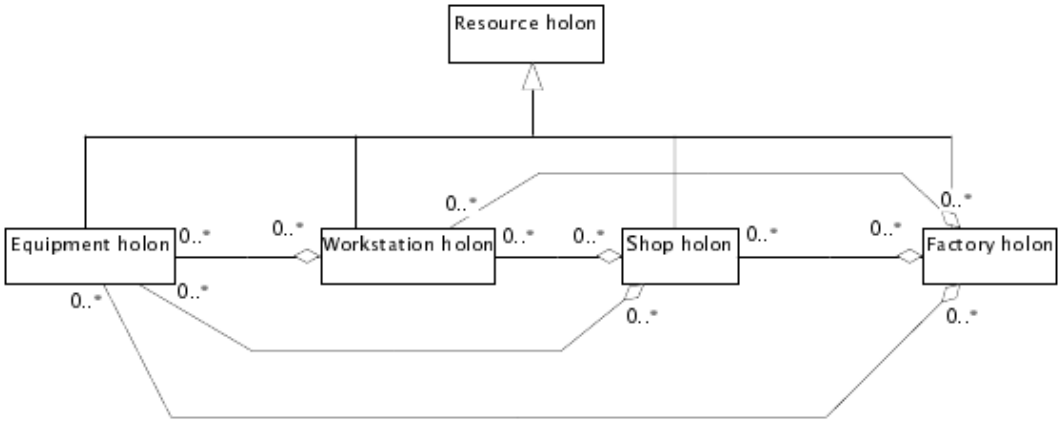


FIGURE A.6 – Agrégation de holons avec type spécialisé « ressource ». Reprise de (Van Brussel et al., 1998)

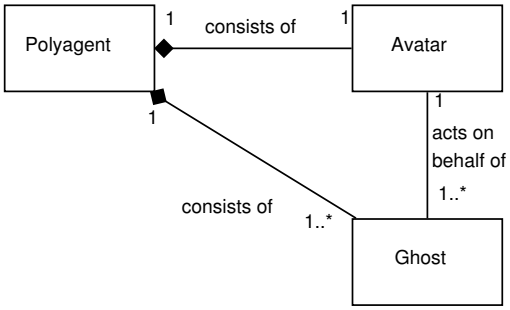


FIGURE A.7 – Un polyagent représente une entité du domaine avec un avatar et plusieurs fantômes.

Parunak and Brueckner proposent comme solution au problème de l'exploration une nouvelle entité de modélisation « polyagent ». Une entité du domaine modélisé est représentée par plusieurs agents dans l'approche polyagent. Cependant, dans la simulation multi-agent traditionnelle un agent représente une entité du domaine modélisé. Un avatar persistant est chargé de la correspondance entre le domaine modélisé et le polyagent ainsi que du contrôle d'un essaim de fantômes temporaires. La figure A.7 illustre à l'aide d'UML, les relations entre polyagents, avatars et fantômes. L'activité principale des fantômes est l'exploration de différents comportements possibles pour l'entité modélisée. Chaque fantôme interagit avec d'autres fantômes à travers des phéromones numériques. Un exemple de l'application de fantômes est illustré dans la figure A.8

Morphologie

La « morphologie » telle qu'elle est proposée par (Campagne et al., 2005; Campagne, 2005; Camus and Cardon, 2006) est une approche pour contrôler un système multi-agent en se basant sur la forme globale du système. Dans cette approche, le système cible est défini en fonction d'objectifs et de senseurs. Chaque objectif a une forme et une valeur spécifiques pour chaque senseur associé à un objectif. Au niveau global, le système est caractérisé par une forme qui résulte de l'agrégation des différents objectifs. Le but de cette approche est de diriger le comportement du système cible vers une forme ou une figure associée à un objectif.

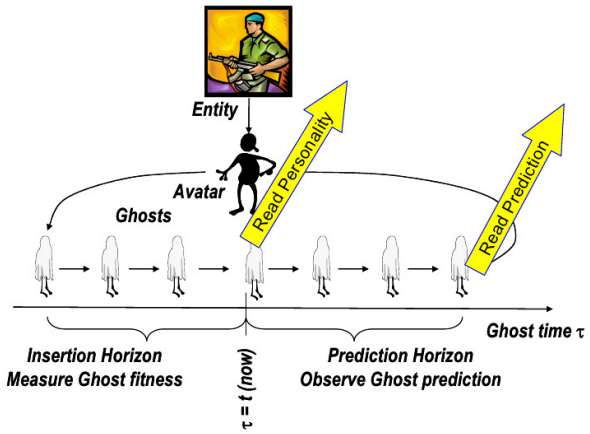


FIGURE A.8 – Dans l'approche polyagent, chaque avatar crée un flux de fantômes qui va explorer l'espace de la personnalité de l'entité qu'il représente. Les fantômes évoluent dans le sens opposé au comportement récemment observé et seuls les fantômes les plus aptes pourront courir vers le futur pour produire des prédictions. Reprise de (Parunak and Brueckner, 2006a)

Ingénierie Émergente

L'ingénierie émergente est un cadre méthodologique pour déployer des systèmes réseaux à grande échelle. Le cadre est exemplifié dans le contexte de la sécurité auto-organisée (Doursat and Ulieru, 2008; Ulieru and Doursat, 2011). Dans l'ingénierie émergente il existe un modèle abstrait d'un réseau qui peut être programmé pour s'auto-construire. Le modèle s'appuie sur la définition d'entités basiques du système ainsi que sur la définition des mécanismes qui permettent la construction d'éléments structurels fiables. Le contrôle est réparti parmi les entités du système. Dans le cadre de l'ingénierie émergente le génotype d'un agent correspond aux règles de comportement de l'agent. Le phénotype est le comportement global résultant du comportement collectif.

On compte deux principes essentiels dans cette approche :

- La création « bottom-up » des entités du système. Les entités ont des micro-contrôleurs à l'intérieur, au lieu d'une spécification de quoi faire à chaque moment. Chaque entité a des règles de comportement qui peuvent être de l'un des deux types suivants :
  - Rétro-action positive. Ce type de règles amplifie les petites fluctuations au niveau local et provoque dans le système une tendance à créer des nouvelles structures macroscopiques.
  - Rétro-action négative. Ce type de règles corrige la réponse de l'agent et ajuste son comportement plus finement.
- Co-évolution du système avec la dynamique environnementale. La co-évolution est possible grâce à la spécification de la manière dont le génotype peut varier ainsi que grâce à la spécification de la façon dont le phénotype peut être sélectionné.

Contrôle avec techniques d'apprentissage par renforcement

Cette approche s'adresse au contrôle d'un système multi-agent réactif avec des techniques d'apprentissage par renforcement. L'apprentissage par renforcement est utilisé pour apprendre

le comportement global du système cible et pour sélectionner les actions de contrôle à exécuter (Klein et al., 2005, 2008; Klein, 2009). Dans cette approche, les différents états possibles du système cible sont identifiés au niveau global. Les actions considérées dans l'approche peuvent être locales ou globales. Le type d'actions dépend des capacités du contrôleur.

L'approche se décompose en trois phases.

1. Caractériser le comportement global du système ainsi que définir la façon automatique de mesurer le comportement.

2. Sélectionner les meilleures actions de contrôle.

3. Déterminer une politique de contrôle qui indique les actions de contrôle à appliquer. La décision de quelle action de contrôle appliquer est prise de façon dynamique : elle dépend de l'état global du système cible.

A.2.6. Synthèse

Nous retenons des différentes approches présentées une série de caractéristiques que nous présentons groupées selon : la manière d'évaluer l'état du système cible, la façon de prendre en considération les niveaux local et global et, l'origine de mécanismes de contrôle.

Evaluation de l'état du système. Nous avons vu parmi les différents applications du paradigme multi-agent au problème du contrôle différentes manières d'explorer la dynamique du système cible. Pour (Klein, 2009) par exemple, il est question d'apprendre la dynamique avec des outils d'apprentisage par renforcement. Pour les approches PROSA et PolyAgent, il est question de explorer les différents futures possibles en même temps que le système est en cours d'exécution. Avec ce type de démarche la sensibilité aux conditions initiales et la non-linéarité des dynamiques de systèmes complexes sont directement traitées.

Niveaux local et global. Les différentes approches présentées s'adressent à la difficulté de la modélisation du challenge de contrôle de systèmes complexes avec le paradigme multi-agent. Cette solution permet de prendre en considération en même temps le niveau local et le niveau global dans la modélisation du système cible.

Origine des mécanismes de contrôle. Nous pouvons identifier deux origines de mécanismes de contrôle par rapport au moment de leur conception.

- L'origine est endogène si les mécanismes ont été construits avec le système cible. Ce type de mécanismes fait l'hypothèse de pouvoir modifier le comportement interne des entités du système cible. Une conséquence de cette hypothèse est que dans les systèmes où pour de raisons légales, techniques ou étiques il n'est pas possible de changer le comportement des entités du système il n'est pas possible d'implanter ce type de mécanismes de contrôle. De ce fait, ce type de mécanismes est plus adapté à des systèmes complexes artificiels.
- L'origine est exogène si les mécanismes ont été construits en dehors de la conception du système cible. Ce type de mécanismes font l'hypothèse que la modification directe des entités du système n'est que très rarement possible. De ce fait, nous considérons ce type de mécanismes plus adaptés à être appliqués sur des systèmes complexes aussi bien naturels qu'artificiels.

Certains approches ont le point de vue de l'ingénierie, concernant l'accès au comportement interne des entités du système cible. Notamment l'informatique ubiquitaire et PROSA. Nous identifions deux hypothèses fortes liées à ce point de vue. La première est qu'un accès non-restreint aux informations du système cible est possible. La deuxième est que les mécanismes de contrôle sont endogènes.

Directions à explorer

Modélisation et actions locales avec des effets globaux. Nous allons nous concentrer sur la modélisation equation-free de systèmes complexes, plus particulièrement avec des modèles microscopiques de type multi-agent. De plus, cette approche nous permettra de prendre en compte le niveau local et global de la dynamique du système cible.

Systèmes pré-existants. Nous allons nous concentrer sur la piste de création de mécanismes de contrôle exogènes. Ceci nous permettra de pouvoir implémenter les mécanismes depuis l'extérieur d'un système cible.

Autonomie des entités. Nous allons nous concentrer sur des mécanismes de contrôle adaptatif pour prendre en compte les structures dynamiques et le changements de comportements internes des entités.

A.3. Proposition

A.3.1. Principes

L'architecture de contrôle que nous proposons est basée sur trois principes : une boucle de contrôle de rétro-action explicite, un mode d'opération exogène et une estimation de l'état du système cible obtenu avec l'approche equation-free.

Boucle de contrôle de rétro-action explicite. Nous empruntons à la notion de boucle de contrôle de rétro-action les concepts suivants :

- Nous avons deux systèmes : C - l'architecture de contrôle et T - le système cible.
- C va excercer une influence sur T pour faire en sorte que la sortie y de T soit proche de la référence r. La sortie de T sera utilisé par C comme entrée et au même temps, la sortie de C deviendra l'entrée de T.

Une première représentation schématique de la boucle de contrôle est illustré dans la figure A.9.

Implémentation Exogène. Le système C est un système indépendant de T. De ce fait, si C n'est pas en exécution, T peut continuer à être opérationnel.

Modélisation equation-free. Les modèles utilisés pour estimer l'état de T, ainsi que les effets des actions de contrôle sont du type equation-free, et en particulier, basés multi-agents. Plus loin dans ce chapitre nous identifions les fonctions du planificateur macroscopique dans notre proposition.

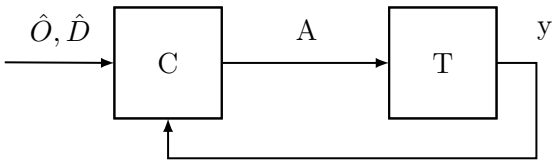


FIGURE A.9 – Schéma de la boucle de retro-action formé avec les entrées et le sorties de l'architecture et le système cible.

A.3.2. Vue Générale

Nous utilisons le langage SysML pour présenter dans cette section une vue de type boîte noire de l'architecture. Nous nous servons de SysML car il permet de définir l'architecture de façon non-ambiguë.

Notation SysML utilisée

SysML est un langage de modèlisation spécifiquement conçu pour modéliser des systèmes dynamiques (SysML Specification, 2012). Il est un raffinement du langage de modélisation de logiciels UML. Dans notre définition nous allons utiliser les diagrammes SysML suivants :

Diagramme de définition de bloc. Les blocs sont l'unité modulaire à la base de la description d'un système. Chaque bloc est composé d'un ensemble de propriétés qui caractérisent un élément d'intérêt. Si les caractéristiques sont structurelles elles sont appelés « propriétés ». Si les caractéristiques sont fonctionnelles, elles sont appelés « opérations ». L'opération principale d'un bloc est appelé « comportement classificateur ».

Diagramme interne de bloc. Ce type de diagramme explique la structure interne d'un bloc avec des connexions entre les propriétés.

Diagramme d'activité. Ce type de diagramme spécifie l'ordre dans lequel les actions à l'intérieur d'une opération sont exécutées.

Diagramme de définition de bloc de l'architecture

L'architecture est représentée au plus haut niveau par le bloc « Architecture ». Il est composé des blocs suivants :

- Observation du système cible, fournit à l'architecture informations sur le système cible.
- Estimation de l'état du système cible, exécute la simulation de(s) modèle(s) utilisé(s) pour estimer l'état du système cible.
- Simulation d'actions de contrôle, exécute la simulation de(s) modèle(s) utilisé(s) pour estimer les possibles effets des actions de contrôle.
- Appliquer les actions de contrôle.

Le diagramme A.10 montre la hiérarchie des blocs principaux de l'architecture.

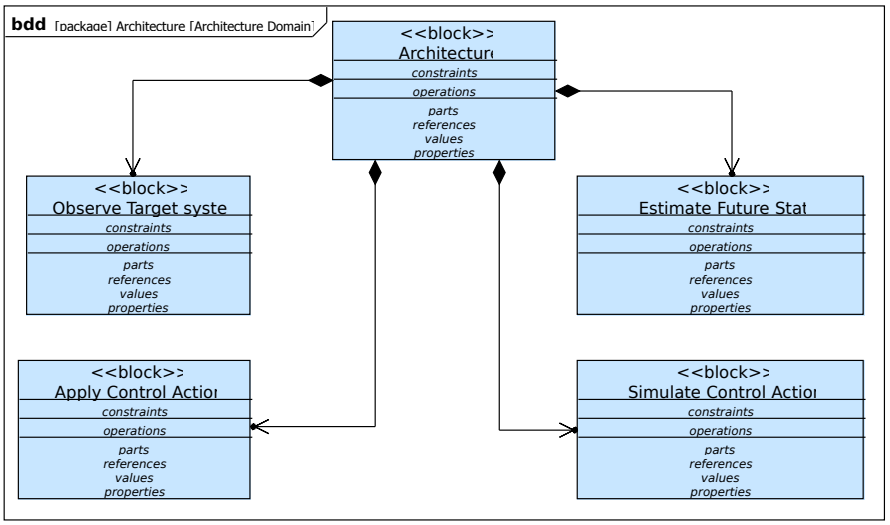


FIGURE A.10 – Diagramme de définition de blocs de l'architecture.

Entrées et sorties de la boucle de contrôle pour C

Entrées.
- Nous allons appeller $\hat{O} = \{o_1, o_2, \ldots, o_n\}$ les observations du système T.
- La réference r dans la boucle de contrôle sera $\hat{D} = \{d_{t_a}, d_{t_b}, d_{t_c}\}$ dans l'architecture. t_a, t_b, t_c représentent des horizons de temps. Ainsi, d_{t_a} est l'objectif de contrôle attendu pour l'horizon de temps t_a.

Sorties.
- La sortie de C sont les actions de contrôle $A = \{a_1, a_2, \ldots, a_n\}$.

A.3.3. Flux d'exécution de l'architecture

Le comportement clasificateur de l'architecture (identifié dans les diagrammes comme « Main Control Loop ») est l'endroit où a lieu le flux d'exécution principal. La première activité exécutée par l'architecture est d'estimer l'état de T à un horizon de temps correspondant à l'objectif de contrôle courant. Après, l'état estimé est comparé avec l'objectif de contrôle pour décider s'il faut appliquer des actions de contrôle. S'il est nécessaire d'appliquer des actions de contrôle, les actions seront déterminées par le bloc « simulation d'actions de contrôle » et appliquées par le bloc « appliquer actions de contrôle ». Si par contre d'actions de contrôle ne sont pas nécessaires, la boucle de contrôle redemarre. Cet flux d'exécution est illustré dans les diagrammes A.11 et A.12.

A.3.4. Des principes à l'implémentation

L'architecture que nous proposons est basée sur l'hypothèse de l'existence de certains éléments que nous explicitons :

- **Modèles multi-agents.** Les modèles que nous considérons comme aptes pour être utilisés dans l'architectures doivent être validés, prêts à être simulés. Par rapport à la hiérarchie de spécification de systèmes présenté en A.2.1.0 nous considérons que les modèles doivent être au niveau de composants couplés (niveau 4 de la hiérarchie). Nous supposons que la durée de simulation est assez courte pour utiliser les résultats dans contexte pertinent. Cela signifie que les résultats devraient être disponibles avant un changement d'état.

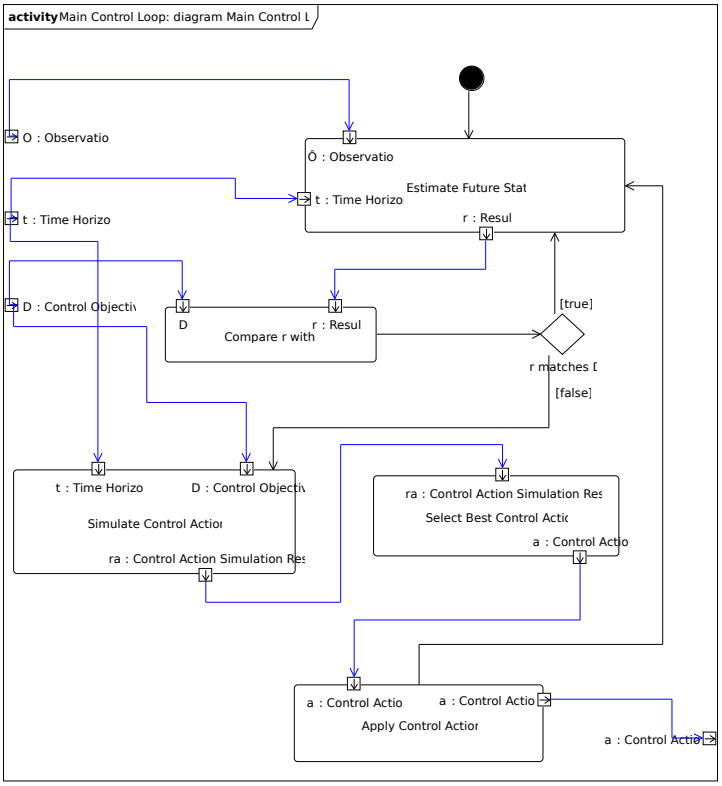


FIGURE A.11 – Flux d'entités entre les différents blocs de l'architecture.

- **Actions de contrôle.** Les actions de contrôle doivent être complétement spécifiées et disponibles. Elles doivent être définies dans le même niveau de description que les modèles qui devront les tester et avoir un effet validé dans le système cible.

- **Système T pré-existant.** Des moyens pour pouvoir observer le système cible doivent être disponibles à l'architecture.

- **Objectif de contrôle.** L'objectif de contrôle doit être défini à l'avance ainsi que les moyens pour comparer les résultats de simulations avec celui-ci.

Jusqu'ici nous avons décrit la suite d'opérations exécutées dans l'architecture pour obtenir le contrôle d'un système cible conformément à la supposition que le contrôle est obtenu. Considérons maintenant la situation où la comparaison entre le comportement estimé (avec ou sans actions de contrôle) et le comportement observé diffèrent significativement des observations. Si nous avons de mauvaises estimations cela signifie que les résultats des simulations ne sont pas corrects.

- Une supposition importante de notre proposition est que les modèles sont valables au niveau de composant de couplage. D'autre part, dans le contexte de systèmes complexes, nous supposons que les structures des systèmes cibles aussi bien que le comportement des entités sont dynamiques. En conséquence de ceci, la validité des modèles peut aussi évoluer.

- De plus, les modèles ont des simplifications inhérentes et dans le contexte de systèmes complexes, l'hypothèse qui supporte les simplifications peut aussi changer au fil du temps.

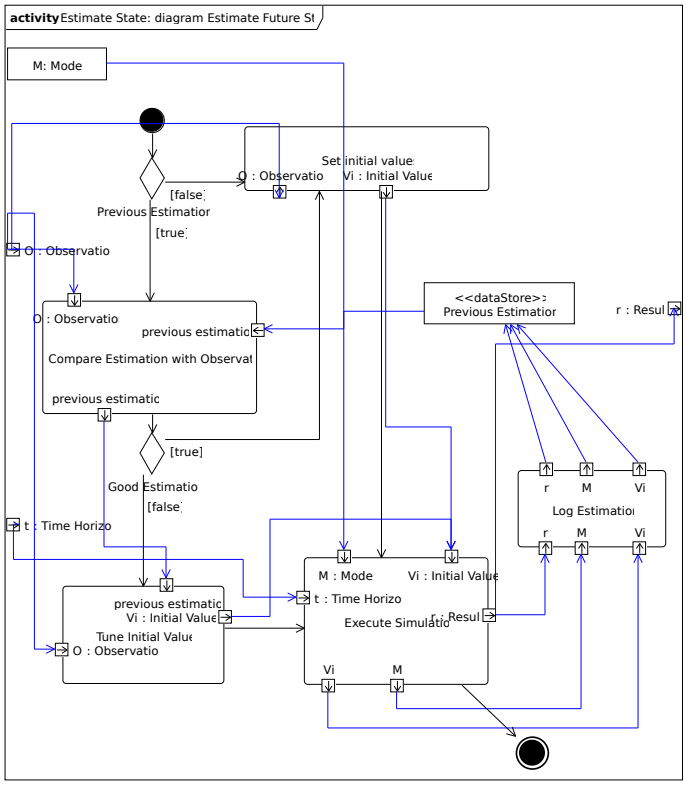


FIGURE A.12 – Diagramme d'activité du comportement classificateur du bloc « estimer l'état du système cible ».

- Notre architecture est exogène au système cible. Il est raisonnable de considérer que nous pouvons observer les interactions différentes entre les entités. Cependant, il pourrait être plus difficile d'observer des aspects liés au couplage des entités avec l'environnement. Le couplage se réfère au régime de mise à jour et l'horizon de perception des agents dans un modèle de multi-agent.

- Du point de vue de systèmes complexes, nous pouvons considérer que les actions de contrôle définies au moment de mettre en œuvre l'architecture étaient valables, mais que dans le temps elles sont devenus moins efficaces.

Nous considérons que pour traiter ces aspects il est nécessaire d'explorer les résultats possibles des multiples simulations. Des simulations différentes pourraient rapporter des résultats différents pour des aspects de couplage différents ou pour des structures différentes du système cible. Ceci est un choix à faire en mettant en œuvre l'architecture.

Dans le tableau A.2 nous présentons une liste de choix à faire lors de la mise en œuvre de l'architecture. Ces choix doivent être pris comme une liste de vérifications à faire avant le déploiement complet de l'architecture.

Choix	Où l'implementer	Exemple
Quoi observer dans le système cible	Bloc Observer système cible	Valeurs macroscopiques, valeurs microscopiques
Comment traduire les observations faites à des valeurs pour le paramètres des modèles multi-agent.	Boucle de contrôle principale.	
Comment calibrer les paramètres des modèles multi-agent.	Blocs de simulation	Algorithmes génétiques, méthodes statistiques, régressions, algorithmes évolutionnaires.
Comment comparer l'état global observé avec les résultats des simulations.	Boucle de contrôle principale	

TABLE A.2 – Choix à faire dans l'architecture pour améliorer sa performance.

A.4. Preuve de concept

A.4.1. Introduction

L'objectif de cette implémentation est d'évaluer la faisabilité de notre proposition. Cette implémentation a pour domaine d'application les réseaux d'échanges de fichiers pair à pair (p2p) et un phénomène collectif présent dans ces réseaux : le « free-riding ».

A.4.2. Les réseaux pair-à-pair comme systèmes complexes

Nous nous intéressons aux réseaux p2p tels que BitTorrent (Cohen, 2003), Kademlia (Maymounkov and Mazières, 2002) ou eDonkey (Heckmann et al., 2004) car ils ont des caractéristiques propres aux systèmes complexes (Shahabi and Banaei-Kashani, 2007) : les réseaux sont composés d'entités autonomes, de millions d'utilisateurs et finalement, à cause de la nature auto-organisée et ouverte de ces réseaux, ils exhibent des comportements non-linaires et enfin il est difficile d'obtenir de l'information sur ces réseaux (Aidouni et al., 2009; Stutzbach and Rejaie, 2005).

Le free-riding dans les réseaux pair-à-pair

Le *free-riding* dans un réseau p2p de partage de fichiers se caractérise par le téléchargement de fichiers sans en partager en retour (Androutsellis-Theotokis and Spinellis, 2004).

Le *free-riding* a déjà fait l'objet de recherches (Hardin, 2008) sous forme de dilemmes sociaux comme *La tragédie des communs* (Glance and Huberman, 1994; Hardin, 1968). Ramaswamy and Liu (2003) et Ge et al. (2003) sont parmi les premiers à s'être intéressés aux effets du *free-riding* sur la performance du réseau bien que l'existence du phénomène ait été identifiée pour la première fois en 2000 par Adar and Huberman. Il existe des observations empiriques qui suggèrent que le free-riding est un phénomène très répandu dans ce type de réseaux (Saroiu et al., 2002; Makosiej et al., 2004; Yang et al., 2005; Zghaibeh and Harmantzis, 2008). La présence importante de free-riders sur un réseau peut dégrader sa performance au point de le rendre inutilisable, d'où l'intérêt de combattre son apparition.

Système cible

Comme il est difficile (ne serait-ce que pour des questions légales) de faire des expérimentations directement dans les réseaux pair-à-pair réels, nous simulons notre système cible ainsi que le phénomène du free-riding.

Aussi, utiliser un simulateur nous permet de mesurer chaque paramètre du système cible et évaluer de la performance de l'architecture. Un autre avantage est la possibilité de pouvoir reproduire les expériences, impossibles autrement dans un réseau pair à pair réel.

Le simulateur choisi est « PeerSim » , un simulateur de réseaux pair-à-pair proposé par les projets européens « Bison » et « DELIS » dont la maturité a orienté notre choix.

Le système cible utilisé dans les expériences est caracterisé par quatre paramètres principaux :

1. N La quantité de pairs dans le réseau.
2. Le comportement interne de pairs.
3. Les pairs sont organisés dans un réseau.
4. x_{init} La proportion initiale des pairs qui partagent dans le réseau.

Comportement des pairs dans le réseau

Touts les pairs dans le système ont un comportement de type « marché libre » inspiré de (Feldman et al., 2006). Un pair ou utilisateur (i) est caractérisé par sa générosité (θ_i), et par un état (*partage*) booléen indiquant s'il partage ou non. Un utilisateur partagera si sa générosité θ_i est supérieure au coût de contribution, sachant que ce dernier correspond à $\frac{1}{x}$ (intuitivement plus il y de gens qui partagent, moins cela coute de partager). Il existent d'autres modèles de comportement : :(Vassilakis and Vassalos, 2007; Krishnan et al., 2004; Jian and MacKie-Mason, 2008) mais nous avons choisi ce modèle car il est rapidement implémentable sous le paradigme multi-agent.

En supposant i) une répartition particulière de la générosité dans le réseau (la distribution de la valeur θ est uniforme entre 0 et un maximum θ_{max}), ii) une décision synchrone des agents et iii) une perception complète du réseau ; il est possible de résoudre exactement l'équation correspondante. Nous renvoyons le lecteur à Feldman et al. (2006) pour les détails de cette résolution.

Cette résolution analytique donne une dynamique de point fixe avec deux points particuliers : x_1 et x_2. Le niveau de contribution final x_f du système avec un niveau de contribution initial x_{init} est donné par :

$$x_f = \begin{cases} x_1 & \text{si } x_{init} \geq x_2, \\ 0 & \text{si } x_{init} < x_2. \end{cases}$$

La figure A.13 illustre le comportement attendu du niveau de contribution tel que prédit par le modèle analytique de (Feldman et al., 2006).

A.4.3. Scénario expérimental

Configuration du système cible

N est établi à l'avance et constant tout au long de la simulation.

Comportement de pairs. Tous les pairs dans le système ont un comportement de type « marché libre » décrit précédemment. Au long des expériences montrées dans ce chapitre nous avons choisi d'utiliser une valeur arbitraire de $\theta_{max} = 10$.

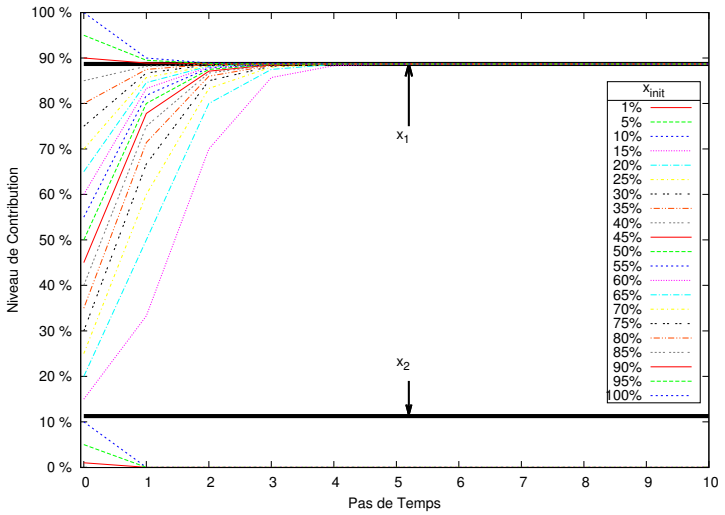


FIGURE A.13 – comportement attendu du niveau de contribution du modèle analytique avec $\theta_{max} = 10$ et différentes valeurs de x_{init}.

Réseau Les utilisateurs sont organisés dans un réseau de type *Small-World* suivant l'algorithme décrit par Watts and Strogatz (1998). L'algorithme a deux paramètres : p et k. Le paramètre p permet d'établir le degré aléatoire du réseau : si $p = 0$ le réseau est un treillis circulaire, si $p = 1$, le réseau est un réseau complétement aléatoire. Les valeurs intermédiaires permettent d'avoir de réseau de type small-world. Le paramètre k permet d'établir la quantité de voisins de chaque nœud dans le réseau.

d le degré de profondeur des perceptions des pairs dans le graphe. Les pairs dans le système cible seront restreints à observer uniquement ses voisins de profondeur d.

x_{init} L'état initial partager ou pas partager des pairs est établi au début de l'exécution du système cible de manière aléatoire.

Objectif de contrôle

L'objectif est d'éviter que le système cible soit dans un état où la majorité de pairs ne partage pas. Quantitativement, nous traduisons l'objectif comme : faire que le niveau de contribution finale x_f soit égal ou supérieur à 50% à la fin de la simulation. Pour faire que le système cible évolue vers un état où la majorité de pairs ne partagent pas (x_f inférieur à 50%) nous avons donné au paramètre x_{init} une valeur inférieur à 2% (cf. tableau A.4).

A.4.4. Implémentation de l'architecture

Les différents blocs de l'architecture sont implémentés comme suit.

1. Observation du système cible. *Le système est échantillonné de façon aléatoire.*

2. Estimation de l'état du système cible. *Le modèle multi-agent « topologique » est initialisé avec les observations et simulé pour obtenir une estimation du niveau de contribution.*

3. Comparaison entre objectif de contrôle et état. *Si le niveau de contribution obtenu dans le bloc précédent est inférieur à 50%, des actions de contrôle sont simulées. Sinon, la boucle recommence.*

4. Simulation d'actions de contrôle. *Le même modèle utilisé pour estimer l'état du système cible est utilisé pour simuler l'application de différentes actions de contrôle.*

5. Sélection de l'action de contrôle. *L'action de contrôle qui dans la simulation l'augmentation la plus importante du niveau de contribution est sélectionnée.*

6. Application des actions de contrôle. *Les pairs concernés par l'action de contrôle sélectionnée sont détournés directement dans le système cible.*

Les différentes entrées et sorties des blocs de l'architecture implémentée sont présentées dans le tableau A.3.

- **Observation de T.** Un pair est sélectionné de façon aléatoire dans le système cible. Son état (partage ou partage pas), sa générosité (θ_n) ainsi que la liste de ses voisins dans le réseau sont observés. Les voisins sont mis dans une liste de pairs à explorer. Le processus continue jusqu'à avoir observé q pairs.

- **Modélisation.** Le modèle utilisé pour simuler l'évolution du système est un modèle construit avec les mêmes hypothèses comportementales que le système cible. Nous considérons le modèle comme validé d'une part parce que les agents dans le modèle ont un comportement identique aux agents du système cible et d'autre part car nous avons déjà fait une étude dans (Navarrete Gutiérrez et al., 2011, 2010). Les détails du comportement sont présentés dans l'algorithme 2.

- **Actions de contrôle.** L'action de contrôle que nous utilisons consiste à détourner la perception d'un pair. Un pair détourné va percevoir un environnent où tous ses voisins partagent et sera donc amené à partager lui même.

- **Critère de sélection d'action de contrôle.** L'action de contrôle à appliquer sera celle qui donnera comme résultat dans les simulations l'augmentation du niveau de contribution le plus important.

Mesures

Dans les expériences nous avons mesuré le niveau de contribution final x_f après 200 pas de temps d'exécution du système cible. L'architecture est en exécution parallèle au système cible dès le premier pas de temps. Pour chaque expérience nous avons mesuré le nombre de fois où le niveau de contribution x_f a atteint un niveau de contribution égal ou supérieur à 50%.

Les valeurs données aux paramètres du système cible et de l'architecture sont résumées dans le tableau A.4.

A.4.5. Expériences

Nous avons effectué des expériences groupées selon deux scénarios.

1. Le système cible est configuré avec $N = 1000$ pairs.

Paramètre de l'architecture	Notation	Valeurs dans les expériences
Information	$\hat{O} = \{o_1, o_2, o_3, o_4\}$	$o_1 = \{q$ Nœuds du réseau de $T\}$ $o_2 = \{$Liens réliant les nœuds de$o_1\}$ $o_3 = \{\theta_q$ Générosité de chacun des q nœuds de $o_1\}$ $o_4 = \{S_q$ État de partage de chacun des q nœuds de$o_1\}$
Actions de Contrôle	$A = \{a_1, a_2, \ldots, a_{10}\}$	$a_n = \{$Détourner n nœuds pour leur faire croir que tous ses voisins partagent$\}$
Modèles	$\{m_1\}$	Modèle topologique (cf. A.4.2)
Paramètres du modèle	$p_{1m_1}, p_{2m_1}, p_{3m_1}$	$p_{1m_1} = \{q$ nœuds dans le modèle$\}$ $p_{2m_1} = \{$Graphe avec$V = o_1$ et $E = o_2\}$ $p_{3m_1} = \{$Profondeur$d = 1\}$
Résultas	$\hat{R} = \{r_{m_1}\}$	$r_{m_1} = \{x_f$ pour$t_a\}$
Horizons de temps	t_a	$t_a = 1$ pas de temps dans le future
Objectif de contrôle	$\hat{\mathrm{D}} = \{d_{t_a}\}$	$d_{t_a} = x_f \geq 50\%$

TABLE A.3 – Entrées et sorties de l'architecture dans les expériences.

Nom	**Change dans les expériences**	
	Oui	Non
N	$1000, 10000$	
$\theta \sim U(0, \theta_{max})$		$\theta_{max} = 10$
p	0, 0.25, 0.50, 0.75, 1	
d		1
k		10
x_{init}		1.8%
q		6
nombre maximal de pairs à détourner		10
stratégie pour choisir les pairs à détourner	aléatoire, betweenness, hits	

TABLE A.4 – Valeurs données au paramètres du système cible (en couleur ■) et de l'architecture (en couleur ■) dans les expériences.

Algorithme 1: Fonction de décision de l'agent

si *générosité* $\geq$ *coût* **alors**
  partager $\leftarrow$ Vrai;
sinon
  partager $\leftarrow$ Faux;
fin

Algorithme 2: Modèle topologique

pour tous les *pasDeTemps* **faire**
  pour tous les *agent* **faire**
    $x \leftarrow$ calculerContributionLocale(profondeur d);
    coût $\leftarrow \frac{1}{x}$;
    agent.decider();
  fin
fin

2. Le système cible est configuré avec $N = 10000$ pairs.

Pour chaque scénario, nous avons défini cinq familles de réseaux de voisinage pour les pairs. Chaque famille correspond à une valeur du paramètre $p = 0, 0.25, 0.50, 0.75, 1$. Pour les deux scénarios nous avons généré 1000 instances du système cible par famille de réseau. Pour chaque scénario, nous avons testé trois stratégies différentes pour sélectionner les pairs à détourner. La première stratégie consiste à sélectionner les pairs à détourner de façon aléatoire. Les deux autres choisissent le pair à détourner en fonction d'une mesure d'importance du nœud dans le sous-réseau obtenu dans l'observation. Les algorithmes pour calculer l'importance retenus sont : betweenes (Steen, 2010) et HITS (Kleinberg, 1999).

Comportement nominal du système cible

Dans les familles de réseaux $p = 0$ et $p = 1$, le système cible (sans action de contrôle) a évolué systématiquement vers un état ou x_f est inférieur à 50%, à l'exception d'une exécution sur 1000 du scénario avec $N = 1000$. Pour la famille $p = 0.50$, l'état final du système cible est dans 95% des exécutions inférieur à 50% pour le scénario $N = 1000$. Pour la famille $p = 0.25$ avec le scénario $N = 1000$, le niveau de contribution final été dans 53% des exécutions inférieur à 50%. Cependant, pour le scénario avec $N = 10000$ le niveau de contribution final été environ 80% des éxecutions inférieur à 50%. Le tableau A.5 présente un résumé du comportement nominal pour chaque scénario.

A.4.6. Résultats

Les résultats sont présentés dans le tableau A.6.

Par famille de réseau. Nous constatons que pour la famille de réseaux de type « Small-World » ($p = 0.25$) nous avons eu le plus de succès à contrôler le système, aussi bien pour le scénario avec $N = 1000$ que pour celui avec $N = 10000$. Ceci constitue des bonnes nouvelles si l'on considère qu'il y a des études empiriques qui montrent que certaines réseaux pair à pair ont ce type de structure (Jovanović et al., 2001; Stutzbach et al., 2008), ainsi que d'autres

	$x_f \geq 50\%$				
	$p = 0$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 1$
$N = 1000$	1	468	41	0	0
$N = 10000$	0	195	0	0	0

TABLE A.5 – Récapitulatif du comportement nominal du système cible. Nombre d'expériences (sur 1000) où le niveau de contribution x_f du système cible a atteint une valeur égal ou supérieure à 50%.

	$x_f \geq 50\%$				
	$p = 0$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 1$
$N = 1000$					
random	8	955	688	13	0
betweenness	8	962	653	4	0
hits	9	917	641	4	0
$N = 10000$					
random	0	578	0	0	0
betweenness	0	570	0	0	0
hits	0	429	0	0	0

TABLE A.6 – Résumé des résultats. Nombre d'exécutions (sur 1000) où le niveau de contribution final x_f a été supérieur ou égal à 50% pour chaque familles de réseaux et pour chaque scénario.

réseaux (Watts, 1999; Jovanović et al., 2001; Stutzbach et al., 2008; Buchanan, 2002; Park et al., 2007; Bassett and Bullmore, 2006; Wang et al., 2009). Par contre nous constatons que pour les autres familles, même si le contrôle a eu lieu dans certains cas, le nombre d'expériences dans lesquelles nous avons eu succès n'est pas suffisant pour considérer que c'est systématique.

Par stratégie de sélection de pairs à détourner. Nous constatons que la stratégie aléatoire est légèrement plus efficace.

A.4.7. Discussion

Les détails sur comment implémenter l'action de contrôle que nous avons testé sont d'une part dépendants de l'application (type de réseau, protocole, version logiciel du client pour se connecter au réseau). D'autre part, l'intérêt de nos expériences est d'évaluer notre architecture et donc les détails techniques pour implémenter les actions de contrôle sont hors de la porté de ce travail de thèse.

Nous considérons que le modèle multi-agent utilisé est valide car il correspond exactement au comportement programmé dans les pairs du simulateur. Cela reste une simplification que nous avons choisi d'avoir pour pouvoir se concentrer sur l'évaluation de l'architecture. Cependant, à notre connaissance, il n'y pas de modèles comportementaux des utilisateurs de réseaux pair à pair amplement considérés valables.

Dans les cas ou nous avons réussi à contrôler le système cible, nous l'avons fait avec très peu de ressources. Moins de 10% des nœuds ont été échantillonnés pour estimer l'état du système pour le scénario $N = 1000$ et moins de 1% pour le scénario $N = 10000$. Dans le deux scénarios, nous avons détournés moins de 1% et 0.1% respectivement, de nœuds du réseau.

A.4.8. Conclusions

Nous avons conduit une série d'expériences qui nous permettent de considérer que l'implémentation de l'architecture a rempli son objectif, au moins pour une famille du système cible (celle avec le réseau de type « Small-world ») dans les deux scénarios.

Par contre, la performance démontrée dans les autres familles n'est pas satisfaisante.

En revenant sur les tableau A.2, où nous évoquions différents aspects à tenir en compte pour améliorer la performance de l'architecture, nous identifions les pistes suivantes à explorer :

La qualité des observations n'est pas satisfaisante. La question est de savoir si d'autres manières d'observer le système cible ou bien la même manière mais avec plus de données aurait un impact sur la qualité des prédictions de l'évolution de l'état du système cible.

Les actions de contrôle ne sont pas correctes. Sous l'hypothèse d'avoir des estimations de l'évolution du système cible parfaites, on voudrait savoir si les actions de contrôle considérées jusqu'à présent sont les plus efficaces ou si elles sont correctement appliquées.

Les stratégies utilisés dans les actions de contrôle ont eu un impact équivalent dans la performance de l'architecture. Il est question ici de savoir si avec plus d'information la performance des actions de contrôle change.

Calibration du modèle. Le modèle utilisé pour estimer l'évolution de l'état dus système et les effets des actions de contrôle ont étés initialisés directement avec les informations issues du bloc d'observation du système cible. Il est question de déterminer si un traitement de l'information avant de l'utiliser pour initialiser le modèle peut améliorer la performance.

A.5. Questions de la simulation multi-agent dans l'architecture

Introduction

Nous nous concentrons ici sur les questions spécifiques à la simulation multi-agent, à savoir : la validité des modèles, la calibration et la correspondance entre entités du système cible et éléments du modèle.

Nous explorons divers manières d'envisager l'initialisation de modèles, dans une première partie. Nous allons nous intéresser à la question relative à la sélection d'un modèle quand nous en avons plusieurs au même temps.

Nous reprenons certains des éléments de la configuration expérimentale précédente : le système cible sera le même et l'architecture sera instanciée dans la plupart des blocs de la même manière.

A.5.1. Scénario expérimental

Nous allons comparer une méthode d'initialisation directe avec une méthode indirecte. Ce qui nous intéresse ici est de savoir, quelle est la différence au niveau de la qualité de prédictions faites par les modèles initialisés de manière différente.

Configuration du système cible

Dans cette série d'expériences nous travaillons avec deux configurations du système cible différentes. Les systèmes seront configurés de la même manière que dans le scénario expérimental précédent. Nous présentons dans le tableau A.7 les valeurs données aux paramètres.

		Système cible A	Système cible B
Paramètre	N	1000	5000
	θ	$U(0, \theta_{max})$	$U(0, \theta_{max})$
	d	1	1
	G	W&S (p, k, n)	W&S (p, k, n)
Conditions Initiales	θ_{max}	10	6
	G	$p = 0.25$,$k = 10$,$n = N$	$p = 0.15$,$k = 10$,$n = N$
	x_{init}	10%	20%

TABLE A.7 – Résumé de valeurs données aux paramètres de deux systèmes cibles utilisés ainsi que les conditions initiales.

Objectif de contrôle

Nous gardons le même objectif de contrôle que celui du chapitre précédent : faire que le système ai un état où la majorité des pairs partagent.

A.5.2. Implémentation de l'architecture

Les différents blocs de l'architecture sont mis en œuvre de la même manière comme celui du chapitre précédent. Cependant, étant donné que dans cette nouvelle série d'expériences nous sommes intéressés par l'évaluation de la performance de l'architecture en fonction de la manière d'initialiser les modèles, nous nous concentrerons sur les changements faits aux blocs « Observation du système cible » et « Estimation de l'état de système cible ».

- Observation du système cible. Un pair est aléatoirement choisi dans le système cible. L'état (partage ou pas), la générosité (θ) et la liste des voisins du pair sont enregistrées. Les voisins du pair sont mis dans une liste de pairs pour échantillonner. Le pair suivant de la liste est échantillonné en ainsi de suite, jusqu'à avoir échantillonné q pairs.

- Estimation de l'état du système cible.

Initialisation directe

Pour initialiser les valeurs des paramètres du modèle, nous rassemblons d'abord des informations du système cible. Ensuite, le modèle M est initialisé comme suit :

- le nombre d'agents dans le modèle est le même que le nombre de pairs échantillonnés ($i = j$).

- Chaque agent a_i dans le modèle aura les caractéristiques de son pair homologue p_i :

  $\theta_{a_i} = \theta_{p_i}$

  $s_{a_i} = s_{p_i}$

- la profondeur d est supposé connue pour le système cible (d), tel que $\forall \text{agent}_i \; d_{a_i} = d$

- les liens E_M du graphe G_M sont identiques aux liens du graphe G_{est}, E_{est}.

Initialisation indirecte

Après que le processus de collecte d'informations a lieu, le modèle topologique M est initialisé comme suit.

- Le nombre de participants du réseau est supposé connu. Sa valeur sera la taille du système cible N.
- Il est supposé connu que la générosité dans le système cible suit une loi uniforme, en $(0, \theta_{max})$, avec θ_{max} étant un entier. La valeur de θ_le de Max est évaluée à partir de l'échantillon : la valeur maximale de l'échantillon, arrondie à la valeur d'entier suivante.
- On suppose que la structure du graphe du système cible a été produite avec l'algorithme Watts and Strogatz. Le graphe du modèle est créé en utilisant le même algorithme. Les paramètres de l'algorithme sont initialisés comme suit.

  Paramètre p de l'algorithme. Il est supposé connu du système cible.

  Paramètre k de l'algorithme. Il est évalué par le processus d'observation comme le nombre moyen de voisins par nœud dans le graphe échantillonné.

  Paramètre n de l'algorithme. Établi comme égal à N (précédemment défini)

Actions de contrôle

Les mêmes actions de contrôle que celles utilisés dans le chapitre précédent : détourner un pair pour lui faire croire que tous ses voisins partagent et le faire ainsi partager. Le même critères pour choisir les actions de contrôle sont utilisés : l'action de contrôle donnant l'augmentation la plus forte du niveau de contribution est choisie.

A.5.3. Expériences

Comportement nominal

Système cible A La figure 5.1 montre le comportement nominal du système cible A. Le système a deux comportements différents pour des conditions initiales identiques : dans 70% des cas il y a un majorité de free-riders et dans 30% des cas il y a une majorité de pairs qui partagent.

Dans 71% des exécutions du système cible A, le niveau de contribution après 30 pas de temps est compris entre 0.006 et 0.055 avec une moyenne de 0.021746 et un écart-type de 0.011241. Dans le 29% d'exécutions restant le niveau de contribution est entre 0.639000 et 0.908000 avec une moyenne de 0.875759 et un écart-type de 0.049379.

Système cible B À la différence du système cible A, le système cible B a seulement un type de comportement : une majorité systèmatique de free-riders. Le niveau de contribution après de 30 pas de temps est compris entre 0.11900 et 0.15760 avec une valeur moyenne de 0.140074 et un écart-type de 0.007574. La figure 5.1 montre le comportement nominal du système cible B.

A.5.4. Résultats

Les résultats concernant l'objectif de contrôle sont résumés dans le tableau A.8.

A.5.5. Discussion

Il est important de souligner, que pour le système cible B, les valeurs données aux paramètres θ_{max} et x_{init} sont différents par rapport à ceux de systèmes cible du chapitre précédent.

Si nous regardons le tableau A.8 l'initialisation indirecte n'a pas été significativement plus efficace que l'initialisation directe. Par contre si nous nous concentrons sur la qualité des estimations faites avec l'initialisation indirecte (voir figures 5.2 et 5.3) le cas est autre : pour le système cible B, l'initialisation indirect est plus précise (moins d'écart type) que l'initialisation directe.

A.5.6. Conclusions

Nous avons exploré deux manières d'initialiser les modèles : avec les valeurs directement observés et avec un traitement préalable.

Étant donnée la différence au niveau de la configuration du réseau entre le système cible B et les autres systèmes cibles étudiés, il est possible que le réseau ai une influence plus importante sur l'évolution du système que ce que nous pensions.

A.5.7. Selection des modéles

Nous nous intéressons ici aux différents choix à faire quand nous avons plusieurs modèles dans l'architecture. Nous allons définir un scénario expérimentale ou nous allons utiliser l'architecture pour contrôler un système cible avec multiple modèles, mais par la même occasion, différents horizons de temps ainsi que différents quantités d'information recopié sur le système cible, et différents actions de contrôle.

- L'objective de contrôle restera le même que le précédent : $D = x_f \geq 50\%$.
- L'implémentation du système cible sera la même que celle du système cible A (définit auparavant).
- Ce qui changera dans l'architecture sera que nous allons définir cette fois-ci un critère pour sélectionner quelle modèle et par conséquence quelle prédiction de l'état futur du système utiliser. Pour cette implémentation en particulier, nous allons utiliser comme critère l'erreur entre les prédictions passés et l'état observé actuel.

$|(\overline{x_f} - \overline{Prediction})/\overline{x_f}|$

	Target System A		Target System B	
Sample Size	Direct	Indirect	Direct	Indirect
$q = 1$	1.28 ± 3.81	1.24 ± 3.50	1.59 ± 1.48	1.81 ± 7.69
$q = 10$	1.27 ± 3.18	0.92 ± 1.89	1.26 ± 0.91	1.87 ± 5.24
$q = 20$	1.12 ± 2.60	0.82 ± 1.44	0.88 ± 0.60	1.68 ± 3.60
$q = 100$	0.58 ± 0.53	0.47 ± 0.53	0.35 ± 0.25	1.21 ± 1.52

TABLE A.8 – Erreur proportionelle moyen entre les prédictions de modèles et le niveau de contribution global réel.

A.6. Conclusion

Dans ce travail de thèse, nous nous sommes concentrés sur le défi suivant : faire qu'un système complexe adopte un comportement souhaité. Plus précisément, nous avons étudié la pertinence d'utiliser la simulation des modèles multi-agent dans le contexte du contrôle de systèmes complexes.

Nous avons proposé une architecture cohérente qui intègre la simulation multi-agent dans une boucle de contrôle utilisant l'approche « equation free ». Cette architecture a la forme d'un motif générique, où plusieurs simulations multi-agent peuvent être utilisées pour évaluer l'état futur du système à contrôler, ainsi que, les effets des actions de contrôle locales.

Cette architecture a été mise en œuvre dans une plate-forme expérimentale et appliquée au problème du « free-riding » de réseaux de partage de fichiers pair-à-pair. Cette mise en œuvre a illustré comment chaque composant de l'architecture peut être instancié. Nous avons démontré que l'architecture peut contrôler un système dans des conditions où le comportement souhaité ne peut pas être observé de manière naturelle dans le système

Le problème que nous avons utilisé, même s'il a été simplifié, est un exemple archétypique d'une famille de problèmes avec la dynamique de points fixes et l'émergence. Le prototype de problème peut être lié à la tragédie des communaux ou aux réseaux booléens, par exemple.

L'utilisation de la simulation multi-agent dans notre proposition entraîne l'apparition de questions concernant la façon d'établir la relation entre le modèle et le système cible, à savoir : la validité, la calibration de modèles et la traduction des entités du système cible aux éléments du modèle. Nous avons montré que la conception modulaire de l'architecture permet d'examiner de telles questions et évaluer la décision faite pour y répondre. Cette conception modulaire permet aussi d'améliorer de façon incrémentale une mise en œuvre spécifique.

Au terme de notre travail de recherche, certains aspects sont restés inexplorés mais mériteraient d'être étudiés dans de futurs travaux.

Dans le court terme, l'étude sur l'exemple applicatif que nous avons utilisé pourrait être approfondi à travers des dimensions différentes pour mieux évaluer l'architecture : un système cible ouvert, des comportements hétérogènes de pairs, du bruit dans les observations, etc. Nous pourrions également envisager l'étude d'autres domaines d'application.

Du point de vue de l'architecture, nous avons examiné des contrôleurs de type « on-off ». La poursuite du travail de recherche permettrait d'examiner des contrôleurs plus avancés, comme le proportionnel ou le proportionnel intégro différentiel et d'étudier les implications sur la relation entre les modèles et le système cible.

De même, il serait intéressant d'étudier des façons différentes de faire évoluer les modèles dans l'architecture, comme l'apprentissage automatique, ou les algorithmes génétiques par exemple.

Enfin, comme perspective à long terme, nous pourrions utiliser le résultat des études précédemment envisagées afin d'obtenir des éléments pour dégager une approche systématique à l'application de notre architecture sur un système cible donné.

Dans notre travail de recherche, nous avons toujours utilisé une instance de l'architecture, mais il serait tout à fait envisageable d'utiliser des instances multiples et de les faire coopérer. Ceci soulèverait de nouvelles questions comme la coordination des instances ou encore la résolution des influences mutuelles entre les instances, (etc.) et amènerait de nouveaux défis.

Résumé

Les systèmes complexes sont présents partout dans notre environnement : internet, réseaux de distribution d'électricité, réseaux de transport. Ces systèmes ont pour caractéristiques d'avoir un grand nombre d'entités autonomes, des structures dynamiques, des échelles de temps et d'espace différentes, ainsi que l'émergence de phénomènes. Ce travail de thèse se focalise sur la problématique du contrôle de tels systèmes. Il s'agit de déterminer, à partir d'une perception partielle de l'état du système, quelle(s) actions(s) effectuer pour éviter ou au contraire favoriser certains états globaux du système. Cette problématique pose plusieurs questions difficiles : pouvoir évaluer l'impact au niveau collectif d'actions appliqués au niveau individuel, modéliser la dynamique d'un système hétérogène (plusieurs comportements différents en interaction), évaluer la qualité des estimations issues de la modélisation de la dynamique du système.

Nous proposons une architecture de contrôle selon une approche « equation-free ». Nous utilisons un modèle multi-agents pour évaluer l'impact global d'actions de contrôle locales avant d'appliquer la plus pertinente.

Associée à cette architecture, une plateforme a été développée pour confronter ces idées à l'expérimentation dans le cadre d'un phénomène simulé de « free-riding » dans les réseaux d'échanges de fichiers pair à pair. Nous avons montré que cette approche permettait d'amener le système dans un état où une majorité de pairs partagent alors que les conditions initiales (sans intervention) feraient évoluer le système vers un état où aucun pair ne partage.

Nous avons également expérimenté avec différentes configurations de l'architecture pour identifier les différents moyens d'améliorer ses performances.

Mots-clés : simulation multi-agent, contrôle de systèmes complexes, equation-free, free-riding, pair à pair.

Abstract

Complex systems are present everywhere in our environment: internet, electricity distribution networks, transport networks. This systems have as characteristics: a large number of autonomous entities, dynamic structures, different time and space scales and emergent phenomena. This thesis work is centered on the problem of control of such systems. The problem is defined as the need to determine, based on a partial perception of the system state, which actions to execute in order to avoid or favor certain global states of the system. This problem comprises several difficult questions: how to evaluate the impact at the global level of actions applied at a global level, how to model the dynamics of an heterogeneous system (different behaviors issue of different levels of interactions), how to evaluate the quality of the estimations issue of the modeling of the system dynamics.

We propose a control architecture based on an "equation-free" approach. We use a multi-agent model to evaluate the global impact of local control actions before applying the most pertinent set of actions.

Associated to our architecture, an experimental platform has been developed to confront the basic ideas or the architecture within the context of simulated "free-riding" phenomenon in peer to peer file exchange networks. We have demonstrated that our approach allows to drive the system to a state where most peers share files, despite given initial conditions that are supposed to drive the system to a state where no peer shares. We have also executed experiments with different configurations of the architecture to identify the different means to improve the performance of the architecture.

Key words: multi-agent simulation, complex system control, equation-free, free-riding, peer to peer.