Chapter 1

Introduction

Autonomous agents and multi-agent systems (MAS) are a recent approach to analysing, designing and implementing complex software systems. Using agents as a key abstraction provides a large and powerful collection of metaphors, methodologies and tools that have allowed conceiving and implementing many innovative types of software [Jennings et al., 1998].

One of the open questions posed in [Jennings et al., 1998] about MAS was: "How to avoid or mitigate harmful overall system behaviour, such as chaotic or oscillatory behaviour?". One possible answer to this question is researchers' proposal to regulate MAS by arranging agents in organisations or institutions.

Organisations and institutions are a key metaphor to regulate interactions of self-interested agents because of the following properties:

openness – agents may enter and leave the MAS at runtime.

regulation – a MAS limits the range of agent behaviour accepted at runtime. That is, all the actions agents can perform are not always permitted.

social structure – Agents are distinguished by their role or their goals.

activity structure – Analogously, each multi-agent activity is also classified by the protocol, *i.e.* possible sequence of actions, that agents have to follow to fulfil certain goals.

For instance, in a virtual market agents are usually classified as providers or customers, and purchasing activities may be classified as different types of auctions with their particular rules. These auctions guide and restrict the behaviour of agents in order to acquire goods. Furthermore, in a virtual market new providers and customers may appear or the existing ones may decide to cease their trading behaviour for a long period of time.

Norms are an intrinsic part of human organisations and institutions. Norms have subdued human societies for centuries thanks mainly to their fear of punishments when violating them. Although agents cannot feel fear, they can emulate it by reasoning for instance about the money loss of the fine(s) when transgressing norms. Initially, agents were regimented following standard software methodologies where software executes a given set of commands in a predetermined order. However, as the agent community agreed autonomy is an essential feature of agents, modelling agent behaviour with norms has been gaining popularity as they allow the partial characterisation of desirable actions of agents.

The purpose of this thesis is to explore how to computationally realise norms in open, regulated, and structured MAS.

1.1 Motivation

In this section, using a motivating scenario, we pose the questions that this thesis tries to answer. The scenario used throughout this thesis is a supply-chain scenario in which companies and individuals come together in a virtual (electronic) marketplace to conduct business.

Consider a wire factory *WireMaking Ltd.* that starts an auction to find suppliers of copper. A buyer agent starts an auction for *WireMaking Ltd.* for copper to which supplier agents may respond. A bid consists of the number of kilograms of the given prime material. Then, after receiving bids the auctioneer assesses the best offer(s).

To discourage unfulfilled promises, buyers may establish economic sanctions or other persuasive measures to be applied in case of delivery problems. Thus, the first setting is the regulation of one activity with norms of behaviour and the following question arises: What kind of language should be used to specify these norms of behaviour?

Some attempts to answer this question have been made, *e.g.* [Esteva, 2003] or [López y López, 2003][1]. However, an essential feature has not been thoroughly treated by these approaches while establishing desirable agent behaviour, *i.e.* time requirements. For instance, the wire factory may want a certain amount of copper to be delivered by the end of the week after the auction takes place. Therefore, after winning an auction, the suppliers are expected to deliver the promised quantity of copper on time. Although the management of time requirements is a desirable feature of the language we are searching for, we notice that dealing with further constraints is also necessary. For instance, when a supplier wins an auction it is expected to deliver the goods by the deadline, but also to fulfil the quantity requirements claimed in the bid. Thus, a question that this thesis tries to answer are:

Q.1 How to specify norms and make them operational to regulate a multi-agent activity?

Retaking our supply chain example, *WireMaking Ltd.* may settle the bill from time to time for the delivery of the promised quantity of copper. That is, apart from the auction activity, agents may participate in a delivery activity

[1]See Chapter 2 for a more comprehensive list of work on norm languages.

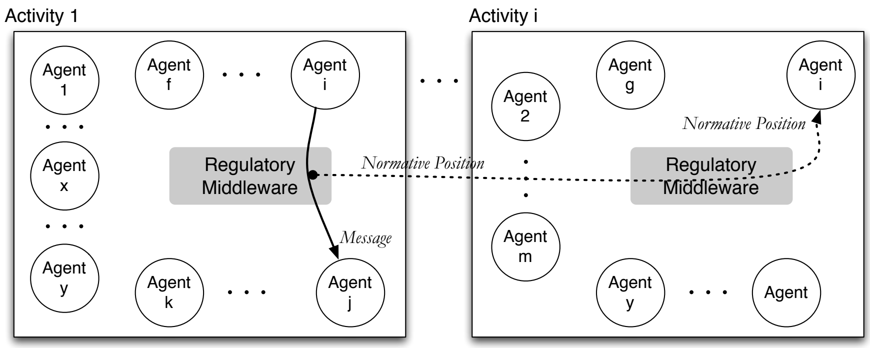

Figure 1.1: Propagation of normative positions

where goods and money are changed according to the agreement established by successfully finishing an auction. Thus, when a supplier wins an auction, it is expected to deliver the promised goods by participating in the delivery activity.

As figure 1.1 shows, in this setting actions in one activity may have effect on other activities, *e.g.* a bid in an auction may generate the expectation of copper deliveries. Furthermore, when the MAS is composed of a large number of activities, it is desirable to distribute them among several computers to reduce their workload and provide a smoother and faster evolution of the activities. Thus, the second setting we shall consider is the regulation of multiple distributed activities. From the previous statements, a question that arises is:

Q.2 How to specify norms and make them operational to handle multiple concurrent activities?

In the literature it is admitted that norms may be contradictory [Sartor, 1992]. Let us consider that a norm may allow all suppliers in an auction to bid at a given moment. However, another norm may state that *Shining-Copper Co.*, a specific supplier agent, is forbidden to bid because of previous unfulfilled deadlines on delivery. In [Kollingbaum, 2005], agents are provided with mechanisms to resolve conflicts among norms. However, the norms have to be *conflict-free* in order to apply them. For instance, what should the MAS do? Should the MAS allow the bid or should it punish the agent? Thus, in our opinion, the resolution of conflicts among norms should be applied in the activity by the MAS.

Since our main goal is to regulate with norms MASs with multiple distributed activities, the following questions also arise:

Q.3 How to computationally enact distributed regulation?

The questions posed above are the main concerns of this thesis. Figure 1.2 shows where the problem addressed in this thesis are. The main concern is to

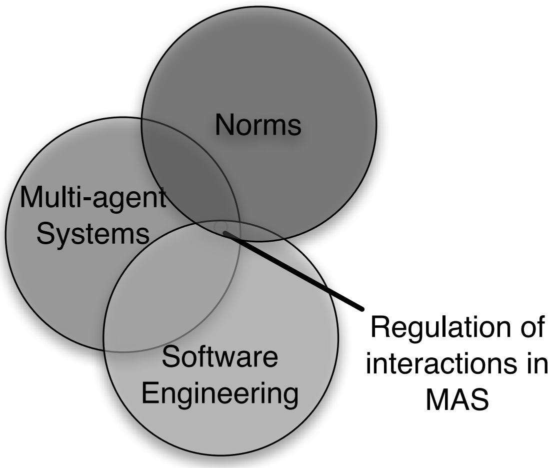

Figure 1.2: This thesis as intersection of research fields.

provide a means to build a computational realisation of a class of MASs regulated by norms. To accomplish this task we will benefit from studies on norms like *deontic logics* [von Wright, 1951] and from already built, regulated MAS such as *electronic institutions* [Rodríguez-Aguilar, 2001].

1.2 Contributions

In this section we introduce the key contributions of this thesis. We envisage the software cycle of a norm as the translation from some requirements represented in natural language to some executable language.

Figure 1.3 shows this process. We start with a representation of requirements in natural language. The representation of requirements as norms is studied deeply in Law. Then, at design time, we envisage the representation of norms with computer languages that it is also deeply studied in the field of Artificial Intelligence and Law. Afterwards, during development, we translate norms into a computer executable language and, in run-time, we feed the system with normative positions and speech acts.

We use as starting point the work in [Noriega, 1997] [Rodríguez-Aguilar, 2001] [Esteva, 2003] that proposes and implements the Electronic Institution metaphor and software tools to establish an agent framework where norms are implemented. We will explore this notion in Chapter 2.

Later on, we will use a broader notion of institution based on [Searle, 1995] where some unprocessed facts, called *brute facts*, are taken into consideration and constitute by convention following the principles of *constitutive rules* new

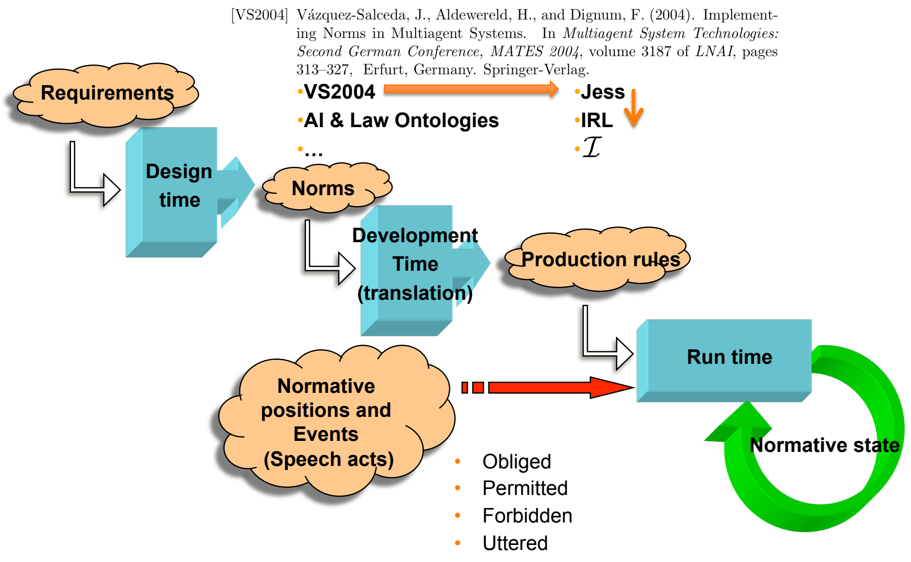
[VS2004] Vázquez-Salceda, J., Aldewereld, H., and Dignum, F. (2004). Implementing Norms in Multiagent Systems. In *Multiagent System Technologies: Second German Conference, MATES 2004*, volume 3187 of *LNAI*, pages 313–327, Erfurt, Germany. Springer-Verlag.
Requirements
Design time
•VS2004
•AI & Law Ontologies
•...
•Jess
•IRL
•$\mathcal{I}$
Norms
Development Time (translation)
Production rules
Run time
Normative positions and Events (Speech acts)
Normative state
• Obliged
• Permitted
• Forbidden
• Uttered

Figure 1.3: Implementation process of a norm

facts that are called *institutional facts*. That is, constitutive rules establish what real facts *count as* some other virtual facts. For example, some pieces of paper may count as a 5 euro bill if it follows pre-defined features as being issued by certain authorities. Furthermore, *regulative rules* establish the restrictions on the use of institutional facts. For instance, they defined the legal (and illegal) use of bills, *e.g* it may be exchanged by items as a car but it is forbidden to acquire certain substances considered illegal.

Initially, we focused on the regulation of an activity. The initial contributions of this thesis are the translation of a norm language into a computer executable language, namely a standard rule production system[2], and the implementation of two rule languages enhanced with norms and constraints[3]. Then, we focused on the regulation of multiple distributed activities. The latter contribution of this thesis is the implementation of a distributed architecture that models and implements norm propagation and the resolution of normative conflicts that may appear during norm propagation.

In order to answer our research question about how to specify norms that

[2]Upper horizontal arrow in Figure 1.3.
[3]Vertical arrow in Figure 1.3

regulate a multi-agent activity, we provide three norm languages. First, we propose a high-level language that allows the user to specify temporal aspects of norms, *e.g.* activation, deactivation and deadlines. Second, we propose a rule language to specify norms with arithmetic constraints. Third, we propose a language with different types of rules to specify norms (with temporal aspects and arithmetical constraints) over agents' simultaneous speech acts and to specify preventive and corrective actions that the system has to perform in each case.

Features	Approach		
	Jess Norms (Ch. 3)	IRL (Ch. 4)	$\mathcal{I}$ (Ch. 5)
Constraints	time	management	management
Distribution	centralised	one activity	one activity
Concurrent Behaviour	activities	actions	actions
Concurrent Regulation	one action	one action	simultaneous actions
Rule execution	forward-chaining	no forward-chaining	regulative rules

Table 1.1: Comparison of the different approaches of chapters 3, 4, and 5

Table 1.1 shows a comparison of features of the three languages. In order to compare normative languages we use the following features:

Constraints – This feature depicts the degree of constraint management. We distinguish no specification (–), specification only of time constraints (time), specification of constraints (specification) and specification and modification (management).

Distribution – This feature reflects the degree of distribution of norms. We distinguish no distribution of norms (centralised), norms distributed in each agent (agents) and norms distributed in each activity (activities).

Concurrent Behaviour – This feature shows the degree of concurrency on actions. We distringuish no concurrency (–), concurrent actions in one or no activity (actions), and concurrent actions in concurrent activities (activities).

Concurrent Regulation – This feature depicts the degree of regulation on concurrent actions. We distinguish:

- no regulation of actions (–),
- no regulation of actions but regulation of goals (goals),
- just monitoring and sanctioning of actions (monitoring),
- monitoring, sanctioning and prevention of one action at a time (one action),
- monitoring, sanctioning and prevention of simultaneous actions (simultaneous actions).

Rule execution – This feature shows how rules are triggered. We distinguish rules that are triggered until no new rule can be triggered (forward-chaining), one execution per rule with different parameters (no forward-chaining) and (regulative rules) different types of rules that change the execution of forward-chaining and one-execution rules.

Jess norms is the first norm language dealing with time in electronic institutions. However, it does not manage arithmetic constraints. IRL is the first language to include constraint management and the specification of the effects of valid events. Notice that it has improved the aspect of constraint management. However, it does not regulate a set of simultaneous actions. Finally, $\mathcal{I}$ is the first language to include that aspect allowing, *e.g.*, to prevent simultaneous actions from being performed. For instance, we may avoid the modification by different agents of the same variable at the same time. As for rule execution, Jess norms uses a standard production system with forward chaining. However, IRL is a rule-based system without forward chaining. Finally, language $\mathcal{I}$ provides several types of rules: standard production rules with forward chaining, standard reactive rules without forward chaining and rules to modify the execution of previous rules, *e.g.* by ignoring agents' actions or by preventing certain states.

In order to answer our research question about how to make norms operational to regulate a multi-agent activity, we provide an implementation for the automatic translation of norms in our high-level language of chapter 3 into rules of a standard production system. Using this automatic translation, we also implemented a norm service for electronic institutions that complements the regulation of activities. Furthermore, we provide an implementation of interpreters for the non-standard rule languages proposed in chapters 4–6.

In order to answer our research question about how to specify norms that handle multiple concurrent activities, we propose a rule language for the propagation of normative positions of agents among activities that we refer to as the language of the normative structure.

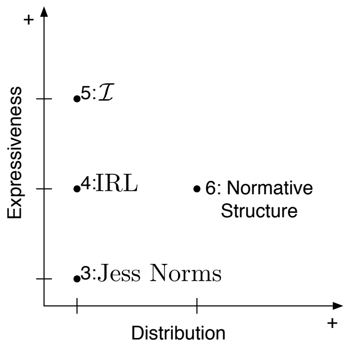

Figure 1.4: Comparison of the approaches proposed in each chapter

Figure 1.4 shows a comparison of the proposed languages in terms of expressiveness and distribution. On the one hand, the languages of chapters 3 - 5 grow in expressiveness without dealing with distribution. On the other hand, the language of chapter 6 deals with distribution of activities in the same degree of expressiveness as that of the language presented in chapter 4.

In order to answer our research question about how to make norms operational to regulate multiple concurrent and distributed activities, we provide normative structures, a computational model for the propagation of normative positions, and an algorithm to manage conflicting norms at runtime to complement formal verification techniques. Our algorithm resolves conflicts among norms at runtime (possibly enriched with arithmetical constraints). If the system does not resolve a given conflict in the normative structure, agents can use conflict resolution techniques to decide which conflicting normative position to comply with. Supplementing formal verification techniques with practical correction techniques used at runtime is not new in software engineering. Usually, imperative programming languages have constructs to detect and repair runtime errors or exceptions. However, these techniques have not been applied before at runtime to norms with constraints. The use of our proposed algorithm enables software designers to correct the specification of desired behaviour of software components at runtime.

Finally, in order to address our research question about how to computationally enact distributed regulation a distributed architecture is presented for the enactment of a regulated MAS. The architecture supports:

- the regulation of agent activities with norms activated as result of agent behaviour,
- the activation of norms among activities and
- the resolution of conflicts among norms in an activity at runtime.

This architecture establishes the basis for building MAS enriched to enforce, propagate, and resolve conflicts in, sets of norms at runtime. Furthermore, by providing this architecture to electronic institutions we enable these to incorporate all the conceptual contributions we have proposed.

1.3 Structure

This thesis is organised as follows:

Chapter 2 surveys the work on the topics dealt with in this thesis. The chapter is divided into an overview on norms in deontic logics and multi-agent systems, on computational normative languages and on regulated multi-agent systems.

Chapter 3 starts addressing **Q.1** (normative language and computational model) in a centralised manner and introduces a language for the representation of norms in electronic institutions and its translation into Jess rules to give them an operational semantics. These norms are complemented with temporal operators to establish deadlines and the time of activation and deactivation.

In chapters 4 and 5, we start by regulating a single activity:

Chapter 4 presents IRL, a language for the representation and explicit management of normative positions, *i.e.* permissions, prohibitions, and obligations active at runtime. This language replaces the language of chapter 3 in the pursuit of **Q.1** (normative language and computational model). This language is supplemented with constraints conferring normative positions with more expressiveness since not only temporal constraints can be represented with this language. In this chapter, we introduce the notion of normative positions, differentiate various types of prohibitions and obligations and provide the means to implement activities without taking into account relations among them.

Chapter 5 describes $\mathcal{I}$, a language for the representation of normative positions, *i.e.* permissions, prohibitions, and obligations active at runtime, and the system behaviour given these normative positions. The system, following its specification, ignores, forces, expects events or prevents states of affairs. In this chapter, we establish how to enforce normative positions by providing the specification of the system behaviour. Furthermore, we classify the usual behaviour of systems that enforce normative positions and also we extend it with the notion of preventing a state or ignoring, forcing, expecting or sanctioning sets of simultaneous events. This language replaces the language of chapter 4 in the pursuit of **Q.1** (normative language and computational model).

Then, in chapters 6 and 7, we regulate multiple distributed activities:

Chapter 6 addresses **Q.2**, introducing the normative structure, an additional layer in the MAS model that propagates the effects of actions among activities, and presenting an algorithm for the resolution of normative conflicts. In this chapter, we extend the behaviour of the system with the actions of propagating and resolving conflicts among normative positions, *i.e.* active norms. In addition, we propose that activities specified in different languages can propagate normative positions by means of the normative structure.

Chapter 7 addresses **Q.3** by describing AMELI$^+$, an extension of the agent middleware for electronic institutions that incorporates the enactment of the normative structure propagating formulae, including normative positions, among several activities. These activities are regulated by normative positions and the specification of their behaviour using languages as the ones presented in chapters 4 and 5. The architecture also embeds the algorithm presented in chapter 6 for the resolution of normative conflicts.

Chapter 8 discusses the contributions of this research and how it can be extended in the future.

1.4 Publications

The work in Chapter 3 has been published in:

- [García-Camino et al., 2005a] García-Camino, A., Noriega, P., and Rodríguez-Aguilar, J.-A. Implementing Norms in Electronic Institutions.

In *Proceedings of 4th International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'05)*, pages 667–673, Utrecht, The Netherlands.

- [García-Camino et al., 2005b] García-Camino, A., Noriega, P., and Rodríguez-Aguilar, J.-A. Implementing Norms in Electronic Institutions (Extended Abstract). In *3rd European Workshop on Multi-agent Systems (EUMAS'05)*, Brussels, Belgium.

The work in Chapter 4 has been published in:

- [García-Camino et al., 2008] García-Camino, A., Rodríguez-Aguilar, J. A., Sierra, C., and Vasconcelos, W. (2008). Constraint rule-based programming of norms for electronic institutions. *Journal on Autonomous Agents and Multi-Agent Systems.* (In press).
- [García-Camino et al., 2006a] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. A Distributed Architecture for Norm-Aware Agent Societies. In Baldoni, M. et al., editors, *Declarative Agent Languages and Technologies III*, volume 3904 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 89–105. Springer, Berlin Heidelberg.
- [García-Camino et al., 2006b] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. A Rule-based Approach to Norm-Oriented Programming of Electronic Institutions. *ACM SIGecom Exchanges*, 5(5):33–40.
- [García-Camino et al., 2006c] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. Norm Oriented Programming of Electronic Institutions. In *Proceedings of 5th International Joint Conference on Autonomous Agents and Multiagent Systems. (AAMAS'06).*
- [García-Camino et al., 2007b] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. Norm-Oriented Programming of Electronic Institutions: A Rule-based Approach. In *Coordination, Organization, Institutions and Norms in agent systems II*, volume 4386 of *Lecture Notes in Computer Science*, pages 177–193. Springer-Verlag.
- [García-Camino et al., 2006d] García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W. Norm-Oriented Programming of Electronic Institutions (Extended Abstract). In *Fourth European Workshop on Multi-Agent Systems (EUMAS'06)*, Lisbon, Portugal.

The work in Chapter 5 has been published in:

- [García-Camino, 2007] García-Camino, A. Ignoring, Forcing and Expecting Concurrent Events in Electronic Institutions. In *COIN III: Coordination, Organization, Institutions and Norms in Agent Systems. Revised Selected Papers from the 2007 Workshop Series*, volume 4870 of *Lecture Notes in Computer Science*, pages 15–26. Springer.

The work in Chapter 6 has been published in:

- [García-Camino et al., 2007a] García-Camino, A., Noriega, P., and Rodríguez-Aguilar, J.-A. (2006) An Algorithm for Conflict Resolution in Regulated Compound Activities. In *Engineering Societies in the Agents World VII*, volume 4457 of *Lecture Notes in Computer Science*, pages 193–208.

- [Gaertner et al., 2007] Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J.-A., and Vasconcelos, W. Distributed Norm Management in Regulated Multi-agent Systems. In *Proceedings of 6th International Joint Conference on Autonomous Agents and Multiagent Systems. (AAMAS'07).*

- [Gaertner et al., 2008] Gaertner, D., García-Camino, A., Noriega, P., Rodríguez-Aguilar, J. A., and Vasconcelos, W. (2008). Normative structures for regulating open multi-agent systems. *Journal on Autonomous Agents and Multi-Agent Systems.* (submitted).

- [Kollingbaum et al., 2007a] Kollingbaum, M. J., Vasconcelos, W. W., García-Camino, A., and Norman, T. J. Conflict resolution in norm-regulated environments via unification and constraints. In *Declarative Agent Languages and Technologies V*, volume 4897 of *Lecture Notes in Artificial Intelligence*, pages 158–174. Springer.

- [Kollingbaum et al., 2007b] Kollingbaum, M. J., Vasconcelos, W. W., García-Camino, A., and Norman, T. J. Managing conflict resolution in norm-regulated environments. In *Engineering Societies in the Agents World VIII*, volume (In press) of *Lecture Notes in Artificial Intelligence.* Springer.

The work in Chapter 7 has been published in:

- [García-Camino et al., 2007c] García-Camino, A., Rodríguez-Aguilar, J. A., and Vasconcelos, W. (2007). A Distributed Architecture for Norm Management in Multi-Agent Systems. In *COIN III: Coordination, Organization, Institutions and Norms in Agent Systems. Revised Selected Papers from the 2007 Workshop Series*, volume 4870 of *Lecture Notes in Computer Science*, pages 275–286. Springer.