Garcia Marques, ME.; Miles, S.; Luck, M.; Giret Boggino, AS. (2014). Evaluating how agent methodologies support the specification of the normative environment through the development process. Autonomous Agents and Multi-Agent Systems. 1-20. doi:10.1007/s10458-014-9275-z.

Autonomous Agents and Multi-Agent Systems manuscript No.
(will be inserted by the editor)

Evaluating how agent methodologies support the specification of the normative environment through the development process

Emilia Garcia · Simon Miles · Michael Luck · Adriana Giret ·

Received: date / Accepted: date

Abstract Due to the increase in collaborative work and the decentralization of processes in many domains, there is an expanding demand for large-scale, flexible and adaptive software systems to support the interactions of people and institutions distributed in heterogeneous environments. Commonly, these software applications should follow specific regulations meaning the actors using them are bound by rights, duties and restrictions. Since this normative environment determines the final design of the software system, it should be considered as an important issue during the design of the system. Some agent-oriented software engineering methodologies deal with the development of normative systems (systems that have a normative environment) by integrating the analysis of the normative environment of a system in the development process. This paper analyses to what extent these methodologies support the analysis and formalisation of the normative environment and highlights some open issues of the topic.

Keywords Multi-agent systems · Normative systems · Agent methodologies

1 Introduction

Currently many domains, such as health and commerce, demand complex, dynamic and decentralised systems in which different entities and institutions interact and interchange services in order to achieve their objectives. Since these stakeholders are usually heterogeneous and autonomous, there is no central authority that designs or ensures their behaviour. Although neither of

E. Garcia, A. Giret
Universitat Politecnica de Valencia, Spain
E-mail: {mgarcia,agiret}@dsic.upv.es

S. Miles, M. Luck
King's College London, UK
E-mail: {simon.miles,michael.luck}@kcl.ac.uk

them have power or authority to control the behaviour of the others, the expected behaviour of each stakeholder should be known in order to ensure the stability of the system and to achieve the success of the interchanges between entities.

Moreover, these domains are usually regulated by governmental legislation and the internal regulations of each institution involved in the system. The behaviour of each entity is bound by rights, duties and restrictions derived from the global legislation, the requirements of the system and the specific regulations of each institution. One example of application is the software supporting the collaboration between health clinics and research institutions in order to allow researchers to find eligible patients to their clinical trials [35]. Each stakeholder involved in the collaboration must follow different regulations. Each clinic must follow their internal regulations and the governmental regulations attached to the management of clinical data. Researchers must follow the regulations of their own research institution and the governmental regulations about clinical trials. Moreover, clinics and research institutions software may have been developed by different stakeholders using different technologies. Therefore, in such domains, it is necessary to develop systems where the desired and forbidden behaviour of each entity is formally specified. Any undesired behaviour can compromise the stability of the system, avoid the correct communication between parties or violate a legal restriction.

In the literature, regulated systems that deal with dynamic regulations in a social environment are called *normative systems* [48]. The advantage of a norm-based design approach is that there is a ready way for developers to specify these regulations explicitly in the development process, such that they become part of the design. Implementing the system in a norm-aware platform can ensure their fulfilment, even if the system has been externally implemented by different providers.

The analysis and design of systems of this kind are complex tasks. Governmental legislation and the specific legislation of each institution and entity involved in the system should be considered, bringing a need to integrate different normative environments. Since these restrictions on the behaviour of the entities may determine the final design of the system, they must be identified and analysed in the early phases of the development process. In this paper, we consider the *normative environment* of a system to be the set of norms that regulate the behaviour of each entity and the set of contracts that formalise the relationships between these entities. The *normative environment* of each entity is specified by the set of norms that directly affect the behaviour of this entity. The *normative context* of an organisation[1] is the set of norms that only affect the entities that are part of this organisation.

Over the last decade, multi-agent technology has been used for the implementation of academic and industrial applications, and there are currently several agent methodologies that deal with the development of normative sys-

[1] In this paper the term *organisation* is considered a synonym of *institution*.

tems [23,26]. In this paper, we wish to answer the following question: *To what extent do agent methodologies support the development of normative systems?*.

In this paper the term "methodology" is understood as a framework that offers a set of design abstractions and guidelines that guides the process of developing a system by offering: (1) mechanisms to specify the design of the system, (2) a set of tasks that must be performed in order to develop the system, (3) and advice about when and how these tasks must be executed.

First, we analyse the methodological requirements for developing normative systems (Section 2). These requirements are specified in Section 3 as a set of questions that help to evaluate to what extent agent methodologies support the analysis and design of normative systems. Section 3 presents an overview of the state of the art and selects four agent methodologies, taking into account their support for analysing and designing normative systems. These methodologies are analysed in more detail in Section 4. Section 5 presents an analysis of some open issues in this context.

In order to illustrate some common characteristics of normative systems the following subsection presents a case study based on a virtual water market. This normative system will be used through the paper as running example.

1.1 Running example: mWater case study

mWater is an institutional, decentralised framework where users with water rights are allowed to voluntarily trade their rights with other users, in exchange for some compensation, economic or otherwise, but always fulfilling some pre-established rules [11,37].

The whole system is regulated by the National Hydrological Plan of the country that establishes the creation of one *basin institution* for each water basin. These institutions are responsible for monitoring the quality and level of the basin's water and for controlling the transfer of water rights within their basin. Each basin institution is an autonomous organisation. In addition to the National Hydrological Plan, these institutions must follow the specific regulations of their region. Basin institutions have the autonomy to decide whether to participate in the virtual market and they are able to abandon the system at any time. Each basin institution offers a set of software services and resources to its members. These applications have been developed by different developers using different technology. Therefore, the mWater virtual market system should be able to integrate the different basin institutions and to allow communication and service interchanges between them. In order to participate in the system each basin institution must fulfil a set of requirements: (1) Every basin internal structure should have a *notary* who will be responsible for validating each water interchange performed inside the basin institution. It also should have a *jury* who will solve the conflicts among the members of the basin. (2) Every basin should provide specific services to the government authority. (3) Every basin should specify internal regulations that must be coherent with the current governmental legislation.

The *government authority* is the person or group of persons that represent the government in the system. This authority can revoke any agreement established in the system and is responsible for verifying that the governmental legislation is fulfilled in every basin institution.

In this virtual market based environment, different autonomous entities, representing individuals, groups of irrigators, industries, or other water users, get in contact in order to buy and sell water rights. They are able to negotiate the terms and conditions of the transfer agreement following the specific regulations of the basins involved. In order to perform an inter-basin transfer, the agreement should be authorised by the government of the country.

This case study was implemented using the Electronic Institutions framework [29,31] as a testbed for agreement technologies [20]. During its analysis, design, implementation and maintenance we have had to deal with some of the most common issues related to normative open systems. We found the analysis and design of the normative environment particularly challenging because of the complexity explained above. We therefore believe this to be a good running example to be used throughout the paper to illustrate some features and attributes. However, it must be noted that this paper does not analyse the specific implementation of the case study with Electronic Institutions; such an analysis is out of the scope of the paper. Instead, the paper's main goal is to analyse the support offered by agent methodologies for analysis and design of the normative environment of a system. The case study is only used to give an example of some common features of normative systems that would be difficult to understand without concrete examples.

In addition, it is important to state that the requirements and criteria presented in the following sections are not only based on the mWater case study, but on the broader literature concerning normative systems, and also on our experience designing normative multiagent systems applications in different domains, such as health [36].

2 Requirements for designing the normative environment

This section describes the requirements at design time for developing normative systems. In this paper, we consider normative systems as systems where the entities' behaviour and relationships are restricted by a set of norms [7]. Now, in the literature there is no consensus about the terminology that must be used to specify normative systems [30,39]. Therefore, in this section we analyse the requirements for designing normative systems from a semantic perspective and associate these semantics to specific terms in order to reuse them in the following sections. The specific terms are highlighted in bold.

The rest of the section is organised as follows: First, Section 2.1 analyses the metamodel constructions that are necessary to represent the normative environments of a system. Second, Section 2.2 analyses the support during the development process that is necessary in order to completely analyse and

formalise these normative environments. Finally, Section 2.3 analyses how the final design of the system should be validated.

2.1 Design abstractions

Due to the increase in collaborative work and the decentralisation of processes in many domains, there is an expanding demand for large-scale, flexible and adaptive software systems to support the interactions of people and institutions distributed in heterogeneous environments [50]. Each one of these institutions may have specific legislation and internal regulations associated with them, meaning that the actors inside them are bound by rights, duties and restrictions [33]. In this paper, the norms that regulate the behaviour inside a specific institution, or group of entities, are called **institutional norms**. The interaction of several institutions and the integration of different subsystems into a common one brings a need to specify different normative environments inside the same system. Therefore, the design architecture should allow the definition of several **normative environments** inside a system. Here, the *normative environment* of a system is considered to be the set of norms that regulate the behaviour of each entity and the set of contracts that formalise the relationships between entities and institutions. The *normative environment* of each entity and institution is thus specified by the set of norms that directly affect the behaviour of this entity or institution.

For example, in the mWater case study each *basin institution* and *irrigator community* has its own regulations, i.e. its own normative environment. These normative environments can have associated with them different and even contradictory norms, but all these normative environments should be compliant with the National Hydrological Plan. When there is an interbasin interchange of water the regulations of all the institutions involved should be fulfilled.

In the agent literature the functionality of the system is usually divided between different roles [1,2]. Any entity that tries to participate in the system should play at least one of these roles, and any agent that plays a specific role acquires a set of rights, duties and restrictions related to this role. Therefore, it is necessary that the design architecture allows the definition of the set of rights, duties and restrictions associated with each specific role. In this paper, these types of norms are called **role norms**.

Moreover, many current developments are supposed to be *open*. In this context, the term *open* means that external entities can interact and become part of the system at runtime [25]. As a consequence, any external entity that wants to enter in the system must acquire a specific role of the system. These external entities can be heterogeneous and can be developed outside the scope of the system. However, once a stakeholder enters a normative system its behaviour is restricted by the rights and duties of the roles that it is playing. Contracts have been used in many domains in order to formalise these restrictions without compromising the autonomy of the entities [21]. This is because contracts are expressive and flexible. They allow agents to operate with expectations of

the behaviour of other agents based on high-level behavioural commitments, and they provide flexibility in how the autonomous agents fulfill their own obligations [60]. Contracts also allow the negotiation of specific terms of the engagement between a stakeholder and a role. Although contracts should be specified and negotiated at runtime, at design time contract templates should be defined in order to specify contract patterns that any contract of this type should fulfil [34]. In this paper, the contract templates that establish the terms under which an entity can play a role are called **play role contracts**.

For example, in the mWater case study basin institutions can participate in the system and leave it at any time. When a basin institution enters the system it acquires a set of rights and duties. Every basin institution is obliged by its play role contract to provide a set of services to the government authority. However, if the contract is not established how should such services be implemented. In this sense, the general behaviour of each basin institution is known but these institutions keep their autonomy regarding how to implement it.

Most normative systems involve a complex social structure. The use of norms to specify the structure allows the entities to reason about the structure of their system, and allows the structure to be updated dynamically at runtime. Such a social structure implies a set of rights and duties between the entities of the system [44]. In this paper, the norms that define the structure of the system are called **structural norms**. The structure of a system is usually specified by the relationships between the organisations within it (if we are dealing with an organisational system), and the roles of the system. Therefore, structural norms apply to organisations, roles, and the agents playing a role.

The description of system architectures imply the specification of the relationship between several entities of the system [47]. In dynamic and flexible systems the specific terms of the social relationship between entities can be negotiated between the entities involved. The use of contract templates to specify these relationships provides flexible architectures and maintains the autonomy of the system about how to implement their commitments [57]. In this paper these kinds of contracts are called **social relationship contracts**. For example, in the mWater case study each basin institution is under the supervision of the government authority, but the specific terms of this supervision can be negotiated for each basin institution. Without the specification of such a contract the relationship between these entities would be fixed at design time and it would not be possible to specify different agreements regarding the individual features of each basin institution.

Also, each individual entity may have special rights or restrictions associated to its own design and implementation. These norms are not related to the general structure of the system or the roles that this entity plays, but to the specific features of each individual entity. For example, although *water users* are allowed to trade with all their water rights, a specific implementation of a *water user* can restrict the quantity of water rights that this agent is allowed to trade. In this paper these kinds of norms are called **agent norms**.

Interchanges of services and resources are formalised at runtime [27], but in regulated systems, it may be necessary to specify at design time which kind of relationships are allowed and under which terms. Therefore, it is necessary to specify contract templates that formalise these restrictions and maybe establish the interaction protocols that should be executed in order to negotiate, execute and resolve conflicts related to these contracts. In this paper these types of contract templates are called **contractual agreements**.

Some normative systems allow their entities to violate the norms and contracts of the system (*enforcement*). Other normative systems do not allow any entities to perform an action that is not permitted (*regimentation*). The methodology should thus offer some guidelines to help designers to decide which option is best suited to their particular system context. Moreover, in the case of enforcement, the methodology should provide a design abstraction that allows specification of the consequences of violating a norm. We refer to such a design abstraction in this paper as **norm violation**.

2.2 Support during the development process

In the previous subsection a set of different types of norms that should be formalised at design time were presented. As shown in the literature [54, 45, 6] and in many studied case studies such as [36], these restrictions can be derived from: (1) the specific requirements of the system (e.g. a system in which the main goal is to increase productivity during a specific period would forbid any entity from taking a holiday in this period); (2) legal documents that formalise governmental law or internal regulations of each institution (e.g. the National Hydrological Plan, the governmental law about water right interchanges); and (3) design decisions. The identification of the normative environment of a system is not trivial because: (1) the descriptions of the requirements of the system provided by domain experts might be incomplete; (2) individual entities might have their own goals that conflict with the goals of the system; (3) in systems composed of different institutions, each could have its own normative environment that needs to be integrated into an overall system; and (4) legal documents are written in plain text, which means that the terminology of the domain expert and these legal documents could be different.

A poor or incomplete specification of the normative environment can produce a lack of trustworthiness and robustness in the system [55]. In open systems in which each entity could be developed by a different institution, if the rights and duties are not formally specified, an entity that tries to join a system would not know how to behave. Entities could perform actions that harm the stability of the system (e.g. in a non-monopoly system, a client could buy all the resources of one type).

Therefore, methodologies should include in their analysis and design phases guidelines to identify and formalise these restrictions in order to be sure that the normative environment is completely specified.

First, specific guidelines should be added to the **requirements analysis** stage in order to identify and formalise the norms that are directly related to the requirements of the system [6,54]. Also, specific guidelines for identifying the norms that should be implemented in a system, derived from the **legal documents** associated with the system, should be provided. This identification is a complex process because such documents are usually written in plain text and the semantic meaning of the concepts described in the legal documents and in the system design can be inconsistent.

Obviously, the norms derived from the requirements analysis and those associated with legal documents may determine the final design of the system [13]. Therefore, these norms should be analysed before the design of the system is performed and they should be taken into account during the process. Once a specific design is determined, in order to ensure that this design is implemented properly, the **structure of the system** and the **relationship between the roles and entities of the system** should be formalised by means of norms and contracts. So, the methodology should provide specific guidelines for identifying institutional, role and agent norms, as well as guidelines to formalise play role and social relationship contracts.

The identification of when two entities must collaborate and the formalisation of these interchanges can be a complex task. So, it is necessary to offer specific guidelines for the **identification and formalisation of contractual agreements**.

Furthermore, methodologies should offer guidelines that help the designer to select the most appropriate **negotiation, execution and conflict resolution protocol** for each specific contract regarding its requirements. The restrictions to consider a protocol as an appropriate one are: (1) to allow achieving the objectives of the interaction, (2)to respect the norms of the system. One metric that could be used to evaluate which protocol is more appropriate than another is the number of interactions that this protocol includes. However each methodology would specify its own metrics.

2.3 Evaluation of the final design

Developing normative systems is a very complex task because it requires specification of the global behaviour of the system, the individual behaviour of each agent, the legal context of each entity, and the social and contractual interactions [45]. Also, many conflicts can arise from the potential combination of institutional norms and the specific restrictions of each agent derived from the commitments of their signed contracts [42]. It is necessary to ensure that each single normative environment has no conflicts, and also that the composition of all the normative environments is itself conflict-free. As is presented in [32], conflicts in norms arise for four different reasons: (1) the obligation and prohibition to perform the same action; (2) the permission and prohibition to perform the same action; (3) obligations of contradictory actions; (4) permissions and obligations of contradictory actions. Therefore, guidelines for

verifying the **coherence of the normative environment** should be offered by the methodology and integrated into the development process.

Requirements traceability is an important feature in any kind of system [63], and refers to the ability to describe and follow the life of a requirement, in both forward and backward directions. Traceability improves the quality of software systems [46], and facilitates verification and validation analysis, control of changes, as well as reuse of software systems components and so on. The ability to follow the life of a requirement associated with a norm is even more important due to the dynamism of the normative environments of a system. For example, in the mWater case study, the whole system should follow the National Hydrological Plan legislation. Without traceability, any change in this law would imply the revision of the whole system. However, if it would be possible to trace each norm individually, only the norms that had changed should be revised and only the parts of the system affected by these norms should be redesigned. Therefore, **traceability of the normative environment** is a desired feature in a methodology for developing normative systems.

3 Evaluation criteria

Due to the differences in the terminology and semantics of each methodology, the evaluation and comparison of methodologies is a complex task. In this section, a set of questions guiding the evaluation and comparison of agent methodologies is presented. These questions are focused on the evaluation of the extent to which agent methodologies support the analysis and design of normative systems. These questions are based on the requirements identified in the previous section, and the criteria divided into three categories: (1) design abstractions, (2) support during the development process, (3) evaluation of the final design. A general overview of the state of the art of agent methodologies regarding these criteria is also presented.

3.1 Regarding the design abstractions

- *Institutional norms:* Does the methodology support the specification of norms that only affect the scope of a specific institution?
- *Normative environments:* Does the methodology support the specification of different normative environments in the system?
- *Role norms:* Does the methodology support the specification of the norms that are associated with a specific role?
- *Agent norms:* Does the methodology support the specification of the norms that are associated with a specific agent?
- *Play role contract:* Does the methodology support the formalisation of the rights and duties that an agent acquires when playing a specific role in the system by means of contracts?
- *Structural norms:* Does the methodology support the formalisation of the structure of the system by means of norms?
- *Social relationship contract:* Does the methodology support the formalisation of the structure of the system by means of contracts?
- *Contractual agreements:* Does the methodology support the formalisation of the interchange of resources and services between different actors of the system?

Currently, many agent methodologies integrate norms into their metamodels in order to formalise the restrictions on the behaviour of the actors within systems [9, 22, 28, 3]. Most offer formalisations of norms that allow the specification of the consequences of violating a norm. However, in these approaches it is the designer who decides, without any guideline, whether the normative system is to be enforced or regimented. This decision depends on the system requirements and also on the functionality offered by the execution platform on which the system will run.

Many of methodologies also allow the specification of organisational systems. These agent methodologies are able to describe different normative environments by means of specifying norms whose scope is limited to one particular organisation of the system [59, 27, 16].

Almost every agent methodology uses the concept of *role* to specify the different functionalities that an actor can have in the system. Therefore, the use of *role norms* is common and well supported by most methodologies for normative systems.

Some organisational methodologies, like OperA [26], do not support the specification of individual agents. However, those that support the design of individual agents usually allow the specification of agent norms [4].

Over the last few years, the integration of electronic contracts in multi-agent systems (MAS) has become increasingly more important to system architectures for agent behaviour regulation [51, 16, 38].

3.2 Regarding the support during the development process

- *Requirement norms:* Does the methodology provide any guideline to identify and formalise the norms of the system during the requirements analysis?
Does the methodology provide any guideline to identify which requirements should be specified as norms?
- *Legal documents:* Does the methodology provide any guideline to identify and formalise the norms that should be implemented in the system derived from legal documents associated with the system?
- *System design:* Does the methodology consider the normative environment of the system as an important factor in the design of the system?
- *Structure considers norms:* Is the normative environment of the system analysed before specifying its structure? Is this normative environment integrated into the guideline to define the structure of the system?
- *Contract protocols:* Does the methodology provide any guideline to formalise the negotiation, execution or conflict resolution protocol associated with each contract regarding its requirements?

Although some methodologies include in their metamodel and development process the description of the normative environment of a system, only a few provide guidelines to actually identify the normative environment of the system. Work by Boella and Rotolo et al. [6, 54] offers several guidelines that focus the attention of the system designer on important issues when developing a normative system, but they cannot be used as an artefact for designers to identify the norms that regulate the system. Kollingbaum et al. [41] present a framework called Requirement-driven Contracting (RdC), for automatically deriving executable norms from requirements and associated relevant information, but this framework only derives system norms from the description of the goals of the system. A more complete guideline that includes the analysis of each entity's goals, and the resources and the relationships between entities is still needed.

Breaux et al. [12, 14] present a methodology for extracting and prioritising rights and obligations from regulations. They show how semantic models can be used to clarify ambiguities through focused elicitation, thereby balancing rights with obligations. [13] continues this work, investigating legal ambiguity and what constitutes reasonable security. This methodology identifies obligations and restrictions derived from the analysis of the complaints, agreements and judgments of the system. It seems to address existing systems and needs runtime information to derive the norms. The methodology is not, however, focused on the analysis and design of multiagent systems, although some of these guidelines could be combined with an agent methodology to adapt the system at runtime and increase its security.

Siena et al. [56] study the problem of generating a set of requirements, which comply with a given law, for a new system. They propose a systematic process for generating law-compliant requirements by using a taxonomy of legal concepts and a set of primitives to describe stakeholders and their strategic

goals. This process must be combined with an agent methodology in order to completely design the system.

Saeki and Kaiya [55] propose a technique to elicit regulation-compliant requirements. In this technique, the regulations are semantically checked against requirements sentences to detect the missing obligation acts and prohibition acts in the requirements.

3.3 Regarding the evaluation of the final design

- *Coherence of the normative environment*: Does the methodology offer guidelines to verify the coherence of the normative environment?
Does the methodology offer guidelines to verify the coherence between the system and agent's goals and the normative environment?
- *Traceability*: Does the methodology support traceability of the normative environment?

Regarding the verification of the models and the consistency and coherence of norms and contracts inside an organisation, some work has been done but it is still an open problem. Most work here is focused on offline verification of norms by means of model checking [61].

The application of model-checking techniques to the verification of contract-based systems is an open research topic. Some work like [58] models contracts as finite automata that model the behaviour of the contract signatories, while other work represents them as Petri nets [40]. These representations are useful to verify safety and liveness properties. However, adding deontic clauses to a contract allows conditional obligations, permissions, and prohibitions to be written explicitly [54]. In [53] and [32] a deontic view of contracts is specified using the *CL* language, while the work in [53] uses model-checking techniques to verify the correctness of the contract and to ensure that certain properties hold. The work in [32] presents a finite trace semantics for *CL* that is augmented with deontic information as well as a process for automatic contract analysis for conflict discovery. In the context of Service-Oriented Architectures, model checkers have recently been used to verify compliance of web-service composition. In [43] a technique based on model checking is presented for the verification of contract-service compositions.

In the context of verification techniques for MAS, there are some important achievements using model checking. In [62], the SPIN model checker is used to verify agent dialogues and to prove properties of specific agent protocols, such as termination, liveness, and correctness. In [10] a framework for the verification of agent programs is introduced, that translates MAS that are programmed in the logic-based agent-oriented programming language AgentSpeak into either PROMELA or Java. It then uses the SPIN and JPF model checkers to verify the resulting systems. In [64], a similar approach is presented but it is applied to an imperative programming language called MABLE. In [52],

the compatibility of interaction protocols and agents' deontic constraints is verified. However none of these approaches considers organisational concepts.

There are only some efforts that deal with the verification of systems that integrate organisational concepts, contracts, and normative environments. The most developed approach is presented in the context of the IST-CONTRACT project [51], which offers contract formalisation and a complete architecture, and uses the MCMAS model checker to verify contracts. However, as far as we know, it does not define the organisational normative environment or verify the coherence of this context with the contracts.

Little work ensures traceability of requirements [17] and none of them is focused on the traceability of the normative environment attributes.

4 Related work: AOSE methodologies

The general study of the state of the art presented in the previous section provides an overview of which methodologies provide extensive support for developing normative systems. In this section we have selected four in order to analyse and compare them (OperA, O-MaSE, Tropos and GORMAS).

It is important to note that there are other approaches like Electronic Institutions [29,8] or InstAL [19,42] that also support the development of normative systems, but their support is less complete than those selected in terms of the detail of specification of the phases of the methodology and the guidelines that they provide for analysing and designing these kinds of systems.

4.1 OperA

Organisations per Agents (OperA) [26] is a framework for the specification of normative open MAS that includes a formal metamodel, a methodology and a CASE tool.

4.1.1 Design abstractions

The OperA model describes a MAS as an organisational structure regulated by norms and contracts, where norms specify obligations, permissions and prohibitions of the roles of the system. The OperA model uses the concept of *group* to specify different normative environments inside a system. Here, groups are used to collectively refer to a set of roles and to specify norms that hold only for all roles in the group. OperA does not include the design of the individual agents, but it assumes that agents can understand the society ontology and communicative acts, and are able to communicate with the society.

OperA defines two types of contracts: *social contracts* and *interaction contracts*. These abstractions respectively match with the *play contract* and *contractual agreement* concepts detailed in Section 2.1. Social contracts establish an agreement between the agent and the organisation model and define the

way in which the agent will fulfil its roles. In this sense, the structure of the society is defined by the social contracts specified in the system. Interaction contracts establish an agreement between agents, i.e., they define agent partnerships, and fix the way a specific interaction scene is to be played.

A contract definition includes its parties (the agents involved), the clauses (described as norms), and the communication protocol to be followed. The clauses of each contract specify the rights, duties and restrictions that the agents acquire when signing the contract.

4.1.2 Support during the development process

The OperA methodology is structured in three steps, as follows.

- *Organisational model design:* This phase specifies the *OperA Organisational Model* for an agent society. This model is composed of three levels: (1) Coordination Level: Specifies how the structure of the society is determined. (2) Environment Level: The society model determined in the previous step is further refined with the specification of its social structure in terms of roles, global requirements and domain ontology. (3) Behaviour Level: The organisational model of an agent society is completed with the specification of its interaction structure, which results from the analysis of the interaction patterns and processes of the domain. This process is supported by a library of interaction patterns.
- *Social model design:* This phase describes the agent population in the Social Model that will enact the roles described in the structure. It describes the roles specified in the previous phase, the role negotiation scenes and the characteristics of the agents that apply for society roles. In other words, during this phase the social contracts that define the structure of the system are detailed.
- *Interaction model design:* This phase describes the concrete interaction scenes between agents. Interaction contracts are used to formalise these interaction scenes.

As can be seen, the OperA methodology integrates in the development process the specification of the contracts and norms that regulate the system and its entities. OperA offers guidelines to select the most appropriate organisational structure and to specify interaction protocols by means of patterns. However, this methodology does not offer guidelines to capture the clauses (norms) that each contract should contain.

4.1.3 Evaluation of the final design

OperA models can be implemented using the Operetta tool [49]. Although OperA methodology does not integrate the verification of the system as a step of the methodology, the Operetta tool integrates model checking techniques in order to verify the coherence of the system design. This verification includes the validation of the coherence of the normative environment of the system.

4.2 O-MaSE

O-MaSE [23] provides a customisable agent-oriented methodology based on a metamodel, a set of methods fragments and a set of method construction guidelines. It also offers a CASE tool [24].

4.2.1 Design abstractions

The O-MaSE metamodel covers most basic MAS concepts such as agents, roles, organisations, and interactions. The normative environment of the system is represented by means of a set of norms called *policies* in this metamodel. These policies describe how an organisation, role or agent may or may not behave in particular situations. The organisational structure allows creation of different normative environments inside the same system.

The O-MaSE metamodel supports the development of open systems. Roles define positions within an organisation whose behaviour is expected to achieve a set of goals, but this metamodel does not integrate the concept of contract. Therefore, any agent that would like to play a specific role would have to revise the description of the role and search in the set of policies for those that would affect the agent if it played this role.

Interactions between agents are represented by means of protocols, but these interactions are not formalised with contracts. This means that agents interacting with other agents in the system should know in advance the interaction protocols and the policies of the system in order to know the expected behaviour of the others.

4.2.2 Support during the development process

The O-MaSE methodology explicitly defines activities and tasks but does not define specific phases. O-MaSE provides a set of guidelines to organise these activities in different ways based on project need. These activities include the analysis of the requirements; the design of the system by means of organisations and roles; the architecture design by means of defining agent classes, protocols and policies; the low level design in which specific plans, capabilities and actions are described; and the code generation.

Although the methodology includes a specific task where the policies of the system are formalised, there is no guideline that helps the designer to identify these policies from the requirements of the system, and this identification relies on designer expertise.

4.2.3 Evaluation of the final design

The O-MaSE methodology framework is supported by the aT^3 integrated development environment, which supports method creation and maintenance, model creation and verification, and code generation and maintenance. The aT^3 verification framework allows selection from a set of predefined rules those

that should be checked against the model. This allows one to verify specific properties of the model and process consistency. However, as far as we know, there is no tool for verifying the coherence of the normative environment.

4.3 Tropos

The initial version of the Tropos methodology was focused on supporting the agent paradigm and its associated mentalistic notions throughout the entire software development life cycle from requirements analysis to implementation [15]. This version of the methodology does not support the concepts of norms or contracts. However, Telang at al. [59] enhance Tropos with commitments, and propose a metamodel based on commitments and a methodology for specifying a business model.

4.3.1 Design abstractions

None of these versions of Tropos include norms in their metamodel. In Telang at al. [59] commitments represent contracts between entities. However, these contracts only represent duties that the entities acquire. They do not establish limits on the behaviour of the entities.

4.3.2 Support during the development process

In Telang at al. [59] a methodology is proposed in order to analyse and design the system. One of the steps of the methodology consists in the identification of the commitment. However, no guideline is specified to identify these commitments and the restrictions on the behaviour are not considered.

4.3.3 Evaluation of the final design

Chopra et al. [18] propose a technique to verify that an agent can potentially achieve its objectives playing a specific role, and that an agent is potentially able to honour its commitments. However, they do not provide any guideline or technique to verify the coherence of the normative system.

Tropos offers support for requirements traceability but does not consider the normative environment.

4.4 GORMAS

GORMAS (Guidelines for ORganisational Multi-Agent Systems) [5] defines a set of activities for the analysis and design of organisational systems, including the design of the norms that restrict the behaviour of the entities of the system.

4.4.1 Design abstractions

The GORMAS metamodel describes a MAS as an organisational structure regulated by norms and whose functionality is expressed by means of services. GORMAS allows the specification of institutional, role, agent and structural norms. Entities interact using standard services and following the restrictions of the system. However, GORMAS does not include the abstraction of contract.

4.4.2 Support during the development process

The GORMAS methodology is focused on the analysis and design processes, and is composed of four phases covering the analysis and design of a MAS: the first phase is *mission analysis*, which involves the analysis of the system requirements, the use cases, the stakeholders and the global goals of the system; the *service analysis phase* specifies the services offered by the organisation to its clients, as well as its behaviour, and the relationships between these services; the *organisational design phase* defines the social structure of the system, establishing the relationships and restrictions that exist in the system; and finally, at the *organisation dynamics design phase*, communicative processes between agents are established, as well as processes that control the acquisition of roles along with processes that enable control of the flow of agents entering and leaving the organisation. Additionally, some norms that are used to control the system are defined. Finally, the organisation dynamics design phase is responsible for designing guides that establish a suitable reward system for the organisation.

Norms are present from the early beginning of the development process, but GORMAS does not offer any specific guideline to identify the norms that restrict the system. Their identification rests on the expertise of the designer.

GORMAS also does not offer any guideline for specifying the most appropriate interaction protocol regarding the specific requirements.

4.4.3 Evaluation of the final design

GORMAS does not offer any tool for the verification of the coherence of the system or the traceability of the normative environment.

5 Discussion: Gap analysis

This section presents a gap analysis of the state of the art of agent methodologies supporting the analysis and design of normative systems. As presented in Section 3, some approaches offer partial solutions to the problem. However, the combination of these partial solutions in order to obtain a complete development methodology that developers can follow is not an easy task because

Comparative				
FEATURES	OMASE	OPERA	TROPOS	GORMAS
Design abstractions:				
Institutional norms	Supported	Supported	Not supported	Supported
Normative contexts	Supported	Supported	Not supported	Supported
Role norms	Supported	Supported	Not supported	Supported
Agent norms	Supported	Not supported	Not supported	Supported
Structural norms	Not supported	Supported	Not supported	Supported
Play Role contracts	Not supported	Supported	Partially supported	Not supported
Social relationship contracts	Not supported	Supported	Partially supported	Not supported
Contractual agreements	Not supported	Supported	Partially supported	Not supported
Support during the development process:				
Identification: Requirement norms	Partially provided	Partially provided	Not provided	Partially provided
Identification: Legal documents	Not provided	Not provided	Not provided	Not provided
Structure consider norms	Partially provided	Partially provided	Not provided	Not provided
System design	Considered	Considered	Not considered	Considered
Structure considers norms	Part of the normative system is analysed before but it is not integrated in the guideline.	Part of the normative system is analysed before but it is not integrated in the guideline.	Not considered	Partially supported
Contract protocols	Not provided	Partially. It offers a library of patterns for interaction protocols.	Not provided	Not supported
Evaluation of the final design:				
Coherence	Partial verification in the case tool	Partial verification in the case tool	Not supported	Not supported
Traceability	Not supported	Not supported	Not supported	Not supported

Table 1 Comparative methodologies

each approach uses different terminology, semantics and metamodel constructions. Section 4 analyses some of the most complete agent methodologies for developing normative systems. Table 1 summarises this analysis in terms of the evaluation criteria presented in Section 3.

The results of the analysis of the state of the art can be summarised as follows.

- Most well-known agent methodologies integrate into their metamodels the concepts of organisations and norms. This allows designers to specify and formalise institutional, role and agent norms, as well as to specify different normative environments inside the same system.
- Only a few methodologies integrate the concept of contracts into their metamodel. Some methodologies integrate into their metamodel the specification of contractual agreements, but the use of structural norms and

contracts to define the structure of the system is only supported by a small subset of methodologies.
- Most methodologies provide specific guidelines for selecting the most suitable organisational typology and for distributing the functionality of the system in the most appropriate way between the parties involved. However, only a small subset consider the normative environment when selecting the organisational structure.
- No methodology integrates into the development process guidelines that completely support the identification of norms from the analysis of the requirements, nor from legal texts.
- Although there is some work related to validation and verification of the designed models, it is still an open problem. Verification using any development approach is important, but in normative open systems is even more so due to the high risk of incoherence resulting from interference between different normative environments, and between the global goals of the system and the individual goals of each party.
- Traceability of norms from requirements is not well supported by current methodologies.

In what follows, we consider the effects of designing a normative system with a methodology that does not include the abstractions and guidelines analysed above. The mWater case study presented in Section 1.1 is used to illustrate some of these effects.

- If the methodology does not specify norms, the restrictions on the behaviour of the entities of the system should have been internally specified in the implementation of each entity. Therefore, the integration of entities that had been implemented outside the scope of the system would not be secure. For example, the *Basin institutions* software, which was used before implementation of the mWater system, should have been revised and reimplemented before allowing entities to be integrated in the system. Moreover, any change in the norms of the system (e.g. new legislation of the National Hydrological Plan) would require the system to be stopped and reimplemented before restarting it again.
- If the methodology does not specify contracts, the *water users*, *basin institutions* and all the entities of the system should know in advance the rights, duties and restrictions that they acquire when enter the system. In this case, it is not possible to negotiate specific conditions for each entity. Moreover, the relationships between entities are not formalised, which implies that there are no negotiations over the terms, and that no specific norm could be attached to any specific interaction.
- Many normative systems have a complex normative environment derived from several legal documents and the internal legislation of the institutions participating in the system. Developing such a system without a complete development process and without guidelines that help designers to identify and formalise the normative environments of the system requires significant

expertise on the part of the designer. Even for an expert designer it is easy to miss a set of norms that could be critical for the system.
- If the methodology does not include methods to verify the coherence of the different norms of the system, the designer must verify them manually. However this is a complex task because of the huge number of norms that must be taken into account. For example, in the mWater system the verification of the normative environment must consider the internal regulation of each *basin institution* and check that all of them are coherent with the Hydrological National Plan.

Although, as explained previously, the analysis of the mWater implementation is outside the scope of this paper, we consider it important to clarify that Electronic Institutions do provide the necessary design abstractions to develop a normative system. However, it lacks detailed development process guidelines such as those for identifying the normative environment of a system.

6 Conclusions

The novelty of our work, as described in this paper, is an examination of the extent to which agent methodologies support the analysis and design of normative systems from a software engineering perspective. This paper has identified a set of specific requirements for systems of this kind, and these requirements have been translated into a set of evaluation criteria that can be used as guidelines to evaluate the support that agent methodologies offer for the design of such systems. These criteria have been used to analyse four agent methodologies.

Although the analysis of the state of the art shows that there are few very mature agent methodologies that can be used to develop normative systems, there are still open issues on the topic. Section 5 summarises the main conclusions relating to the state of the art on this topic. As future work we plan to develop a methodology for normative open systems that deals with the open issues identified in this paper. We are also working on the specification and evaluation of quantitative metrics to compare methodologies, about which there is as yet no consensus.

References

1. *Handbook on Agent-Oriented Design Processes*. Springer, 2014.
2. O. Akbari. A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance. volume 1, pages 14–28. 2010.
3. E. Argente, V. Botti, C. Carrascosa, A. Giret, V. Julian, and M. Rebollo. An Abstract Architecture for Virtual Organizations: The THOMAS approach. *Knowledge and Information Systems*, pages 1–35, 2011.
4. E. Argente, V. Botti, and V. Julian. GORMAS: An Organizational-Oriented Methodological Guideline for Open MAS. In *Proc. AOSE'09*, pages 440–449, 2009.

5. E. Argente, V. Botti, and V. Julian. Organizational-oriented methodological guidelines for designing virtual organizations. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, volume 5518 of *Lecture Notes in Computer Science*, pages 154–162, 2009.
6. G. Boella, G. Pigozzi, and L. van der Torre. Normative systems in computer science - ten guidelines for normative multiagent systems. In G. Boella, P. Noriega, G. Pigozzi, and H. Verhagen, editors, *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, 2009.
7. G. Boella, L. Torre, and H. Verhagen. Introduction to normative multiagent systems. *Computational and Mathematical Organization Theory*, 12(2-3):71–79, 2006.
8. A. Bogdanovych, M. Esteva, S. Simoff, C. Sierra, and H. Berger. A methodology for developing multiagent systems as 3d electronic institutions. In M. Luck and L. Padgham, editors, *Agent-Oriented Software Engineering VIII*, volume 4951 of *Lecture Notes in Computer Science*, pages 103–117. Springer Berlin Heidelberg, 2008.
9. O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vazquez-Salceda. *Coordination, Organizations, Institutions and Norms in Multi-Agent Systems*, volume 3913 of *LNCS (LNAI)*. 2006.
10. R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. In *Autonomous Agents and Multi-Agent Systems*, volume 12, pages 239–256, Hingham, MA, USA, 2006. Kluwer Academic Publishers.
11. V. Botti, A. Garrido, A. Giret, and P. Noriega. The Role of MAS as a Decision Support Tool in a Water-Rights Market. In *Post-proceedings workshops AAMAS2011*, volume 7068, pages 35–49. Springer, 2011.
12. T. Breaux. Exercising due diligence in legal requirements acquisition: A tool-supported, frame-based approach. In *Proc. IEEE Int. Requirements Engineering Conference*, pages 225–230, 2009.
13. T. D. Breaux and D. L. Baumer. Legally reasonable security requirements: A 10-year ftc retrospective. *Computers and Security*, 30(4):178 – 193, 2011.
14. T. D. Breaux, M. W. Vail, and A. I. Anton. Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, RE '06, pages 46–55, Washington, DC, USA, 2006. IEEE Computer Society.
15. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
16. H. L. Cardoso and E. Oliveira. A contract model for electronic institutions. In *COIN'07: Proceedings of the 2007 international conference on Coordination, organizations, institutions, and norms in agent systems III*, pages 27–40, 2008.
17. A. Castor, R. C. Pinto, C. T. L. L. Silva, and J. Castro. Towards requirement traceability in tropos. In *WER*, pages 189–200, 2004.
18. A. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Modeling and reasoning about service-oriented applications via goals and commitments. *ICST Conference on Digital Business*, 2009.
19. O. Cliffe, M. Vos, and J. Padget. Specifying and analysing agent-based social institutions using answer set programming. In O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vzquez-Salceda, editors, *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*, volume 3913 of *Lecture Notes in Computer Science*, pages 99–113. Springer Berlin Heidelberg, 2006.
20. N. Criado, E. Argente, A. Garrido, J. A. Gimeno, F. Igual, V. Botti, P. Noriega, and A. Giret. Norm enforceability in Electronic Institutions? In *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, volume 6541, pages 250–267. Springer, 2011.
21. C. Dellarocas and M. Klein. Contractual agent societies. In R. Conte and C. Dellarocas, editors, *Social Order in Multiagent Systems*, volume 2 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, pages 113–133. Springer US, 2001.
22. S. A. DeLoach. Developing a multiagent conference management system using the o-mase process framework. In *Proc. Int. Conf. on Agent-oriented software engineering VIII*, pages 168–181, 2008.

23. S. A. DeLoach and J. C. Garcia-Ojeda. O-mase; a customisable approach to designing and building complex, adaptive multi-agent systems. *Int. J. Agent-Oriented Softw. Eng.*, 4(3):244–280, 2010.
24. S. A. DeLoach, L. Padgham, A. Perini, A. Susi, and J. Thangarajah. Using three aose toolkits to develop a sample design. In *International Journal Agent-Oriented Software Engineering*, volume 3, pages 416–476, 2009.
25. F. Dignum, V. Dignum, J. Thangarajah, L. Padgham, and M. Winikoff. Open agent systems ? *Eighth International Workshop on Agent Oriented Software Engineering (AOSE) in AAMAS07*, 2007.
26. V. Dignum. *A model for organizational interaction:based on agents, founded in logic.* PhD thesis, Utrecht University, 2003.
27. V. Dignum, J. Meyer, F. Dignum, and H. Weigand. Formal Specification of Interaction in Agent Societies. *Formal Approaches to Agent-Based Systems*, 2699, 2003.
28. V. Dignum, J. Vazquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In R. Bordini, M. Dastani, J. Dix, and A. Seghrouchni, editors, *Programming Multi-Agent Systems*, volume 3346 of *Lecture Notes in Computer Science*, pages 181–198. Springer Berlin / Heidelberg, 2005.
29. M. d'Inverno, M. Luck, P. Noriega, J. Rodriguez-Aguilar, and C. Sierra. Communicating open systems. volume 186, pages 38–94. 2012.
30. C. Elsenbroich and N. Gilbert. Agent-based modelling. In *Modelling Norms*, pages 65–84. Springer Netherlands, 2014.
31. M. Esteva, B. Rosell, J. A. Rodriguez, and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *AAMAS04*, pages 236–243, 2004.
32. S. Fenech, G. J. Pace, and G. Schneider. Automatic conflict detection on contracts. In *Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing*, ICTAC '09, pages 200–214, 2009.
33. C. Garbay, F. Badeig, and J. Caelen. Normative multi-agent approach to support collaborative work in distributed tangible environments. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, CSCW '12, pages 83–86, New York, NY, USA, 2012. ACM.
34. E. Garcia, A. Giret, and V. Botti. Regulated open multi-agent systems based on contracts. In *Information Systems Development*, pages 243–255, 2011.
35. E. Garcia, G. Tyson, S. Miles, M. Luck, A. Taweel, T. V. Staa, and B. Delaney. An Analysis of Agent-Oriented Engineering of e-Health Systems. In *13th International Workshop on Agent-Oriented Software Engineering (AOSE - AAMAS)*, 2012.
36. E. Garcia, G. Tyson, S. Miles, M. Luck, A. Taweel, T. V. Staa, and B. Delaney. Analysing the Suitability of Multiagent Methodologies for e-Health Systems. In *Agent-Oriented Software Engineering XIII*, volume 7852, pages 134–150. Springer-Verlag, 2013.
37. A. Garrido, A. Giret, V. Botti, and P. Noriega. mWater, a Case Study for Modeling Virtual Markets. In *New Perspectives on Agreement Technologies*, volume Law, Gover, pages 563–579. Springer, 2013.
38. B. Gteau, O. Boissier, and D. Khadraoui. Multi-agent-based support for electronic contracting in virtual enterprises. *IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, 150(3):73–91, 2006.
39. C. D. Hollander and A. S. Wu. The current state of normative agent-based systems. *Journal of Artificial Societies and Social Simulation*, 14(2):6, 2011.
40. F.-S. Hsieh. Automated negotiation based on contract net and petri net. In *E-Commerce and Web Technologies*, volume 3590 of *Lecture Notes in Computer Science*, pages 148–157. 2005.
41. M. Kollingbaum, I. J. Jureta, W. Vasconcelos, and K. Sycara. Automated requirements-driven definition of norms for the regulation of behavior in multi-agent systems. In *Proceedings of the AISB 2008 Workshop on Behaviour Regulation in Multi-Agent Systems*, Aberdeen, Scotland, U.K., April 2008.
42. T. Li, T. Balke, M. Vos, K. Satoh, and J. Padget. Detecting conflicts in legal systems. In Y. Motomura, A. Butler, and D. Bekki, editors, *New Frontiers in Artificial Intelligence*, volume 7856 of *Lecture Notes in Computer Science*, pages 174–189. Springer Berlin Heidelberg, 2013.

43. A. Lomuscio, H. Qu, and M. Solanki. Towards verifying contract regulated service composition. *Autonomous Agents and Multi-Agent Systems*, pages 1–29, 2010.
44. F. Lopez, M. Luck, and M. d'Inverno. A normative framework for agent-based systems. In *Computational and Mathematical Organization Theory*, volume 12, pages 227–250, 2006.
45. F. y. Lpez, M. Luck, and M. dInverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12(2-3):227–250, 2006.
46. P. Mader and A. Egyed. Assessing the effect of requirements traceability for software maintenance. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 171–180, Sept 2012.
47. X. Mao and E. Yu. Organizational and social concepts in agent oriented software engineering. In *AOSE IV*, volume 3382 of *Lecture Notes in Artificial Intelligence*, pages 184–202, 2005.
48. J.-J. C. Meyer and R. J. Wieringa, editors. *Deontic logic in computer science: normative system specification*. John Wiley and Sons Ltd., Chichester, UK, 1993.
49. D. Okouya and V. Dignum. Operetta: A prototype tool for the design, analysis and development of multi-agent organizations (demo paper). In *AAMAS*, pages 1667–1678, 2008.
50. M. T. W. S. J. B. Olson, G. M. *Coordination theory and collaboration technology*. Mahwah, NJ: Lawrence Erlbaum Associates, 2001.
51. N. Oren, S. Panagiotidi, J. Vázquez-Salceda, S. Modgil, M. Luck, and S. Miles. Towards a formalisation of electronic contracting environments. *COIN*, pages 156–171, 2009.
52. N. Osman, D. Robertson, and C. Walton. Run-time model checking of interaction and deontic models for multi-agent systems. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 238–240, New York, NY, USA, 2006. ACM.
53. G. Pace, C. Prisacariu, and G. Schneider. Model checking contracts a case study. In *Automated Technology for Verification and Analysis*, volume 4762 of *Lecture Notes in Computer Science*, pages 82–97. 2007.
54. A. Rotolo and L. van der Torre. Rules, agents and norms: Guidelines for rule-based normative multi-agent systems. In *RuleML Europe*, volume 6826, pages 52–66, 2011.
55. M. Saeki and H. Kaiya. Supporting the elicitation of requirements compliant with regulations. In *CAiSE '08*, pages 228–242, 2008.
56. A. Siena, J. Mylopoulos, A. Perini, and A. Susi. Designing law-compliant software requirements. In *Proceedings of the 28th International Conference on Conceptual Modeling*, ER '09, pages 472–486, 2009.
57. M. P. Singh. Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects.
58. E. Solaiman, C. Molina-Jimenez, and S. Shrivastav. Model checking correctness properties of electronic contracts. In *Service-Oriented Computing - ICSOC 2003*, volume 2910 of *Lecture Notes in Computer Science*, pages 303–318. Springer Berlin / Heidelberg, 2003.
59. P. R. Telang and M. P. Singh. Conceptual modeling: Foundations and applications. chapter Enhancing Tropos with Commitments, pages 417–435. 2009.
60. J. Vázquez-Salceda, R. Confalonieri, I. Gomez, P. Storms, S. P. Nick Kuijpers, and S. Alvarez. Modelling contractually-bounded interactions in the car insurance domain. *DIGIBIZ 2009*, 2009.
61. F. Viganò and M. Colombetti. Symbolic model checking of institutions. In *ICEC*, pages 35–44, 2007.
62. C. D. Walton. Verifiable agent dialogues. *Journal of Applied Logic*, 5(2):197 – 213, 2007. Logic-Based Agent Verification.
63. S. Winkler and J. Pilgrim. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling (SoSyM)*, 9(4):529–565, Sept. 2010.
64. M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multi-agent systems with mable. In *AAMAS02*, pages 952–959. ACM, 2002.