8
Conclusion

Three main assumptions underlie this research. Firstly, MAS has emerged as a concrete solution to develop complex software systems in which monolithic architectures (based on objects) have been replaced by distributed ones (based on agents). Secondly, with the advent of the Semantic Web, agents will be able to process information from different sources and, so, they will be able to move around other MAS looking for resources and/or services not found locally. In this scenario, openness will be an intrinsic and mandatory characteristic of upcoming systems. However, openness without control leads to chaotic scenarios. The use of norms in MAS is a promising approach for achieving openness in a reliable way. So, the final assumption of this work is that MAS should be normative.

In the second chapter of this thesis, it was outlined some drawbacks of current solutions for NMAS. The achievement of a solution to resolve those drawbacks turned up the objectives of this thesis. Below, the drawbacks outlined are remembered and the work realized to resolve them is analyzed in order to discuss the main findings related to the objectives of this thesis.

"Solutions for MAS modeling are too *agent-centric* or too *organizational-centric*." DynaCROM eases[13] both types of solutions, being the responsibility of the system developer not to mainly focus on the model of single agents nor on the society one. DynaCROM supports the modeling of a NMAS regarding domain concepts in which agents' interactions can be regulated at different levels of abstractions, softening the solution. This means that agents' interactions do not need to be completely fixed in rigid protocols to be regulated.

"*Roles* and *agents* are usually treated without an explicit distinction." DynaCROM is based on an ontology instance created for the application domain of a developed NMAS. In the domain ontology, the system developer can represent the roles and agents of his NMAS as distinct instances of the *Role* and *Agent* concepts, respectively. This way, the differences between the values of the or-

[13] '*Easy*' meaning "*to reduce in tension, pressure, or rigidity*" [OnLineDictionary, URL].

ganization and its individuals (*i.e.*, agents) are established in two separated concepts from different levels of abstractions.

"Normative aspects are not often considered or, when considered, they are either too theoretical or too practical. Few agent methodologies cover normative aspects and they usually do it by trying to model the whole normative environment in only one level of abstraction, either too theoretical (by means of computationally hard logics) or too practical (by means of the usage of policies or protocols)." The normative aspects drive the DynaCROM approach, which is neither too theoretical nor too practical. The logic suggested to be used in DynaCROM is OWL DL (for the reasons given in subsection 3.1.3.1 of this thesis). Two usage scenarios were presented in this text by using OWL DL, which provides expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems. The practicality of DynaCROM for dealing with norms is given by the possibility to use different third-party enforcers, each of them providing their specific advantages.

"Ontologies are seen as an external (accessory) component, while in fact they are tightly coupled with the rest of the system when used to model most of its elements." As said before, DynaCROM is based on a created domain ontology instance, which is tightly coupled with the rest of the system and holds the representation of its elements and data.

8.1. Thesis Contributions

Motivated by the research questions of this thesis, seeking for solutions in NMAS that can apply and reflect changing norms in open systems, some goals for norm management in NMAS were defined. Then, those goals were limited mainly according to the scope for the areas of software engineering and regulative norms. A research methodology was also defined in order to conduct to the DynaCROM solution. Research questions, objectives, scope and methodology were all presented in chapter 1 of this text.

Following, a summary of each achievement of this thesis is outlined. Those achievements reached the goals of this thesis, answering its posed research questions. The work was done surrounded by the scope defined. The work remained by consequence of the scope defined is presented as future work, in next.

Thesis contributions:

TC.1. Definition of a top-down classification for contextual norms, which facilitates the tasks of elicitation, organization and management of norm information.

TC.2. Development of a contextual normative ontology to explicitly represent the semantics of classified norms in a meaningful way (*i.e.*, with a common understanding) for heterogeneous agents.

TC.3. Definition of a norm composition process, based on ontology-driven rules, that makes it easy to update the system regulation by both evolving norms in a unique resource (an ontology) and by customizing particular rules for different compositions of contextual norms.

TC.4. Development of a solution for informing and/or enforcing contextual norms.

The thesis contributions TC.1, TC.2 and TC.3 reached the thesis research objective RO.2; the thesis contributions TC.2 and TC.3 reached the thesis research objective RO.4; and, finally, the thesis contribution TC.4 reached the thesis research objectives RO.1, RO.3 and RO.5.

The list of publications reporting the contributions of this thesis is presented in the Appendix A of this text.

8.2. Future Work

This thesis introduces an approach to operationalize regulative norms in MAS. The approach supports the design and development of NMAS. Nonetheless, there is still work to be done, as summarized by the next five main points.

Firstly, DynaCROM does not encompass a formal method amenable to rigorous verification of the system developer's specifications. DynaCROM provides an ontology structure in which semantics for normative data representation can be interpreted by heterogeneous agents. However, the responsibility to create correct specifications for MAS remains to the system developer.

In [Silva, 2008], an automatic approach conceived to generate rules from norms is presented. Such approach claims to be amenable to formal verification, thus, the work can be used as a reference for the DynaCROM formalization.

Secondly, DynaCROM is currently dealing with *regulative norms* (*i.e.*, *permissions*, *obligations* and *prohibitions*), however, other types of norms exist. *Constitutive norms* are norms that define what is classified as institutional facts in a NMAS; and, *procedural norms* are norms which aims is to achieve the social order specified by using regulative and constitutive norms [Boella and Torre, 2007]. *Conditional norms* are norms concerning a condition 'P' or an action 'A' under some circumstance 'C'. 'The norm is conditional under C' means that the activation of the norm is detected when the condition/circumstance 'C' is true and the deactivation of the norm is detected when a predicate 'P' or an action 'A' is fulfilled, or 'C' does not hold anymore [Vázquez-Salceda *et al.*, 2004].

DynaCROM can also deal with constitutive, procedural and conditional norms, as follows. Constitutive norms can be represented by organization norms (*i.e.*, norms that are represented by instances of the DynaCROM *Organization* concept). Procedural norms can be represented by compositions of the defined regulative and constitutive norms, as states its definition. Conditional norms can be represented by instances of any DynaCROM concept that is enhanced with a string attribute for holding the conditional clause.

Thirdly, DynaCROM does not consider time restrictions (*e.g.*, deadlines) in norms. Present norms are all norms applicable at a given moment. The main difficulty of considering time restrictions in DynaCROM is due to the fact that it does not have a centralized solution that can hold a clock for providing the time for all system's entities. Besides that, this difficulty increases when interacting agents are running in networks distributed geographically and, so, with different time zones.

An inspiration for resolving the time restriction can be found in [Paes *et al.*, 2005] by investigating the '*Clock*' element of the XMLaw conceptual model.

Fourthly, DynaCROM is not currently dealing with possible conflicts that exist among norms from different normative contexts. It is assumed that it is the responsibility of the system developer to maintain the coherence of the norms of his system. Some experiments with conflicting norms can be planned in order to verify if DynaCROM can detect and resolve them. If it is possible, DynaCROM can also be used as a mechanism for providing conflict-free norm inputs to norm enforcement solutions. This way, DynaCROM could work as a fine-grained approach for resolving normative conflicts.

Conflict-free norms can be reached by adopting one of the three classic strategies for resolving deontic conflicts, all presented in [Vasconcelos *et al.*, 2007], as follows: *legis posterior* (the most recent norm takes precedence), *legis superior* (the norm imposed by the strongest power takes precedence) and *legis specialis* (the most specific norm takes precedence).

DynaCROM support the implementation of the *legis superior* and *legis specialis* strategies. The *legis posterior strategy* cannot be trivially implemented in a DynaCROM NMAS because the sequence of norm additions in its solution is not currently registered. For the *legis superior* strategy, the system developer needs to specify, in the DynaCROM code, that prohibitions override both obligations and permissions, and that obligations override permissions. For the *legis superior* strategy, the system developer needs to specify, in the DynaCROM code, that norms from lower level concepts take precedence (*e.g.*, role norms take precedence of organization and environment norms). Both solutions were also pointed out in [Vasconcelos *et al.*, 2007].

Finally, DynaCROM is currently implemented as an active JADE behavior. JADE behaviors are executed in a *round-robin non preemptive* way. That is, behaviors are never interrupted and, so, they are executed until they return. Hence, it is important that the system developer defines simple behaviors in their JADE agents, being careful to avoid infinite loops (in this case, since a behavior never returns, no other agent behavior will be executed). Meanwhile, DynaCROM should have a mechanism to limit the execution time of JADE behaviors.