6
Comparison with Related Work

In the following subsections, some related work done in the field of MAS modelling and norm enforcement are outlined in order to better clarify the advantages of DynaCROM to the area of NMAS.

Because SCAAR and MOSES were already presented in the last two chapters of this thesis, as the current DynaCROM solution for norm enforcement, thus, those solutions will not be further described in this chapter.

6.1.
OMNI

As described in chapter 2 of this thesis (more specifically in its subsection 2.1.1.1), the OMNI framework (meaning *Organizational Model for Normative Institutions*) is proposed as a solution for modeling electronic agent-based organizations.

OMNI is composed of the following three dimensions: *Normative*, *Organizational* and *Ontological*; furthermore, each dimension also can be considered at three abstraction levels, which are differentiated by increasing implementation details. The *Abstract Level* has the statutes of the organization to be modeled, the definitions of terms that are generic for any organization and the ontology of the model itself. The *Concrete Level* refines the meanings defined in the previous level, in terms of norms and rules, roles, landmarks and concrete ontological concepts. The *Implementation Level* has the *Normative* and *Organizational* dimensions implemented in a given multi-agent architecture with the mechanisms for role enactment and for norm enforcement.

In order to illustrate the architectural similarities and differences of DynaCROM compared to OMNI, Figure 30 was created for DynaCROM based on Figure 29, which is originally presented in [Vàzquez-Salceda *et al.*, 2005]. The architectures of OMNI and DynaCROM are defined at the *Abstract*, *Concrete* and *Implementation* Levels, however, in DynaCROM, the *Normative Dimension* of OMNI is defined as a *Contextual Normative Dimension* and the *Organizational Dimen-*

sion as a *Domain Dimension*. This is because, in DynaCROM, the focus is on normative domain concepts instead of on the organizational ones.

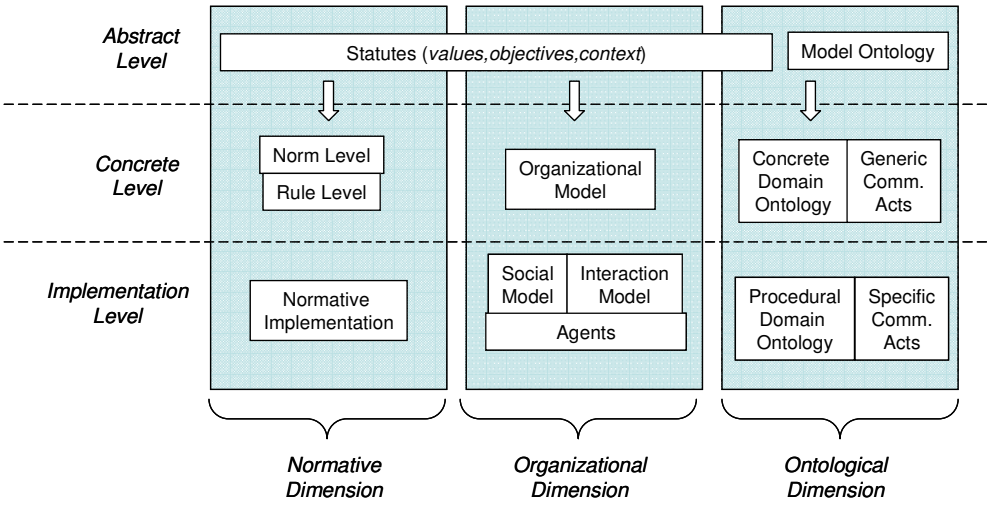

Figure 29 – Levels and dimensions of the OMNI framework, from [Vàzquez-Salceda *et al*., 2005]

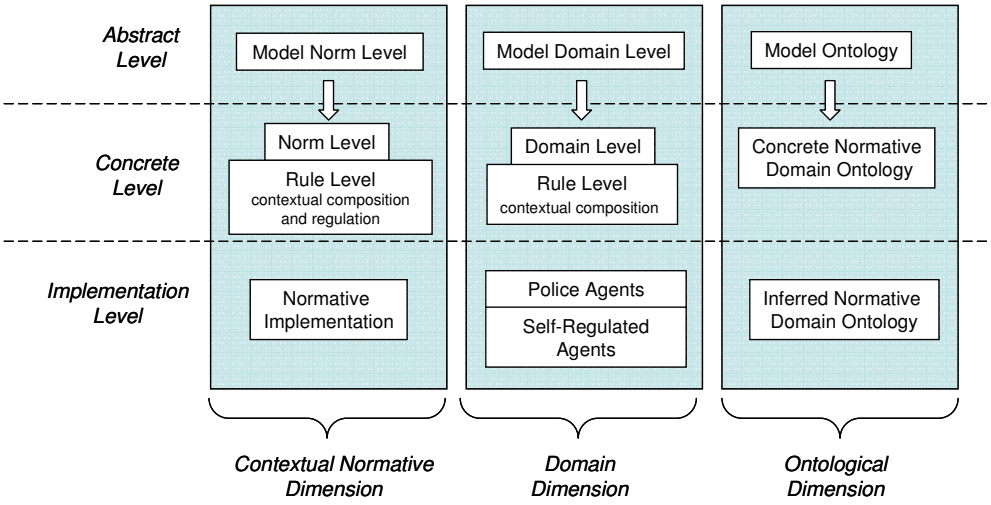

Figure 30 – Levels and dimensions of DynaCROM, based on Figure 29

In the *Contextual Normative Dimension* of DynaCROM, its *Abstract Level* defines the 'Model Norm Level' instead of the statutes of an organization, which is defined in OMNI. This is because, the statutes of an organization is not a concern of the DynaCROM methodology, but the definition of the contextualized abstract norms of the application domain. Those abstract norms should be concretized with domain values in the 'Norm Level' and, then, rules should be specified in the 'Rule Level', both levels defined by the *Concrete Level* (as in OMNI). In

DynaCROM, besides the refinement process from norms (without operational semantics) to rules (capable to be computed), the 'Rule Level' also holds rules written to compose contextual related norms. The explicit use of contexts as part of the norm definition process differentiates DynaCROM from OMNI. Thus, it makes necessary to employ those extra (*contextual*) rules to give flexibility for the norm composition process.

In the *Domain Dimension* of DynaCROM, its *Abstract Level* defines the 'Model Domain Level', *i.e.*, the domain concepts that will have their actions regulated. Those concepts should be concretized with instances values in the 'Domain Level' and, then, rules should be specified in the 'Rule Level' for composing related concepts and, consequently, their data. Both 'Domain and Rule Levels' are defined in the *Concrete Level*.

In the *Ontological Dimension* of DynaCROM, its *Abstract Level* defines the 'Model Ontology', *i.e.*, the DynaCROM ontology extended for representing the domain concepts that will have their actions regulated. The ontology should be concretized with both domain values and norms in the *Concrete Level*, resulting in a 'Concrete Normative Domain Ontology'.

All the tasks presented above should be done by the system developer of the DynaCROM NMAS.

In the *Implementation Level* of the DynaCROM *Domain Dimension*, 'Police Agents' or 'Self-Regulated Agents' (depending on a system developer's decision) enforce the norms of the *Contextual Normative Dimension* based on the 'Inferred Normative Domain Ontology', which is specified in the *Ontological Dimension*. The ontology is automatically inferred by DynaCROM according to the rules applied in the 'Norm Level' and 'Domain Level' (all from the *Concrete Level*).

DynaCROM does not have an explicit support to: the 'Interaction Model' and other agents besides the (MOSES) police agents and (SCAAR) self-regulated agents (model and agents from the *Implementation Level* of the *Organizational Dimension* of OMNI); 'Generic Comm. Acts' from the *Concrete Level*; and, 'Procedural Domain Ontology' and 'Specific Comm. Acts' from the *Implementation Level*, the last two levels from the *Ontological Dimension* of OMNI.

In the OMNI subsection (2.1.1.1), presented in chapter 2 of this thesis, it was pointed out some limitations of OMNI and also mentioned that DynaCROM could be used as a possible solution for those limitations. In next, those limitations are rewritten in order to describe possible DynaCROM solutions for them.

In the OMNI subsection of chapter 2, it was pointed out that: "In order to support the development of closed systems and open, flexible environments,

OMNI presents a rigid specification of its structure, defining particular fields for the description of scenes, roles and groups of roles."

In DynaCROM, particular fields for the description of domain concepts can be specified and implemented as concepts' fields represented in a DynaCROM domain ontology. DynaCROM is implemented as an active agent's behavior, which continuously reads its domain ontology. So, each time any structural information is updated in the DynaCROM domain ontology (*e.g.*, updates in concepts, concepts' fields or concepts' relationships), it is automatically forwarded to the application agents that spontaneously incorporate the DynaCROM behavior, seeking to receive updated system data.

"There are no normative aspects further than the ones for organizations, roles, group of roles, agent interactions and agents (only norms for roles, group of roles, scene and transition can be specified)." This limitation of OMNI can be solved by additions in the DynaCROM domain ontology of new concepts for representing new normative aspects and, then, with instantiations of their respective norms. This flexibility in the DynaCROM solution is due to the perception that specific domain aspects are also important for regulation in NMAS.

"The organization entity is not explicitly present. An organization is formed by listing all its institutional roles (*e.g.*, managers, directors, president, etc.) and represented when agents play these roles." This limitation of OMNI is solved by the creation and instantiation of the *Organization* concept in the DynaCROM ontology. The advantage of having a concept exclusively for representing organization instances is that their specific data (*e.g.*, norms and relationships) can be more easily implemented and managed.

"Currently, OMNI does not provide a solution for the implementation and integration of its specifications in a given MAS." This limitation of OMNI can be solved by DynaCROM, as will be exemplified in the following subsection.

6.1.1. DynaCROM_OMNI at Work

Currently, OMNI does not provide a tool for implementing its specifications. In order to exemplify how DynaCROM can be used to implement OMNI specifications, some of them presented in [Vàzquez-Salceda *et al.*, 2005] are considered.

Table 8 presents an example of a role description in OMNI. The two obligation norms are classified independently of the contexts in which they are, *i.e.*, the norms hold in any organization around the world.

Table 8. PC member role description in OMNI, from [Vàzquez-Salceda *et al.*, 2005]

Id	PC_member
Objectives	paper_reviewed (Paper, Report)
Sub-objectives	{read(P), report_written(P, Rep), review_received(Org, P, Rep)}
Rights	access-confmanager-program (me)
Norms and rules	PC_member is OBLIGED to understand English IF paper_assigned THEN PC_member is OBLIGED to review paper BEFORE given deadline IF author of paper_assigned is colleague THEN PC_member is OBLIGED to refuse to review ASAP
Type	external

The phases of the DynaCROM methodology for contextualization and concretization of the OMNI norms are based on their (OMNI) descriptions. The remaining phases for the effective implementation of the role description in a NMAS can be carried out by representing the norms in a DynaCROM domain ontology, as illustrated in Figure 31.

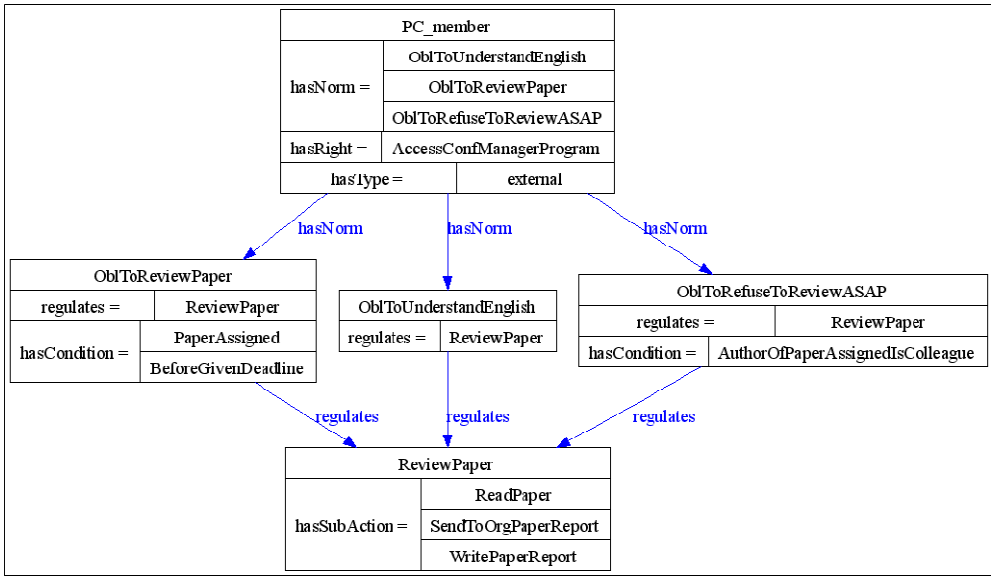

Figure 31 – Representation in DynaCROM of a OMNI role description

The OMNI *PC_member* role is represented by a DynaCROM *Role* instance, which has norms (*OblToUnderstandEnglish*, *OblToReviewPaper* and *OblToRefuseToReviewASAP*), rights (*AccessConfManagerProgram*) and a type (*external*). The objective (*paper_reviewed (Paper, Report)*) and its sub-objectives, both specified in OMNI, are implemented in DynaCROM as a regulated action (*ReviewPaper*) and its sub-actions, respectively.

The norms of the *PC_member* role are enforced every time that an agent playing the role informs the review of its assigned paper to the program chair

agent (*i.e.*, when the agent executes the *ReviewPaper* regulated system action). More precisely, the enforcement occurs when *PC_member* agents execute the '*reviewAPaper(...)*' method presented in Code 30.

Code 30. Part of a method to be implemented by *PC_member* agents

```
(1)public abstract class APlanForPCMemberAgt{
(2) public Object reviewAPaper(Object assignedPaper,
                               String programChair,…){
(3)  Object reviewedPaper[] = new Object[1];
(4)  reviewedPaper[] = reviewPaper(assignedPaper);

(6)… return reviewedPaper[] } }
```

Figure 32 illustrates an example in which the norm for accepting paper reviews before the defined deadline is enforced and Code 31 presents the example codified in MOSES. The *ProgramChair* agent sends an INFORM message to the *PC_member* agent with the paper to be reviewed by him. When the message arrives at the *ProgramChairPolice* agent, he verifies the parameter value of the 'startsWith' performative. As the message is an informative message (*i.e.*, the parameter value of the 'startsWith' performative is equal to "INFORM", see line 2 from Code 31) and no obligation is defined for the message, then, the *ProgramChairPolice* just forwards the received message to its recipient (the *PC_member* (line 3)).

When the message arrives at the *PC_memberPolice* agent, he verifies the parameter value of the 'startsWith' performative. As the message is an informative message (line 7) and no obligation is defined for the message, then, the *PC_memberPolice* just delivers the received message to its recipient (*PC_member* (line 8)).

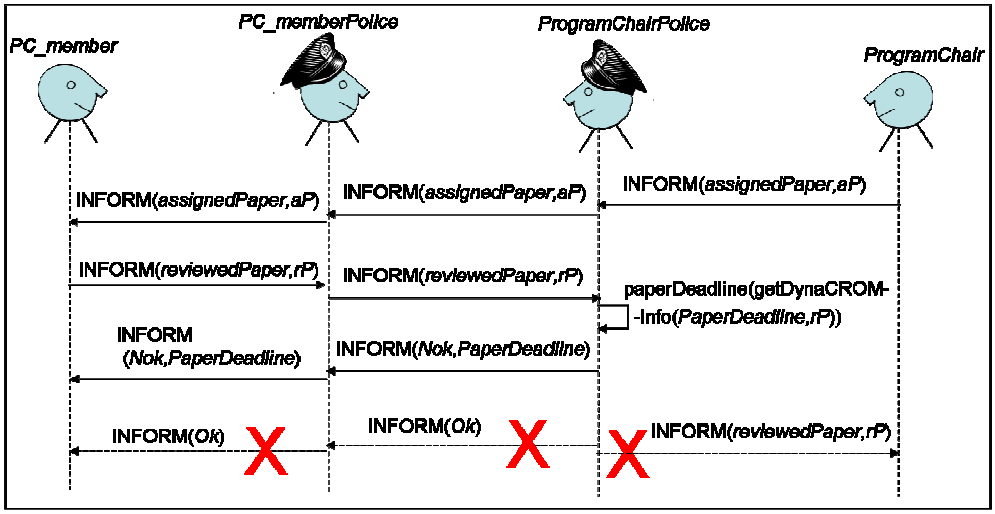

Figure 32 – DynaCROM_MOSES enforcing the paper deadline norm

Code 31. Part of a MOSES law for enforcing the paper deadline norm

```
(1)  public void sent(…){
(2)   if (message.startsWith(“INFORM(assignedPaper") OR
          message.startsWith(“INFORM(reviewedPaper")){
(3)    doForward();
(4)    return;
(5)  }}

(6)  public void arrived(…){
(7)   if (message.startsWith("INFORM(assignedPaper")){…
(8)    doDeliver();
(9)    return;
(10)  }

(11)  if (message.startsWith("INFORM(reviewedPaper")){…
(12)   doAdd("paperDeadline(" + getDynaCROMInfo(PaperDead-
            -line, rP + ")");
(13)   if “OblToReviewPaper” isIn domainRole.hasNorm{…
(14)    doImposeObligation("reviewPaper",1,"sec");
(15)   return;
(16)  }}

(17)  public void obligationDue(Term obligationTerm){
(18)  if (obligationTerm.equals("reviewPaper")){…
(19)   if (paperDeadline.less(currentDay)){
(20)    doForward(CS.toString(),“INFORM(Nok,PaperDeadline)",
                 sourceAddress); …}
(21)   else{
(22)    doDeliver;
(23)    doForward(CS.toString(),“INFORM(Ok)”,sourceAddress);
(24)  …} …} …}
```

When the *PC_member* sends an INFORM message to the *ProgramChair* with the paper reviewed by him, then, the *PC_memberPolice* verifies the parameter value of the ‘startsWith’ performative. As the message is an informative message (line 2) and no obligation is defined for the message, then, the *PC_memberPolice* just forwards the received message to its recipient (the *ProgramChair* (line 3)).

When the message arrives at the *ProgramChairPolice* agent, he asks DynaCROM the information about the paper deadline and adds the returned value in the ‘*paperDeadline*’ variable (line 12). Then, the *ProgramChairPolice* checks the obligation to review the paper before its deadline (lines 13 and 14).

The norm is enforced if the paper deadline is less than the current day (informed by DynaCROM). In this case, an INFORM(*Nok*,*PaperDeadline*) message is sent by the *ProgramChairPolice* to the *PC_member* in order to report him that his message was not delivered because an error occurs with the paper deadline.

When the *PC_member* sends the paper before or in the day of its deadline, then, the message with the paper reviewed is delivered to its recipient (the *Pro-*

gramChair (lines 21 and 22)) and an INFORM(*Ok*) message is sent by the *ProgramChairPolice* to the *PC_member* (line 23) in order to inform him that his message was delivered.

In order to enforce the paper deadline norm according to the SCAAR solution, the control hook specific of the action to review a paper (represented by the execution of the '*reviewAPaper(...)*' method) triggers the agent's enforcement core for verifying if the execution of the action is compliant to its norms.

Figure 33 illustrates an example in which the paper deadline norm is enforced and Code 32 presents the example codified in SCAAR. A system developer wrote the SCAAR norm to regulate the *PC_member* agents from his NMAS when they send back to the *ProgramChair* agent the assigned papers with their reviews. SCAAR verifies if the deadline of the reviewed paper, informed by a *PC_member* agent via the '*agtReviewedPaper.hasPaperDeadline*' variable, is less than the current day – informed by DynaCROM via the '*currentDay*' variable – all in line 3. The norm is enforced every time that it is presented in the analyzed role due to the verification occurred in lines 4 and 5.

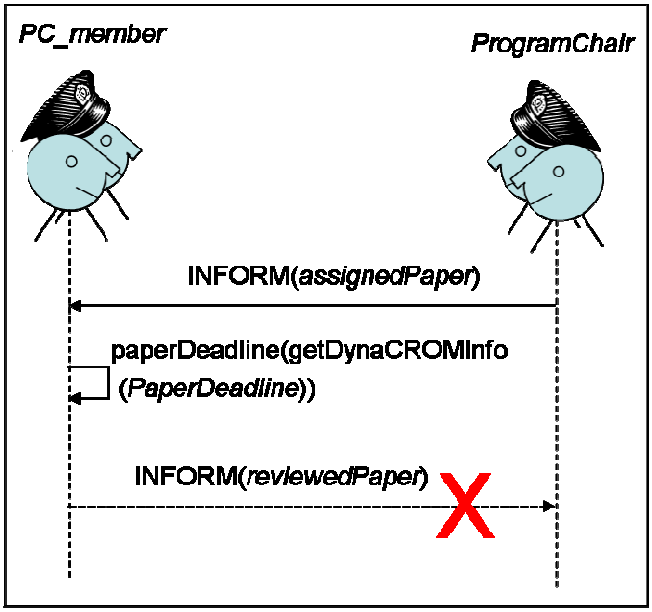

Figure 33 –DynaCROM_SCAAR enforcing the paper deadline norm

Code 32. Part of a SCAAR code for enforcing the paper deadline norm

```
(1) SCAARNorm_OblPCMemberToReviewPaperBeforeDeadline:
(2) [OBLIGED(agt DO reviewAPaper(…) AND
(3)  (agtReviewedPaper.hasPaperDeadline =< currentDay))
(4)  IF(agt BE in Role AND (agtRole == domainRole) AND
(5)   (“OblToReviewPaper” isIn (domainRole.hasNorm))]
```

Another example of specification in OMNI given in [Vàzquez-Salceda *et al.*, 2005] and that can be implemented by DynaCROM is group descriptions. Table 9 presents an example of a group description in OMNI and Figure 34 illustrates the group description represented in a DynaCROM domain ontology instance. The DynaCROM ontology (originally presented in Figure 6) was extended, for the example, with a concept named *GroupOfRoles* for implementing the groups described in OMNI. Then, the concept was instantiated with the *Organizers* group, which is represented by a set of roles (*e.g.*, *GeneralChair* and *PCChair*) and its norm (*PrhToSumitAPaper*).

Table 9. Description of the organizer group in OMNI, from [Vàzquez-Salceda *et al.*, 2005]

Group id	Organizers
Roles	{PC-Chair, website manager, general chair, local organizer}
Norms and Rules	IF author is member of Organizers THEN author is FORBIDDEN to submit a paper

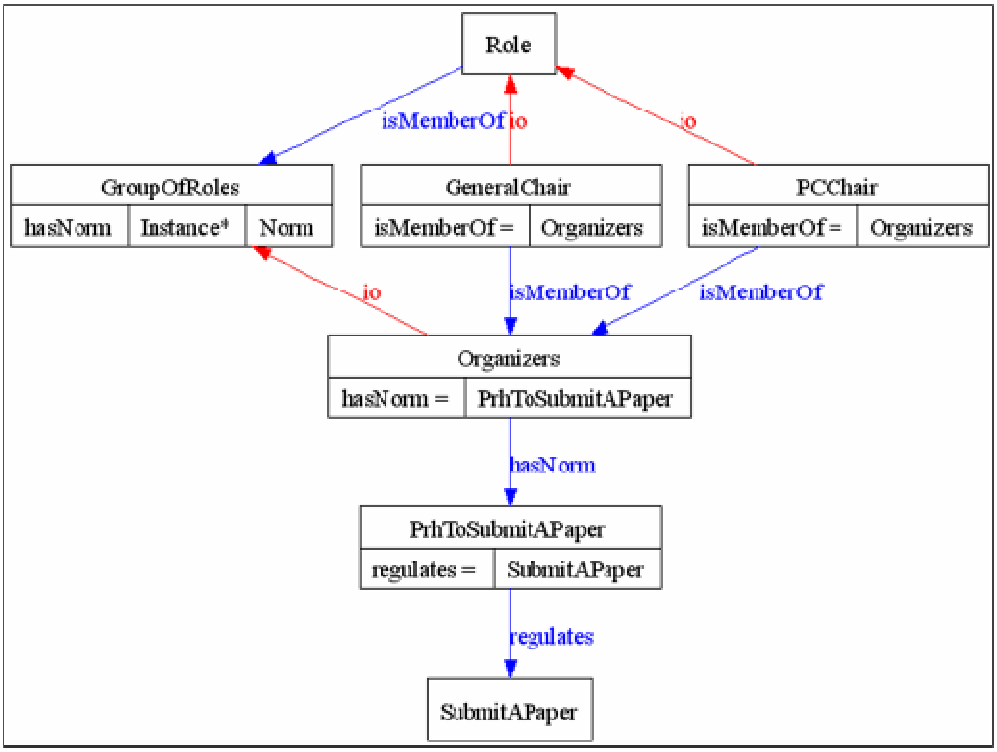

Figure 34 – Representation in DynaCROM of a OMNI group description

Code 33 presents an example of a DynaCROM rule that states that a given role will have its norms composed with the norms of its group. More precisely, considering *GeneralChair* as an example of the given role, the following composition process is executed, according to the domain ontology instance illustrated in Figure 34: in (4), the '*?GroupOfRoles*' variable is instantiated with the *Organizers*

inferred value, when the '*?Role*' variable is instantiated with the *GeneralChair* given value; in (3), the '*?GNorms*' variable is instantiated with the *PrhToSubmitAPaper* inferred value; and in (2), the inferred norm is added as a new norm of *GeneralChair*.

The result of the norm composition process is that, when agents are playing the *GeneralChair* role (or any other role from the *Organizers* group of roles), they are prohibited to submit a paper.

Code 33. Adding the norms of a group in its roles

```
(1)[DynaCROMRule_RoleWithGroupOfRolesNorms:
(2) hasNorm(?Role,?GNorms)
(3)  <- hasNorm(?GroupOfRoles,?GNorms),
(4)     isMemberOf(?Role,?GroupOfRoles)]
```

The norms of a group of roles are enforced in its roles likewise any other contextual norms are enforced in a DynaCROM NMAS, *i.e.*, by using MOSES, SCAAR or other third-party enforcer integrated with DynaCROM.

6.2. ISLANDER and AMELI

As described in chapter 2 of this thesis (more specifically in subsection 2.1.1.2), EI (meaning *Electronic Institutions*) is proposed as a solution for modeling electronic agent-based institutions.

The EI model is based on four elements: *dialogic framework*, *scene*, *performative structure* and *norm*. In this sense, a MAS is understood as a type of *dialogical system*.

The *dialogic framework* element of an EI is defined as a tuple DF = {*O*, *L*, *I* R_I, R_E, R_S} where:

- *O* stands for an ontology, *i.e.*, the vocabulary that defines the possible values for the concepts in a given domain;
- *L* stands for a content language that allows for the encoding of the knowledge to be exchanged among agents using the vocabulary offered by the ontology;
- *I* is the set of illocutionary particles composed by: (i) an agent variable or identifier, (ii) a role variable or identifier, (iii) the addressee(s) of the message which can be an agent or group of agents, (iv) an expression of the content language, and (v) a time variable or constant (*e.g.*, timeouts, which provoke transitions between states in a scene).

- R_I is the set of internal roles;
- R_E is the set of external roles;
- R_S is the set of relationships over roles.

The *scene* element of an EI is, in broad terms, a conversation protocol played by a group of agents. The protocol is specified by a finite state directed graph where the nodes represent the different states of the conversation and the directed arcs connecting the nodes are labeled with the actions that make the scene state evolve. The graph has a single initial state and a set of final states representing the different endings of a conversation. For the correct evolution of a conversation protocol, the minimum and maximum numbers of agents per role in it have to be defined.

In a scene, contextual information gives the interpretation of a received message. This interpretation is based on what has been said by the sender to the recipient, and restricts the valid messages in a certain moment (state) of the conversation.

The *performative structure* element of an EI can be viewed, from a structural point of view, as a *network of scenes* mediated by transitions. It specifies how agents navigate from one scene to another constrained by rules, which define the relationships among scenes. More concretely, a performative structure defines how agents, depending on their role, can move among different scenes and when new conversations will be started, taking into account the relationships among the different scenes. In short, the performative structure defines what participating agents are *permitted* to do within the institution depending on their role.

Some agent's actions within scenes may have consequences that either limit or expand its possible subsequent actions to be performed outside the scope of the scene. Those consequences are captured in EI by using norms.

In order to implement the EI model, AMELI, an agent-based middleware for EI, is proposed in [Esteva *et al.*, 2004]. The middleware is also an infrastructure that mediates agents' interactions while enforcing institutional norms.

An infrastructure for EI has to satisfy the following requirements, all outlined in [Esteva *et al.*, 2004]: it must (i) facilitate agent's participation within the institution; (ii) enforce institutional rules; (iii) prevent participating agents from jeopardizing the functioning of institutions; (iv) be architecturally neutral; (v) interpret any specification to guarantee re-usability and domain independence; and, (vi) be scalable.

Considering a DynaCROM NMAS implemented as an EI, then, concerning the normative aspect, the requirements outlined above for AMELI are also satisfied by DynaCROM. DynaCROM satisfies: (i) because its solution is automatically applied in the agents of an institution when they incorporate the DynaCROM behavior; (ii) because DynaCROM is enhanced with a third-party solution for enforcing institutional rules; (iii) because it enforces institutional rules; (iv) because its solution was developed as an active behavior that can be encapsulated by using the implementation unit provided by the chosen agent platform; (v) because the specification represented in a DynaCROM domain ontology can be reusable and it can be written for any domain; and, (vi) because its solution is incorporated in each agent, so, the risk of overload due to the regulation in the system does not exist and it is scalable for the number of agents that are enhanced with the DynaCROM behavior.

Currently, AMELI implements its EI architecture by using four types of agents: (i) *Institution Manager*; (ii) *Transition Manager*; (iii) *Scene Manager*; and, (iv) *Governor*.

Institution Manager Agents are in charge of starting an EI, authorizing agents to enter the institution, as well as managing the creation of new scene executions. Those agents keep information about all participants and all scene executions. *Transition Manager Agents* are in charge of managing transitions for controlling agents' movements between scenes. *Scene Manager Agents* are responsible for governing a scene execution. *Governor Agents* are devoted for mediating the participation of an external agent within the institution. There is one: *Institution Manager* per institution execution; *Transition Manager* per transition; *Scene Manager* per scene execution; and, *Governor* per participating agent.

Concerning the architectural aspect, the four types of agents used by AMELI can be identified, in the DynaCROM solution, by their respective functionalities. Because, in AMELI, the norm enforcement is based on the agents' external behavior, then, the integration of DynaCROM with MOSES is used in the comparison, instead of the integration with SCAAR (based on the agents' internal behavior).

A DynaCROM NMAS does not explicitly implement the scene element of the EI model. The DynaCROM methodology suggests that the system developer specifies the abstract classes and methods of his NMAS and, then, agent developers can freely implement their agents according to those specifications. This way, the agents of the types *Institution Manager*, *Transition Manager* and *Scene Manager* of AMELI are not necessary in the DynaCROM solution.

Because a DynaCROM NMAS is in essence an open system, agents can enter or leave it without any restriction or need to ask for permission. The only obligation specified is, when the system developer decides to enforce the system norms (*i.e.*, by using a third-party norm enforcer integrated with DynaCROM, as the SCAAR or MOSES one), then, the addition of the DynaCROM behavior in the agents that will perform in the system is mandatory. This obligation is due to guarantee the correct execution of the norm enforcement solution in the system as a whole, *i.e.*, in all agents that are currently playing in the system.

The only agent type of AMELI that has a respective agent in DynaCROM is the *Governor* one. Table 10 summarizes the actions contained in a message that an agent can send to its governor, both agents from AMELI. Three types of actions are permitted: *illocutionary* (illocutions that agents try to utter within scenes), *motion* (movements between scenes and transitions, and the other way around) and *information request* (scenes reachable from a transition, transitions reachable from a scene, agent's obligations, scene's state and scenes' participants).

The *illocutionary* actions that an agent can perform in a DynaCROM NMAS can be compared to the abstract classes and methods specified by the system developer. In DynaCROM NMAS, it is the responsibility of the agent developer to implement those classes and methods according to their specifications. This way, agents can freely perform illocutionary actions in a DynaCROM MAS because the normative layer of the system is in charge of regulating those actions.

The *motion* actions that an agent can perform in a DynaCROM NMAS do not need to be specified. As a DynaCROM NMAS is in essence an open system, then, agents can move to any type of environment created for the system.

Information requests can be made, at any time, in a DynaCROM NMAS, by agents simply calling the provided *getDynaCROMInfo(…)* method. The method must have its parameter value filled with the agent's desired information, which is represented in the system's domain ontology instance.

Table 10. Possible requests from an agent to its governor, from [Esteva *et al.*, 2004].

Action	**Description**
enterInstitution	Request to enter the institution
moveToTransition	Request to move from a scene to a transition
moveToScenes	Request to move from a transition to several scenes
saySceneMessage	Request to state a message in a scene
accesScenes	Ask for the scenes the agent can join from a transition
accesTransitions	Ask for the transitions the agent can join from a scene

agentObligations	Ask for pending obligations
sceneState	Ask for a scene's current state
scenePlayers	Ask for agents in a scene

As presented above, DynaCROM satisfies the requirements to be an EI infrastructure. Moreover, concerning the normative and architectural aspects, DynaCROM can also be integrated with AMELI in order to enforce institution (contextual) norms. The main advantage of using DynaCROM in EI is that contextual norms can be used in the NMAS and those norms can be dynamically managed in the system by simply updating the domain ontology instance and/or the rule file, which specifies customized compositions of contextual norms.

Other advantages of using DynaCROM in EI can be outlined. In the EI subsection (2.1.1.2) presented in chapter 2 of this thesis, some limitations of EI were pointed out. Now, those limitations are rewritten for presenting the advantages of using DynaCROM to resolve them.

As written in [Esteva *et al.*, 2004], "*an EI defines a normative environment that shapes agents' interactions at execution time.*" Hence, in an EI: (i) there are no normative aspects further than the ones for roles, agent interactions and agents; (ii) the specification of an EI is often too *society-centric* in the sense that it completely fixes agent interactions in rigid protocols and interfaces; (iii) external agents have no room for autonomous behavior, *i.e.*, they blindly follow defined protocols with the only autonomy to accept or reject them; (iv) all possible interactions among agents have to be defined; (v) it is difficult, if not impossible, to describe indirect interactions; this is due to the fact that all interacting activity taking place in an EI is purely dialogic by means of direct communication between the agents; and, (vi) the structure of an EI is static and, so, cannot evolve at system runtime.

Those limitations can be minimized by using the DynaCROM domain ontology in which specific domain abstractions can be represented and, then, regulated. The advantages of using DynaCROM in EI are: for (i), domain normative aspects (*e.g.*, political, economical or religious ones) can also be implemented; for (ii), the specification of an EI does not need to fix agent interactions; for (iii), agents can violate norms, however, depending on the decision of the system developer about the implementation of a mechanism for *a priori* or *a posteriori* norm enforcement, a violation may not have any effect or a punishment can be given, respectively; for (iv), only the interaction norms of the system need to be defined; for (v), indirect interactions can be regulated by the specification of rules that

compose related normative contexts; and for (vi), the structure of an EI is able to evolve at system runtime by updating the ontology concepts and/or by customizing different compositions of contextual norms.

6.3. $\mathcal{M}$OISE$^+$ and $\mathcal{S}$-$\mathcal{M}$OISE$^+$

In chapter 2 of this thesis (more specifically, in its subsection 2.1.1.1), the $\mathcal{M}$OISE$^+$ model is presented as an important work for the modeling of MAS based on electronic organizations. The model defines structural, functional and deontic dimensions of a MAS in such a way that those dimensions can be specified independently of one another.

Figure 35 illustrates an example of representation of the structural dimension for a soccer game, according to the $\mathcal{M}$OISE$^+$ model. There, the *team* group is composed by its *attack* and *defense* sub-groups. Roles and role relations are defined according to their specific groups. For instance, the coach role is defined by the *team* group and it has authority (*i.e.*, a relationship of superiority among roles) over the *player* role, which is defined by the *middle*, *leader* or *attacker* sub-roles of the *attack team*.

Figure 36 gives an overview of a social scheme to score a soccer goal. For each role to be played by agents, missions are defined. A *mission* is a set of coherent goals that an agent can commit to. For instance, an agent playing the *middle* role is committed to *m7*, a mission defined by the set: {g7,g6,g9,g2,g0}.

For the *m7* mission, in the deontic dimension of the $\mathcal{M}$OISE$^+$ model for the soccer game, it is defined: {per($\rho_{goalkeeper}$,m7,Any)}, meaning that an agent playing the *goalkeeper* role is permitted to commit to the *m7* mission at any time.

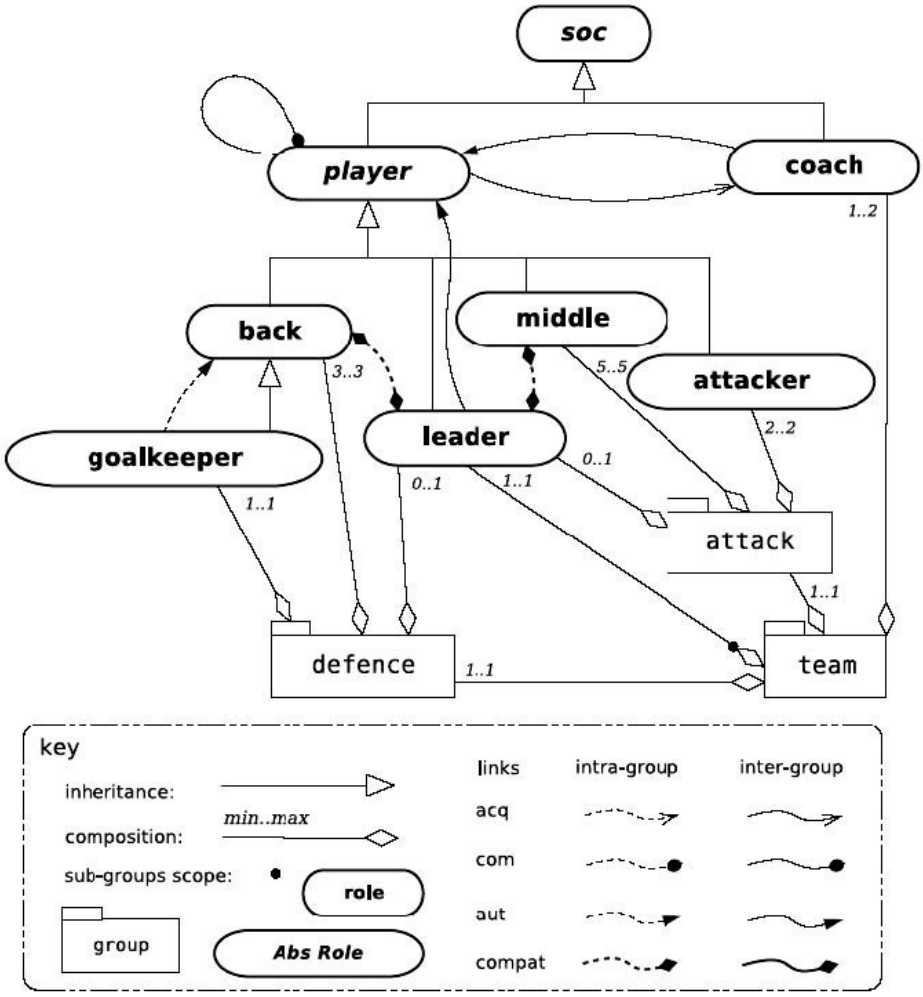

Figure 35 – Structure of a soccer team, from [Hübner *et al.*, 2002]

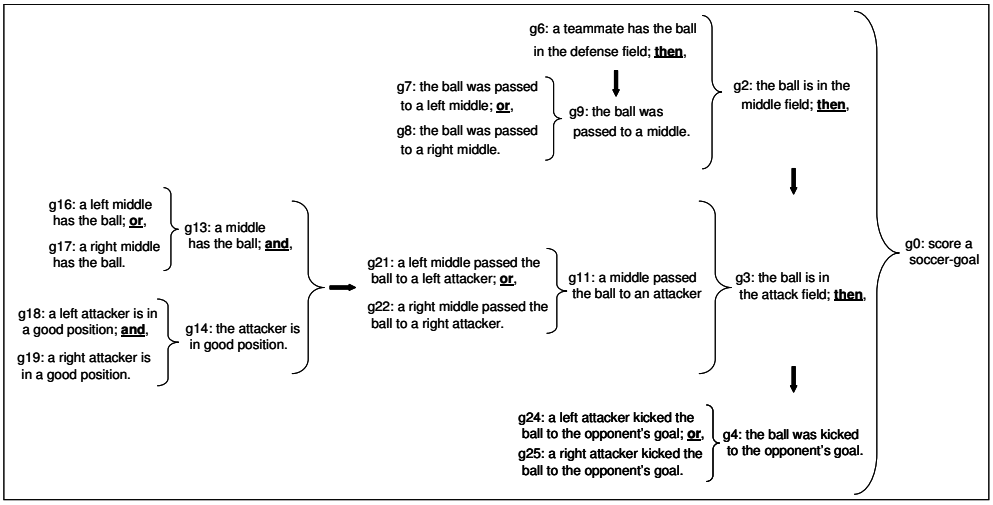

Figure 36 – A social scheme to score a soccer goal

In order to summarizes the three dimensions of the $\mathcal{M}$OISE$^+$ model in the individual, social and collective levels, Figure 37 was created inspired by Figure 29. Then, Figure 38 was created for enabling the comparison of DynaCROM with the $\mathcal{M}$OISE$^+$ model.

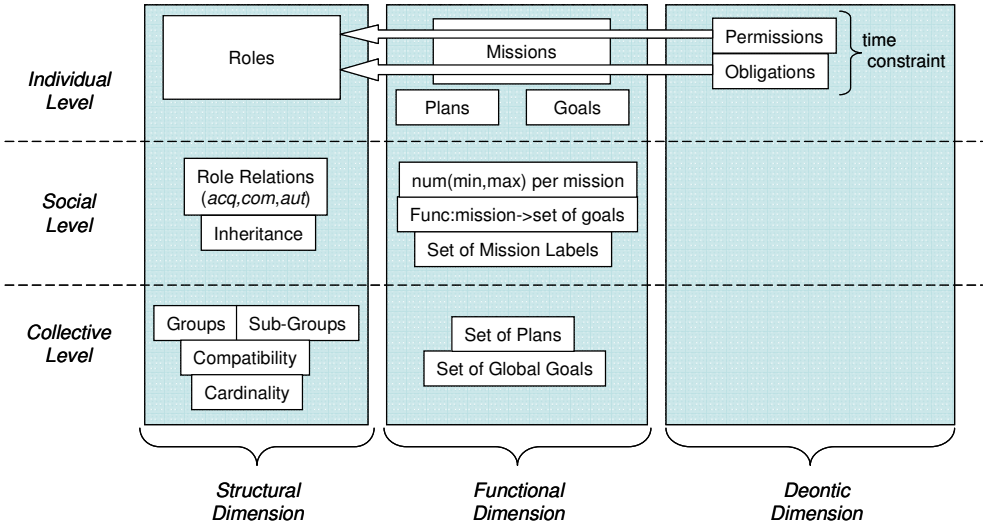

Figure 37 – Levels and dimensions of *M*OSES$^+$

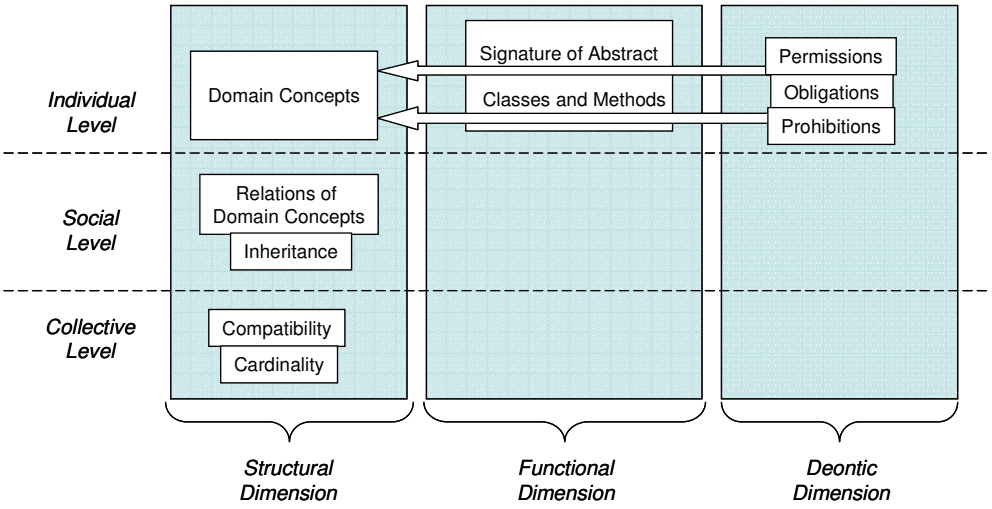

Figure 38 – Levels and dimensions of DynaCROM compared to *M*OSES$^+$

In the *Structural Dimension*, DynaCROM relaxes the individual level of the $\mathcal{M}$OISE$^+$ model by letting other domain concepts, besides roles, to be defined. This way, the structure of domain concepts can be refined and, moreover, regulated by the deontic dimension. The *Social Level* of the $\mathcal{M}$OISE$^+$ model is also

relaxed by DynaCROM in the sense that role relations are not limited to three types, moreover, relations of domain concepts can also be defined.

In the *Functional Dimension*, DynaCROM does not specify all possible plans to achieve a goal. DynaCROM defines abstract classes and methods, letting agents to freely create their plans. The assumption is that agents know how to act in the MAS in which they want to perform to achieve a goal, so, the dimension should support them and not impose to them only choices in the process.

In the *Deontic Dimension*, DynaCROM also deals with prohibitions instead of only permissions and obligations. However, DynaCROM does not deal with time constraints as the $\mathcal{M}$OISE$^+$ model does.

In order to support the implementation of the specifications done by following the $\mathcal{M}$OISE$^+$ model, the $\mathcal{S}$-$\mathcal{M}$OISE$^+$ organizational middleware is proposed in [Hübner *et al.*, 2006]. The middleware permits external agents to access the organizational layer through a special agent provided by the middleware, called *OrgManager*, which is in charge to ensure organizational constraints. Two kinds of constraints are supported by the middleware: *hard* and *soft* constraints.

Hard constraints are those that must be enforced to maintain the organizational entity in a consistent state. Since these constraints cannot be violated by any agent, they should be implemented in the middleware.

Soft constraints are related to the deontic dimension and are not guaranteed by the middleware, since agents are supposed to autonomously decide to follow them or not. This way, those constraints are normally enforced by agent reasoning capabilities and a sanction system may also be used when soft constraints are violated.

Some limitations of AMELI, the middleware developed for enforcing interaction norms in EI, can be similarly outlined for $\mathcal{S}$-$\mathcal{M}$OISE$^+$. Those limitations of $\mathcal{S}$-$\mathcal{M}$OISE$^+$ are: (i) there are no normative aspects further than the ones for group of roles, roles, agent interactions and agents; (ii) the specification of an organization is too *society-centric* in the sense that it completely fixes agent interactions in rigid protocols and interfaces; (iii) external agents have no room for autonomous behavior, *i.e.*, they blindly follow defined missions with the only autonomy to accept or reject them; and, (iv) all possible plans, goals and interactions among agents have to be defined.

Those limitations can be minimized by using the DynaCROM domain ontology. The advantages for $\mathcal{S}$-$\mathcal{M}$OISE$^+$ can be deducted by their corresponding advantages for the AMELI limitations, all presented in subsection 6.2 of this thesis.

6.4. XMLaw and M-Law

In [Paes *et al.*, 2005], it is presented how interaction laws can be specified by using XMLaw. XMLaw encompasses a declarative language and a software implementation. The language supports a conceptual model for developing laws in open MAS. The model is composed by static and dynamic definitions. The implementation is to allow the enforcement of laws through the interception of messages changed between interacting agents. Regulation takes place at the level of interaction laws in order to achieve higher degrees of predictability.

Comparing DynaCROM with XMLaw, three main differences can be pointed out. The first difference is about the definition of regulatory contexts in both solutions. In XMLaw, regulatory contexts are defined by interaction laws and, so, regulations are restricted to this level. DynaCROM supports the definition of interaction laws (*i.e.*, interaction norms), which can also be composed with other domain laws (*e.g.*, environment, organization and role norms). Thus, others regulatory contexts, besides the interaction one, can be defined and used in the DynaCROM solution.

The second difference between DynaCROM and XMLaw is how the enforcement is carried out when agents do not act according to the defined laws. In XMLaw, law enforcement is only carried out *a priori*, *i.e.*, the messages exchanged between interacting agents are intercepted for checking law compliance and, then, enforced. DynaCROM also supports *a priori* norm enforcement and, moreover, the *a posterio*ri one.

Finally, the third difference between DynaCROM and XMLaw is that, in DynaCROM, the regulated entities of a MAS (*e.g.*, traffic signals) have to exist and be perceived by acting entities, whether police or participating (*e.g.*, driver) agents. As the law enforcement in XMLaw is concentrated on the interception of changing messages, then, the regulated parts of the system does not need to be known.

As previously mentioned in chapter 2 of this thesis (more specifically, in its subsection 2.2.5), M-Law [Paes *et al.*, 2006 and 2007a] is a middleware devel-

oped to enforce, at agents' runtime, interaction laws, which are specified by following the XMLaw conceptual model.

Some limitations of AMELI, the middleware developed for enforcing interaction norms in EI, can be similarly outlined for M-Law, the middleware developed for enforcing interaction norms in XMLaw *Scenes*. Those limitations of M-Law are: (i) there are no normative aspects further than the ones for roles, agent interactions and agents; (ii) XMLaw elements (all presented in subsection 2.2.5) often have their specifications, for an application domain, too *society-centric* in the sense that it completely fixes agent interactions in rigid protocols and interfaces; (iii) external agents have no room for autonomous behavior, *i.e.*, they blindly follow defined protocols with the only autonomy to accept or reject them; (iv) all possible interactions among agents have to be defined; (v) it is difficult, if not impossible, to describe indirect interactions; this is due to the fact that all interacting activity taking place in a XMLaw *Scene* is purely dialogic by means of direct communication between the agents; and, (vi) the structure of a XMLaw *Scene* is static and, so, cannot evolve at system runtime.

Those limitations can be minimized by using the DynaCROM domain ontology. The advantages for M-Law can be deducted by their correspondingly advantages for the AMELI limitations, all presented in subsection 6.2 of this thesis.

6.5. Discussion

In this chapter, some implementing solutions for MAS modeling and others for norm enforcement are compared to DynaCROM.

For OMNI, the comparison includes the levels and dimensions of both solutions, and also the DynaCROM solution for implementing OMNI specifications.

For ISLANDER and AMELI, the comparison includes normative and architectural aspects considering a DynaCROM NMAS implemented as an EI, integration of DynaCROM with AMELI in order to enforce institution (contextual) norms and advantages of using a DynaCROM domain ontology for resolving some limitations of EI.

For $\mathcal{M}$OISE$^+$ and $\mathcal{S}$-$\mathcal{M}$OISE$^+$, the comparison includes the levels and dimensions of the $\mathcal{M}$OISE$^+$ model and DynaCROM, limitations of $\mathcal{S}$-$\mathcal{M}$OISE$^+$ and the advantages of using a DynaCROM domain ontology for resolving those limitations.

For XMLaw and M-Law, the comparison includes the main differences between DynaCROM and XMLaw, limitations of M-Law and the advantages of using a DynaCROM domain ontology for resolving those limitations.

Furthermore, in chapter 2 of this thesis (more specifically, in its subsection 2.1.1.4), some research questions were proposed for a comparative study conducted among some modeling solutions for MAS engineering. Those questions are rewritten below (in the exact way that they were presented in chapter 2) for presenting, in Table 11, the results of DynaCROM as a modeling solution for MAS engineering. The only question not answered by DynaCROM (the last one) is pointed out as one of the future works proposed in this thesis.

rq.i. Does it explicitly support the organizational normative dimension?

rq.ii. Does its conceptual model have an implemented solution for it?

rq.iii. Does it support the management of norms to be done at system runtime?

rq.iv. Does it provide ways for norm representation with a common understanding for heterogeneous agents?

rq.v. Does it have an editor, preferably a graphical one, to support the writing of its specifications?

rq.vi. Does it have a semi-/automatic solution for the verification of its specifications?

Table 11. Results of DynaCROM as a modeling solution for MAS engineering

	rq.i	rq.ii	rq.iii	rq.iv	rq.v	rq.vi
OMNI	V	X	–	V	X	X
MOISE+	V	V	V	V	X	X
ISLANDER	X	V	V	V	V	V
DynaCROM	**V**	**V**	**V**	**V**	**V**[12]	**X**

[12] DynaCROM specifications can be written by using any third-party ontology editor for representing its norms and by using any third-party rule editor for composing its norms.