IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 2, 2021, accepted January 17, 2022, date of publication January 27, 2022, date of current version February 8, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3147144

Creating a Modeling Language Based on a New Metamodel for Adaptive Normative Software Agents

MARX VIANA[1], PAULO ALENCAR[2], (Member, IEEE), EVERTON GUIMARÃES[3], ELDER CIRILO[4], AND CARLOS LUCENA[5]

[1]Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro 22541-041, Brazil
[2]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[3]Engineering Division, Pennsylvania State University, Malvern, PA 19355, USA
[4]DCOMP, Federal University of São João del-Rei, São João del Rei, Minas Gerais 36307-352, Brazil
[5]Department of Informatics, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro 22541-041, Brazil

Corresponding author: Marx Viana (mleles@inf.puc-rio.br)

This work was supported in part by the Coordination for the Improvement of Higher Education Personnel (CAPES), in part by the National Council for Scientific and Technological Development (CNPq), and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

ABSTRACT The demand for creating increasingly dynamic, autonomous and proactive software systems is challenging for the traditional Multi-agent Systems (MASs) approaches. Such requirement has given rise to adaptive software agents approaches. At the same time, norm is an essential and challenging feature that still tends to be addressed in adaptive MAS. In fact, norms to regulate agent behavior is still a vague concept that has not been properly investigated in terms of modeling and implementation. Even though many researchers have proposed modeling languages to deal with different abstractions, these languages fail to support the modeling of abstractions, such as adaptation and norms. Even more severe is the fact that little has been done to support the systematic design of Adaptive Normative Multi-Agent Systems (ANMASs). To facilitate the design and development of ANMASs, this paper presents a new metamodel, as well as language support, as means to provide tools to enable software developers. The proposed metamodel fosters a better understanding of the way agents are able to change their behaviors to deal with norms and captures interactions between agent's norms and adaptation. To this end, our research is organized into five steps: (i) a literature review to identify the limitations of existing approaches related to ANMAS modeling; (ii) propose a new metamodel to support adaptative and normative concepts; (iii) propose a new language for modeling ANMASs; (iv) perform a qualitative and quantitative evaluation of the proposed language using a real case scenario, and (v) an empirical evaluation. The proposed metamodel and its associated modeling language advances the state of the art in modeling MASs and the approach is assessed in terms of correctness, time and difficulty. Our initial results revealed that our approach can be feasibly applied in a real world application, and is less difficult to apply and requires less time in comparison with a traditional approach. As software applications become more dynamic and adaptive, we believe it is essential to support developers to model MASs with abstractions such as adaptive agents, norms and their relationships. Such information can be foundational to steer future research on modeling adaptive agents capable of understanding and dealing with norms and adaptation.

INDEX TERMS Multi-agent systems, software modeling, normative systems, software adaptation.

I. INTRODUCTION

Software agents emerged as a new technology for building complex systems [1]. These systems are characterized by the distribution and composition of autonomous entities that interact with each other [2]. Multi-agent Systems (MASs) are societies in which these heterogeneous and individually designed entities work to accomplish common or independent goals [3]. Thus, the use of agents to develop complex systems is considered a promising approach in many areas, such as

The associate editor coordinating the review of this manuscript and approving it for publication was Emre Koyuncu.

the Internet of smart Things (IoT) [4], smart traffic lights [5], [6], smart cities [7], energy [8], second life [9] and healthcare systems [10]. Because of the pervasiveness of adaptation and some forms of norms in modern software applications, the proposed approach is relevant not only for MAS, but also for various other domains, such as Systems of Systems (SoS) and the Internet of Things (IoT). Because of their complexity, the successful and widespread deployment of large-scale MASs requires a unifying set of agent-related abstractions to support modeling languages and their respective methodologies.

Modeling languages are defined as languages that support the design and construction of models in a specific domain. In general, a model is a simplification of reality that can help software developers to better understand the systems that they are developing. These models help developers to: visualize the system as it is, or as they wish it to be; specify its structure or behavior; provide guidance during system development, and have a way to record their design decisions. Moreover, models of complex systems need to be built because in most cases even the software professionals who interact with these systems on a daily basis cannot understand the systems in their entirety. For all these reasons, a modeling language is an indispensable element in agent-based software technology as it is an essential tool for designing software systems models.

To reduce the risk of adopting a new technology, it is convenient to present a modeling language as an incremental extension of known and trusted approaches as well as to provide explicit engineering tools that support industry-accepted methods of technology deployment [11]. As a standard, the UML modeling language has been used for object-oriented modeling in both industry and academia although it does not provide the abstractions required for modeling MASs [7]. In addition, there are several MAS modeling languages [12]–[14] that extend UML to model agent-oriented systems but do not provide abstractions that support the modeling of adaptative or normative agent concepts.

However, the ongoing research efforts towards defining more comprehensive and expressive MAS modeling approaches and languages has faced numerous challenges, especially in the case of agents that need to exhibit both normative and adaptive behavior. An agent is considered normative if its behavior is regulated by norms, and it is considered adaptive when it is able to respond to changes in other agents or in the environment. The first challenge is that there is currently a lack of consensus about the understanding of these two key abstractions, namely norms and adaptive behavior. Furthermore, most modeling languages do not define the interaction between these abstractions at design-time and runtime concepts for both the agent's internal and external aspects [12].

Tackling these challenges becomes increasingly relevant as many modern software applications need, for one, to (self-) adapt their behavior and dynamically react to changes in their environments. In addition, they need to be able to define normative (i.e. regulatory) behavior of individual and collective agents. For example, smart traffic lights, an emergent application in the area of the Internet of Things (IoT), require agent models that are able to sense and adapt to changes in other agents and in the environment and that are also subject to norms aiming to regulate their behavior. Another example is the modern healthcare systems, which need agent models that are both adaptive and normative; these systems require adaptations in different dynamic contexts and are often highly regulated. In both examples not only do the models need to capture isolated normative and adaptive behavior, but they also need to understand the changes that affect their adaptive behavior, and vice-versa.

A. SPECIFIC PROBLEMS AND GOAL

Although some modeling languages involve abstractions of adaptation and / or normative concepts ([11], [12], [15], [16], [17]–[20], [21]), these approaches do not explicitly present the adaptive and normative factors that can influence MASs, such as:

- (**#F1**) traditional agent architectures and their associated languages do not provide a mechanism for an agent to adapt to the norms that restrict its behavior ([22]);
- (**#F2**) due to limitation (**#F1**), these languages cannot be used to model many essential aspects in the representation of agents, such as aspects involving adaptation, norms and the interactions between them ([12], [18]);
- (**#F3**) both at design and runtime, changes in agents' behavior and their reactions to changes in the environment need to be represented ([12]);
- (**#F4**) there is a lack of models to capture how agents can adapt their behavior to comply with the new norms adopted ([18]).

Given the above, there is a need for creating approaches that enable the development of multi-agent systems capable of: (**#F1**) understanding and adapting the norms addressed to them in the environment; (**#F2**) supporting a language that allows to represent the concepts of norms and adaptation explicitly; (**#F3**) verifying how these abstractions are related, and (**#F4**) representing mechanisms that assist in the construction of agents capable of adapting to deal with norms.

Based on the challenges and limitations of current approaches, our main goal is to provide a novel MAS modeling approach by: creating a metamodel according to which agents can understand and adapt the norms addressed to them in the environment; creating a modeling language which allows representing the concepts of norms and adaptation explicitly; verifying how these abstractions relate to each other, and representing mechanisms to help the construction of agents capable of adapting to deal with norms.

B. OVERVIEW OF THE PROPOSED APPROACH

An overview of the proposed MAS modeling approach, which can support adaptation and norms, is presented in Fig. 1. The MAS modeling approach involves four phases: Phase 1, which aims at the development of a novel MAS metamodel that supports adaptation and norms; Phase 2,

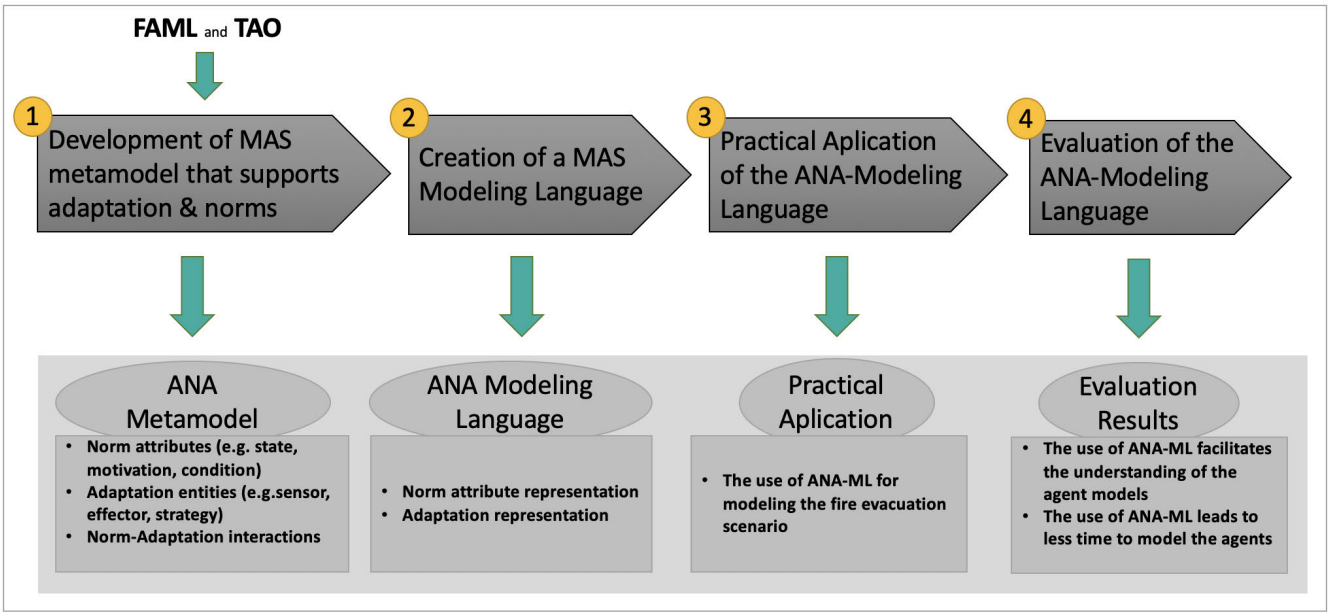

FIGURE 1. An overview of the proposed MAS modeling approach.

which aims at the creation of a MAS modeling language based on the new metamodel; Phase 3, which aims at modeling a real-world practical application, and Phase 4, which aims at evaluating the MAS modeling language.

In Phase 1, we develop a new MAS metamodel called Adaptive Normative Agent (ANA) by extending TAO [23] and FAML [12] metamodels. We extend TAO because it only introduces basic relationships between roles and organizations of agents. In addition, we extend FAML because it already gives support to modeling at different meta-levels, namely: (i) system-level – agent-external in design-time; (ii) environment-level – agent-external in runtime; (iii) agent definition-level – agent-internal in design-time, and (iv) agent-level – agent-internal in runtime. These two metamodels already include the basic concepts that we needed, namely basic agent-based entities and relationships as well as the support for modeling at different levels (i.e., at design-time and runtime). The extension allows us to take advantage of existing modeling constructs and focus our modeling approach on the novel adaptation and normative-oriented constructs. The ANA metamodel introduces new abstractions that are not provided in neither FAML or TAO, such as normative attributes (e.g., state, condition, motivation) and adaptation attributes (e.g., *sensor*, *effector*, *strategy*, *policies*). This metamodel defines the static and dynamic aspects of agents, both internally and externally, in design and run times. The static aspect of ANA captures entities, their properties and relationships. The entities defined in ANA are agents, organizations, environments, roles, mental state and norms. The dynamic aspect of ANA is directly related to the relationships between the MAS entities and define the domain's independent behaviors associated with the interaction between MAS entities. In summary, the ANA metamodel captures new modeling language constructs to represent norm and adaptation abstractions as well as their interactions.

In Phase 2, based on the ANA metamodel, we develop an Adaptive Normative Agent Modeling Language (ANA-ML) [24] by extending the MAS-ML modeling language [23] because it already gives support to modeling: (i) the main MAS entities: agents, organization and environments; (ii) the static and dynamic properties of a MAS; (iii) agent roles, which are important while defining agent societies, and (iv) proactive agents. However, ANA-ML introduces new modeling language constructs to represent the ANA metamodel norm and adaptation abstractions.

In Phase 3, we characterize a fire scenario involving evacuation from hazardous areas [21], which is modeled using ANA-ML. Our goal is to describe the entities and relationships composing the evacuation plan, which take into consideration people's location given a specific disaster management situation.

In Phase 4, we conduct a quantitative experimental evaluation of ANA-ML by comparing it to another modeling language. This evaluation addresses three main aspects, namely we evaluate ANA-ML in terms of its ability to support: (i) the correct understanding of the creation and maintenance of the models; (ii) the time to correctly understand the creation and maintenance of the models, and (iii) the difficulty to correctly understand the creation and maintenance of the models. The experimental evaluation has shown that, in comparison with an existing modeling language, the use of

ANA-ML facilitates the correct understanding of the creation and maintenance of models that represent adaptation and norms, and requires less time from the developers for model creation and maintenance.

Furthermore, we note that many types of practical applications of multi-agent systems need to support adaptive and normative mechanisms in design and run times, as for example, in emergency services and natural disasters (i.e. floods, fires, landslides, earthquakes) [25], [26]; homes and smart [4], [7], cities [8]; the construction of virtual assets (i.e., Second Life [9]) and health (i.e. diseases or conditions involving the immune system [10], [27]).

C. CONTRIBUTIONS

This article proposes and evaluates the ANA-ML modeling language, and its underlying metamodel. Its main contributions can be summarized as:

- An extended metamodel, which allows representing the structural and adaptive behavior of agent-based software that can deal with norms. The proposed metamodel aims to express entities and relationships, both internal and external to the software agent;
- A modeling language, which addresses the particular features of MASs as means to cope with the limitations of existing approaches documented in the literature;
- A qualitative analysis of the proposed modeling language through a real case scenario related to a fire scenario in which people need to be evacuated.
- An initial evaluation of the proposed modeling language, in which we compare the creation and maintenance of models based on ANA-ML in contrast to MAS-ML. As criteria for this evaluation, we have considered: the time spent; the number of errors made by the participants when performing the required activities (the quality of the models), and the difficulties encountered.

D. PAPER STRUCTURE

The remainder of this paper is organized as follows. Section II presents research background and related work. Section III discusses the proposed ANA metamodel and Section IV presents the ANA modeling language (ANA-ML). Sections V and VI present a real case scenario and a quantitative experimental evaluation of ANA-ML, respectively. Finally, Section VII presents our conclusions and future work.

II. BACKGROUND AND RELATED WORK

This section introduces the main concepts needed to understand this work, including the concepts related to software agents, adaptive behaviour, normative systems, and MAS metamodels. The section also describes related work. This section is based on a survey we have developed to assess existing approaches and identify the gaps we are addressing in this paper [28].

A. SOFTWARE AGENTS AND THE BDI ARCHITECTURE

Software agents are autonomous entities able to perform tasks without human intervention [2]. The behavior of a software agent depends on both the state of the environment and its mental state. The mental state of an agent is composed of its state and behavior at a given time, i.e., goals, beliefs, decisions and intentions linked to its plans and actions. When it executes actions, the agent can change its mental state, introducing new perceptions about the environment, and by sending and receiving messages from other agents. The agent behavior is characterized based on its plans, actions and environment norms, which influences the agent autonomy and interactions [29]. Nevertheless, software agents can send and receive messages in the environment of other agents. The agent autonomy is given through its capacity of acting proactively. Finally, agents are adaptive entities, that may adapt its state and behavior regarding changes and restrictions in the environment.

One of the widely known architectures for designing and implementing cognitive agents is the belief-desire-intention (BDI) architecture [29], [30], following a model initially proposed by Bratman [31], which consists of beliefs, desires and intentions as mental attitudes that influence human action. Beliefs represent the characteristics of the environment, desires captures information about objectives and their costs and priorities, and intentions represent the current action plan [28].

B. ADAPTIVE AGENTS

The term adaptation means ''any change in the structure or functioning of an organism that makes it better suited to its environment'' [32]. Software adaptation refers either to the software being adapted or the evolutionary process leading to the new adapted software. More specifically, in the context of MASs, adaptations can be observed at two levels, which involve structural and behavioral changes. These changes aim at making applications more adaptable to their evolutionary context [33]. A promising solution for software adaptation is introducing self-adaptive software systems that can manage change dynamically while running efficiently and reliably [34]. According to [35], self-adaptation systems are able to modify their behavior or structure in response to their perception of the environment, the system itself, and their goals. One of the main advantages of self-adaptive software is its ability to manage the complexity arising from highly dynamic and non-deterministic operating environments.

For a software system to be considered self-adaptive [10], it should support a specific set of features, which include having: (i) the ability to observe changes in its operating environment; (ii) the ability to detect and diagnose changes in the operating environment and assess its own behavior; (iii) the ability to change its own behavior to adapt to new changes; and (iv) support for dynamic behavior (i.e. its internal and external behavior should be able to change intentionally and automatically). Even though many approaches [10],

TABLE 1. Properties of norms.

Property	Description
Addressee	It is used to specify the agents or roles responsible for fulfilling a norm.
Activation	It defines the activation condition for a norm to become active.
Expiration	It defines the expiration condition for a norm to become inactive.
Rewards	It is used to represent the set of rewards given to an agent for fulfilling a specific norm.
Punishments	It is the set of punishments given to an agent for violating a norm.
Deontic Concept	It is used to indicate if a norm states an obligation, a permission or a prohibition.
State	It is is used to describe the set of states being regulated.

[34], [35] describe systems' self-adaptation, they do not focus on modeling and implementing software agents. Therefore, although properties related to self-adaptive systems are considered important (i.e. autonomy, reasoning, proactivity), they are not explicitly addressed by existing approaches [17], especially when both adaptation and norms have to be considered.

C. NORMATIVE SYSTEMS

Norms are mechanisms that enable an agent to require other agents behave in a certain way [36]. In other words, norms can regulate the agents' behavior by representing the way in which an agent understands the responsibilities of others [3]. In a normative system, the software agents work under the belief that other agents will behave according to prescribed norms. The use of norms is a precondition in MASs in which the members are autonomous, but not self-sufficient and, therefore, cooperation is required and needs to be assured by specific mechanisms to support it.

Moreover, norms can be used to achieve different purposes, ranging from the construction of a simple agreement between software agents to more complex cases involving legacy systems [37]. Therefore, different aspects can be used to characterize norms. Table 1 presents the properties of norms. First, norms are always created to be obeyed by a set of agents to achieve specific social goals. Second, norms are not always applied, and their activation conditions depend on the context of the software agents involved in a specific interaction. The type of entity a norm regulates must be known, and it can be an agent's action or the state of the environment. Finally, in some cases, norms may provide a set of sanctions to be imposed when agents fulfill or violate them [3], [36], [37]. Nevertheless, norms and adaptation are indispensable to MAS models in domains such as the Internet of Things (IoT) [7] and in both internal and external and external design and run times [12], [13].

D. RELATED WORK

Having described relevant background, in this section we discuss works related to (i) metamodels and (ii) modeling languages, both in the context of MAS systems. The literature reports very distinct and varying sets of abstractions suitable for different domains [12], [13], [21], [25], [38], [39], [40]. Each methodology has incorporated its own abstractions for modeling the different multi-agent systems concepts, and there is no agreement about a common group of abstractions that can be used across different methodologies. Several authors recognized the importance of modeling agents in their environment both in design time and in runtime [8], [12], [41]. In general, these authors also observed that most modeling languages do not represent some important concepts that are present in MASs, such as adaptive behavior and norms [7], [12]. Nevertheless, norms and adaptation are relevant and indispensable to the software agent's internal and external design and run times.

1) MAS METAMODELS AND FAML

Several MAS metamodeling approaches have been introduced in the literature [8], [12], [23]. A metamodel involves a collection of concepts such as things and terms associated with a domain, which basically include relevant entities and relationships. In the case of MAS metamodels entities are, for example, agent, message, role, action, and plan. Instead of describing specific metamodels individually, we describe in this section a generic metamodel called FAML.

FAML [12] is a process-independent metamodel for an agent-oriented modeling language. It allows the description of software components of any multiagent system, as it captures problem-independent concepts (e.g., agents, resources) involved in multiagent system requirement description and system design with different abstractions. FAML's suitability for supporting modeling language development is demonstrated through a comparison with existing methodology-specific metamodels.

The FAML metamodel is constructed by a combination of bottom-up and top-down analysis and best practices. The expressiveness of the FAML metamodel concepts and their relationships were evaluated in [12] in comparison with two agent-oriented metamodels, namely TAO [42] and Islander [43]. The FAML metamodel has two layers: design-time and runtime layers. Each layer may have two scopes: system-related or agent-related. The design-time layer captures entities and relationships related to the ''system as developed'' and the runtime layer captures those related to the ''system as being executed''.

We briefly describe in this section a generic metamodel called FAML. A more detailed description of FAML as well as other multiagent models and languages is provided in [28]. However, concepts such as adaptation and norms are not included in the metamodel structure, neither at design nor execution times. In doing so, the agents cannot reason about the norms nor adapt to changes in the environment.

Cernuzzi [20] provided a comprehensive description of the GAIA, a methodology initially developed to model small-scale multiagent systems [2]. After undergoing several modifications [44], the authors seek to show the advantages of using GAIA to create models in the FAML social and

agent level [12]. In addition, various efforts have been made to model organizational structures and the norms governing the agents' overall behavior in the organization. Even after the modifications to the GAIA methodology, it still lacks support to model how the agent understands those norms and how it deals with them. The authors mainly addressed the relation between the norms and the roles of the agents in the organization. Further, this methodology does not consider possible adaptations of the agent to deal with norms in the environment; we introduce these adaptations in our modeling language.

In contrast, the metamodels presented in this section do not include a description of the internal architecture of normative agents and how they adapt their behaviors to deal with norms.

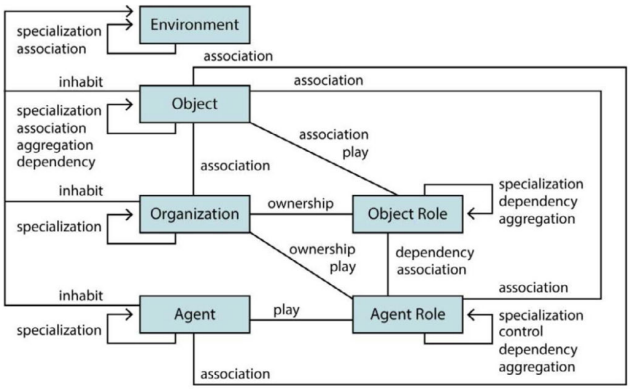

FIGURE 2. Abstractions and relationships of TAO [14].

2) MAS MODELING LANGUAGES

Several languages have been proposed for the modeling of MASs ([2], [11], [12], [15], [17], [21], [23], [44], [45], [46]–[49], [50]). Each of these approaches will be described in the following paragraphs. Additional information on MAS metamodels and languages can be found in [28].

Bernon [46] proposed the ADELFE methodology, which is capable of guiding and helping designers to build adaptive MASs. ADELFE 3.0 [19] can be used to design non-collaborative situations an agent can find or create. For each of these situations, an agent must leave the actions to be performed and ensure that the agent resumes these actions while staying in a state of cooperation with other agents and with itself.

Silva *et al.* [23] presented the modeling language MAS-ML, which extends UML and was based on the conceptual framework (metamodel) called TAO [51]. The metamodel Taming Agents and Objects (TAO) provides an ontology that covers the fundamentals of Software Engineering based on agents and objects and makes possible the development of large-scale MASs [14]. This metamodel connects consolidated abstractions, such as objects and classes, and new abstractions, such as agents, roles and organizations, which are the foundations for agent-oriented software engineering. TAO presents the definition of each abstraction as a concept of its ontology and establishes the relationships between them. Fig. 2 depicts the abstractions and relationships proposed in TAO. The TAO abstractions are defined as follows:

- Object: It is a element that can be passive or reactive and has state and behavior, and can be related to other elements;
- Agent: It is an autonomous and interactive element that has a mental state. Its mental state has the following components: (i) beliefs — everything the agent knows; (ii) goals — the future states that the agent wants to achieve; (iii) plans — the sequences of actions that achieve a goal, and (iv) the actions themselves;
- Organization: It is an element that groups agents and sub-organizations, which play roles and have common goals. An organization has intra characteristics, properties and behaviours represented by the agents inside it. It may restrict the behaviour of their agents and their sub-organizations through the concept of axiom, which characterizes the global organizational constraints that agents and sub-organizations must obey;
- Object role: It is an element that guides and restricts the behaviour of an object in the organization. An object role can add information, behaviour and relationships that an object instance executes;
- Agent role: It is an element that guides and restricts the behaviour of the agent playing the role in the organization. An agent role defines (i) duties as actions that must be performed by the agent playing the role, (ii) rights as actions that can be performed by the agent playing the role, and (iii) a protocol that defines the interaction between agent roles;
- Environment: It is an element to which agents, objects and organizations belong. Environments have state and behaviour.

In [14], the authors define the following relationships in the TAO metamodel: Inhabit, Ownership, Play, Specialization/Inheritance, Control, Dependency, Association and Aggregation/Composition. The concepts are presented through a semi-formal approach that uses templates to formalize Objects, Agents, Organization, Object Role, Agent Role, Environment and their relationships.

The great difference between this approach and others is the clear definition and representation of abstractions and behaviors that make up a MAS. However, the approach in [23] is incomplete because important abstractions such as adaptation and norms are not described neither in the TAO nor in the MAS-ML modeling language.

Gonçalves and colleagues [11] proposed an extension of MAS-ML [23] called MAS-ML 2.0, which bring, as a main novelty, mechanisms that support modeling proactive agents, and also the ability to model different internal architectures of agents, such as those presented in [52]. These different architectures were created to deal only with agent behavior based on reactive and proactive fundamentals.

The modeling language NormML [21] based on UML was developed to specify norms that restrict the behavior of entities in MASs. The NormML metamodel provides a language to model roles, permissions, actions, resources, and authorization constraints along with the relationships between permissions and roles, actions and permissions, resources and actions, and constraints and permissions. This approach allows the modeling of the static aspects of norms, but it is not possible to define dynamic aspects of a norm or to define norms in an interactive context. In addition, it is not possible to identify the rules that are active or those that have been violated; the norms described in NormML are not able to undergo changes throughout their activation time in the environment. Other approaches for modeling normative agents based on the BDI model have been proposed in the literature [53], [54].

The work presented in [55] describes the AUML language. This modelling language aims to provide semi formal and intuitive semantics through a friendly graphical notation. AUML does not provide elements to represent the next-function, planning, formulate problem function, formulate goal function and utility function. For instance, this language does not define the environment as an abstraction, so it is not possible to model agents that are able to move from one environment to another. Further, AML [13] is a modelling language based on a metamodel that enables the modelling of organizational units, social relations, roles and role properties. AML agents are composed of attribute list, operation list, parts and behaviors, and sensors and actuators.

In contrast, the approaches previously described do not support modeling concepts related to both adaptation and norms and their respective interactions [18], [19]. Indeed, norms and adaptation are relevant and indispensable to the internal and external design and run times of MAS modeling. Even though the literature reports several languages [4], [7], [8], [9], [10], [12], with capabilities for modeling MASs, there is still a need for a modeling language to describe concepts related to adaptation and norms as first-class abstractions through an explicit MAS metamodel that can be used to model the structural and dynamic aspects often described in MASs for these concepts, and promote the refinement of these models from design into code.

Current organizational norms are mostly used to restrict the behavior of agents and are implemented in the following ways: (i) addressing a given role; (ii) as a set of states to be regulated by the norm which also affects the agent's role; (iii) as valid restrictions when the norm is active in the environment; (iv) in many modeling languages and methodologies using the deontic concept of obligation [8], [25], [38], and (v) through the rewards and punishments associated with a particular norm [36].

Regarding the adaptation concept, we make explicit in the metamodel atomic concepts such as: (i) monitoring sensors; (ii) effectors, which are mechanisms needed to run the agent's actions and plans; (iii) events of the specific system — for example, to show when the system objectives are violated; (iv) strategies to adapt to environmental conditions; (v) different tactics performing plans in each strategy to carry out the adaptation, and (vi) adaptation policies that should always be checked during the environmental monitoring [10], [34].

Table 2 presents a comparison among the expressiveness of the main multi-agent architectures and models found in the literature ([12], [21], [25], [38], [39], [44]) based on the atomic abstractions related to the concepts of software agents, norms and adaptation. Some modeling languages and methodologies, such as AUML [55] and MAS-ML [23], Adelfe [46], FAML [12] and GAIA [20], do not support the modeling of norms. Approaches such as NormML [21] also make the modeling of several elements of a norm possible. From the set of modeling languages [10]–[13], [21], [22], and methodologies [15], [17], [20], [46], we have reviewed, just one of them (NormML) is able to model all the properties of the elements described in Section II.B However, NormML does not support the representation of adaptation-related abstractions (Section II.A) and, as a consequence, does not support modeling the interactions between norms and adaptation. In terms of abstractions for adaptation, although only Adelfe [46] is able to represent adaptive agents using adaptive workflows, it does not introduce the atomic elements an agent needs to adapt its behavior.

III. ANA METAMODEL

The ANA metamodel (Adaptive Normative Agent) enables the structural and adaptive behavior of agent-based software to support norms. Based on the distinction proposed by Beydoun [12] about design and run times, this metamodel aims at representing the internal and external entities and relationships of a software agent.

The metamodel presented in this section has four parts: (i) the external agent design-time representation, which describes the relationships among agents and other entities in the environment; (ii) the internal agent design-time representation, which describes the agent internal properties and their relationships; (iii) the external agent runtime representation, which describes the classes related to the Environment at runtime and includes events and messages; and (iv) the internal agent runtime representation, which describes the internal entities and relationships at runtime and includes plans, actions and messages.

A. EXTERNAL AGENT DESIGN-TIME REPRESENTATION

First, to create a multi-agent system, agents, organizations and their norms must be in an environment, but these agents cannot belong to more than one environment at the same time [12], [14]. Fig. 3 shows an external agent at design-time, which aims at representing the relation among the agents and other entities of the environment. This meta-level representation is adapted from TAO [14] and FAML [12]. The entities presented in Fig. 5 are represented by blue and green rectangles. The new classes in our model (i.e., the ones that were not defined in existing models) are shown as blue rectangles and the modified classes (e.g., the ones that result

TABLE 2. Comparing the expressiveness of multi-agent modeling approaches using abstract atomic elements related to agents, norms and adaptation.

		AUML	AML	GAIA	ADELFE	FAML	MAS-ML	MAS-ML 2.0	NormML	ANA-ML
Agent	Belief		X		X	X	X	X		X
	Desire					X				X
	Intention					X	X	X		X
	Action		X	X	X	X	X	X		X
	Plan		X		X	X	X	X		X
	Goal		X	X	X	X	X	X		X
	Role		X	X	X	X	X	X		X
Adaptive	Sensor				X					X
	Effector				X					X
	Event				X					X
	Strategy				X					X
Norm	Policies				X					X
	Addressee								X	X
	State								X	X
	Condition								X	X
	Deontic Concept		X	X					X	X
	Motivation								X	X

from modifications of classes that were defined in existing models) are shown as green rectangles.

A software **Agent** is an autonomous entity able to perform tasks without human interference [56]. Agents transform their own **Environment**; therefore, it is necessary to consider the environment state and the ''mental states'' of each agent to understand how they work. In addition, a software agent may communicate with other agents to achieve the common goals they would otherwise not achieve by themselves.

Agent behavior is characterized by its plans, actions, reasoning type and active norms in the environment. These features are related to the agent's general characteristics such as interaction, autonomy, and adaptation [36], [39], [56]. In addition, agents can interact by sending or receiving messages from other agents within the environment. The agent's autonomy is provided through its ability to act proactively, without any need for external input while the agents are executing their goals. Finally, agents are adaptive entities, since they can adapt their status and behavior according to changes and constraints in the environment.

The **MentalState** of a software agent is composed of its state and behavior at a given time, i.e., the *goals*, *beliefs*, *decisions*, and *intentions* linked to its *plans* and *actions*. When agents execute actions, they can change the mental state either by introducing new perceptions about the environment or sending and receiving messages from other agents. These concepts were included in the ANA metamodel to make it possible for the agent to use the BDI architecture.

An agent's **Role** guides and constrains the agent behavior by describing the goals it needs to achieve when playing a certain role [37], [56]. If an agent agrees to play a role, it will add the goals and beliefs of that role to its knowledge base. These new goals and beliefs can describe constraints in its behavior. Moreover, we cannot guarantee that an agent, which is an autonomous entity, will always execute all actions associated with a specific role, since these actions may be contrary to the agent's individual goals.

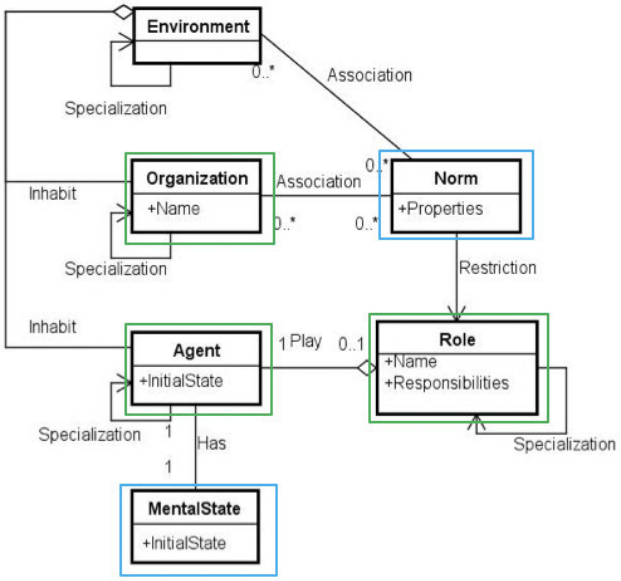

FIGURE 3. External agent design-time representation.

TAO [14] has introduced the concepts of duty and rights that are related to the role an agent is playing. However, these concepts are not enough to show that an agent needs to comply with the goals of the role because the software agent has no advantage if it decides to fulfill, or violate, the duties of that role. If an agent does not find any advantage in achieving the goals of the role, it will only fulfill its individual goals. For this reason, the concept of norms has been introduced (see Section II.B) in the ANA metamodel, in order to restrict the behavior of agent roles. **Norm** properties include rewards and punishments, leaving to the agent the task of deciding whether or not it should comply with the goals of that role.

An **Organization**, in turn, arranges the agents of a MAS into groups and roles, both defining the structure of software

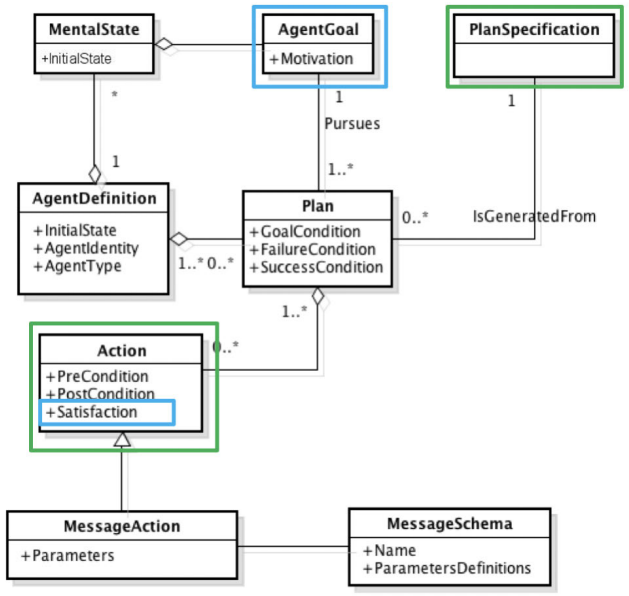

FIGURE 4. Internal agent design-time representation.

agents within an organization [13], [25]. Establishing an organization involves the specification of social plans, objectives and norms. Normative specifications are defined by using norms (see Section II.B) that connect agent roles to the organization's goals, and restrict an agent behavior while performing a certain role. TAO defines the axiom concept, which agents and sub-organizations must satisfy. In FAML, the convention concept has been established as an agreement between the organization and the agent. However, the concepts used by both approaches are not provided to regulate the behavior of agents while keeping their autonomy.

Regarding the external agent relationships at design-time, there are two kinds: specialization and association. The specialization relationship defines sub-entities by specializing super-entities; the sub-entities inherit the properties and relationships defined in the super-entity. Inherited properties can also be reset by a sub-entity [42]. The association relationship, on the other hand, specifies a semantic relationship between typified instances. Thus, if an entity is associated with another entity, it will become aware of the other entity with which it can interact [42].

B. INTERNAL AGENT DESIGN-TIME REPRESENTATION

In Fig. 4 the internal agent design-time representation explicitly describes the agent internal properties and their relationship. To express the mental state of an agent, the model needs to capture its mental components, such as beliefs, goals, intentions, plans and actions [26].

The **AgentDefinition** entity has generic functions in the system, which are used, for example, to initialize all agents and specify a function of the role an agent intends to play. The **MentalState** entity consists of components such as beliefs, goals, intentions, desires, plans and actions. This is characterized by the belief an agent has about itself and the environment, its intentions and individual goals. Further, the MentalState is responsible for checking the effects of a norm over the beliefs and desires of an agent. This entity is also used to capture the state and behavior of an agent at a given moment.

A **Plan** is composed of actions and it is related to the set of goals that an agent can access and run. When an action is taken, the agent can change its mental state. For example, it can send a message or perform specific adaptation steps. An agent may be able to choose a plan based on its goals and the norms responsible for restricting its behavior. The *GoalCondition* attribute characterizes the goals which the plans can be applied to. The *FailureCondiction* attribute characterizes a plan that cannot achieve a desired goal. The *SucessCondition* attribute describes a plan that can be deemed to have successfully achieved a goal.

An **Action** entity comprises the following attributes: (i) *PreCondition* — it refers to the states (system events) that must be met before an action is performed; (ii) *PosCondition* — it refers to the states (system events) that must be met after an action is performed, and (iii) *Satisfaction* — it describes the level of satisfaction an agent will have by performing a given action. Software agents should be able to support adaptive plans, and therefore, through the **PlanSpecification** entity we can set plans to monitor, analyze, decide and effect internal changes of the agent so that it can reason about norms.

In **AgentGoal**, the *Motivation* attribute describes the level of motivation that an agent has to accomplish a specific goal. An agent cannot call the enforcement actions of another agent that is sending a message to it. The goals of an agent are related to the future states or desires it intends to reach or accomplish. It is important to note that these goals are associated with at least one plan that the agent can perform to accomplish it.

The different relationships represented in Fig. 4 describe how the agent entities are connected. The connection of these entities defines an agent's mental state, with its beliefs, goals, intentions, actions, and plans. Each agent instance can change its initial mental state by adding, modifying or removing the beliefs, goals, actions, and initial plans.

C. EXTERNAL AGENT RUNTIME REPRESENTATION

Fig. 5 depicts the classes related to the **Environment** where agents "live" at runtime, that is, the external agent runtime representation. These classes coexist with instances of the agent design-time representation (see Figs. 3 and 4). The classes related to Environment focus on features that exist only in the runtime environment.

The abstractions supported by the metamodel at runtime enable different functions: (i) the historical data about the environment involves capturing of event and message registrations, in chronological order by using **MessageSchema** and **MessageEvent**; (ii) the **Event** element supports different types of events, such as send or receive a message

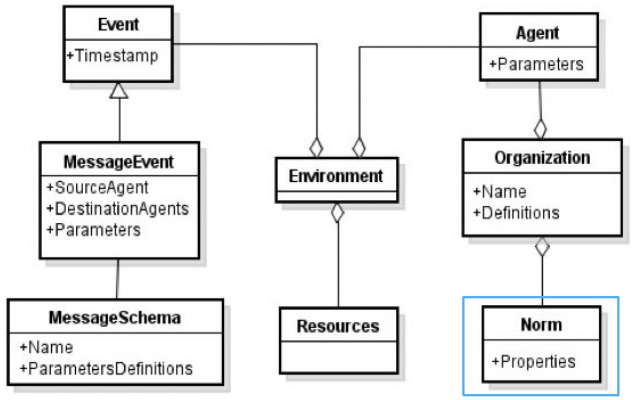

FIGURE 5. External agent runtime representation.

(**MessageEvent**); (iii) access points connect the system and relationships with events, resources, organizations and its norms, and (iv) agents and norms have an ''inhabit'' relationship with organizations.

D. INTERNAL AGENT RUNTIME REPRESENTATION

In turn, Fig. 6 depicts classes related to the internal agent runtime representation, which provide support for representing: (i) plans and actions; (ii) relationships between actions and messages; (iii) communication and the relationship between messages and protocols; (iv) mental states and the relationship with the BDI architecture; (v) the relationship between the previous abstractions and the states of the environment, and (vi) the adaptive reasoning performed by the agent to adapt the norm-related activities in the system.

In contrast with other modeling approaches, the proposed internal agent runtime representation provides several new features. In fact, the proposed metamodel extends TAO relationships [14] in different ways. These extensions include, for example, the possibility of using agents that support cognitive reasoning based on BDI architecture, represented by the classes **MentalState** and *AgentGoal*, and the introduction of a more generic class for communication. Further, in contrast with the FAML metamodel [12], we have introduced an adaptive form of reasoning that an agent can perform to adapt to changes and constraints (e.g., norms) of the environment.

The internal agent adaptation abstractions are represented by the following classes: (i) **Collect**, which has the Sensors attribute used to monitor the environment; (ii) **Analyze**, which represents the different tactics to deal with changes in the environment, given the Strategy type was created; (iii) **Decision**, which is responsible for making adaptation decisions by choosing the best set of plans given a specific situation; (iv) **Effector**, which has the *Actuator* attribute responsible for applying the actions in the environment.

The metamodel allows agents to perform cognitive reasoning, as well as recognize norms active in the environment. This form of reasoning is possible because the relationships between the beliefs, desires and intentions, are made explicit. Agent desires can be determined independently of its beliefs, but they can also be updated as its beliefs change. The intentions are derived from a set of desires based on the agent knowledge of the environment. Thus, agents can learn from their experience as they deal with norms, being able to keep and update their knowledge. These features not only allow agents to learn but also makes it possible for them to create and adopt adaptive plans as means to interact with the norms imposed by the environment.

IV. ANA-ML

In this section, we introduce the proposed modeling language, herein called ANA-ML (Adaptive Normative Agent Modeling Language) as means to provide support for modeling adaptive normative agents. ANA-ML introduces adaptation and norm-related abstractions in the modeling language MAS-ML [42] through a UML extension [39]. The language complies to a metamodel [38], which supports both structural and behavioral software agent adaptations. One of the mechanisms adopted by ANA-ML was the addition of a profile focusing on the definition of new classes, restrictions and stereotypes based on MAS-ML.

Modeling languages are created to describe the graphical representation of new abstractions, their semantics and relationships [14]. In addition, a modeling language for MASs should include diagrams to model the structural aspects of a system. Structural diagrams should be able to model: (i) the entities usually defined in a MAS; (ii) the properties of these entities and their association, and (iii) the relationships between the entities. In contrast to ANA-ML, the modeling languages proposed in the literature do not model adaptations and norms explicitly, and therefore, relationships between agents and entities and relationships that model these concepts are not supported.

Moreover, the development of appropriate approaches to implement agent-based systems is a key issue that needs to be addressed when agent technology is used in the development of a software system. To develop multi-agent systems using a specific modeling language, it is necessary to transform the MAS design models into code. These models are composed of high-level agent-related abstractions. In turn, as a means to derive source code out of those models, the agent-related abstractions must be mapped into abstractions defined in a certain programming language.

There are many advantages of using ANA-ML, which include the ability to: (i) represent all abstractions associated with a MAS application both at design and runtime; (ii) specify adaptation and norm-related static relationships; (iii) represent adaptation and norm-related dynamic interactions; and (iv) represent the dynamic interactions involving abstractions related to adaptation, norms, agents, organizations and environments. Indeed, to the best of our knowledge, it is not possible to model explicitly abstractions such as agent adaptation and norms, and their relationships and interactions using any existing modeling approach.

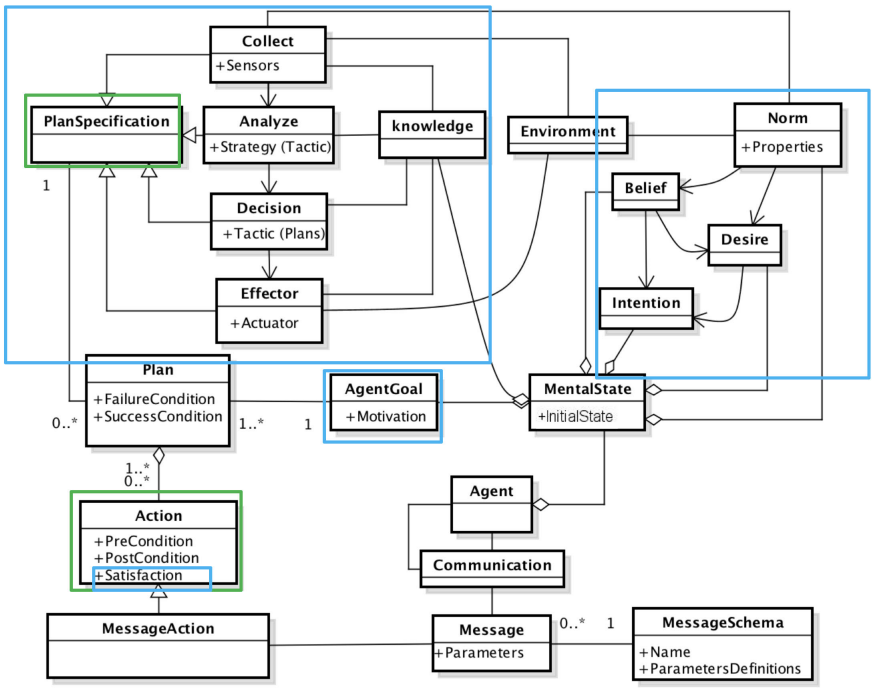

FIGURE 6. Internal agent runtime representation.

Fig. 7 presents the main classes, restrictions and stereotypes of ANA-ML. New classes include Agent Role, Organization, Plan, Goal, and Action. These classes are described in the following paragraphs.

A. AGENT ROLE

An agent role guides the agent behavior by describing the goals it needs to reach within an organization. On the other hand, a role restricts the agents' behavior so its tasks can be performed more efficiently. If an agent agrees to play some role, it will add the corresponding goals and beliefs to its knowledge base. These new goals and beliefs can describe constraints in its behavior. Moreover, we cannot guarantee an agent, which is an autonomous entity, will always execute all the actions associated with that role, because some actions may be contrary to its individual goals. TAO concepts are not enough to present an agent the need to comply with the goals of the norm. This is because the agent has no advantage if it decides to fulfill or violate the duties of that role. If the agent does not see any advantage in achieving the goals of the role, it will only fulfill its individual goals. For this reason, the concept of norms has been introduced in the ANA metamodel [57], as means of restricting the behavior of agent roles. Norm properties include rewards and punishments, leaving to the agent the task of deciding how much it is interested in complying with the goals of the role.

B. ORGANIZATION

An organization classifies the agents of a MAS into groups and roles, both defining the structure of different groups of agents and sub-groups within an organization [23]. Establishing an organization involves the specifications of social plans, objectives and norms. Normative specifications are defined using norms that connect agent roles to the organization's goals and restrict an agent behavior when it is performing a certain role.

C. PLAN

A plan is composed of actions and it is related to a set of goals the agent can access and run. When an action is taken, the agent can change its mental state. For example, it can send a message or perform specific adaptation steps. An agent may be able to choose a plan based on its goals and the norms that restrict its behavior.

D. GOAL

The goals of an agent are the future states or desires it intends to reach. It is important to note these goals are associated with, at least one plan the agent can perform to accomplish it. The *Motivation* attribute describes the level of motivation the agent has to accomplish a specific goal. Each agent instance can change its initial mental state by adding, modifying or removing the beliefs, goals, actions, and initial plans. In order to represent goals in the MAS-ML language, Silva [23] created the metaclass Goal, which describes a tag linked to the plans and restrictions. However, we observed the need to include the concept of motivation, which is required so the agent can reach a goal. The motivation for reaching a goal is then defined by the stereotype *motivation*, which maps the goal to an integer for representing the desire's intensity for

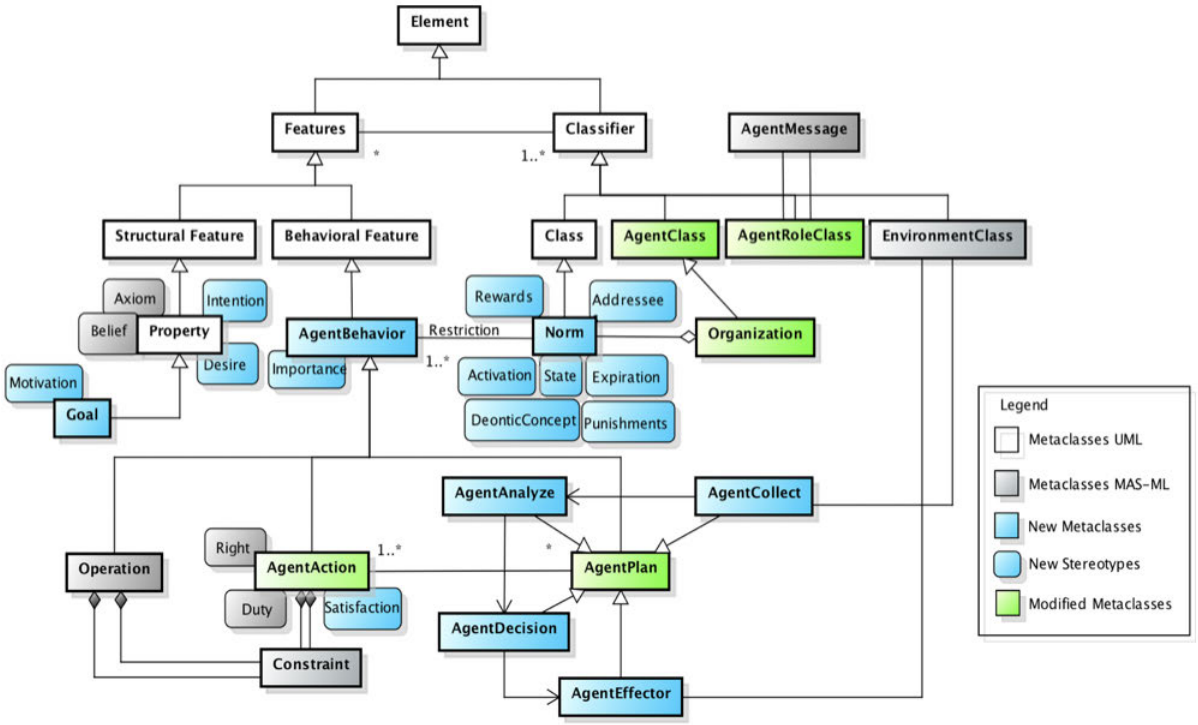

FIGURE 7. ANA-ML diagram.

reaching a given goal. The stereotype is based on the concept of motivation described in [29].

We kept the *belief* stereotype not specifying any type of restriction, but only identifying attributes that represent beliefs. In turn, for ANA-ML we defined new concepts observed in the entity **MentalState** of the conceptual model [57] to deal with agents Belief-Desire-Intention. Therefore, we created new stereotypes for **desire** and **intention**, where the first describes the desires an agent expect to reach, while the second describes states an agent prioritized when reaching its goals.

E. ACTION

It represents behavioral characteristics of a software agent. Actions are associated with agents and never called by other agents, but only executed within the agent's own domain. Thus, a metaclass called **AgentBehavior** has been created and it extends **Behavioral Feature** aiming to represent the agent's behavior. The metaclass **AgentAction**, from MAS-ML, is now an extension of the **AgentBehavior**, and it keeps the same purpose of representing actions performed by a software agent. In addition, we associated the metaclass **Constraint** to **AgentAction** aiming to define pre-and post-conditions that must be true to execute an action.

Besides the aforementioned characteristics, an action defined within ANA-ML has a new stereotype *satisfaction* associated with the metaclass **AgentAction**. The satisfaction of a software agent is defined based on its rewards and punishments when executing an action. Therefore, Satisfaction is defined by the function satisfaction that maps an action of the type Action to an integer, which represents the intensity of the agent's satisfaction when executing the given action. The satisfaction function is defined at design time and it is based on the work developed in [33]. In addition, the behavioral characteristics of an agent is also composed by its plans. The metaclass **AgentPlan** of the MAS-ML has been extended so that the creation of adaptive plans is supported. A plan is associated with a goal and it can be represented as a sequence of actions executed by a software agent to reach a goal. Other metaclasses have been created to specify the adaptive behavior of an agent, and include: (i) **AgentCollect** to monitor the norms in the environment; (ii) **AgentAnalyze** to analyze what possible plans can deal with the norms addressed to the agent; (iii) **AgentDecision** to make decisions based on different kinds of strategies and; (iv) **AgentEffector** to execute the actions and plans in the setting of a specific agent behavior considering the way an agent adapts to the environment based on the active norms.

V. REAL CASE SCENARIO

In this section, we present a real case scenario that illustrates how ANA-ML can be used to model a fire scenario involving evacuation from hazardous areas [58]. This case scenario emerged from the need to model situations in which people need to be rescued because of floods, landslides or other natural phenomena caused by climate change. The case is a result

of research based on our earlier work on using developing agent-based models and tools for simulating planned evacuations [58], [59], which led to agent-oriented models involving normative agents [41]. This kind of simulation was relevant especially for cities with a large concentration of population. Landslides, for example, are difficult to predict since they depend on many factors such as climate, soil properties and humidity and their specific relationship [60]. The annual number of landslides globally is estimated to be in the thousands, and the associated infrastructure damage is billions of dollars [58]. Planning evacuations from these areas of risk can be assisted by simulations using the JSAN framework [41]. For example, these simulations can be used to implement different scenarios in which firefighter agents regulated by norms rescue civilians at risk. Our goal is to describe the entities and relationships that comprise an evacuation plan, which take into consideration people's location given a disaster management situation (e.g. fire, floods, landslides, earthquakes). To represent the MAS entities, we have used the elements defined in section 3. Moreover, we have provided an activity diagram to describe many different dynamic aspects involved in this scenario.

When executing the evacuation plan, considering the best scenario, firefighters should ideally rescue and give first aid to the largest number of injured people. To do so, firefighters have a set of limited resources regulated by the firefighters chief. The regulation is performed through the definition of norms that restrict the firefighter's behavior so that the rescue happens in a coordinated way, considering the best usage of the available resources. These resources are, for example, aerial vehicles such as helicopters, land vehicles, troops, as well as excavation equipment. If the environment conditions change during the rescue of civilians due to weather instability - landslide, among other factors - there is a need for the firefighters to adapt.

As such, the goal for each group would be to get more resources to save people in areas of risk. Finally, we assume each group has its own reputation, and each group can keep or increase its reputation based on whether the goals are performed successfully. This scenario takes into consideration the firefighters' behavior is defined by a set of norms, as well as their ability to adapt under certain circumstances. The importance of performing a behavior is evaluated based either on what motivated the agent to reach the goal, or on the feedback (satisfaction) it must provide regarding the execution of an action. Particularly, specific norms are considered according to tables 3, 4 and 5.

The main goal of the firefighter chief is to provide assistance to the group of firefighters that will rescue people, and his/her motivation is scored as 10. Although the resources have high costs associated to their usage, they are essential for the firefighters to perform the rescue. Therefore, the firefighter chief is scored according to the satisfaction of the provided service. For example, this satisfaction is scored 3, 5 and 7 when firefighter chief requests the use of aerial vehicles, land vehicles, and excavation equipment, respectively.

TABLE 3. Scenario description for norm 1.

Norm #1	If firefighters are going to rescue civilians from hazardous areas, the firefighters chief is obligated to provide resources.
Reward	#1. The chief's reputation is increased. #2. More firefighters are made available to help people in a given hazardous area.
Punishment	#1 If resources are not provided, the chief's reputation is decreased.

TABLE 4. Scenario description for norm 2.

Norm #2	If the Fire Department is not able to provide the resources required to guarantee the minimum safe conditions, the firefighters are not eligible to rescue civilians.
Punishment	#1 In case the rescue is performed without the safe conditions, the reputation of the Fire Department is decreased.

TABLE 5. Scenario description for norm 3.

Norm #3	If during a rescue operation the number of people in the hazardous area is higher than the number of firefighters, the Fire Department is obligated to assist them with aerial vehicles.
Reward	#1. If assistance is provided, the reputation of the Fire Department is increased.
Punishment	#1. If assistance is not provided, the reputation of the Fire Department is decreased.

Next, we introduce the user scenario where firefighters rescue people from hazardous areas [3]. The entities in this scenario will be identified in order to create both structural and dynamic diagrams.

A. STRUCTURE DIAGRAMS

This section introduces the structural diagrams, which in this case, illustrate the environment and **Firefighter Organization** involved in the usage case (see Fig. 8). The main purpose of this view is to represent organizational and environmental classes, along with the elements and roles. The diagram represents the organization class **Hazardous Area**. There are two different kinds of agents in the fire scenario: **Civilian Agent** and **Firefighter Agent**. The diagram also represents the role of the **Person at Risk**, **Rescuer** and **Firefighter Chief**. The role of the person at risk is played by the civilian agent, given the roles of the rescuer and the Firefighter chief are played by the firefighter agent. In addition, we defined **norms** for the main organization aiming to restrict the firefighters' behavior in the rescue mission.

1) PARTIAL DESCRIPTION OF ORGANIZATION AND ENVIRONMENT

When analyzing the described scenario, we have noticed that the organization Firefighter Organization is included in the environment **Hazardous Areas**. Fig. 9 presents the environment modeled as an active environment where information is added over time so attributes, such as weather forecast and landslide, can be captured by the system. This class implements the following methods: (i) *addPerception* for

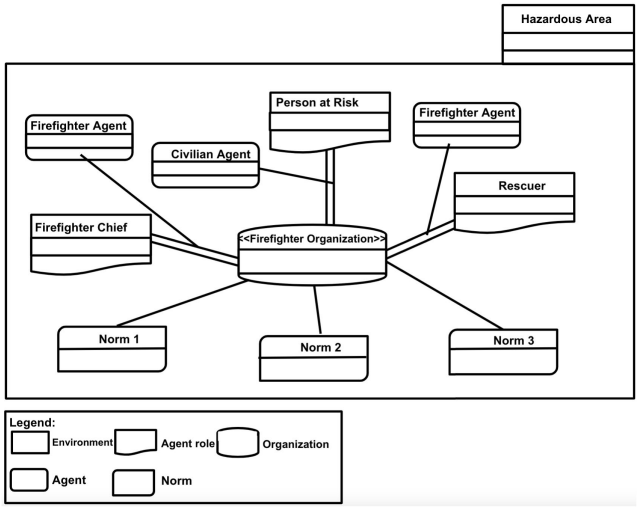

FIGURE 8. Diagram of the fire department.

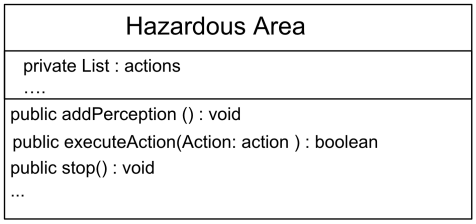

FIGURE 9. Partial description of the class hazardous areas.

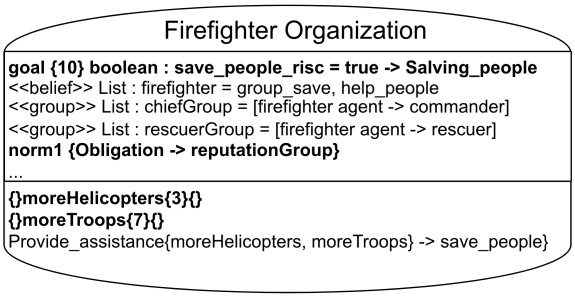

FIGURE 10. Partial description of the class firefighter organization.

new settings in the environment; (ii) *executeAction* to define what actions the environment will execute, and (iii) *stop* to end the execution of the MAS.

Fig. 10 shows the **Firefighter Organization** as the main organization of the system. Only one instance of this class can be created for each environment. The organization is mainly concerned with rescuing people from hazardous areas, and therefore, its motivation is a score of 10. To reach this goal, the main organization defines a rescue plan to managing the equipment, updating the environment to inform a civilian has been saved, and evaluating the mission.

Moreover, the firefighter agents are organized into different groups, depending on their role. That is, groups named *rescuerGroup* and *chiefGroup*, have been created to represent agents playing a role of a rescuer or firefighter chief, respectively. Remember that each group has an associated reputation score, and that the goal of a group is to keep or increase this score.

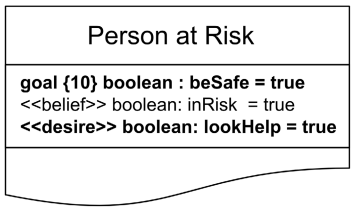

FIGURE 11. Partial view—the role of a person at risk.

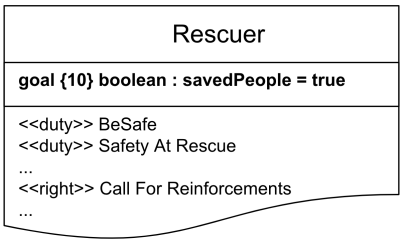

FIGURE 12. Partial view—the role of the rescuer.

2) DESCRIPTION OF THE ROLES PLAYED BY THE AGENTS IN THE MAIN ORGANIZATION

As previously mentioned, the organization defines the roles of: (i) the person at hazardous areas; (ii) the firefighters, and (iii) the firefighter chief. It also defines which goals are related to saving people and managing limited resources based on the reputation of each role. To reach these goals, the firefighters interact with the norms to restrict their behavior, i.e., the duty of a firefighter is to save peoples' lives. The norms managing the use of limited resources, as well as the agent's behavior must be extended to the firefighter agent.

Fig. 11 represents the role of a **Person at risk**, whose main objective is to be safe, and whose motivation is to receive the maximum score. In addition, a Person at risk has the right to ask the firefighters for help.

In turn, Fig. 12 illustrates the role **Rescuer** of a firefighter agent. Its motivation is to receive a score with the maximum value. The duties related to this role specifically are: (i) be safe, and (ii) keep people safe, i.e., not to put human life at risk. Agents playing this role can ask for resources during the execution of a rescue plan at hazardous areas.

Finally, Fig. 13 illustrates the role **Firefighter Chief**, whose main goal is to manage resources during the execution of a rescue plan. The motivation is to receive a score with the maximum value. That is, providing assistance to the firefighters is not the agent's main goal. The goals related to this role are to keep people alive and create norms. In this sense, the agent chief should not put human life at risk and should restrict the behavior of his/her subordinates.

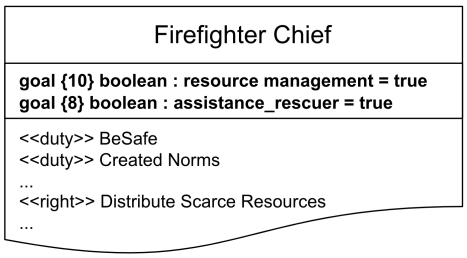

FIGURE 13. Partial view—firefighter chief class.

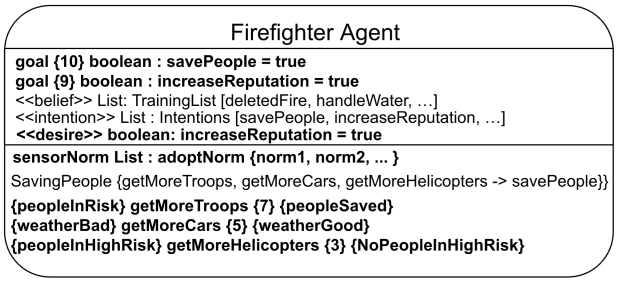

FIGURE 14. Partial view—the class firefighter agent.

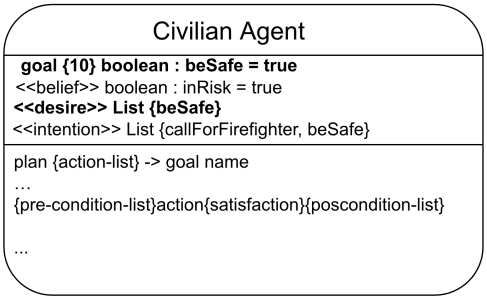

FIGURE 15. The civilian agent class.

3) AGENT DESCRIPTION

Some agents play the role of a firefighter. To represent a software agent playing this role, we use the class Firefighter Agent as illustrated in Fig. 14. The **Firefighter Agent** contains the following properties: (i) the *goals* that the agent has in this scenario; (ii) the *beliefs* of an agent on how to deal with problems the agent is trained to deal with on a daily basis, taking into consideration rescue operations; (iii) the *desire* to increase its reputation; and (iv) the list of *intentions* to be performed.

The last section in the Firefighter Agent class presents norms identified by the sensor agent and extended by the agent playing the role performed by the organization. The plans the agent needs to perform to save people and the level of satisfaction resulting from possible actions are associated to the tasks the Firefighter Agent needs to execute. For example, in case of a bad weather, more vehicles should be requested, and the satisfaction would have a score of 5, which is an intermediate medium value defined for the case scenario.

In turn, Fig. 15 shows **Civilian Agent** class representing people at risk according to our study scenario. This class describes only one goal and has the score 10 for motivation, which reflects a high priority due to the danger of situation. When the Civilian Agent is at risk, the belief is added to its base, and the agent, whose main goal is to be safe and stay alive, starts creating its list of intentions, such as asking for help.

4) INTERACTION BETWEEN NORMS AND ADAPTATIONS

In fact, adaptation and norms may be conceded simultaneously, and adaptation may impact norms and norms may impact adaptations. For example, consider **Norm #2** in an emergency rescue operation to be conducted by some agents represent by **Firefighter Agent**. This norm states that ''If the fire department is not able to provide the resources required to guarantee the minimum safe conditions, the firefighters are not eligible to rescue civilians.'' This norm may impact the behaviour of the firefighter agents. In case these agents are not provided the necessary resources, they are prevented from rescuing the civilians. Normally, the agent adaptive reasoning tends to decide to avoid the rescue operation if the conditions are not safe. In this case, the norm does not impact the firefighter agent behavior and, therefore, does not lead to an agent adaptation. However, the agent adaptive behaviour may decide based on its rewards that it should conduct the rescue operation even in the absence of appropriate safety conditions. In this case, the norm impacts adaptation in the sense that although the norm is triggered, the agent decides not to fulfill it and adapt.

B. DYNAMIC DIAGRAMS

A dynamic diagram is introduced to capture the activities of the normative adaptive agents and their sequences. The proposed activity diagram is a variation of the one proposed by [11]. This diagram, which represents the behavior of the agents, is shown in Fig. 16. Each activity is represented as a rounded-corner rectangle, whereas the characteristics are represented as squares, with the identification of what stereotype corresponds to a specific agent. The goal of the activity diagram is to represent the agent's behavior when it extends the norm addressed to it. The execution of the normative adaptive reasoning is possible through the modifications performed in the process of the normative application.

The behavior of an adaptive normative agent starts when it identifies the norms in the environment and performs activities of the normative process, i.e., agents are able to reason about norms [41]. The first activity, **NormAwareness**, the software agent identifies the existing active norms in the environment, and these norms are addressed to specific agents. The second activity, **NormDeliberation**, the agent recognizes the norms where its responsibility is specified. The third activity, **NormCompliance**, norms are realized through the agent's reasoning to define which plans could be performed when a given norm is accepted. In the last activity, **NormImpact**, the agent will execute the norm and the goals

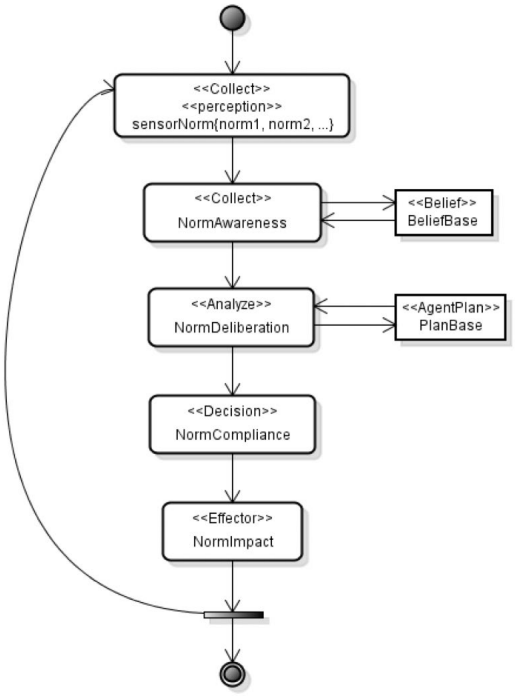

FIGURE 16. Activity diagram—normative adaptive agent.

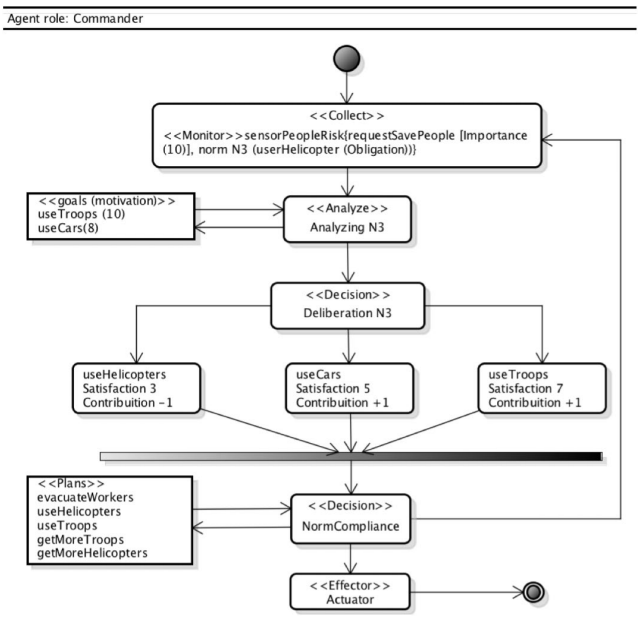

FIGURE 17. Activity diagram of ANA—detailed actions for norm deliberation.

of the agent will be updated. After that, the cycle of the agent's reasoning continues, and the agent starts identifying other norms to be adopted.

A diagram associated with the Firefighter Agent playing the role of a Firefighter Chief is illustrated in Fig. 17. For instance, the norm N3 regulates if the number of people to be saved is higher than the number of firefighters (Rescuer), so the Firefighter Chief has to continue the rescue operation by sending aerial vehicles. Given the fact resources are reduced and insufficient, the self-adaptation process regarding the norm N3 will be executed by the Firefighter Chief agent. This process searches other ways to rescue people from hazardous areas without the use of aerial vehicles, based on the data the agent has perceived by sensing the environment. Therefore, the agent will need to collect and analyze data for the decision-making process, before an action is taken to adapt the agent's behavior to deal with the norm N3.

VI. EMPIRICAL EVALUATION

The main goal of this section is to investigate whether the different modeling languages (ANA-ML and MAS-ML) pose some influence on the correct creation and maintenance of the agent models. Similar to related efforts [61], [62], three dimensions were evaluated in this empirical study: (i) correctness (errors); (ii) time, and (iii) difficulty. We evaluated whether the subjects could create and maintain the agent models correctly, how fast they got the information they needed, and the difficulties they faced. We have defined the following research questions:

- **#RQ1**: Does the availability of specific modeling language increase the correct understanding of the creation and maintenance of the agent models?
- **#RQ2**: Does the availability of specific modeling language reduce the time that is needed to correctly understand the creation and maintenance of the agent models?
- **#RQ3**: Does the availability of specific modeling language decrease the difficulty to understand the creation and maintenance of the agent models?

Associated with the first three research questions are three null hypotheses:

- $\mathbf{H1}_0$: The correct understanding of the creation and maintenance of the agent models **does not depend** on the different modeling languages.
- $\mathbf{H2}_0$: The time to correctly understand the creation and maintenance of the agent models **does not depend** on the different modeling languages.
- $\mathbf{H3}_0$: The difficult to correctly understand the creation and maintenance of the agent models **does not depend** on the different modeling languages.

For the sake of completion, the defined three alternative hypotheses **H1**, **H2** and **H3**. They are, respectively, defined by replacing "does not depend" to "depend" in $\mathbf{H1}_0$, $\mathbf{H2}_0$, and $\mathbf{H3}_0$.

Our initial results revealed that our approach is feasible and less difficult to be applied in a real-world application, and requires less time than a traditional approach. We discuss the results of our empirical evaluation at the end of this section.

In order to analyze the hypotheses previously described, we considered the following independent variables: (i) the participants; (ii) the training sessions using UML, MAS-ML, and ANA-ML; and (iii) the activities that the participants performed to create the models. Moreover, we considered as dependent variables (i) the time (effort) spent to perform the activities; (ii) the number of errors made by the participants

when performing each activity – the quality of the models (correctness), and (iii) the difficulties encountered when performing the activities. This section describes how we conducted the empirical evaluation concerning the creation and maintenance of the agent models, presents and discusses the evaluation results, and describes threats to validity.

A. APPLICATION DOMAIN

The chosen application domain was crime prevention [63]. This domain is an alternative to traditional approaches as criminologists have started to collaborate with Computer Science and Artificial Intelligence researchers to explore the benefits of social simulations that use agents to investigate the spatial distribution of crimes. This new approach aims at using simulation environments to predict the dynamics of crime distribution in the future and to analyze past distributions.

The simulation of crime prevention was modeled as an organization that dwells in the environment of a city and introduces the following roles as people agents: citizen, criminal and police. Agents playing the role of citizens move around different locations in the city to go to shopping malls, museums, big events, that is, places with a high concentration of people. In contrast, agents playing the role of criminals such as thieves usually try to find these places in order to commit the highest number of robberies as possible. The police agents move around different areas of the city aiming to prevent the highest number of crimes as possible and imprison the criminals.

B. EMPIRICAL EVALUATION PROCESS AND DESIGN

The empirical evaluation involved 14 participants who had different knowledge and experience levels with respect to MASs. These participants were graduate students from different post-secondary academic institutions. Further, all of them had taken a graduate course on MASs. Therefore, they were previously exposed to methods for modeling and developing multiagent-based applications. The participants were divided into two groups: Group A (G-A); and Group B (G-B). While one group underwent a phase and had to perform the activities using the MAS-ML to answer the questionnaire, the other group underwent the same phase but had to answer the questionnaire performing the activities using the ANA-ML. The designed questionnaires included questions about the creation and maintenance of models based on MAS-ML and ANA-ML. The questionnaires were conducted by all participants at different times on the same day without tool support. The participants agreed not to share any information about the questionnaire so that they would not influence one another, and the results would not be compromised. G-A and G-B answered the questionnaires in different rooms.

1) QUESTIONNAIRE PROCESS

The questionnaire application involved six phases. In the first phase, G-A was trained to create models using MAS-ML and UML elements (e.g., stereotypes and comments). In contrast, G-B was trained in ANA-ML. In the second phase a questionnaire was applied to both groups. The first part of the questionnaire contains questions about each participant and includes questions about their education, specific knowledge, and experience in modeling MASs. The second part of the questionnaire is divided into two sub-parts, each one involving two modeling activities that require the different groups to create diagrams based on MAS-ML and ANA-ML. In the third phase the focus is on the inverted training of the groups, that is, G-A was trained in ANA-ML and G-B was trained in MAS-ML. The training session took from 30 to 60 minutes.

In the fourth phase another questionnaire was applied to both groups and it required G-A to create models based on ANA-ML and G-B to create models based on MAS-ML. The structure and complexity of this questionnaire is similar to the questionnaire applied in the second phase. To avoid any adverse influence in the questions posed in the questionnaires three experts in modeling and experimental software engineering validated them. In the fifth phase, each of the participants was interviewed to identify the easy parts and difficulties faced when creating the requested diagrams. Finally, in the sixth phase, the participants provided the data that were gathered and analyzed.

TABLE 6. Latin square.

	G-A	G-B
First Step	MAS-ML	ANA-ML
Second Step	ANA-ML	MAS-ML

2) DESIGN

We designed our empirical evaluation with the Latin Square [64] to maximize statistical power with a relatively small number of subjects. Table 6 presents the configuration of the Latin Square design. The Latin square design also alleviated some effects of learning and heterogeneity among subjects [65]. The size of the Latin square is 2×2, in which the x-axis are the subjects and the y-axis are the modeling languages. Therefore, the experimental design gave us a random allocation of questionnaire in such a way that they questions were answered once by each subject and once with each modeling language. As the study involved 14 subjects, we were able to replicate the 2×2 Latin square 7 times, obtaining 28 independent observations for each comprehension task.

This design also allowed a comparison of the results considering the same phases (horizontal comparison). That is an analysis involving the activities performed by the G-A and G-B using the different modeling languages. A vertical comparison was also conducted (between phases). In this case, the analysis considered the results across phases.

C. RESULTS

The analysis of the data and the presentation of the results were divided into two subsections. The first subsection analyzes the profile of the participants. The second subsection analyzes the Latin Square in a vertical manner by phase

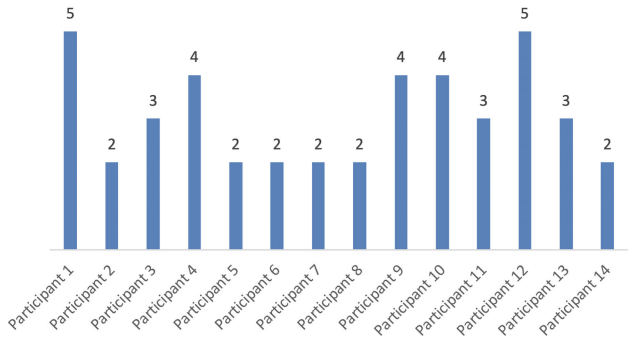

FIGURE 18. Modeling expertise of each participant in years.

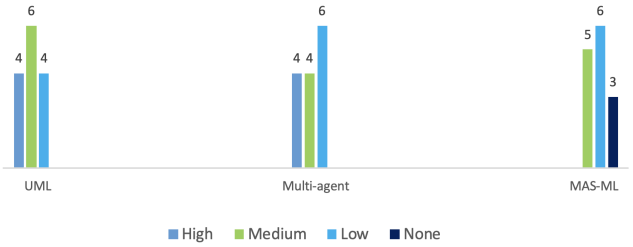

FIGURE 19. Participants profile.

and in a horizontal manner by group. Both quantitative and qualitative aspects are examined in the analysis.

1) VARIABLES AND ANALYSIS

The metrics averages were initially calculated by (Table 7): (i) the percentage of errors per activity (correctness); (ii) the time spent on the activities in minutes, and (iii) the difficulty level of each activity. To test our hypotheses, we first test whether the sample distribution is normal (Shapiro-Wilk) and has equal variance (Levene). If these tests pass, we use the ANOVA test to evaluate our hypotheses, that is, whether there is any evidence that the means of the populations differ; otherwise, we use the non-parametric Kruskal-Wallis test. If these tests lead to a conclusion that there is evidence that the group means differ, we then are interested in investigating which of the means are different. To this end, we use the Tukey multiple comparison test in the case of the sample distribution is normal and have equal variance. This test compares the difference between each pair of means with appropriate adjustment for the multiple testing. Otherwise, we use the Nemenyi-Damico-Wolfe-Dunn test. For the correctness, difficulty and time variables, we maintain a typical confidence level of 95 percent (alfa = 0.05).

a: BACKGROUND OF THE PARTICIPANTS

Fig. 18 presents the modeling experience of each participant in years and Fig. 19 presents the knowledge of the participants based on their knowledge of: (i) the UML modeling language; (ii) multi-agent systems, (iii) and the modeling language MAS-ML. We consider expertise as a value ranging from 1 to 4 in a Likert scale, with 1 being a beginner (no expertise) and 6 being an expert.

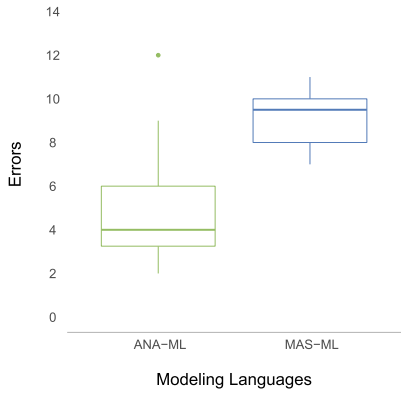

FIGURE 20. Boxplot errors.

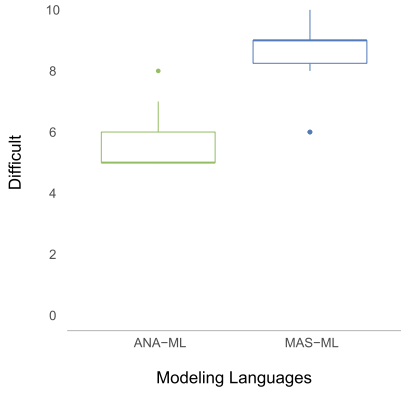

FIGURE 21. Boxplot difficult.

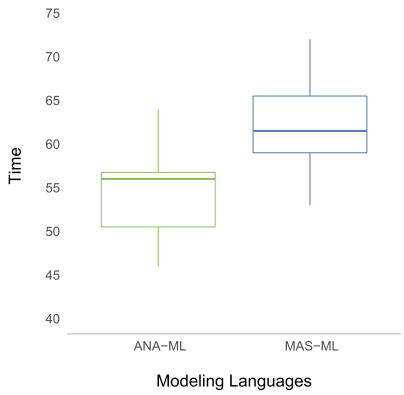

FIGURE 22. Boxplot time.

In general, the 14 selected subjects have at least the basic skills. They have the necessary expertise on software modeling with UML and MAS-ML, and the development of Multi-agent systems. The most of subjects claimed to have a medium degree of expertise in UML, and low to medium expertise in development of Multi-agent systems and MAS-ML.

b: ANALYSIS AND INTERPRETATIONS

In the empirical evaluation we answer the questions **#RQ1**, **#RQ2** and **#RQ3**. This section presents the results and discusses some general observations.

TABLE 7. Descriptive statistics of the experimental results.

	Error		Time		Difficult	
	ANA-ML	MAS-ML	ANA-ML	MAS-ML	ANA-ML	MAS-ML
Mean	4.4285	9.2142	54.2857	61.7857	5.7142	8.5714
Difference		-51.9%		-12.2%		-33.4%
Min	7	16	100	110	10	15
Max	12	21	120	131	14	19
Median	9	18	107	127	11	18
Stdev.	1.2089	1.3301	5.1065	5.7375	1.1435	1.2403

TABLE 8. ANOVA test for participants answers—error.

Source	SS	DF	F	Sig.
Replica	23.93	6	0.596	0.72852
Activity - groups	0.57	1	0.085	0.77509
Technique	120.14	1	17.957	0.00115
Replica: participants	22.50	7	0.480	0.83126
Residuals	80.29	12		

i) Descriptive Statistics and Hypotheses Testing

We start our discussion by presenting descriptive statistics and hypothesis testing. The Fig. 20, Fig. 22, and Fig. 21 shows box plot for the scores that were obtained by the subjects, considering their overall scores. They shows that there is a difference in terms of time and difficulty to model agents using ANA-NL or MAS-ML. The box plots are complemented by Table 7, which shows descriptive statistics of the measurements.

We start by testing null hypothesis $\mathbf{H1}_0$, which states the errors per activity to understand the creation and maintenance of the agent models do not depend on the different modeling languages. Fig. 20 depicts a box plot for the scores that were obtained by the subjects. The Shapiro-Wilk and Levene tests succeeded for the answers data, which means that ANOVA may be used to test $\mathbf{H1}_0$.

Table 8, therefore, shows the results for ANOVA. The average correctness by the ANA-ML was clearly lower and the p-value = 0.00115 is lower than 0.05, which means that $\mathbf{H1}_0$ can be rejected in favor of the alternative hypothesis **H1**, stating that the correctness to modeling normative MASs depends on the different modeling languages.

Next we test null hypothesis $\mathbf{H2}_0$, which states the time to correctly understand the creation and maintenance of the agent models does not depend on the different modeling languages. Fig. 22 shows box plot for the scores that were obtained by the subjects.

Because the Shapiro-Wilk and Levene tests succeeded for the answered data, ANOVA may be used to test $\mathbf{H2}_0$. Table 9 shows the ANOVA results. The average time spent by the ANA-ML was clearly lower and the p-value = 0.00315 is lower than 0.05, which means that $\mathbf{H2}_0$ can be rejected in favor of the alternative hypothesis **H2**, stating that the time to correctly understand the modeling of normative MASs depends on the different modeling languages.

We also test the null hypothesis $\mathbf{H3}_0$, which states that the difficulty to correctly understand the creation and maintenance of the agent models does not depend on the different modeling languages. Fig. 21 shows box plot for the scores that were obtained by the subjects. Note that we consider overall scores rather than scores per questions. The box plot shows that there is no explicit difference in terms of correctness. The Shapiro-Wilk test did not succeed for the answers data, which means that ANOVA cannot be used to test $\mathbf{H3}_0$.

TABLE 9. ANOVA test for participants answers—time.

Source	SS	DF	F	Sig.
Replica	145.2	6	0.833	0.56726
Activity - groups	78.9	1	2.714	0.12540
Technique	393.7	1	13.544	0.00315
Replica: participants	116.2	7	0.571	0.76638
Residuals	348.9	12		

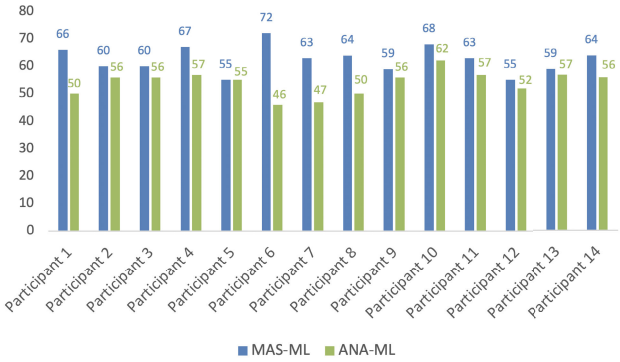

FIGURE 23. Measure of time answers.

Consequently, we used the nonparametric Kruskal-Wallis test. As a chi-squared = 17.266, degree-of-freedom = 1, and a p-value = 0.335e-05, the Kruskal-Wallis indicates that there is a statistically significant difference among the investigated modeling languages in terms of difficulty, meaning that $\mathbf{H3}_0$ can be rejected in favor of our alternative hypothesis **H3**, and that the difficulty to model the agent, indeed, depends on the different modeling languages.

ii) MODELING LANGUAGES VS. CORRECT ANSWERS

Based on these results, we also observe that hypothesis **H2** might hold, but only with respect to some of the modeling languages. By comparing the time spent to answer questions correctly with ANA-ML and MAS-ML (Fig. 23), we observed the use of the modeling agents that are capable of adapting their behavior to understand normative systems. However, it seems that there is no significant difference between ANA-ML and MAS-ML in terms of the number of errors for modeling adaptive behaviors and norms.

iii) EXPERTISE AND DIFFICULTY

Finally, we analyzed whether the modeling expertise of participants was essential to correctly answer the questionnaires. Fig. 24 depicts the chart that relates the degree of difficulty of each participant and his/her number of errors (wrong answers) per activity. Each bar in this chart indicates the difficulty (y-axis) of the participants (x-axis) with respect to the modeling languages (lines). The bullets exhibit the total number of incorrect answers of each participant. Note that there is a dependence between the difficulty and the number of incorrect answers.

FIGURE 24. Measure of difficulty and errors answers.

For MAS-ML, a high degree of difficulty and incorrect answers in modeling the questions was not fundamental to correctly answer the questionnaire. As a result, we could observe that there is indeed a correlation between the difficulty and the number of incorrect answers.

iv) REASONS FOR DIFFERENT DIFFICULTY

The lower difficulty achieved by ANA-ML subjects can be attributed to several factors. First, all information required for correctly understanding the modeling language was shown in the training of models, which eliminates the need to interpret the model examples.

The participants considered the use of ANA-ML easier than the original MAS-ML, both being based on UML. One of the main reasons for this conclusion was that MAS-ML does not provide support to model the adaptive and normative concepts that the participants were requested to create. In addition, some participants who had better knowledge about the concepts of norms and adaptation considered the ANA-ML approach more accessible and more intuitive to use.

Next, we provide some examples of sentences mentioned by the participants during the interview that happened after answering the questionnaires:

Participant #1: "Using ANA-ML it was possible to create norms and understand the adaptation process, although I had some doubts the characteristics of these abstractions."

Participant #2:"I noticed that some requests were difficult to be performed using only the available resources, that is, the resources shown in the presentation about MAS-ML."

Participant #3:"The problems I had in my first modeling phase with MAS-ML were solved with the use of the models available in ANA-ML to model norms and the adaptive process of the agent."

v) ERRORS FOUND IN THE ANSWERS

Table 10 presents the errors found in the answers reported by the participants in the study. The most common errors reported when modeling using MAS-ML were related to the lack of abstractions to support modeling norms and how norms were interpreted internally by the agents. In contrast, the most common errors related to the ANA-ML models were related to the lack of concepts to support the notions of norms and adaptation.

TABLE 10. Errors found.

Problem with MAS-ML and UML	Use of ANA-ML
Not model any requested point	Not model any requested point
Use of comments that are not clear to provide some kind of information in the diagrams	ANA-ML syntax
Not follow a pattern in the produced models, generating different results for similar situations	Modeling wrongly some information such as the norm or adaptation characteristics
MAS-ML syntax	
Modeling wrongly some information such as the norm or adaptation characteristics	

c: THREATS TO VALIDITY

This subsection discusses the study constraints. For each category we describe both the possible threats and the procedures we have used to alleviate their risks.

i) CONCLUSION VALIDITY

The major external risk was related to the engagement of the subjects to be part of the experiments, due to the length (time) of the questionnaires, which took almost two hours for each participant. However, there was a rotation in the order of the approaches given that we adopted the Latin square. Another threat is the heterogeneity of participants. We have not used any mechanism to select the participants and so they may represent random choices. Although the heterogeneity of the subjects can also be considered a threat to the conclusion validity, it helps to promote the external validity of the study. Finally, the quality of the investigated modeling languages is also a risk related to conclusion validity. However, we did not observe bugs that hampered the understanding of specifications or forced the participants to spend more time answering a question.

ii) CONSTRUCT VALIDITY

We identified the following threats to the construct validity: confounding questions, and insufficient training session. To minimize these problems, we answered the questions from participants as they were arising. To avoid biases in the experiment results, we limited the explanations of the representation models to what was demonstrated during the training session and limited the questions to include only the clarifications that were absolutely necessary.

iii) INTERNAL VALIDITY

Threats to internal validity involve the specification of adaptive agents and normative systems with different modeling languages. We have made sure that each adaptive agents and normative systems had specified following the same patterns in all models by triple-checking each specification and by using the model developers to model the system.

iv) EXTERNAL VALIDITY

The major external risk was related to the user scenario. The selected modeling language might not be representative of industrial practices. To reduce this risk, we selected a modeling language that was a UML extension, given that UML is heavily used on industry.

VII. CONCLUSION AND FUTURE WORK

Agent-based software engineering (AOSE) is a powerful paradigm for software design and implementation [11], [30]. Systems developed with AOSE technology require methodologies, modeling languages, development platforms and programming languages that explore their benefits and their own characteristics. After analyzing many of the MAS modeling languages published in the literature [11], [12], [15], [16], [17]–[20], [21] we have realized that there is a lack of modeling languages that support abstractions related to norms and adaptation, and promote the refinement of design models into code.

In addition, most modeling languages do not model the structural aspects (entities and relationships) neither the dynamic aspects (domain independent behavior) normally described in MASs and defined in the ANA conceptual model, such as those related to norms and adaptation. Some existing proposals only describe a subset of these abstractions, and others do not model the interaction between the defined entities. In order to define a modeling language for MASs that encompasses all the concepts described in ANA, the ANA-ML modeling language [66] was proposed. ANA-ML is a modeling language that extends MAS-ML based on the structural and dynamic aspects presented in the ANA conceptual model.

The presented approach exhibits a very interesting feature, that is, agents are able to adapt to deal with restrictive norms. This means that an agent can be modeled in a way that achieves a balance between the agent's desires and the organizational goals of the environment he resides by having its behavior and adaptation regulated by explicit norms.

Regarding limitations, the modeling language ANA-ML presented in this paper was based on the ANA metamodel and, as consequence, it is necessary to extend all concepts defined by ANA. There are a lot of new abstracts defined in the metamodel – to extend all definitions, relationships and different types of interactions is a complex task. Furthermore, ANA-ML diagrams can become very large when large systems are modeled and all elements of the diagram are expressed. For example, the activity diagram can become very complex given the number of actions and plans that are realized taking into account the conditions and constraints of the environment.

For future work, we plan to create an automated modeling tool to help the developers of multi-agent systems during the modeling and implementation of their applications. The goals of the modeling tool are to simplify and accelerate the designs of ANA-ML diagrams. In addition, the formalization of the ANA-ML language would bring forth several benefits of a precise semantics: clarity, equivalence, consistency, ability to extend, refinement and proof [67].

We also believe that including the concept of governance systems [51] could bring a greater transparency to the interaction between agents, and also help to regulate the system resources by means of the norms that can be created by entities of the system. Furthermore, it would be possible to ensure that agents are punished based on other agents, because the evaluation of an agent's behavior could be based on the testimonies it receives from other agents about violations of norms.

Last but not least, we could differentiate between organizational norms and individual norms. The organizational norms that are defined by the organization restrict the behavior of agents who play roles in the organization and are punished for their violations. The individual norms are related to the expectations agents have about the behavior of other agents. Therefore, it becomes important to understand how these norms are defined and how to identify the agents that best meet these expectations. In this case, norms can be defined by an agent and their violations would not receive the punishments coming from the organization.

In conclusion, as software applications become more dynamic and adaptive, we believe it is essential to support developers to model MASs with abstractions such as adaptive agents, norms and their relationships. Such information can be foundational to steer future research on modeling adaptive agents capable of understanding and dealing with norms and adaptation.

REFERENCES

[1] N. Jennings and M. Wooldridge, "Software agents," *IEE Rev.*, vol. 42, no. 1, pp. 17–20, Jan. 1996.
[2] M. Wooldridge, *An Introduction to Multiagent Systems*. Hoboken, NJ, USA: Wiley, 2009.
[3] F. L. Lopez, "Social power and norms," Ph.D. dissertation, Dept. Electron. Comput. Sci., Univ. Southampton, Southampton, U.K., 2003.
[4] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
[5] N. Moraes do Nascimento and C. J. P. de Lucena, "Engineering cooperative smart things based on embodied cognition," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst. (AHS)*, Jul. 2017, pp. 109–116.
[6] O. K. Tonguz, "Red light, green light—No light: Tomorrow's communicative cars could take turns at intersections," *IEEE Spectr.*, vol. 55, no. 10, pp. 24–29, Oct. 2018.
[7] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
[8] D. J. Cook, "How smart is your home?" *Science*, vol. 335, no. 6076, pp. 1579–1581, 2012.
[9] A. Bogdanovych, J. A. Rodríguez, S. Simoff, A. Cohen, and C. Sierra, "Developing virtual heritage applications as normative multiagent systems," in *Proc. Int. Workshop Agent-Oriented Softw. Eng.* Berlin, Germany: Springer, 2009, pp. 140–154.
[10] T. Laroum and B. Tighiouart, "A multi-agent system for the modelling of the HIV infection," in *Proc. KES Int. Symp. Agent Multi-Agent Syst., Technol. Appl.* Berlin, Germany: Springer, 2011, pp. 94–102.
[11] E. J. T. Gonçalves, M. I. Cortés, G. A. L. Campos, Y. S. Lopes, E. S. S. Freire, V. T. da Silva, K. S. F. de Oliveira, and M. A. de Oliveira, "MAS-ML 2.0: Supporting the modelling of multi-agent systems with different agent architectures," *J. Syst. Softw.*, vol. 108, pp. 77–109, Oct. 2015.

[12] G. Beydoun, G. Low, B. Henderson-Sellers, H. Mouratidis, J. J. Gomez-Sanz, C. Gonzalez-Perez, and J. Pavon, "FAML: A generic metamodel for MAS development," *IEEE Trans. Softw. Eng.*, vol. 35, no. 6, pp. 841–863, Nov. 2009.

[13] I. Trencansky and R. Cervenka, "Agent modeling language (AML): A comprehensive approach to modeling MAS," *Informatica*, vol. 29, no. 4, pp. 1–241, 2005.

[14] V. Silva, A. Garcia, A. Brandao, C. Chavez, C. Lucena, and P. Alencar, "Taming agents and objects in software engineering," in *Proc. Int. Workshop Softw. Eng. Large-Scale Multi-Agent Syst.* Berlin, Germany: Springer, 2002, pp. 1–26.

[15] R. Gowri, S. Kanmani, and D. Punitha, "Tropos based adaptation framework for self adaptive system," *J. Theor. Appl. Inf. Technol.*, vol. 63, no. 3, pp. 790–799, 2014.

[16] A. Gómez-Rodríguez, R. Fuentes-Fernández, J. C. González-Moreno, and F. J. Rodríguez-Martínez, "Ingenias with the unified development process," in *Handbook Agent-Oriented Design Processes* Berlin, Germany: Springer, 2014, pp. 371–405.

[17] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Auton. Agents Multi-Agent Syst.*, vol. 8, no. 3, pp. 203–236, May 2004.

[18] M. B. Van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks, "A semantic framework for socially adaptive agents: Towards strong norm compliance," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, Princeton, NJ, USA: Citeseer, 2015, pp. 423–432.

[19] W. Mefteh, F. Migeon, M.-P. Gleizes, and F. Gargouri, "Adelfe 3.0 design, building adaptive multi agent systems based on simulation a case study," in *Computational Collective Intelligence*. Cham, Switzerland: Springer, 2015, pp. 19–28.

[20] L. Cernuzzi, A. Molesini, and A. Omicini, "The Gaia methodology process," in *Handbook Agent-Oriented Design Processes*. Springer, 2014, pp. 141–172.

[21] K. da Silva Figueiredo, V. T. da Silva, and C. de Oliveira Braga, "Modeling norms in multi-agent systems with normml," in *Proc. Int. Workshop Coordination, Organizations, Institutions, Norms Agent Syst.* Berlin, Germany: Springer, 2010, pp. 39–57.

[22] F. R. Meneguzzi and M. Luck, "Norm-based behaviour modification in BDI agents," in *Proc. AAMAS*, 2009, pp. 177–184.

[23] V. T. Da Silva, R. Choren, and C. J. De Lucena, "MAS-ML: A multiagent system modelling language," *Int. J. Agent-Oriented Softw. Eng.*, vol. 2, no. 4, pp. 382–421, 2008.

[24] M. Viana, P. Alencar, D. Cowan, E. Guimarães, F. Cunha, and C. Lucena, "The development of normative autonomous agents: An approach," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, vol. 2, Dec. 2015, pp. 9–16.

[25] B. M. Beamon and S. A. Kotleba, "Inventory management support systems for emergency humanitarian relief operations in South Sudan," *Int. J. Logistics Manage.*, vol. 17, no. 2, pp. 187–212, May 2006.

[26] M. Janssen, J. Lee, N. Bharosa, and A. Cresswell, "Advances in multi-agency disaster management: Key elements in disaster research," *Inf. Syst. Frontiers*, vol. 12, no. 1, pp. 1–7, Mar. 2010.

[27] S. Zhang, J. Yang, Y. Wu, and J. Liu, "An enhanced massively multi-agent system for discovering HIV population dynamics," in *Proc. Int. Conf. Intell. Comput.* Berlin, Germany: Springer, 2005, pp. 988–997.

[28] M. Viana, P. Alencar, and C. Lucena, "Towards an adaptive and normative multi-agent system metamodel and language: Existing approaches and research opportunities," 2021, *arXiv:2111.13084*.

[29] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in *Proc. ICMAS*, vol. 95, 1995, pp. 312–319.

[30] M. Wooldridgey and P. Ciancarini, "Agent-oriented software engineering: The state of the art," in *Proc. Int. Workshop Agent-Oriented Softw. Eng.* Berlin, Germany: Springer, 2000, pp. 1–28.

[31] M. Bratman, *Intention, Plans, Practical Reason*, vol. 10. Cambridge, MA, USA: Harvard Univ. Press, 1987.

[32] *Towards a Formal Semantics of UML 2.0 Activities*, E. Dictionary Oxford, Simpson, JA & Weiner, ESC, Oxford, U.K., 1989.

[33] H. K. Dam and M. Winikoff, "Towards a next-generation AOSE methodology," *Sci. Comput. Program.*, vol. 78, no. 6, pp. 684–694, Jun. 2013.

[34] N. Huber, A. van Hoorn, A. Koziolek, F. Brosig, and S. Kounev, "Modeling run-time adaptation at the system architecture level in dynamic service-oriented environments," *Service Oriented Comput. Appl.*, vol. 8, no. 1, pp. 73–89, Mar. 2014.

[35] R. D. Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, and D. Weyns, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 1–32.

[36] B. F. dos Santos Neto, V. T. da Silva, and C. J. de Lucena, "An architectural model for autonomous normative agents," in *Proc. Brazilian Symp. Artif. Intell.*, Cham, Switzerland: Springer, 2012, pp. 152–161.

[37] F. L. y López, M. Luck, and M. d'Inverno, "Constraining autonomy through norms," in *Proc. 1st Int. Joint Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2002, pp. 674–681.

[38] G. Génova, "What is a metamodel: The OMG's metamodeling infrastructure," *Softw. Syst. Model.*, vol. 4, no. 2, pp. 171–188, 2005.

[39] V. C. Gu, Q. Cao, and W. Duan, "Unified modeling language (UML) IT adoption—A holistic model of organizational capabilities perspective," *Decis. Support Syst.*, vol. 54, no. 1, pp. 257–269, Dec. 2012.

[40] V. Nallur and R. Bahsoon, "A decentralized self-adaptation mechanism for service-based applications in the cloud," *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 591–612, May 2013.

[41] M. L. Viana, P. Alencar, E. T. Guimarães, F. J. P. Cunha, D. Cowan, and C. J. P. Lucena, "JSAN: A framework to implement normative agents," in *Proc. Int. Conf. Softw. Eng. Knowl. Eng.*, Jul. 2015, pp. 660–665.

[42] V. T. Da Silva and C. J. P. De Lucena, "From a conceptual framework for agents and objects to a multi-agent system modeling language," *Auto. Agents Multi-Agent Syst.*, vol. 9, no. 1/2, pp. 145–189, Jul. 2004.

[43] M. Dastani, J. Hulstijn, F. Dignum, and J.-J. Meyer, "Issues in multiagent system development," in *Proc. 3rd Int. Joint Conf. On Auto. Agents & Multi Agent Syst. (AAMAS)*, 2004, pp. 922–929.

[44] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Trans. Softw. Eng. Methodol.*, vol. 12, no. 3, pp. 317–370, 2003.

[45] J. Pavón and J. Gómez-Sanz, "Agent oriented software engineering with INGENIAS," in *Proc. Int. Central Eastern Eur. Conf. Multi-Agent Syst.* Berlin, Germany: Springer, 2003, pp. 394–403.

[46] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard, "Adelfe: A methodology for adaptive multi-agent systems engineering," in *Proc. Int. Workshop Eng. Societies Agents World*. Berlin, Germany: Springer, 2002, pp. 156–169.

[47] M. Challenger, B. T. Tezel, V. Amaral, M. Goulao, and G. Kardas, "Agent-based cyber-physical system development with SEA_ML++," in *Multi-Paradigm Modelling Approaches for Cyber-Physical Systems*. Amsterdam, The Netherlands: Elsevier, 2021, pp. 195–219.

[48] G. Kardas, B. T. Tezel, and M. Challenger, "Domain-specific modelling language for belief–desire–intention software agents," *IET Softw.*, vol. 12, no. 4, pp. 356–364, Aug. 2018.

[49] T. Z. Asici, B. T. Tezel, and G. Kardas, "On the use of the analytic hierarchy process in the evaluation of domain-specific modeling languages for multi-agent systems," *J. Comput. Lang.*, vol. 62, Feb. 2021, Art. no. 101020.

[50] O. F. Alaca, B. T. Tezel, M. Challenger, M. Goulão, V. Amaral, and G. Kardas, "AgentDSM-eval: A framework for the evaluation of domain-specific modeling languages for multi-agent systems," *Comput. Standards Interface*, vol. 76, Jun. 2021, Art. no. 103513.

[51] V. T. da Silva and C. J. de Lucena, "Modeling multi-agent systems," *Commun. ACM*, vol. 50, no. 5, pp. 103–108, 2007.

[52] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. London, U.K.: Pearson, 2002.

[53] N. Alechina, M. Dastani, and B. Logan, "Programming norm-aware agents," in *Proc. 11th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 2, 2012, pp. 1057–1064.

[54] D. Dell'Anna, M. Dastani, and F. Dalpiaz, "Runtime revision of sanctions in normative multi-agent systems," *Auto. Agents Multi-Agent Syst.*, vol. 34, no. 2, pp. 1–54, Oct. 2020.

[55] J. J. Odell, H. V. D. Parunak, and B. Bauer, "Representing agent interaction protocols in UML," in *Proc. Int. Workshop Agent-Oriented Softw. Eng.* Berlin, Germany: Springer, 2000, pp. 121–140.

[56] M. A. Kalareh, "Evolving software systems for self-adaptation," Ph.D. dissertation, Univ. Waterloo, Waterloo, ON, Canada, 2012.

[57] M. Viana, P. Alencar, D. Cowan, E. Guimarães, F. Cunha, and C. Lucena, "The development of normative autonomous agents: An approach," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, vol. 2, Dec. 2015, pp. 9–16.

[58] B. F. dos Santos Neto, V. T. da Silva, and C. J. de Lucena, "Developing goal-oriented normative agents: The NBDI architecture," in *Proc. Int. Conf. Agents Artif. Intell.* Berlin, Germany: Springer, 2011, pp. 176–191.

[59] S. E. A. Cerqueira, "The georisc platform: Computing engineering applied to the analysis of Geo-environmental risks," Dept. Inform., Pontifical Catholic Univ. Rio de Janeiro, Rio de Janeiro, Brazil, Tech. Rep. 34/09, 2009.
[60] G. Domènech, M. Alvioli, and J. Corominas, "Preparing first-time slope failures hazard maps: From pixel-based to slope unit-based," *Landslides*, vol. 17, no. 2, pp. 249–265, Feb. 2020.
[61] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 721–734, Aug. 2002.
[62] M. C. Ohlsson and C. Wohlin, "An empirical study of effort estimation during project execution," in *Proc. 6th Int. Softw. Metrics Symp.*, Nov. 1999, pp. 91–98.
[63] T. Bosse and C. Gerritsen, "An agent-based framework to support crime prevention," in *Proc. 9th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 1, no. 1, 2010, pp. 525–532.
[64] B. J. Winer, "Latin squares and related designs," in *Statistical Principles in Experimental Design*, B. J. Winer, Ed. 1962, pp. 514–577.
[65] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, and A. Pohthong, "Robust statistical methods for empirical software engineering," *Empirical Softw. Eng.*, vol. 22, pp. 579–630, Apr. 2017.
[66] M. Viana, P. Alencar, and C. Lucena, "A modeling language for adaptive normative agents," in *Multi-Agent Systems and Agreement Technologies*. Cham, Switzerland: Springer, 2016, pp. 40–48.
[67] H. Storrle and J. H. Haumann, "Towards a formal semantics of UML 2.0 activities," *Softw. Eng.*, vol. 1, pp. 1–12, 2005.

MARX VIANA received the B.S. degree in computer science from the Federal University of Viçosa (UFV), in 2010, and the master's degree in informatics and the Ph.D. degree from the Pontifical Catholic University of Rio de Janeiro (PUC-RJ), in 2012 and 2016, respectively.

He is a currently a Researcher at PUC-RJ. His research in computer science spans a wide range of topics, with an emphasis on software engineering, and addresses specifically software frameworks, object-oriented design and programming, software agents, software adaptation, normative systems, nature language processing, and machine learning development processes.

PAULO ALENCAR (Member, IEEE) is a Research Professor with the David R. Cheriton School of Computer Science, University of Waterloo, and the Associate Director of the Computer Systems Group (CSG). He has published over 200 refereed publications and has been a member of program committees of numerous highly-regarded conferences and workshops. His recent research in information technology and software engineering has focused on high-level software architectures, design, components, and their interfaces; software frameworks and application families; software processes, automated workflows and work graphs, and evolution; web-based approaches and applications; open and big data applications; context-aware and event-based systems; software agents; machine learning; cognitive chatbots; artificial intelligence; and formal methods. He has been the principal or co-principal investigator in projects supported by NSERC, ORF-RE, IBM, SAP, CITO, CSER, Bell, and funding agencies in Canada, USA, Brazil, Germany, and Argentina.

He is a member of the Association of Computing Machinery (ACM), Association for the Advancement of Artificial Intelligence (AAAI), Waterloo Water Institute (part of the Global Water Futures initiative), and Waterloo Artificial Intelligence Institute. He has received international research awards from organizations, such as Compaq and IBM.

EVERTON GUIMARÃES received the master's degree in computer science from the Federal University of Rio Grande do Norte, in 2010, and the Ph.D. degree in informatics from the Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, in 2014.

He has been an Assistant Professor at Penn State Great Valley University, since 2018. He has collaborated with many partners in industry and academia, both in Brazil and abroad, over the past ten years. His most recent research investigates problems related to software architecture and source code and their impact on the overall software quality attributes (i.e., maintenance and evolution). He is also interested in research topics related to technical debt, architecture recovery techniques, pattern detection, and mobile computing.

ELDER CIRILO received the B.S. degree in computer science from the Federal University of Juiz de Fora (UFJF), in 2006, and the master's and Ph.D. degrees in informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-RJ), in 2008 and 2012, respectively.

He has been an Assistant Professor with the Department of Computer Science, Federal University of São João del-Rei, since 2013. He has experience in the area of software engineering, with an emphasis on the following themes: configurable software systems, domain specific languages, feature-oriented software development, end-user programming, and experimental software engineering.

CARLOS LUCENA received the B.Sc. degree from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 1965, the Math degree in computer science from the University of Waterloo, Canada, in 1969, and the Ph.D. degree in computer science from the University of California at Los Angeles, in 1974.

He worked at the PUC-Rio as the Vice-Rector of the University, the Dean for the Center for Science and Technology, and several times as the Director of the Computer Science Department. He has been a Full Professor with the Department of Informatics, PUC-Rio, since 1982. He has made extensive and internationally recognized research contributions, being the author or coauthor of over 600 refereed papers and 19 books in the area of software design and formal methods in software engineering, his primary areas of research. Until 2015, he has supervised 45 Ph.D. theses and 114 M.Sc. dissertations. His former Ph.D. students are faculty members at universities in Brazil and abroad (e.g., USA and Canada). He has been a member of the program committees of more than 62 national and international conferences as well as member of the editorial board of some important journals in his area of research. He is also a member of the editorial board of the international journal on *Formal Aspects of Computing* (New York: Springer-Verlag). He is a fellow of the Guggenheim Foundation, an ACM Fellow, and a member of the Brazilian Academy of Sciences. He was awarded the insignia for the Grand Cross of the Order of Scientific Merit of the Presidency of the Republic of Brazil, the Medal of Scientific Merit Carlos Chagas Filho from the Council of Directors of FAPERJ (State Research Council), the Alvaro Alberto Award for Science and Technology (Ministry of Science and Technology), and several times the IBM Innovation Award, among many others.