Designing MAS Organisation through an integrated MDA/Ontology Approach*

Daniel Okouya[1] and Loris Penserini[1] and Sébastien Saudrais[2] and Athanasios Staikopoulos[2] and Virginia Dignum[1] and Siobhán Clarke[2]

[1] Universiteit Utrecht, The Netherlands,{maatari,loris,virginia}@cs.uu.nl
[2] Trinity College Dublin, Computer Science, Ireland
{Athanasios.Staikopoulos, Sebastien.Saudrais, Siobhan.Clarke}@cs.tcd.ie

Abstract. The increasing complexity of distributed applications, software services that can be dynamically deployed, adjusted and composed, paves the way for new challenges in software and service engineering. This paper describes a novel approach that combines the flexibility of MDE techniques to deal with the conceptual modelling of MAS and the expressive power of OWL based ontologies to deal with semantics constraints verification as well as domain knowledge provision of MAS models. We will illustrate these ideas through the modeling of a crisis management scenario, using a first prototype of our future Design tool: OperettA.

1 Introduction

Nowadays' distributed applications based on the notion of service-oriented systems –which can dynamically adapt, organise, and compose to satisfy with their networked stakeholders' needs– are fostering the software engineering research area with new challenges. As these distributed systems have to be deployed within real organisational contexts, adhering with organisational rules, and meeting stakeholders' expectations, it is crucial to characterise the software architectural and functional requirements in terms of their correlations with the actual environment. To deliver on this aim, a promising approach in software engineering has been to build methodologies along with conceptual modelling languages that better reflect and describe the complex social and human organisational context where the system-to-be has to be deployed [1][2][3]. For example, in [3] and [4], an ontology based on the Telos language has been presented to describe the conceptual model of the Tropos methodology.

In this paper, we describe some achievements and future improvements of a MAS development framework, OperettA [5], which is based on the OperA methodology [1]. Using a simplified crisis management scenario, we will illustrate how the Operetta's conceptual modelling language –which adheres to a

* This work has been performed in the framework of the FP7 project ALIVE IST-215890, which is funded by the European Community. The authors would like to acknowledge the contributions of their colleagues from ALIVE Consortium (http://www.ist-alive.eu)

Model Driven Engineering approach– can be integrated with ontology representation languages as OWL. Moreover, this approach allows designers for both the verification of the semantics and the provision of domain specific knowledge of the MAS design models.

Model Driven Engineering (MDE) refers to the systematic use of models as primary artefacts throughout the Software Engineering lifecycle. The defining characteristic of MDE [6] is the use of models to represent the important artefacts in a system, be they requirements, high-level designs, user data structures, views, interoperability interfaces, test cases, or implementation-level artefacts such as pieces of source code. The Model Driven Development promotes the automatic transformation of abstracted models into specific implementation technologies, by a series of pre-defined model transformations.

The paper is organised as follows: Section 2 presents the methodological context of our approach with a motivation example. Section 3 provides an overview of our approach. Section 4 summarises the main characteristics and benefits of the approach adopted, which is partially implemented within the OperettA tool.

2 A Methodological Context

2.1 OperA overview

OperA [1] is an engineering methodology based on organisational abstractions, suitable both to model and study existing societies, as well as to develop new systems that participate in an organisational context. The main focus of OperA enable a suitable balance between global aims and requirements agent autonomy, their coordination needs, and environmental stakeholders' needs.

The development framework for agent societies, proposed in OperA, is composed of three conceptual design models: *Organisational Model*, *Social Model*, and *Interaction Model*, as detailed in [1]. In this paper, we illustrate our ideas by only using some concepts and diagrams belonging to the *Organisational Model*. It contains the description of the roles, relations and interactions in the organisation. It is constructed based on the functional requirements of the organisation. The social model and the interaction model are the link between the organisational description and the executing agents.

2.2 Running scenario

Using an example taken from the Dutch procedures for crisis management, we provide a conceptual model based on organisational and social concepts. The modelling phase is conducted according to the OperA methodology [1] using the OperettA tool [5] for graphical representation and verification of the model. This scenario will be used along the paper to explain the development framework properties.

The structure diagram depicted in Figure 1 represents the crisis situation. It specifies the responsibilities and goals of each role, e.g., each time an emergency call occurs to the Emergency_Call_Center, this role alerts the Fire_Station entity, informing about the location in which the (possible) disaster is taking place. The

Fire_Station is responsible to build the appropriate fire brigade and depends on the established Firefighter_Team in order to achieve the objective extinguish the fire. When the team arrives at the accident location, it has to decide (based on its personal experience) the severity of the disaster. Only after this evaluation is reported, an intervention decision is taken. For example, according to local rules, the evaluation should comply with some standardised emergency levels, as established by the Dutch Ministry of Internal Affairs. For the sake of simplicity, we consider that Firefighter_Team sets up a strategic intervention based on the results of two evaluation criteria: *damage evaluation* and *fire evaluation*. Based on the number of wounded, Firefighter_Team decides on the necessity or not to ask for ambulance_service. Moreover, the Firefighter_Team checks if the damage involves building structures in which case police intervention is necessary to deviate traffic. From the fire evaluation criterion, Firefighter_Team can decide whether it is the case or not to ask Fire_Station for a Firefighting_Truck intervention. As described in [1], the Social Structure is further detailed by interaction structure diagrams to model activities among and within roles in order to achieve their objectives. Such activities are called scenes.

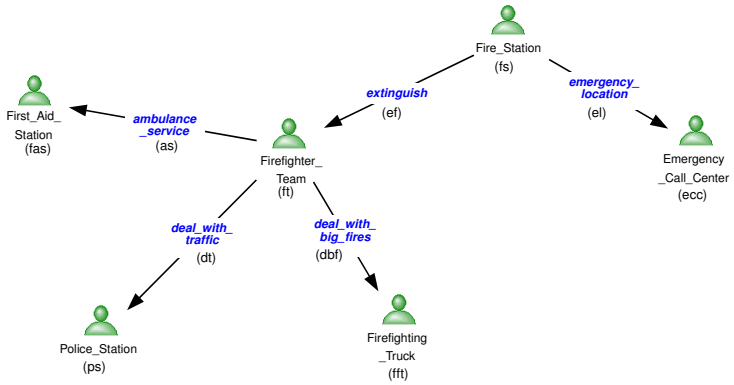

Fig. 1. Social Structure diagram: Fire Station organisation (O) example.

Notice that, the specification provided at this time is not sufficient to give a complete picture about the *know how* required to software systems to achieve the modelled organisational objectives. Nevertheless, the level of abstraction achieved provides enough anchor points for agents to coordinate their activity without fully pre-specifying the capabilities of the agents and therefore limiting flexibility.

3 Approach overview

3.1 A meta-level view of conceptual models

In order to effectively deal with the MAS development, the proposed approach takes into account both the syntax and the semantics of a MAS, through an

integration of MDA with Reasoning and Domain Knowledge specification Based on Ontology. Fig. 2 illustrates the architecture of our approach. The central part corresponds to the OperA metamodel, which provides the syntax of MAS. The right part provides the actual semantics of MAS, which is described by the OperA ontology. The MAS ontology instantiation will be automatically produced from the MAS model, which is created with the OperettA tool. Next, the MAS ontology instantiation will be semantically checked against the OperA ontology, see section 3.2. The left part provides an interaction between existing domain ontologies and MAS models. The interaction is maintained by defining transformations relation between the OperA metamodel and EODM[3]. EODM is an implementation of the ODM standard from OMG, defining metamodels for RDF(S) and OWL, see section 3.3.

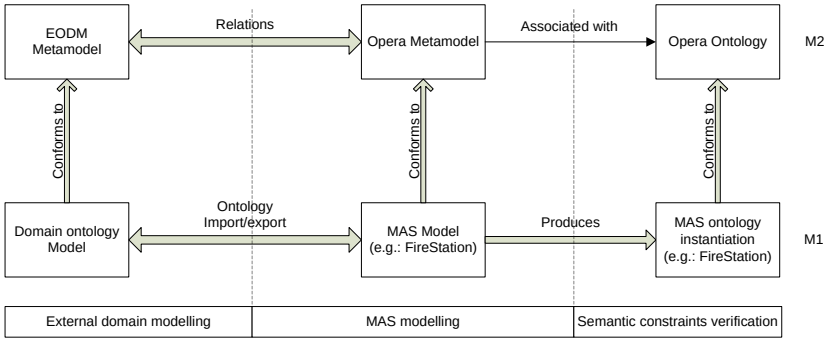

Fig. 2. Overview of our approach.

3.2 Reasoning with models compliant with OperA ontology

The first aspect of our integrated approach is directed toward reasoning on our models using logics, with description logic as our language mainly dedicated at reasoning on the structural aspect of our models. This will provide us with the ability to use the power of descriptive logic along with associated reasoners, combined with techniques to formally analyse our models and enhance their quality.

At the meta-level, the abstract syntax is defined through metamodelling and the semantics is based on description logic. This results in the OperA ontology, which formalises OperA conceptual framework concepts and their relationships, as well as domain independent OperA semantics constraints. Meanwhile, the integration of the metamodelling and ontologies supports domain specific languages and also offers the opportunity to query the models. Indeed, as the OperA metamodel is associated to the OperA ontology, a MAS model is associated to a MAS ontology instantiation. Consequently, the semantics of our models are stored in the MAS ontology instantiation, which is automatically produced using

[3] http://www.eclipse.org/modeling/mdt/?project=eodm

the MAS model (following a straightforward transformation). Hence, it allows for the designer, the verification and validation of models using ontologies; namely, the application of description logics for reasoning about the OperA language. In the following, some examples of semantics constraints, that we are interested in verifying, have been proposed.

- Checking if the objective of a dependency is an objective or a sub-objective of the dependency's initiator. For instance, in the running scenario, this would mean verifying that the role fire-station possesses an objective or sub-objective extinguish fire as it appears to be dependent on Firefighter_Team for it.
- Checking if each dependency is realised by at least one scene.
- Checking if roles are involved in a dependency, do indeed cooperate in at least one realisation scene of that dependency. Again, referring to the scenario, there must be at least one realisation scene of objective deal_with_big_fires dependency in which Firefighter_Team and Firefighter_Truck must cooperate.
- Checking that each role posses at least one dependency link.

The approach described above has been partly implemented in and illustrated by the OperettA prototype [5]. The results obtained with that prototype have encouraged us to move towards a more standardised and accessible version of the tool. The actual version is currently under development using Eclipse.

3.3 Conforming design models to domain ontology

The second aspect of our approach permits the use of ontology within our models as the source of domain specific related knowledge, necessary for the description and support of roles interaction and communication in OperA organisation and at a lesser extend, domain specific semantics constraints enrichment. That is, domain ontologies are integrated at the model level. They are defined, used and imported within the OperA modelling language as well as exported from it. For the latter two, a relation is defined at the meta-level, between the Eclipse Ontology Definition Metamodel and the OperA metamodel enabling the interaction with standard ontology representation languages like OWL. Meanwhile, in the ontological world the integration is at the same level. That is, the OperA ontology and domain ontology are at the same level within logic hierarchy; paving the way for the analysis of the overall structural aspect of the organisation, consisting in querying one knowledge base being the combination of the OperA ontology, the Domain Ontology and the derived MAS ontology instantiation.

Firstly, a vocabulary is available for describing interactions and supporting communications related to a domain. Referring back to our Fire_Station organisation, given the equivalent formal notation for the objective extinguish –e.g., *Extinguish-Fire(L : location)*– of the Firefigther_Team role, the concept of *Location* must be defined in the Domain ontology, otherwise the parameter type will not be available when defining the objective. Furthermore, at run-time, this ontology will be used by members of the organisation in order to communicate.

Secondly, the OperA Ontology can be enlarged by further modelling domain-specific constraints enriching the generic semantic constraints. It provides the opportunity to define modelling rules within the domain ontology as (re)configuration rules e.g., a specific domain could forbid more than 2 dependencies between two specific roles.

This second aspect provides a separation of concerns for knowledge taking into account its domain specific part and enabling at the same time the tool to tailor himself based on it.

4 Conclusions

This paper describes the features of an implemented design tool, OperettA, based on the OperA methodology. This enables the construction of a development framework to support software and services engineering. Besides, this contributes to the achievements of a more general research objective established within the ALIVE project. Specifically, the ALIVE project combines cutting edge Coordination technology and Organisation theory mechanisms to provide flexible, high-level means to model the structure of inter-actions between services in the environment.

The proposed approach (partly) implemented in OperettA deals with techniques to integrate features of the model driven architecture (MDA) with features of the ontology languages (OWL). Main benefits of our approach come from the fact that it provides a model driven approach for the specification of a MAS dealing with its semantics and its provision of domain specific knowledge. Hence, our approach allows designers for powerful reasoning mechanisms to be employed as well as a smart integration of domain specific knowledge that comes first to refine and enrich (along with further constraints) the specification of the design model.

References

1. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. PhD thesis, Universiteit Utrecht, 2004.
2. Brian Henderson-Sellers and Paolo Giorgini, editors. *Agent-Oriented Methodologies*. Idea Group Inc., 2005.
3. A. Fuxman, P. Giorgini, M. Kolp, and J. Mylopoulos. Information systems as social structures. In *FOIS '01: Proceedings of the international conference on Formal Ontology in Information Systems*, pages 10–21, New York, NY, USA, 2001. ACM.
4. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: representing knowledge about information systems. *ACM Trans. Inf. Syst.*, 8(4):325–362, 1990.
5. D. M. Okouya and V. Dignum. A prototype tool for the design, analysis and development of multi-agent organizations. In *DEMO Session: Proc. of the 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-08)*. ACM, 2008.
6. J. Bézivin, N. Farcet, J.-M. Jézéquel, B. Langlois, and D. Pollet. Reflective model driven engineering. In *UML*, pages 175–189, 2003.