2011

Supporting developers in complex systems modelling

Antonio A. Lopez-Lorca
University of Wollongong, aall645@uowmail.edu.au

Ghassan Beydoun
University of Wollongong, beydoun@uow.edu.au

Rodrigo Martinez-Bejar
University of Murcia

Holly Tootell
University of Wollongong, holly@uow.edu.au

Publication Details
Supporting developers in complex systems modelling

Abstract
Development of complex systems often requires building a large number of models with many interconnections and dependencies among them. The success of a project can be compromised by cognitive overload or limits of developers, who might miss relationships between elements of the models. Developing Multi-Agent Systems (MAS) is a typical example of where this may occur. Despite of its potential, this technology has not yet been widely adopted by industry due to its complexity and frequent errors in modelling activities. These errors typically propagate to later phases of the MAS development lifecycle, becoming costlier to fix and then lowering the quality of the final product. Early validation of MAS models can prevent rework efforts or building a system that is non-compliant with the client’s specification. In this paper we propose a process to support developers in modelling tasks using ontologies to validate and improve the quality of requirement analysis models as they are being developed and at the same time bridging the traditional gap between developers and clients. The proposed ontology-mediated validation is easily applicable to other kinds of architectures, however we illustrate this for MAS development as its complexity justifies any additional cost associated with applying it.

Keywords
supporting, developers, complex, modelling, systems

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/2019
Supporting Developers in Complex Systems Modelling

Antonio A. Lopez-Lorca
Ghassan Beydoun
School of Information Systems and Technology
University of Wollongong
Wollongong, Australia
Email: gall645@uowmail.edu.au
beydoun@uow.edu.au

Rodrigo Martínez-Béjar
Faculty of Informatics
University of Murcia
Murcia, Spain
Email: rodrigo@um.es

Holly Tootell
School of Information Systems and Technology
University of Wollongong
Wollongong, Australia
Email: holly@uow.edu.au

Abstract

Development of complex systems often requires building a large number of models with many interconnections and dependencies among them. The success of a project can be compromised by cognitive overload or limits of developers, who might miss relationships between elements of the models. Developing Multi-Agent Systems (MAS) is a typical example of where this may occur. Despite of its potential, this technology has not yet been widely adopted by industry due to its complexity and frequent errors in modelling activities. These errors typically propagate to later phases of the MAS development lifecycle, becoming costlier to fix and then lowering the quality of the final product. Early validation of MAS models can prevent rework efforts or building a system that is non-compliant with the client’s specification. In this paper we propose a process to support developers in modelling tasks using ontologies to validate and improve the quality of requirement analysis models as they are being developed and at the same time bridging the traditional gap between developers and clients. The proposed ontology-mediated validation is easily applicable to other kinds of architectures, however we illustrate this for MAS development as its complexity justifies any additional cost associated with applying it.

Keywords

Ontology, Multi-Agent System, Model Validation.

INTRODUCTION

We advocate using ontologies to support modellers in the development of complex systems. An ontology is used to validate and improve the quality of software workproducts during the development process. Underpinning an automated reasoning development support tool, ontologies help to uncover hidden knowledge and ill-defined relations in evolving software models. Moreover, as an element of joint development with the user, ontologies bridge common communication gaps between users and developers. As intermediary modelling elements, they facilitate and improve requirements elicitation later reducing the development and the maintenance cost of software systems (Sadraei et al. 2007). We propose using domain ontologies to assist modellers throughout the validation processes of requirement models. This is of particular significance in complex domains or system architectures not very well known to the developers such as Multi-Agent Systems (MAS). These are complex software systems situated in some environment capable of autonomous action to affect it to meet its design objective (Beydoun et al. 2009b; Wooldridge 2002). MAS can solve complex problems in which entities are required to work autonomously and collaborate to achieve their goals. Many authors have proposed new MAS specific development methodologies (notable examples are reviewed in Tran and Low (2005)) but despite this, the technology has not yet been widely adopted by industry. Pechoucek and Marik (2008) highlight that the absence of verification and validation in current methodologies is a serious hindrance towards adoption.

Whilst some existing works use ontologies to MAS development, none offer any formal development activities to validate that deployed agent-based systems correspond to original requirements. Some works use ontologies to assist in the development of workproducts in the detailed design phase, as advocated by Tran et al. (2006) and
Tran and Low (2008), where domain ontologies are used during development and run-time. By designing a reusable ontology allowing complex queries on the domain of “MAS development” Girardi and Leite (2008) also propose an ontology-based multi-agent development process that can model all the phases of development of MAS. Others focus on facilitating deployment of the system, such as the work presented by Nyulas et al. (2008), or the deployment of the methodology itself such as Hristozova and Sterling (2002) and Lister et al. (2005). None yet provide detailed and effective support for the validation of MAS models, which is the focus of our proposal. Closest to our focus is Okouya et al. (2008), where a MDA/Ontology approach is proposed to improve a MAS development framework by allowing the creation of models which are automatically transformed to an ontology, that is then verified against an MAS domain ontology. They aim to verify models to assess that they have been properly built, but our purpose goes further, we want to validate the models to assess that we have built the correct product that accurately corresponds to the functional requirements of a client. Our focus is the quality of the MAS workproducts through a domain enriched process rather than the software process itself as those works propose and specifically, the validation and verification of requirement models.

AN ONTOLOGY MEDIATED SOFTWARE MODELS VALIDATION

Our recent work in Lopez-Lorca et al. (2011) highlighted that a single pass over the models is not adequate to validate them. An effective validation rather needs to be interleaved with the actual development of the models. In other words, an iterative process is required. In this section, we present such an iterative ontology-based software models validation process. We introduce the process in two levels to facilitate its description: we first outline the activities required during the process and then we describe the iterative validation process model that organises and combines the activities.

High Level Process Description

There are two main types of activities in the process (Figure 1): the first deals with the preparation of the ontology that constitutes the cornerstone of the process and the second focuses on the iterative validation of the models. These two types of activities are also interleaved, that is, validation activities may lead to subsequent refinement of the ontology. The ontology may not be originally made at an appropriate level of details. The ontology needs to be sufficiently detailed to uncover hidden knowledge leading to new requirements which were overlooked (see the discussion section). Too little detail can lead to incomplete modelling and but too much detail on the other hand can add unnecessary complexity and extra costs to the validation process. It is important to validate the models as soon as they become available, as the cost associated to errors dramatically increases as the software development process proceeds (Westland 2002).

Ontology Acquisition: An ontology needs to represent the domain, its concepts, relationships and axioms as the client perceives it. If possible, a suitable ontology is retrieved from an existing repository, otherwise one is built using the most suitable ontology engineering techniques. Communication with the client has to be initially intensive to model the domain as detailed and conceptualized by the client. If the ontology lacks details then its effectiveness in the validation and modelling assistance to software developers is reduced. Input to this activity comes through interviewing client and acquiring any documents that can describe their business processes.

Ontology Augmentation: Each paradigm has an associated set of concepts and relationships that differentiate it from others. The ontology is augmented to represent features related to the chosen development paradigm. Domain concepts are linked to paradigm concepts. The resultant augmented ontology will always be larger than the original domain ontology that it is based on. The initial ontology contains concepts and relations from the analysis space, for example, Crew pursues Disembark passengers. The augmented ontology adds the scaffold to the domain knowledge modelled in the domain ontology, for example, Crew is a Role, Disembark passengers is...
a Goal and Each Role pursues at least 1 Goal. In next section we illustrate this augmentation process for the paradigm of MAS development. Input to this activity is the ontology acquired in the previous activity. The output is the domain ontology augmented according to the development paradigm.

Ontology Validation: Before using the ontology for validating the software models, the ontology itself is validated with the client by various members of the development team. The goal of this is twofold: to ensure that the ontology is compliant and accommodating of the conceptualisation of the client and to secondly ensure a common understanding of the domain across the development team (between persons responsible for developing and persons responsible for validation). The ontology consistency is also checked and an automatic reasoner is used to uncover implicit knowledge (see discussion section for more details). The input is the augmented domain ontology, and the output will be a refined version of the ontology.

Software Models Validation and Improvement: In this activity, model validation alternates iteratively with model improvement. It has as input the domain ontology, the software models and possibly the client’s feedback. The validation process concludes with this activity, hence, its output is the set of validated software models. Not all the models can be validated to the same extent using the ontology. Some may be very structured and the use of the ontology will provide specific instructions to improve them. Other models may be composed of free text, for which the use of the ontology will only be able to provide a guideline for the analyst to interpret. Depending on the decisions made by the developers upon the assessment of the reports generated in this activity the ontology may have to be improved. Next subsection presents a detailed validation process and in next section the operators for the MAS paradigm are introduced.

Development proceeds with each iteration further along the sequence of workproducts required by the chosen methodology. The development and validation of the software models are intertwined and done concurrently. Problems of reviewed models are fixed before their full development. Any models yet to be commenced in that iteration, will take advantage of the recommendations avoiding compounded errors. The software model development process will follow an iterative, incremental and concurrent development process model (as depicted in Figure 2). Our proposal is an add-on to the core process of creating software models and is intended to be completely independent of the underlying software models or their development methodology.

However, our validation mechanism is affected by the software development process in which the modelling activities are carried out. Some software lifecycles have underlying philosophies that oppose the iterative nature of our proposal, such as the waterfall process model. In this process there are typically no iterations, every phase has to be completed before starting the following and have an initial well-defined and stable set of requirements. As the ontology is a form of requirements specification, it could be considered to be correct from the beginning, needing to undergo no changes during the validation of the models. Under this assumption, our validation process would also be applicable to this software development process. Because of the sequential development of models, compound errors will still be prevented. However, if we accept that the ontology may undergo changes during the validation of the models, then as models cannot be revisited because the software development process prohibits iterations, the final set of validated models may present inconsistencies.
A detailed look into the validation activity

There are significant dependencies between models, as some of them look at the same aspect of the problem from different points of view. Errors in models may be propagated to other dependent models. This means that all the interdependent models will have to be revisited to fix any error. To avoid these compound errors, the validation process has to start as soon as a minimal set of models in an early stage of development is available (as illustrated in Figure 2). Each iteration will deal with a growing set of models, both in number and detail. Because of this, in a given iteration, the various models under consideration may have varying degrees of maturity. Validation of early versions of the models will focus on identifying only serious issues and will be addressed to keep the development of the models on the right track. However, the validation process for mature models will focus on the detail and fine-tune the models. For example, in the scenario depicted by Figure 2, in the validation of iteration 1, while M_2 is mature enough to be examined in detail, M_1 and M_3 are still relatively immature and only serious issues have to be tackled.

The validation of the models is performed through the application of operators defined specifically for the different models under review. In next section we propose operators for a set of MAS models. The operators compare aspects of the models with the corresponding knowledge modelled in the domain ontology. Divergences between models and ontology will be pointed out as recommendations to be reviewed. Further iterations will be necessary as long as the models keep being developed and while they are not fully compliant with the ontology. Although the ontology itself may change during the validation process, at the end of the process the models and ontology will converge. Developers analyse the generated recommendations and decide whether they will be accepted or rejected. Either models or ontology will have to be modified as a result of the decision made. Because of the relation between models and between models and ontology, changes in either of them will have effect in others. Let it be the scenario depicted by Figure 3.

![Interdependencies between models](image)

### Figure 3: Interdependencies between models

Three models M_1, M_2 and M_3 are being reviewed in certain iteration. The validation mechanism detects discrepancies between the ontology and the models and generates recommendations R_1, R_2 and R_3. The developer, after discussing with the client if necessary, decides that the ontology is right with regards R_1 and R_2 but not R_3. Then M_1 is changed according to the ontology to be compliant with regards to R_1 and R_2. The recommendation R_2 affects aspects of the system modelled as well in M_2 and M_3. Consequently, M_2 and M_3 are also modified. The ontology has to be adjusted to be compliant with the models with regards to R_3. These adjustments may be inconsistent with the knowledge already modelled in the ontology, therefore a consistency check is necessary. Moreover, the changes may provoke divergences with other aspects of the models that will be detected by the application of the operators in the following iteration. This example illustrates the consequences of modifications in ontology or models. The situation is aggravated in the development of complex systems where several developers model different models with many inter-dependencies. It is necessary to provide an automatic mechanism to maintain dependencies. We are working on a support tool for our validation process that tackles this challenge (see the discussion section).

Another role of the ontology that deserves further attention is as a requirements elicitation mechanism. The ontology is compared with the models using the operators to detect discrepancies. This can lead to finding new requirements that had been overlooked by clients and developers if the ontology includes knowledge not included in the models. These recommendations have to be discussed with clients to decide if they are relevant for the purpose of the system being developed or if they should be ignored. Applying a reasoner to the domain ontology makes explicit knowledge that was only implicitly included in the conceptualisation from the client. In the discussion section we consider further aspects of ontology reasoning applied to our problem.

**ONTOGRAPHY-MEDIATED VALIDATION FOR THE MAS PARADIGM**

As earlier described, the domain ontology is augmented with paradigm-dependent features. In our MAS case, we identify terms in MAS modelling: Goal (a functional requirement of the system (Sterling and Taveter 2009)), Role (any capacity that the system requires in order to achieve its goal (Sterling and Taveter 2009)), Activity
(some work carried out by a role in order to fully or partially fulfil its goal), *Environment Entity* (any entity which is not part of the system but it is needed by the roles to achieve their goals) and *Agent* (proactive or reactive components of the system which plays one or more roles (Sterling and Taveter 2009)). The terms defined in the domain ontology are associated to these concepts. Agents are situated in time, i.e. they are time-aware. This is very different from traditional objects in Object Oriented technologies, where objects do not have sense of time. Every decision agents make and every action they carry on has to fit in certain order. To specify this order, the relations *precedes* and *follows* establish which activities precede and follow which ones.

Illustrative examples of other relations defined to model the MAS development paradigm are:

- Each Agent plays at least 1 Role
- Each Role pursues at least 1 Goal
- A Goal may need some Environment Entity
- An Agent may use some Environment Entity
- Each Goal is achieved by at least 1 Activity
- Each Role participates in at least 1 Activity
- An Activity may precede some Activity

After defining concepts and relations of the development paradigm, we select a set of MAS requirement models, from the existing MAS development methodologies. A recent survey in Tran and Low (2005) of ten prominent agent-oriented methodologies shows that there is a set of common models across existing methodologies. For instance, we identify the following common models in descending acceptance order: Agent model (90%), goal model (60%), interaction model (60%), organisation model (40%), role model (30%), and environment model (30%). Without loss of generality, we work with all those models. The validation process is based on semantic comparisons between models and ontology elements. For example, the relations *Aircraft transports Luggage* and *Aircraft carries Baggage* are semantically equivalent in our domain.

In Lopez-Lorca et al. (2011) we presented a case study in which we validated MAS models developed for the ROADMAP methodology (Juan et al. 2002; Sterling and Taveter 2009). The case study was the development of a simulator for the *Aircraft Turnaround Activity*. Aircraft Turnaround refers to the process of preparing an arriving aircraft for departure. Typical operations that are involved are: passengers disembark, luggage is unloaded, safety checks performed, then the activities for the new flight, loading food, luggage and embarking passengers are performed. It is highly desirable to minimise the time that the aircraft remains in the airport, as longer stays mean higher costs for the airline. The MAS simulation was expected to identify how to optimise the process, completing a speedier turnaround with fewer resources (staff). Turnaround-related operations vary in duration and in how they are handed over within the sequence of tasks. There was scope for decentralisation and parallelisation. This made the domain an excellent candidate for a MAS simulation. Illustrative examples of the domain ontology that we created for the case study are:

- Prepare Arrival is a sub-goal of Aircraft Turnaround.
- Position Airbridge achieves Prepare Arrival.
- Ground Staff participates in Position Airbridge.
- Ground Staff uses Airbridge.
- Position Airbridge follows Position Wheel Chocks.
- Crew controls Passenger.

The models examples provided in Lopez-Lorca et al. (2011) were real excerpts of the models developed for the case study following the ROADMAP methodology. Those excerpts were only the goal and agent model and were used to illustrate an early version of the operators tuned to the ROADMAP methodology. In this paper we illustrate the versatility of our validation method. We adapt the operators originally defined for the ROADMAP methodology to comply with the MOBMAS MAS development methodology (Tran and Low 2008; Tran et al. 2006). Furthermore, we apply the operators to a complete set of agent analysis model templates (six in total).

**Agent Model Validation**

Agent classes encompass one or more roles, being capable of dynamical change amongst its assigned roles at run-time, thus exhibiting dynamic behaviour (Tran and Low 2008). They describe the goals that each agent
pursues, the events that activate them, interaction channels with other agents and the resources that they need (e.g. Figure 4). The validation consists of the following proposals:

- To add to the model any agents defined in the ontology but not included in the model and removing any agents from the model without corresponding agent defined in the ontology.

- To add to the agent model any goals associated in the ontology to any of the roles played by agents but not listed, and removing any listed goals which are not associated to any of the roles played by the corresponding agent in the ontology.

- To add to any agent an event for each terminal goal (it has activities that achieve it) that is pursued by a role played by that agent if no role played by the agent participates in the activity that immediately precedes the first activity that achieves the goal, and removing the event otherwise.

- To add interaction links between pairs of agents if there is at least one activity in which participate roles played by both agents, and removing interaction links if there is no such activity.

- To add any missing resource used by any of the roles played by a corresponding agent or needed in any of the activities that the roles participate in, and removing any resource not defined in the ontology as used by any of the roles played by the corresponding agent or needed in any of the activities in which that agent participates.

Figure 4: Example of agent model

Goal Model Validation

A goal is a state of the world that an agent wants to achieve (Tran and Low 2008). In a goal model (e.g. Figure 5), the main goal of the system is sub-divided into sub-goals. The ontology can ensure that all the specified goals are accounted for and hierarchy is maintained. The goal model validation consists of the following proposals:

- To add to the model any roles defined in the ontology but not used in the goal model, and removing those not defined in the ontology.

- To add any relation between goals and sub-goals defined in the ontology but not used in the model, and removing those not defined in the ontology.
Interaction Model Validation

Interaction models (e.g. Figure 6) capture the patterns of data exchanges between agents instances (Tran and Low 2008). The labels of the interaction messages do not always correspond exactly with the activities defined in the ontology, therefore this model cannot be validated to the same degree as others using the ontology. The validation consists of adding to the model any agent or role defined in the ontology but not used in any interaction model (taking the report described below as guideline), and removing those not defined in the ontology as participating in the corresponding interaction.

Organisation Model Validation

The organisation model (e.g. Figure 7) reflects the positions, individuals or departments that exist in the organisational context and the interaction channels between them (Tran and Low 2008). The ontology can help checking that relations among roles are correct and that the structure is correct. The validation consists of the following proposals:

- To add to the organisation model any role defined in the ontology but not having been included in it, and removing those roles that do not exist in the ontology.
- To add to the organisation model relationships between roles and its containing department if there exist a subsumption relation in the ontology including the role as subclass and removing any relationships that do not have their role as subclass in a subsumption relation in the ontology.
- To add relationships between roles if there exists at least one activity in the ontology in which both roles participate and removing relationships if there is no such activity.
Role Model Validation

Roles serve as the building blocks for defining agent classes. They refer to the position that the entity has in the organisation and what it is expected to do (Tran and Low 2008). The ontology can check that all roles have been defined in the role model and their tasks correspond to the activities that they participate in. It can also validate the roles hierarchy. Figure 8 depicts an example of a role model. The validation consists of the following proposals:

− To add to the model any role defined in the ontology but without correspondence and removing any role that has not been defined in the ontology.
− To add to roles the activities that have been defined in the ontology as carried out by them that have not been already included as task, and removing those tasks that do not exist in the ontology as activities in which the role participates.
− To add to the model any hierarchical relation (peer, control, controlled by) between roles if such relationship has been defined in the ontology but not in the model, and removing it otherwise.

Environment Model Validation

An environment entity is a non-agent resource that provides system-specific services. They do not belong internally to the system and are available to agents in other systems (Tran and Low 2008). The environment model (e.g. Figure 9) relates agents with the resources they use. The validation consists of the following:

− To add to the model environment entities defined in the ontology but not in the model, and removing those not defined in the ontology.
− To add to the environment model any relationship between resource and agent that is defined in the ontology but not included in the model, and remove those which have no associated relationship defined in the ontology.
To integrate our validation of all models described in this section seamlessly into the development process, we use an iterative, incremental and concurrent development process (as earlier shown in Figure 1). The process iterates over intermediate versions of the model to achieve high quality. It is incremental in nature, not all the models are considered during each iteration. It is concurrent as development and validation activities overlap.

**DISCUSSION, CONCLUSION AND FUTURE WORK**

Applying the validation process can incur additional development cost. This is easily justified in the development of complex systems in which the numerous dependencies between models frequently cause errors. It is also justified in critical software applications where errors can be very costly and disastrous. This cost overhead may also be justified in other scenarios, in which modellers are not sufficiently experienced or when the domain itself is complex or unknown. That said, the cost of the validation can be greatly reduced by more effective reuse of existing ontologies. With the advent of the Semantic Web, more ontologies are made available. More importantly, there is a great scope for generating the amendment proposals automatically.

An important drawback of our proposed method is the overhead that its application causes, not only with regards to the validation of models, but the augmentation of the ontology as well. This task can be arduous for large ontologies. To tackle this problem we are currently studying the capabilities of ontology reasoners. We are developing a formal ontology to model the development methodology (metamodel ontology) that could be reused in any project involving that particular development methodology. This ontology will enable us to apply a reasoner to automatically assign properties to the concepts defined in the domain ontology. Besides speeding up the augmentation process, the reasoner gives us another important advantage. As the metamodel ontology has been formally defined, implicit relations will be automatically extracted from the domain ontology. This hidden knowledge may be significant for developers and clients, helping to identify new requirements. This mechanism can also be used to detect inconsistencies and violations of axioms defined in the metamodel ontology by concepts of the domain ontology. We are developing a support tool to explore all these capabilities and hence speed up the process of augmenting the domain ontology. Preliminary results seem promising. The reasoning mechanisms that we are testing, automatically augment a relatively simple domain ontology with the paradigm concepts and relations. Implicit knowledge is uncovered making the initial ontology to grow several times its original size. It also detects violations in the cardinalities of the relations as defined in the metamodel. Besides of this pre-processing tool, we are also developing the model validation tool as a decision support system. The tool uses the operators defined in this paper to compare the augmented ontology with the models. Then, it produces a report highlighting the differences between model and ontology. This tool will make possible to quickly and efficiently apply our methodology on real contexts.

The definition of the metamodel ontology opens the doors for further research possibilities. We are planning to introduce a further layer of abstraction on top of the metamodel ontology. We will base on the approach presented in Beydoun et al. (2009a) to automatically decide which models are necessary to be created and validated for a given development methodology. This will extend our mechanism and will enable us to examine and validate relations between models and not only relations between concepts within models.

As research progresses in these directions, we will make available the tool that supports an agile validation process. This tool will automatically maintain the dependencies between recommendations and between recommendations and ontology, alleviating the developer of this cumbersome task. We intend to apply our validation process to further case studies to fine-tune it and to test our forthcoming tool. The first case study is a cellar inventory management system, to manage interactions between wine producers, brokers, reviewers and end customers. The second is an automated curriculum updating system, to manage e-learning sources and interactions between academics, students and the sources.
REFERENCES


COPYRIGHT

Antonio A. Lopez-Lorca, Ghassan Beydoun, Rodrigo Martínez-Béjar, Holly Tootell © 2011. The authors assign to ACIS and educational and non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive licence to ACIS to publish this document in full in the Conference Papers and Proceedings. Those documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. Any other usage is prohibited without the express permission of the authors.