2011

An ontology-mediated validation of software models

Antonio Lopez Lorca
University of Wollongong

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

Leon Sterling
Swinburne University of Technology

Tim Miller
University of Melbourne

An ontology-mediated validation of software models

Abstract

When errors in software modelling activities propagate to later phases of software development lifecycle, they become costlier to fix and lower the qual-ity of the final product. Early validation of software models can prevent rework and incorrect development non-compliant with client's specification. In this paper we advocate the use of ontologies to validate and improve the quality of software models as they are being developed, at the same time bridging the traditional gap between developers and clients. We propose a general ontology-mediated process to validate software models that can be adapted in a broad range of software de-velopment projects. We illustrate this for Multi-Agent Systems (MAS) develop-ment providing early evidence of the soundness of our approach. We successfully validate and improve the quality of MAS models for a real-life development pro-ject, illustrating the ontology-mediated models validation in a commercial setting.

Disciplines

Physical Sciences and Mathematics

Publication Details

Lopez-Lorca, A., Beydoun, G., Sterling, L. & Miller, T. (2011). An ontology-mediated validation process of software models. International Conference on Information Systems Development (ISD10) (pp. 1-12). Czech Republic: Springer.

An Ontology-Mediated Validation Process of Software Models

A. Lopez-Lorca[1], G. Beydoun[1], L. Sterling[2,3], T. Miller[3]

[1] University of Wollongong, Australia, {aall645, beydoun}@uow.edu.au

[2] Swinburne University of Technology, Australia, lsterling@groupwise.swin.edu.au

[3] University of Melbourne, Australia, tmiller@unimelb.edu.au, leon@cs.mu.oz.au

Abstract When errors in software modelling activities propagate to later phases of software development lifecycle, they become costlier to fix and lower the quality of the final product. Early validation of software models can prevent rework and incorrect development non-compliant with client's specification. In this paper we advocate the use of ontologies to validate and improve the quality of software models as they are being developed, at the same time bridging the traditional gap between developers and clients. We propose a general ontology-mediated process to validate software models that can be adapted in a broad range of software development projects. We illustrate this for Multi-Agent Systems (MAS) development providing early evidence of the soundness of our approach. We successfully validate and improve the quality of MAS models for a real-life development project, illustrating the ontology-mediated models validation in a commercial setting.

1 Introduction

Ontologies[1] provide a mechanism of representing domain knowledge to a varying degree of formalism [4]. They can be utilised by software developers and at the same time read by future users of a system. Our work is in line with what Guarino [8] calls *ontology-driven information system development.* We advocate the use of ontologies to validate and improve the quality of software workproducts during development processes. As an element of joint development with the user, they can bridge common communication gaps between users and developers. We illustrate using an ontology to check consistency, correctness and completeness of models against initial system requirements. We believe that ontologies are generally faster to develop and easier to understand than most analysis and design models that require specific and in-depth methodological knowledge. As an initial system development step, an ontology engineer interviews a client to capture an ontology reflecting their conceptualisation of their problem and desired features of the solution. We expect that as intermediary modelling elements, ontologies can

[1] Understood as a theory about the structure and behavior of the real world in general.

facilitate and improve the development of software workproducts, potentially reducing the development and maintenance costs of software systems. In this paper, we provide methodology-independent and ontology-based add-on validation processes to facilitate the creation of models for inexperienced modellers and to assist more experienced ones detecting and resolving errors. The ontology can assist modellers throughout the validation processes, which is of particular importance when the domain is complex or not very well known to the modellers. Our proposal is of particular significance for our chosen field of Multi-Agent Systems (MAS). Unlike other disciplines such as Object Oriented Development, MAS development is not so well understood, and due partly to its complexity it has not yet been widely adopted by industry. Whilst the focus of our illustrations is on applying ontologies to improve the development of MAS models, we expect our approach to be easily adaptable to other paradigms such as agile methods.

2 Related Work

The use of ontologies for general software development to validate conceptual models to produce better quality models is not a new idea. However, most existing validation work focuses on using a formal ontology to choose a specific suitable conceptual modelling language for the domain e.g. [14] and more recently [2, 3]. In [2], an Eclipse-based tool is proposed to build and automatically verify conceptual models developed in a language (OntoUML) that uses a foundation ontology to extend UML. In [3], OntoUML conceptual models are automatically transformed to a logic-based language to allow the validation of the modal meta-properties. Our approach is not specific to any modelling language.

Many existing works focus on the use of ontologies to MAS. Of these many focus on the process itself. For example, by designing a reusable ontology allowing complex queries on the domain of "MAS development" in [7] Girardi and her colleagues propose an ontology-based multi-agent development process that can model all the phases of development of MAS. As another example, Nyulas et al. present in [12] an architecture to develop and deploy end-to-end solutions for MAS. They focus on the deployment steps of the system. In [9], a method is given to adapt extreme programming methods to develop a lightweight ontology to help agile development of MAS. It is refined further in [11]. Our focus in this paper is the quality of the workproducts through a domain enriched process rather than the software process itself. Other works use ontologies to assist in the development of workproducts in particular in the detailed design phase. Tran et al. [16] present an ontology-based MAS for the domain of a peer-to-peer (P2P) information sharing community where ontologies are built and used in development-time to create the models and in run-time to exchange information between agents. They use domain ontologies during development and run-time, they do not provide detailed support for the validation of MAS, which is the focus of our proposal. Okouya et al. present [13] a MDA/Ontology approach to improve OperettA, a MAS development framework. They allow the creation of MAS models which are automatically

transformed into an ontology. The semantic constraints of the ontology (and of the MAS models) are verified against a MAS domain ontology. They aim to the verification of the models to assess that they have been built properly, but our purpose goes further: we want to validate the models to assess that we have built the correct product according to the user requirements.

Our approach shares similar goals with the work developed by Brandão et al. [6]. They propose the use of ontologies as a method for the verification of MAS designs. They use an ontology to model the MAS modelling language. These model-diagram mappings enable the automatic validation of the models to check that there are neither intra-model nor inter-model inconsistencies. Again, the main difference with our proposal is that they can validate the models against their theoretical structure and dynamics, but use no information about the specification or application domain and their proposal has not been properly validated. Furthermore, they do not generalise their efforts to outside MAS development.

In conclusion, our work uses ontologies to inform modelling of workproducts and is unique in that it is development methodology independent, is focused on the quality of workproducts and does not depend on any specific modelling language. The rest of the paper is structured as follows: Section 3 presents our ontology-based add-on validation process for MAS models and its key features. Section 4 presents a case study in which we apply our process to the simulation of the aircraft turnaround. Section 5 concludes with future work discussion.

3 Ontology-Mediated Add-on Validation Process of MAS Software Models

In this section we present our ontology-mediated process to validate MAS software models. 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 [18]. Our proposed ontology-mediated MAS software models validation (Fig. 1) consists of five activities that overlap with the development process. Our proposal is an add-on to this core process and completely independent of the underlying software models or their development methodology. Although the model development activity is not in essence part of our proposal, it has been included in Figure 1 to show that it is intertwined with model validation.

In the *Ontology Development* activity 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 conceptualised 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 elicitation techniques such as interviewing clients and acquiring any documents that can describe their business processes. For example, in our case study in Section 4, in addition to the interviews we use diagrams provided by the client to describe the existing timeline for an aircraft turnaround process.

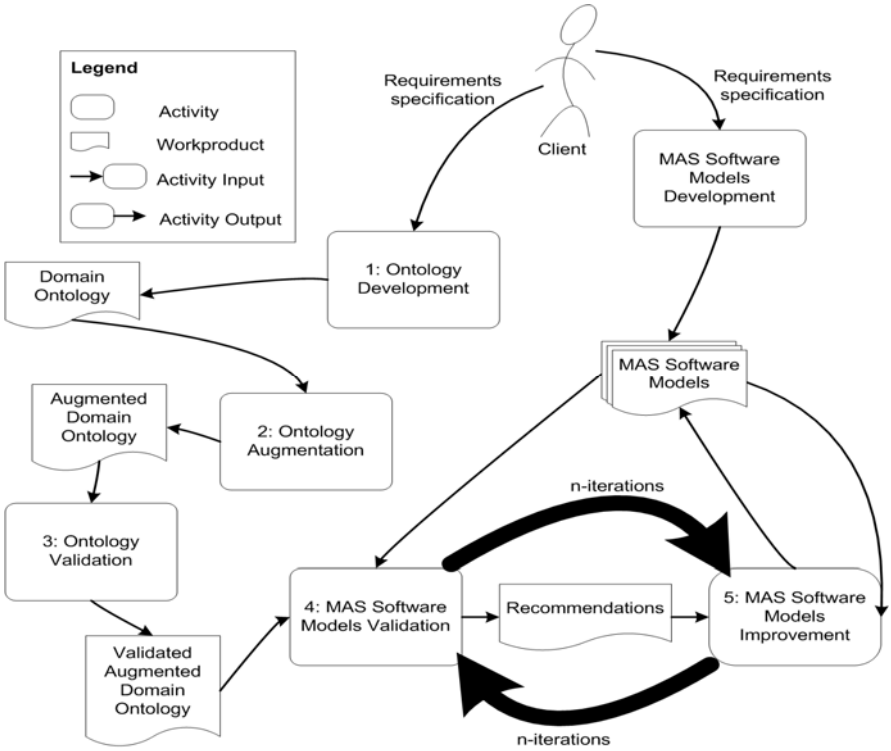

Fig. 1. Ontology-mediated software models validation add-on process overview.

In the *Ontology Augmentation* activity, the ontology is augmented to represent features related to the chosen development paradigm. Domain concepts are linked to paradigm concepts. Domain concepts are annotated and relations between them are created according existing relations defined for the paradigm. For the paradigm of MAS, we identify terms in MAS modelling: *Goal* (a functional requirement of the system [15]), *Role* (any capacity that the system requires in order to achieve its goal [15]), *Activity* (some work carried out by a role in order to fully or partially fulfil its goal), *Environment* (any entity which is not part of the system but it is needed by the roles to achieve their goals) and *Agent* (a proactive or reactive component of the system plays one or more roles [15]). Some domain concepts are annotated with these terms and related properties are also modelled (summarised in Table 1). Moreover, agents are time-aware. Every decision agents make and every action they carry on has to fit in certain sequence. To specify this sequence, the properties *precedes* and *follows* establish which activities precede and follow which ones.

In the *Ontology Validation* activity, before using the ontology for validating the MAS 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 for validation).

Table 1. MAS-dependent properties used to annotate the ontology.

Domain	Property	Range	Domain	Property	Range
Goal	has a	Goal	Role	uses	Environment
Role	responsible for	Goal	Agent	plays	Role
Role	participates in	Activity	Activity	fulfils	Goal
Role	is peer	Role	Activity	needs	Environment
Role	controls	Role	Activity	precedes	Activity
Role	is controlled by	Role	Activity	follows	Activity

In the *MAS Software Models Validation* activity, the MAS models are validated against the augmented ontology for consistency and compliance with the client's specification. This activity provides the control element for new iterations. A new iteration will be necessary as long as any recommendation is made to improve the quality of the 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.

In the *MAS Software Models Improvement* activity, the recommendations are analysed by the developers to choose which to apply and which to ignore. After improving the quality of the MAS models according to chosen recommendations, the new set of models will be used as input for Activity 4 in the next iteration.

Development proceeds with each iteration further along the sequence of work-products required by the chosen methodology. The development and validation of the MAS 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 MAS software model development process will follow an iterative, incremental and concurrent development process model.

In order to perform the validation described in Activity 4, the process has to be instantiated: A MAS development methodology has to be chosen and mechanisms to validate the associated MAS models defined.

A recent survey in [17] of ten prominent agent-oriented methodologies shows that there is a set of common models across existing methodologies. The following models are the most common (in increasing acceptance order): Agent model (90%), goal model (60%), interaction model (60%), scenarios (50%), organisation model (40%), role model (30%), and environment model (30%). Without loss of generality, we work with the ROADMAP methodology [10, 15] which provides all those models. Moreover, authors of ROADMAP availed themselves to develop the models for our case study to simulate aircraft turnaround (the process between an aircraft landing at an airport and taking off again). The validation process is

based on comparing models and ontology elements pairwise, taking into account their semantics. For example, suppose that the relation `Aircraft transports Luggage` is defined in the ontology, while in the environment model it is stated that `Aircraft carries Baggage`. Both are equivalent in our domain.

As an example, we show the mechanisms to validate the two more popular models of the ROADMAP methodology, the goal and the agent model.

A *Goal Model* can be seen as a use case for an open and distributed system [5, 15]. It sub-divides the main goal of the system into sub-goals and specifies roles participating in the fulfilment of each goal (e.g. Fig. 2). The ontology can ensure that all the specified goals are accounted for, the roles integrity and hierarchy is maintained. The goal model validation consists of the following proposals:

1. 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.
2. To add any relation between goals and sub-goals, `Goal has Goal`, defined in the ontology but not used in the model, and removing those not defined in the ontology.

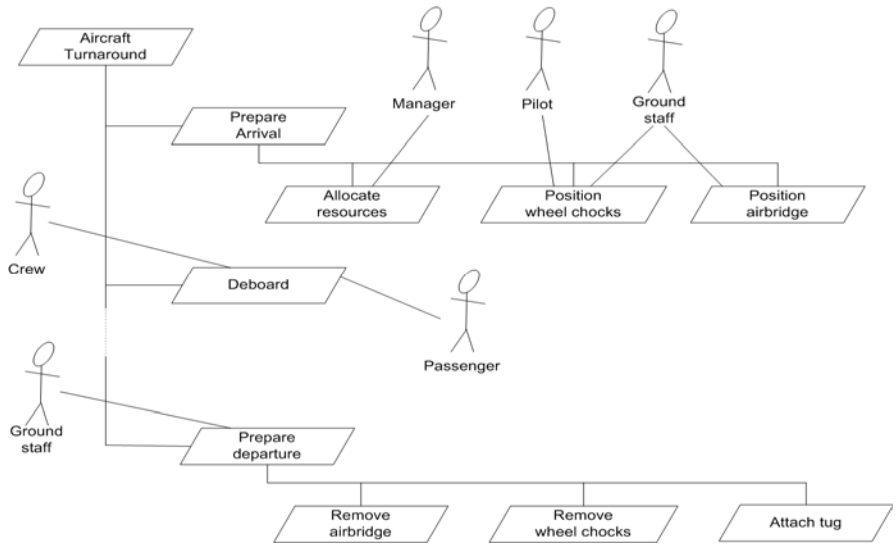

Fig. 2. A goal model decomposing the 'aircraft turnaround' goal.

3. To add to the model any relation between roles and goals, `Role responsibleFor Goal`, defined in the ontology but not used or for which there is no associated role in the model, and removing those not defined in the ontology.

Agent Models (e.g. Fig. 3) transform abstract constructs from analysis, e.g. roles, to design constructs, agent types, which are realised implementation [1, 15]. They describe the activities that each agent is involved in, along with their pre- and postconditions. The ontology validates that activities defined for each agent comply with the specification, that each agent plays the correct roles and partici-

pates in the correct activities using necessary environment entities to fulfil its goals. The validation consists of the following proposals:

1. To add to the model set any agents defined in the ontology but without corresponding models, and removing any agent models without corresponding agent defined in the ontology.
2. To add to every agent model, any missing activities associated with any of the roles (`Role participatesIn Activity`) played by agents (`Agent plays Role`), and removing any listed activities which are not associated to any of the roles played by the corresponding agent (as shown in the ontology).

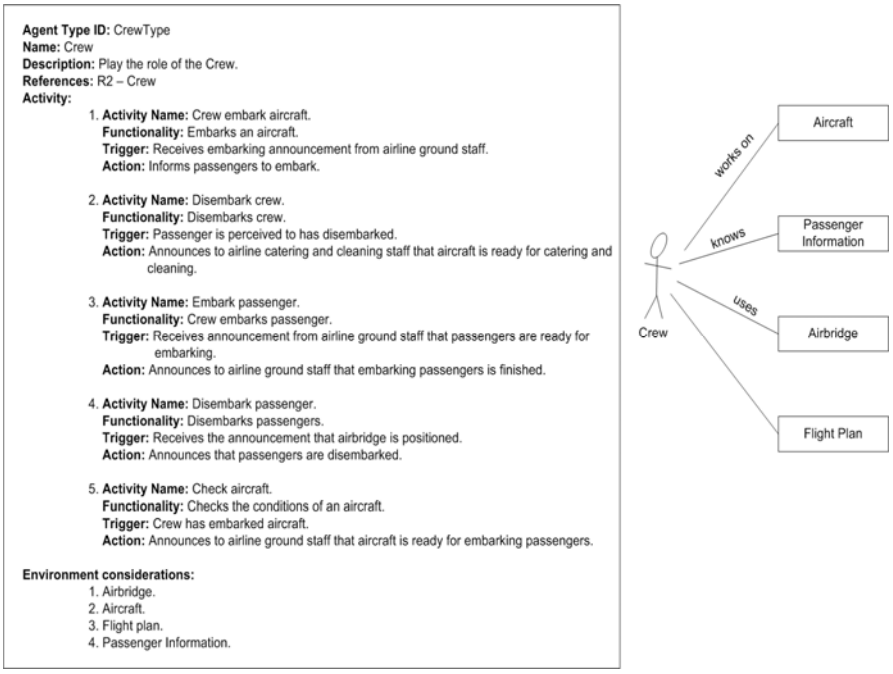

Fig. 3. Example of an agent model.

3. To update the trigger or action fields to correct the pre- and postconditions of any activity in the ontology (`Activity precedes Activity` and `Activity follows Activity` respectively) whose pre- or postconditions do not match any of the ones described by the fields trigger and action (any activity may have several pre- or postconditions). If the fields are incomplete, propose completion with the suitable activities as in the ontology.
4. To add to the environment list in every agent model, any missing environment entities used by any of the agent roles (`Role uses Environment`) or in any of the activities (`Activity needs Environment`) that the roles participate in (`Role participatesIn Activity`), and removing any listed environment entity not defined in the ontology as used by any of the agent roles or needed in any of the activities in which the agent participates.

4 Case Study: An Aircraft Turnaround Simulator

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. The study arises from a Linkage project involving the third author. 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 is 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 is scope for decentralisation and parallelisation. This makes the domain an excellent candidate for a MAS simulation.

We developed an ontology that models the problem as conceptualised by the client. We based on the documentation that the client provided us with, as well as several interviews with them. For the next step, we augmented the domain ontology annotating certain classes with concepts related to the MAS domain (see Section 3). Figure 4 shows an excerpt of the ontology and its augmented version.

Figure 5 shows the current state of the software development process involved and models interaction within this case study. The process is in its second/third iterations. The evolution of the models is clear so far: some models have already reached their final versions while others are expected to do so at the end of the third iteration. The validation process is iterative: models are validated as soon as they are developed and are revisited as soon as amendment proposals are reported by the iterative validation activity. This process proved to be effective as models are interrelated and therefore starting their development using corrected versions of the ones they are based on saves time as avoids rework. Iterations are undertaken until models converge and no further amendments are proposed by the validation activity. Due to lack of space, we cannot go into details for all the recommendations made, so we present illustrative examples of the process.

The initial set of models under validation included: environment, goal, role, organisation, interaction and scenario models. During the second iteration an agent model was added to this set. They evolved as follows:

1. *Environment model*: In *Iteration 1*, this model lacked explicit relations between concepts and was not compliant with the ontology. Rework was proposed to improve it. In *Iteration 2*, it was changed thoroughly to be more faithful to the ontology. But still a few changes were proposed to align it with the ontology.
2. *Goal model*: In *Iteration 1*, this model was close to what was expected, only minor changes were proposed to improve it. In *Iteration 2*, it remained unchanged. Detected discrepancies had been suggested by the client beforehand. We consider this model validated.
3. *Role model*: In *Iteration 1*, this model was close to what was expected of it, so only minor changes were proposed to improve it. In *Iteration 2*, most of the

proposals were accepted. Few remain pending further discussions with the client. We consider this model validated.

4. *Organisation model*: In *Iteration 1*, this model presented some inconsistencies regarding the hierarchical relation between roles. In *Iteration 2*, no improved version of this model has yet been developed.

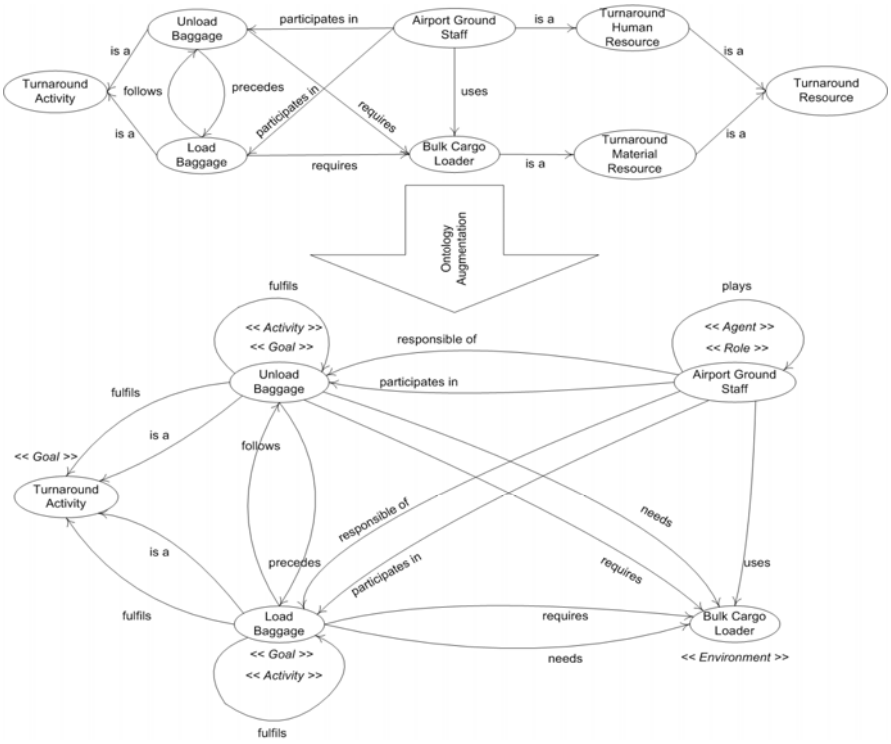

Fig. 4. Example of ontology augmentation: the above snap shot shows concepts and relations that are changed. Classes are annotated according to MAS concepts examined in Section 3.

5. *Interaction model*: In *Iteration 1*, this model presented severe discrepancies with the ontology. Some interactions were not complete and others were not correctly planned (sequential or parallel). Most of the interactions were coordinated by the role Manager. Some of the potentiality of multiple processing by a MAS to be wasted. Agents could not interact autonomously with each other to achieve their goals. Using the ontology, we identified interactions that needed the mediation of the manager. In *Iteration 2*, this model is under development.
6. *Scenario model*: In *Iteration 1*, no recommendations were made to the scenario model, as a very basic version was provided. It included no sub-scenarios, only the main process was sketched. We suspected that upon its extension we would be able to propose amends (as it eventually happened). In *Iteration 2*, an extended version of the scenario was developed which included sub-scenarios detailing the turnaround process. These sub-scenarios reflected the suggestions

made about the role of the manager in the interaction models. The scenario still has some minor flaws but generally speaking is correct. This proves the importance of concurrent work between validation and development activities to avoid rework.

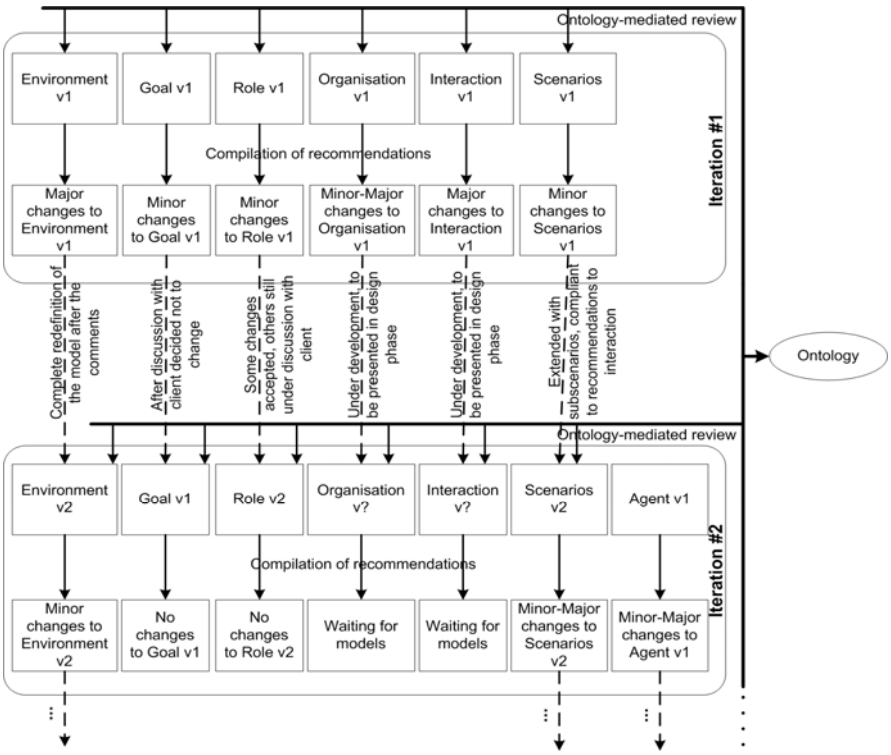

Fig. 5. Current state of development: The arrow head pointing towards the ontology entails that some aspect of the ontology were refined as result of validation process of MAS models itself.

7. *Agent model*: In *Iteration 1*, the agent models were not provided. In *Iteration 2*, this model was included for the first time. Some significant proposals are made. In particular some relations inter-models have to be improved and a few activities need changes regarding their triggers.

A third validation iteration is initiated, but yet to be completed. The number of proposals produced by the validation process has largely converged. The evolution of the models have produced high quality MAS models, and smoother interactions with the client indicating higher degree of satisfaction of client and developers.

5 Conclusion and Future Work

We apply ontologies to improve the quality of software models. Unlike other proposals we take into account the domain as specified by the client's requirements,

filling any communication gap between clients and developers. Models are validated as soon as they are available, fixing errors as they arise and avoiding compounding and propagating errors to later phases of the development. To integrate our validation add-on seamlessly into the development process, we use an iterative, incremental and concurrent development process. The process iterates over intermediate versions of the model to achieve high quality. It is incremental in nature, not all the models are considered for each iteration. It is concurrent as development overlaps validation activities. This process can incur additional development cost and requires a cost justification. It is particularly appealing in critical software application where errors can be very costly and disastrous. This cost overhead may also be justified in the following scenarios:

- In developments of inexperienced modellers to guide them and avoid errors.
- In MAS developments of experienced modellers in any other technology, as agents have many particularities which cannot be found in other paradigms.
- In projects where the domain is complex or unknown, for experienced and inexperienced modellers alike.
- In software product line developments, where models have to be error free, as they will be reused in multiple developments.
- In projects dealing with the same domain, to enable reuse of the domain knowledge generated (i.e. the ontology).

That said, the cost of the validation can be greatly reduced by more effective reuse of existing ontologies. With advent of the Semantic Web, more ontologies are made available. More importantly, there is a great scope for generating the amendment proposals automatically. Indeed, we are now studying this possibility with the expectation to develop a tool that can significantly alleviate the burden of the details of the ontology-mediated validation process. In the future, we also intend to apply the ontology-mediated software model validation process to further cases studies to fine-tune it and to test our forthcoming tool.

Acknowledgement

This work is supported by the Australian Research Council. The authors wish to thank developer Bin Lu for providing software models to support the case study.

References

1. Ashmalla, A., G. Beydoun and G. Low. *Agent Oriented Approach for Call Management*, in *International Conference on Information Systems Development*. 2009. Nanchang, China: Springer.
2. Benevides, A.B. and G. Guizzardi. *A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML*, in *International Conference on Enterprise Information Systems*. 2009. Milan, Italy: Springer-Verlag.
3. Benevides, A.B., G. Guizzardi, B.F.B. Braga and J.P.A. Almeida. *Assessing Modal Aspects of OntoUML Conceptual Models in Alloy*, in *International Workshop on Evolving Theories of Concep-*

tual Modeling, at the International Conference on Conceptual Modeling. 2009. Gramado, Brazil: Springer-Verlag.

4. Beydoun, G., A. Hoffmann, J.T. Fernández-Breis, R. Martínez-Béjar, R. Valencia-García and A. Aurum. *Cooperative Modeling Evaluated.* International Journal of Cooperative Information Systems. 2005. **14**(1): p. 45-71.
5. Beydoun, G., G. Low, B. Henderson-Sellers, H. Mouratidis, J.J. Gómez-Sanz, J. Pavón and C. Gonzalez-Pérez. *FAML: A Generic Metamodel for MAS Development.* IEEE Transactions on Software Engineering. 2009. **99**(1): p. 841-863.
6. Brandão, A.A.F., V.T.d. Silva and C.J.P.d. Lucena, *Observed-MAS: An Ontology-Based Method for Analyzing Multi-Agent Systems Design Models*, in *Agent-Oriented Software Engineering VII*. 2007. Springer Berlin / Heidelberg. p. 122-139.
7. Girardi, R. and A. Leite. *A knowledge-based tool for multi-agent domain engineering.* Knowledge-Based Systems. 2008. **21**(7): p. 604-611.
8. Guarino, N. *Formal Ontology and Information Systems*, in *Formal Ontology in Information Systems*. 1998. Trento, Italy: IOS Press.
9. Hristozova, M. and L. Sterling. *An eXtreme Method for Developing Lightweight Ontologies*, in *Workshop on Ontologies in Agent Systems, at the International Join Conference on Autonomous Agents and Multi-Agent Systems*. 2002. Bologna, Italy: ACM.
10. Juan, T., A. Pearce and L. Sterling. *ROADMAP: Extending the Gaia Methodology for Complex Open Systems* in *International Joint Conference on Autonomous Agents and Multi-Agent Systems*. 2002. Bologna, Italy: ACM.
11. Lister, K., M. Hristozova and L. Sterling. *Reconciling Implicit and Evolving Ontologies for Semantic Interoperability*, in *Ontologies for Agents: Theory and Experiences*, V. Tamma, Cranefield, S., Finin, T., Willmott, S, Editor. 2005. Birkhäuser Basel. p. 121-144.
12. Nyulas, C., M. O'Connor, S. Tu, D. Buckeridge, A. Okhmatovskaia and M. Munsen. *An Ontology-Driven Framework for Deploying JADE Agent Systems*, in *International Conference on Web Intelligence and Intelligent Agent Technology*. 2008. Sydney, Australia: IEEE Computer Society.
13. Okouya, D., L. Penserini, S. Saudrais, A. Staikopoulos, V. Dignum and S. Clarke. *Designing MAS Organisation through an integrated MDA/Ontology Approach*, in *International Workshop on Transforming and Weaving Ontologies in Model Driven Engineering*. 2008. Tolouse, France: CEUR-WS.org.
14. Shanks, G., E. Tansley and R. Weber. *Using ontology to validate conceptual models.* Communications of the ACM. 2003. **46**(10): p. 85-89.
15. Sterling, L. and K. Taveter. *The Art of Agent-Oriented Modeling*. Intelligent Robotics and Autonomous Agents. 2009: MIT Press.
16. Tran, N., G. Beydoun and G. Low. *Design of a Peer-to-Peer Information Sharing MAS Using MOBMAS (Ontology-Centric Agent Oriented Methodology)*, in *International Conference on Information Systems Development*. 2006. Budapest, Hungary: Springer.
17. Tran, N. and G. Low. *Comparison of Ten Agent-Oriented Methodologies*, in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Editors. 2005. Idea Group: Hershey. p. 341-367.
18. Westland, J.C. *The cost of errors in software development: evidence from industry*. Journal of Systems and Software. 2002. **62**(1): p. 1-9.