

Modeling Mental States in the Analysis of Multiagent Systems Requirements

Alexei Lapouchnian
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3G4, Canada
alexei@cs.toronto.edu

Yves Lespérance
Department of Computer Science and Engineering
York University
Toronto, ON, M3J 1P3, Canada
lesperan@cs.yorku.ca

ABSTRACT

This paper presents an agent-oriented requirements engineering approach that combines informal i^* models with formal specifications in the CASL language through the use of Intentional Annotated Strategic Rationale diagrams. In the resulting framework, agent goals and knowledge are represented as their mental states, which allows for the formal analysis of, among other things, agent interactions and incomplete knowledge. CASL models can also serve as high-level specifications for multiagent systems.

Categories and Subject Descriptors

D.2.1 [Software Engineering]: Requirements/Specifications – methodologies; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – Multiagent Systems

General Terms

Theory, Design, Verification.

Keywords

Agent-Oriented Software Engineering, Logics for Agent Systems, Goal-Oriented Requirements Engineering.

1. INTRODUCTION

Agent technology is well-suited for constructing applications that are open, distributed, and adaptable to change. We need software engineering methods that are tailored for designing systems with multiple communicating agents that strive to achieve their assigned goals thereby delivering the desired application.

In requirements engineering (RE), *goal-oriented approaches* (e.g., Tropos [1]) have become prominent. In Goal-Oriented Requirements Engineering (GORE) high-level stakeholder objectives are modeled as goals and refined into fine-grained requirements or environment expectations. Their reliance on goals makes GORE methods and agent-oriented software engineering a great match. Therefore, it is natural to use a goal-oriented requirements engineering approach when developing MAS. With GORE, it is easy

to make the transition from the requirements to the high-level MAS specifications. In this context, while it is possible to informally analyze small systems, formal analysis is needed for any realistically-sized one. Thus, the aim of this work is to devise an agent-oriented requirements engineering approach with a formal component that supports reasoning about agents' goals (and knowledge), thereby allowing for rigorous formal analysis of the requirements expressed as the objectives of the agents in a MAS.

In our approach, we integrate the i^* goal-oriented modeling framework [11] with CASL [8], a formal agent-oriented specification language that supports the modeling of agent mental states. This gives the modeler the flexibility and intuitiveness of the i^* notation as well as the powerful formal analysis capabilities of CASL. To bridge the gap between informal i^* diagrams and formal CASL specifications we propose an *intermediate notation* that can be easily obtained from i^* models and then mapped into CASL. With our i^* -CASL-based approach, a CASL model can be used both as a requirements analysis tool and as a formal high-level specification for a multiagent system that satisfies the requirements. This model can be formally analyzed using the CASLve [9] tool or other tools and the results can be fed back into the requirements model.

2. BACKGROUND

i^* [11] is an agent-oriented modeling framework used for requirements engineering, etc. The i^* framework has two main components. Firstly, the *Strategic Dependency (SD) model* provides an external view of actors and the intentional dependencies between them. There can be four types of dependencies based on what is being delegated – a goal, a task, a resource, or a softgoal dependency. Secondly, the *Strategic Rationale (SR) model* describes the internals of actors (see Figure 1). The same four types of nodes are used in SR models along with task decompositions and means-ends links. Softgoals are used to represent non-functional requirements. Softgoal contribution links in SR models specify how process alternatives fare with respect to quality concerns. In Figure 1, we see a fragment of an (annotated) SR model showing the interactions between a meeting initiator agent (MI) and a meeting scheduling system agent (MS). The MI depends on the MS for information on the suggested meeting date range. The diagram shows how some of the agents' tasks are decomposed.

The Cognitive Agents Specification Language (CASL) [8] is a formal specification language that combines theories of action [6] and mental states [7] expressed in the situation calculus [5] with ConGolog [2], a concurrent, non-deterministic agent-oriented

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS '06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

Copyright 2006 ACM 1-59593-303-4/06/0005...\$5.00.

programming language with a formal semantics. CASL specifications consist of two parts: the model of the domain and its dynamics (the declarative part) and the specification of the agents' behaviour (the procedural part). The domain is modeled in terms of *agents*, *primitive actions* executed by agents, *situations*, which are states of the domain that result from the execution of sequences of actions, and *fluents*, which are predicates and functions that may change from situation to situation. Axioms of the following types are used: *action precondition axioms*, *successor state axioms* (SSA) [6] that specify how primitive actions affect fluents, and *initial state axioms* describing the initial state of the domain and the initial mental states of the agents. Agent behaviour is specified using a rich high-level programming language with recursive procedure declarations, loops, conditionals, non-determinism, concurrency, and interrupts [2].

CASL supports formal modeling of agents' goals and knowledge based on a *possible worlds semantics* incorporated into the situation calculus, where situations are viewed as possible worlds [7]. **Know**(*agt*, ϕ ,*s*) specifies that an agent knows some formula ϕ , while **Goal**(*agt*, ψ ,*s*) indicates that in situation *s* the agent *agt* has the goal that ψ holds. We commonly use the formula **Goal**(*agt*,**Eventually**(ψ),*s*) to state that *agt* has the goal that ψ is eventually true. Primitive communication actions such as *request*, *inform*, and *cancelRequest* are defined, which affect the mental states of the agents involved in the obvious way.

3. THE APPROACH

3.1 Increasing Precision with iASR Models

Our goal is to tightly associate *i** models with formal specifications in CASL. The standard SR diagrams are geared to informal analysis and are imprecise. For instance, they do not provide any information on whether subtasks are to be executed sequentially or concurrently, under certain conditions, etc. However, CASL is a precise language. To handle this mismatch we use Intentional Annotated SR (iASR) models, which are precise graphical representation for the procedural part of CASL specifications.

To develop an iASR diagram for an actor, one starts with the regular SR diagram for that actor. It then can be appropriately transformed to become an iASR model every element of which can easily be mapped into CASL. The main steps for producing iASR models from SR ones are discussed below.

Annotations. The main tool for disambiguating SR models is *annotations* – textual constraints on SR models [10]. *Composition annotations* (sequence (“;”), concurrency (“||”), prioritized concurrency (“>>”), and alternative (“|”)) are applied from right to left and specify how the subtasks/subgoals are to be combined to execute the supertask or achieve the goal. *Link annotations* are applied to subtasks/subgoals and specify how/under what condition they are supposed to be achieved/executed. These annotations are: *while* loop, *for* loop, *if* condition, the *pick* (non-deterministic choice of argument), the *interrupt*, and the *guard* (it blocks execution until its condition becomes true). The latter two are commonly used modify agents' behaviour based on changes in their mental states. *Applicability condition annotations* specify when the goal achievement alternatives are applicable.

Softgoals. Softgoals are imprecise and thus cannot be handled in a formal specifications language. Therefore, softgoals are used to

choose the best process alternatives and then are removed before iASR models are produced. Alternatively, softgoals can be operationalized or metricized, thus becoming hard goals.

Providing agent interaction details. While *i** abstracts from modeling agent interaction details, CASL models requests for services, monitoring for such requests, etc., which are important for formally analyzing MAS. Thus, high-level aspects of agent interaction must be specified in iASR models. In the iASR model in Figure 1 we can see that the interaction between the MS and the MI to exchange the meeting date range information has been specified. Note the use of the interrupt annotation that triggers when the MI acquires the goal to inform the MS about the meeting the date range following the MS's request. We can see here how the goals of individual agents are explicitly modeled. iASR models can also refer to agents' knowledge. For example, the guard annotation on the MS ensures that it can only proceed when it knows the date range.

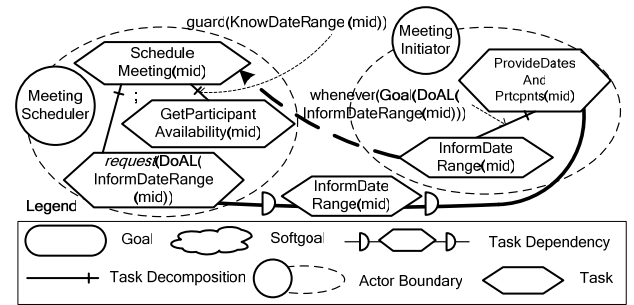


Figure 1. Adding agent interaction details to iASR model.

3.2 Mapping iASR Diagrams into CASL

Once all the necessary details have been introduced into an iASR diagram, it can be mapped into the corresponding formal CASL model, thus making the iASR model amenable to formal analysis.

The modeler defines a mapping **m** that maps every element (except for intentional dependencies) of an iASR model into CASL. This mapping associates iASR model elements with CASL procedures, primitive actions, and formulas so that a CASL program can be generated from an iASR model. Specifically, agents are mapped into CASL procedures that specify their behaviour; roles and positions (these can be played/occupied by agents) are mapped into similar procedures with an agent parameter so that they can be instantiated by individual agents; leaf-level task nodes are mapped into CASL procedures or primitive actions; composition and link annotations are mapped into the corresponding CASL operators, while conditions present in the annotations map into CASL formulas. Non-leaf task nodes are mapped into procedures that are automatically constructed from procedures/actions corresponding to the subtasks, which are combined based on the annotations used. For example, the task *ScheduleMeeting* in Figure 1 will be mapped into the following CASL procedure:

```

proc ScheduleMeeting(mid)
  request(MS, MI, DoAL(InformDateRange(mid))) ;
  guard(KnowDateRange(mid)) do
    GetParticipantAvailability(mid)
  endGuard
endProc

```

A goal node is mapped into a formula (its formal definition) and an achievement procedure that is automatically constructed based on the means for achieving the goal specified in the iASR model.

Intentional dependencies are not mapped into CASL per se – they are established by the associated agent interactions. iASR tasks requesting services from agents will be mapped into actions of the type `request(FromAgt, ToAgt, Eventually( $\varphi$ ))` for achievement goals φ . To ask an agent to execute a certain known procedure, the depender must request it with: `request(FromAgt, ToAgt, DoAL(Procedure))`. This requests the execution of the procedure while allowing ToAgt to perform other activities. The service provider must monitor for its mental state changes with appropriate interrupts (see Figure 1).

3.3 Formal Verification

Once an iASR model is mapped into CASL, it is ready to be formally analyzed. One tool that can be used is CASLve [9], a theorem prover-based verification environment that can be employed to verify executability of programs, their termination, etc.

Other approaches can also be used. However, tools based on techniques like simulation work with much less expressive languages than CASL requiring simplification of CASL specifications before these methods can be used on them. For example, most simulation tools cannot handle mental state specifications; these would then have to be operationalized before simulation is performed. The ConGolog interpreter can be used to directly execute such simplified specifications, as done in [10].

If expected properties of the system are not entailed by the CASL model, it means that the model is incorrect and needs to be fixed. The source of an error found during verification can usually be traced to a portion of the CASL code and to a part of its iASR model since our systematic mapping supports traceability.

4. DISCUSSION

In the approach presented in this paper and in [4], we produce CASL specifications from i^* models for formal analysis and verification. The approach is related to the Tropos framework [1] in that it is agent-oriented and is rooted in RE concepts. Our method is not the first attempt to provide formal semantics for i^* models. For example, Formal Tropos (FT) [3] supports formal verification of i^* models through model checking. Also, in the i^* -ConGolog approach [10], on which our method is based, SR models are associated with formal ConGolog programs for simulation and verification. The problem is that in these methods, goals of the agents are abstracted out of formal specifications. This is done due to the fact that the formal components of these approaches (the model checker input language for FT and the restricted situation calculus theories for the others) do not support reasoning about agent goals (or knowledge). However, most of the interactions among agents involve knowledge exchange and goal delegation since multiagent systems are developed as social structures. Thus, complementing informal modeling techniques such as i^* with formal analysis of agent goals and knowledge is very important in the design of multiagent systems. We propose a framework where goals are not removed from the agent specifications, but are modeled formally and can be updated following requests. This allows agents to rea-

son about their objectives. Information exchanges among agents are also formalized as changes in their knowledge state. In our approach, goals are not system-wide properties, but belong to concrete agents. The same applies to knowledge. This subjective point of view provides support for new types of formal analysis on CASL specifications. Thus, this method is more agent-oriented and allows for precise modeling of stakeholder goals. Modeling of conflicting stakeholder goals, a common case in requirements engineering, and agent negotiations is also possible. All of the above can be useful for the later stages of AOSE since most specifications of agent communications model them as changing agents' mental states. Moreover, many agent implementation languages/tools are based on models of mental states.

The full paper is available at: www.cs.toronto.edu/~alexci.

5. REFERENCES

- [1] J. Castro, M. Kolp and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 27(6):365-389, 2002.
- [2] G. De Giacomo, Y. Lespérance and H. Levesque. ConGolog, A Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, 121:109-169, 2000.
- [3] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, P. Traverso. Specifying and Analyzing Early Requirements in Tropos. *Requirements Engineering Journal*, 9(2):132-150, May 2004.
- [4] A. Lapouchnian. Modeling Mental States in Requirements Engineering - An Agent-Oriented Framework Based on i^* and CASL. M.Sc. Thesis. Department of Computer Science, York University, Toronto, 2004.
- [5] J. McCarthy and P. Hayes. Some Philosophical Problems From the Standpoint of Artificial Intelligence. *Machine Intelligence*, Volume 4, pp. 463-502, Edinburgh University Press, 1969.
- [6] R. Reiter. The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In V. Lifschitz, Ed. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pp. 359-380, Academic Press, 1991.
- [7] R.B. Scherl and H. Levesque. Knowledge, Action, and the Frame Problem. *Artificial Intelligence*, 144(1-2):1-39, 2003.
- [8] S. Shapiro and Y. Lespérance. Modeling Multiagent Systems with the Cognitive Agents Specification Language - A Feature Interaction Resolution Application. *Proc. ATAL-2000*, pp. 244-259, LNAI 1986, Springer, 2001.
- [9] S. Shapiro, Y. Lespérance and H. Levesque. The Cognitive Agents Specification Language and Verification Environment for Multiagent Systems. In C. Castelfranchi and W.L. Johnson Eds., *Proc. AAMAS'02*, Bologna, Italy, July 15-19, 2002, ACM Press.
- [10] X. Wang and Y. Lespérance. Agent-Oriented Requirements Engineering Using ConGolog and i^* . *Proc. AOIS-01*, 2001.
- [11] E. Yu. Towards modeling and reasoning support for early requirements engineering. In *Proc. RE '97*, Annapolis, USA.