1-1-2005

Towards executable specification: combining i* and AgentSpeak(L)

Farzad Salim
University of Wollongong, fsalim@uow.edu.au

Chee Fon Chang
University of Wollongong, cfc@uow.edu.au

Aneesh Krishna
University of Wollongong, aneesh@uow.edu.au

Aditya K. Ghose
University of Wollongong, aditya@uow.edu.au

Towards executable specification: combining i* and AgentSpeak(L)

Abstract

Agent-oriented conceptual modeling(AoCM) approaches in Requirements Engineering (RE) have received considerable attention recently. Semi-formal modeling frameworks such as i* assist analysts in requirements elicitation and reasoning of early-phase RE. AgentSpeak(L) is a widely accepted agent programming language. The Strategic Rationale (SR) model of the i* framework naturally lends itself to AgentSpeak(L) programs. Furthermore, the Strategic Dependency (SD) component of the i* framework prescribes the interaction between the agents in a multi-agent environment. This paper proposes a formal methodology for transforming a SR model to an AgentSpeak(L) agent. The constructed AgentSpeak(L) agents will then form the essential components of a multi-agent system, MAS.

Keywords

Towards, Executable, Specification, combining, AgentSpeak

Disciplines

Business | Social and Behavioral Sciences

Publication Details

Salim, F., Chang, C., Krishna, A. & Ghose, A. K. (2005). Towards executable specification: combining i* and AgentSpeak(L). In W. Wong, W. Chu & N. Juristo (Eds.), International Conference on Software Engineering and Knowledge Engineering (pp. 739-742). USA: Knowledge Systems Institute Graduate School.

Towards Executable Specification: Combining *i** and AgentSpeak(*L*)

Farzad Salim, Chee Fon Chang, Aneesh Krishna, Aditya Ghose

Decision Systems Laboratory
University of Wollongong
NSW 2500 Australia
$\{fs26, c03, ak86, aditya\}@uow.edu.au$

Abstract

Agent-oriented conceptual modeling(AoCM) approaches in Requirements Engineering (RE) have received considerable attention recently. Semi-formal modeling frameworks such as i assist analysts in requirements elicitation and reasoning of early-phase RE. AgentSpeak(*L*) is a widely accepted agent programming language. The Strategic Rationale (SR) model of the i* framework naturally lends itself to AgentSpeak(*L*) programs. Furthermore, the Strategic Dependency (SD) component of the i* framework prescribes the interaction between the agents in a multi-agent environment. This paper proposes a formal methodology for transforming a SR model to an AgentSpeak(*L*) agent. The constructed AgentSpeak(*L*) agents will then form the essential components of a multi-agent system, MAS.*

1 Introduction

It is recognised by [8, 2, 1] that Requirements Engineering(RE) is a vital phase in system development. The *i** modeling framework [8, 7] is a modeling language which supports reasoning in the early-phase of RE. The framework is based on the notion of "distributed intentionality" [8, 7] with the aim of capturing and modeling intentional characteristics such as goals, beliefs, capabilities and commitments assigned to actors in an organisational environment [7]. AgentSpeak(*L*) [5] is a widely accepted agent programming language which deals with the concepts of goals, beliefs, plans, actions and intentions. Even though these two frameworks are employed for two diverse tasks, (*i** models at a very abstract level and AgentSpeak(*L*) at a detailed programming level) there exists a conceptual thread between the two frameworks. In this paper, we propose a methodology for deriving AgentSpeak(*L*) programs from a given *i** model. It is our thesis that the transformation between the two frameworks can be automated hence changes in the *i** diagram can be reflected in the AgentSpeak program and vice-versa. Our goal is to simulate *i** model in a multi-agent system composed of AgentSpeak(*L*) agents. We believe that such a coalition can bridge the gap between early-phase and the late-phase of RE. The ability to execute an *i** model via a multi-agent system permits us to verify a wide range of conditions such as *creation conditions*, *invariant conditions* and *fulfillment conditions* mentioned in Formal Tropos [3]. These constraints can neither be expressed in *i** model notation nor within individual AgentSpeak(*L*) agents. This paper will discuss the methodology that provide us with agents within the envisioned executable specifications *MAS*.

2 The *i** modeling framework

The *i** model for agent-oriented conceptual modelling was designed primarily for early-phase requirements engineering. An *i** model consists of two main modelling components: the Strategic Dependency (*SD*) Model and the Strategic Rationale (*SR*) Model. Both, SD and SR diagrams are graphical representations that describe the world in a manner closer to the users perceptions. The SD diagram consists of a set of nodes and links. Each node represents an "actor", and each link between the two actors indicates that one actor depends on the other for something in order that the former may attain some goal. The depending actor is known as *depender*, while the actor depended upon is known as *dependee*. The object around which the dependency relationship centers is called *dependum*. The SD diagram represents the goals, task, resource, and soft goal dependencies between actors/agents.

The *SR* diagram is the central concept in *i** model. It represents the internal intentional characteristics of each actor/agent via two types of links, the *task decomposition link* and the *means-end link*. The task decomposition link provides details of the tasks and the (hierarchically decomposed) sub-tasks to be performed by each actor/agent while the means-end link relates goals to the resources or tasks required to achieve them. In this paper we will not elaborate

more on the functionality of the *i** model, readers who may want more details are directed to [7].

Formally, an *i** model is a pair $\langle SR, SD \rangle$ where SD is a graph while SR is a set of graphs (one for each actor). The graph SD is a pair $\langle Actors, Dependencies \rangle$. $Actors$ is a set of nodes, one for each actor. $Dependencies$ is a set of edges, and is partitioned into the following sets: *goal dependencies* (denoted by $D_G(SD)$), *task dependencies* (denoted by $D_T(SD)$), *resource dependencies* (denoted by $D_R(SD)$) and *softgoal dependencies* (denoted by $D_S(SD)$). Each edge $e \in Dependencies$ may be viewed as a triple $\langle T_o, T_d, ID \rangle$. Each graph in SR is a triple $\langle$*SR-nodes, SR-edges,ActorID*$\rangle$. The nodes in *SR-nodes* are partitioned into the following 4 sets: *goal nodes* (denoted by N_G(*SR-nodes*)), *task nodes* (denoted by N_T(*SR-nodes*)), *resource nodes* (denoted by N_R(*SR-nodes*))and *softgoal nodes* (denoted by N_S(*SR-nodes*)). The edges in *SR-edges* can be of two kinds: *means-end links* or *task decomposition links*. A means-end link can be further classified into the following three types: *goal-task links* (or *GTlinks*), *resource-task links* (or *RTlinks*) and *softgoal-task links* (or *STlinks*). A task decomposition link (or *TDlink*) can relate a task to another task, goal, resource or softgoal. A means-end link may be viewed as representing one option in an OR-decomposition of its parent goal. All task decomposition links represent components of an AND-decomposition of their parent task.

3 AgentSpeak(L) Programming Language

AgentSpeak(L) [5] is an agent programming framework/language with explicit representations for beliefs and intentions. An AgentSpeak(L) agent is a set $\langle E, B, P, I, A, S_E, S_O, S_I \rangle$ where: E is a set of events, B is a set of base beliefs, P is a set of plans, I is a set of intentions and A is a set of atomic actions. Also, there are three selection functions: S_E selects an event from a set of events, S_O selects a plan from a set of plans and S_I selects an intention from a set of intentions.

There are two types of goals in AgentSpeak(L). An *achievement goal* (a predicate prefixed with "!") states that the agent wants to achieve a state of the world where the associated predicate is true. A *test goal* (a predicate prefixed with "?") states that the agent wants to test if the associated predicate is a true belief.

Events in AgentSpeak(L) might be external or internal. External events represent the changes in the state of the world that should be handled by the agent. On the other hand, internal events are triggered from within the agent as a result of executing a current plan.

Plans are the central concept of the abilities of an agent. They enable the agent to respond to changes in the environment. A plan is composed of two main parts, a *head* and a *body*.

$$e : b_1; \ldots; b_n \leftarrow h_1; \ldots; h_n$$

The *Head* of a plan is a *2-tuple*; *triggering event*, e and *context*, $b_1; \ldots; b_n$. A triggering event is required to identify if the plan is a relevant plan for an event that has been selected from E. The context of a plan consists of beliefs that should hold true for that plan to be applicable. The body of a plan,$h_1; \ldots; h_n$ is a sequence of sub-goals or actions that should be executed for a plan to be successfully completed.

Intentions are formed when an agent commits to a particular plan to achieve a goal. Interested readers may refer to the original AgentSpeak(L) paper [5] for more details.

4 Environment Simulator

A multi-agent simulation (*MAS*) is an executable environment. It consists of AgentSpeak(*L*) agents that are the counterpart of SR diagrams and a special environment-agent that is used to simulate the SD. The environment-agent can be customised to verify a range of conditions such as *creation conditions, invariant conditions, fulfilment conditions* that are clearly defined in Formal Tropos [3].

Definition 1 An *MAS* is a pair $\langle Agents, \mathcal{ESA} \rangle$ where $Agents = \{a_1, \ldots, a_n\}$, each a_i is an AgentSpeak(*L*) agent and $\mathcal{ESA}$ is a specially designed Environment Simulator Agent implemented in AgentSpeak(*L*) .

Note that this paper does not elaborate on the functionality of $\mathcal{ESA}$ but focuses more on the methodology for constructing AgentSpeak(*L*) agents from SR diagrams. Therefor we take a simplistic approach by considering $\mathcal{ESA}$ to be an instance of an AgentSpeak(*L*) agent. It consists of a set of plans that correspond to the actions that other agents (SR) perform within *MAS*. The body of plans may contain two types of atomic actions: 1) The changes that can motivate other agents within *MAS* (i.e., to make an agent perform a task), or 2) Simple notifications to the analyst about the occurance of an inconsistency. For example, $\mathcal{ESA}$ may have a plan p that has a, b as context and $fulfilled(x)$ in its body. Therefore, it sends the notification about the fulfilment condition x, whenever a, b comes true in *MAS*.

5 Customized AgentSpeak(*L*)

Given an AgentSpeak(*L*) agent = $\langle E, B, P, I, A, S_E, S_O, S_I \rangle$, we will proceed with defining the action predicates that will be used while mapping the SR diagram to AgentSpeak(*L*) .

Definition 2 A plan in the plan library of an AgentSpeak(*L*) agent is a 3-tuple $\langle \tau, \chi, \pi \rangle$ where:

$-$ τ is a triggering event.
$-$ χ is the context that must be entailed by the agent's current set of beliefs for the plan to be applicable.
$-$ π is the body of the plan.

Definition 3 Given three goal predicate symbols, $goal$, $task$, $resource$ and a term t:
$-$ $!goal(t)$ is a valid goal iff $t \in N_G$.
$-$ $!task(t)$ is also a valid goal iff $t \in N_T$.
$-$ $resource(t)$ is a valid belief atom iff $t \in N_R$.

Definition 4 Given four action predicate symbols, $RequestAchieve$, $RequestPerform$, $RequestResourse$, $Supply$ and a term t:
$-$ $RequestAchieve(t)$ is a valid action iff $t \in N_G$.
$-$ $RequestPerform(t)$ is a valid action iff $t \in N_T$.
$-$ $RequestResource(t)$ is a valid action iff $t \in N_R$.
$-$ $Supply(t)$ is also a valid action iff $t \in N_R$.

6 Function ϕ: Mapping Rules

In this section we will define the mapping function ϕ and the rules that constraint its behaviour. Given an *i** model, we use the function ϕ to construct AgentSpeak(*L*) agents that will form the main component of *MAS*. However, we assume the *i** models that are used for a simulation satisfy two conditions. Firstly, sub-tasks within the *Task- decomposition links* are ordered from left to right, based on their execution order[1]. Secondly, there do not exist two elements with the same name within an *i** model.

Before we proceed any further, we shall introduce three functions that will assist us in describing the mapping process. Note that these functions are neither part of the mapping function ϕ nor part of the AgentSpeak(*L*) syntax.
$-$ $Trigger(p)$ = τ
$-$ $Context(p)$ = Con
$-$ $Body(p)$ = π
Note that for readability we use subscripts that indicate the ID's for agents/actors and their components, ($i, j \in \mathbb{N}$).

Definition 5 $\phi : \bigcup_I \rightarrow \bigcup_{MAS}$ where $\bigcup_I$ is a class of *i** models and $\bigcup_{MAS}$ is a class of multi-agent systems where each agent implemented in AgentSpeak(*L*) .

Given the function ϕ, for every $m \in \bigcup_{I_n}$ where $m = \langle\langle Actors, Dependencies\rangle, \langle SR_i - nodes, SR_i - edges, actor_i\rangle\rangle$, an $mas \in \bigcup_{MAS}$ is valid with respect to m iff it satisfy the following postulates.

1. $a \in Actors$ *iff* $a \in Agents$.

[1]There is no specific ordering for *Means-end links*, because there is an *"or"* relationship between the sub-tasks

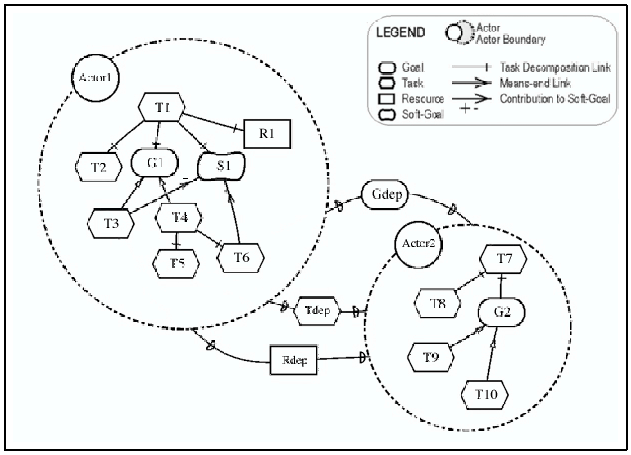

Figure 1. *i Model**

2. For every $g \in N_G(SR_i - nodes)$, there exist an agent $Actor_i \in Agents$ s.t $Actor_i = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and $\exists p \in P_i$ s.t $Trigger(p)$ = $!goal(g)$.

3. For every $t \in N_T(SR_i - nodes)$, there exist an agent $Actor_i \in Agents$ s.t $Actor_i = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and there exist only one $p \in P_i$ s.t $Trigger(P)$ = $!task(t)$.

4. For every $r \in N_R(SR_i - nodes)$, there exist an agent $Actor_i \in Agents$ s.t $Actor_i = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and there exist only $resource(r) \in B_i$.

5. For every $\langle a, b\rangle \in GTlink$, there exist an agent $Actor_i \in Agents$ s.t $Actor_i = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and there exist only one $p \in P_i$ where $Trigger(p)$ = $!goal(a)$ and $!task(b) \in Body(p)$.

6. For every $\langle a, b\rangle \in TDlink$, there exist an agent $Actor_i \in Agents$ s.t $Actor_i = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and there exist only one $p \in P_i$ where $Trigger(p)$ = $!task(a)$ and,
if $b \in N_G$ then $!goal(b) \in Body(p)$
else if $b \in N_T$ then $!task(b) \in Body(p)$
else if $b \in N_R$ then $resource(b) \in Context(p)$.

7. For every $\langle T_o, T_d, ID\rangle \in D_G \cup D_T \cup D_R$, there exist two agents, $T_o, T_d \in Agents$ where $T_o = \langle E_i, B_i, P_i, I_i, A_i, S_{E_i}, S_{O_i}, S_{I_i}\rangle$ and $T_d = \langle E_j, B_j, P_j, I_j, A_j, S_{E_j}, S_{O_j}, S_{I_j}\rangle$:
if $\langle T_o, T_d, ID\rangle \in D_G$
$\exists p \in P_i$ s.t $RequestAchieve(ID) \in Body(p)$, and
$\exists p \in P_j$ s.t $Trigger(p)$ = $!goal(ID)$.
else if $\langle T_o, T_d, ID\rangle \in D_T$
$\exists p \in P_i$ s.t $RequestPerform(ID) \in Body(p)$, and
$\exists p \in P_j$ s.t $Trigger(p)$ = $!task(p)$.
else if $\langle T_o, T_d, ID\rangle \in D_R$

$\exists p \in P_i$ s.t $RequestResource(ID) \in Body(p)$, and
$\exists p \in P_j$ s.t $Supply(ID) \in Body(p)$.

Figure 2 is a snapshot of the prototype, it shows the output of the prototype when the *i** model in Figure 1 is given as input. The current version of the prototype relies on the output from the *i** Organization Modeling Tool (OME)[2].

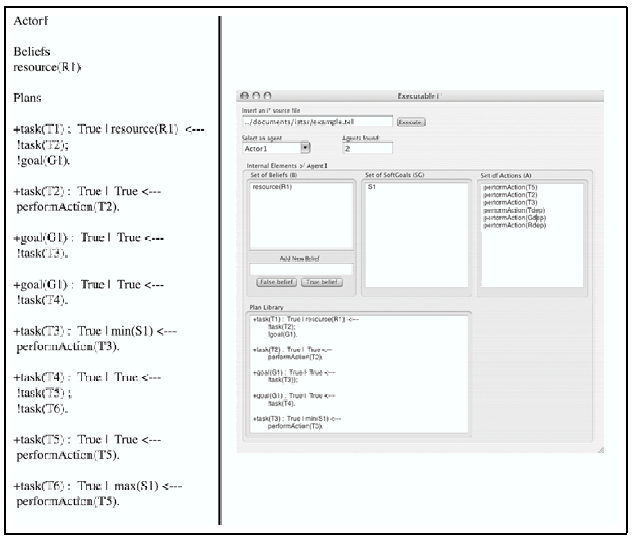

Figure 2. AgentSpeak(L) Agents

One should note that to construct a *MAS*, some other necessary information must be available that are not directly expressed in the *i** model. However, this prototype only focuses on extracting the information available in the *i** model to construct AgentSpeak(L) agents that will be used in the *MAS*.

7 Related Work

It is reported by [3, 6] that *i** models are not expressive enough to be directly used in late-phases of the software engineering life cycle. For example, there is no temporal sequence between the decomposed nodes, so one can assume any order for their execution. Moreover, given an *i** model, it is impossible to determine where a process begins. In resolving such ambiguities, researchers like [3, 6] have annotated the *i** model with new nodes-types and link-types. Although such detailed annotations provide a very fine-grained control-flow an the *i** level, they need a detailed understanding of the system that is being modelled. Moreover, they leave modellers with options to choose among the various control alternatives [3].

Since the *i** model is a tool to assist analysts in reasoning with abstract information to derive the high-level intentional characteristics of a system [8], we believe there must be a distinction between the early-phase modeling using the *i** model versus refining it to accommodate the late-phase of RE. Such a clear distinction allows for co-evolution of both models, which can result in further validations such as those explored in [4].

[2]OME is a tool with graphical interface for creating *i** models

8 Conclusion

In this paper we have shown the conceptual similarity that exists between the *i** model and AgentSpeak(L) . Furthermore, we envisioned how these two approaches can be used in a synergestic fasion lending towards an executable specification, *MAS*. We took the primary step of creating *MAS* and proposed a formal mapping methodology with its implemented prototype for transforming a SR diagram to an AgentSpeak(L) agent.

MAS bridges the gap between the early and late-phases of RE and allow analysts to check for boundary conditions such as creation conditions fulfilment conditions and invariant conditions.

References

[1] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[2] A. Fuxman, J. Mylopoulos, M. Pistore, and P. Traverso. Model checking early requirements specifications in tropos. In *RE '01: Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (RE '01)*, page 174, Washington, DC, USA, 2001. IEEE Computer Society.

[3] G. Gans, G. Lakemeyer, M. Jarke, and T. Vits. Snet: A modeling and simulation environment for agent networks based on i* and congolog. In *CAiSE*, pages 328–343, 2002.

[4] A. Krishna, S. Vilkomir, and A. Ghose. A case study of combining *i** framework and the z notation. Proceedings of ICEIS-2004: The 6th International Conference on Enterprise Information Systems, 2004.

[5] A. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

[6] X. Wang. Agent-oriented requirements engineering using ConG olog and *i**. Master's thesis, Department of Computer Science, University of Toronto, 2001.

[7] E. Yu. Towards modelling and reasoning support for early-phase requirements engineering. pages 226–235, Washington D.C., USA, 1997. Proceedings of 3rd IEEE International Symposium on Requirements Engineering.

[8] E. Yu and J. Mylopoulos. Understanding "why" in software process modelling, analysis, and design. In *ICSE '94: Proceedings of the 16th international conference on Software engineering*, pages 159–168, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.