Integrating Scenarios, i*, and AspectT in the Context of Multi-Agent Systems

Antonio de Padua Albuquerque Oliveira[1, 2, 3]
Luiz Marcio Cysneiros[3]
Julio Cesar Sampaio do Prado Leite[2]
Eduardo Magno Lages Figueiredo[2]
Carlos Jose P. Lucena[2]

[1] Universidade do Estado do Rio de Janeiro – UERJ
[2] Pontificia Universidade Catolica do Rio de Janeiro – PUC-Rio
[3] York University – Atkinson – School of Information Technology -Toronto

Abstract

Developing Multi-Agent Systems (MAS) calls for addressing different concerns. Some of them are general and related to the technology and others are particular to each collaborating agent. Our proposal aims to provide a more holistic approach to the construction of MAS. Integrating three different perspectives for information modeling we have achieved a more comprehensible way of dealing, early on, with the different concerns that are akin to MAS. Scenarios is a well known requirements technique; it produces a structured description of one situation that occurs in the real world; i* Framework models organizational contexts using the strategic relationships among actors and; AspectT is a software framework for MAS implementation. In this paper we report our initial findings in integrating these three perspectives. We used a well known example: the Expert Committee, to illustrate the advantage of dealing with these three different perspectives. Our contribution relies on tackling different concerns in an integrated manner during the requirements definition of a MAS development.

1 Introduction

Multi-Agent Systems (MAS) modeling calls for using concepts and metaphors that reflect the way we understand the world and therefore is a complex process. MAS development needs techniques to deal with a higher level of abstraction for reasoning about how software should behave and what characteristics should they present [20].

In order to produce a MAS, developers need to understand, think and act, using MAS concepts from the definition to the implementation of the system. The complexity starts with the concepts of intentionality that involves a large number of actors with opportunities and vulnerabilities [4]. Since each actor has properties such as goals, beliefs, abilities, commitments, which are intentional in nature [20], intentionality should play a major role in the MAS construction.

Adding to the complexity there is the need for addressing agency properties, such as autonomy, pro-activeness, sociability, adaptation, and interaction as well as additional properties such as collaboration, learning, and mobility [18]. Aside from that, MAS modeling has three additional types of interactions (Actor ↔ Agent, Agent ↔ System, and Agent ↔ Agent) compared with the traditional software modeling (Actor ↔ System) [10] [17].

To face this new level of complexity and explore the appropriated use of agency properties, we need an approach that addresses these concerns early on. In this work we present an integration schema, to tackle these problems.

We have chosen i* [19] as our central perspective, mainly because it deals with intentionality at the early stages of the MAS construction. Scenarios are used to enhance the elicitation of intentionality and AspectT [7] is used to stress the concerns more akin to the general characteristics of agents systems, such as: complexity, autonomy, connectivity and pro-activeness.

Scenarios are mainly used to help eliciting requirements while these requirements will be mainly modeled using the i* framework. Heuristics were created and used to allow the requirements engineer to derive i* models from scenarios. We also have created heuristics to link i* models and agency properties to AspectT models.

We illustrate the proposal by applying it to one example of multi-agent system: "The Expert Committee System" (EC) [5], which is a Web-based system. An Expert Committee is a group of members convened by one Director-General (coordinator) for the purpose of reviewing and making technical recommendations on a subject of interest to one organization or to one conference. In this paper we will focus on conference organization.

A member of an expert committee is an expert appointed by the Director-General to serve at a particular committee. Software agents should be introduced to the EC System in order to assist researchers with time-consuming activities in the paper submission and reviewing processes. EC agents are software assistants, who represent actors in different roles of the conference such as: authors, reviewers, committee members, chairs and coordinators.

The paper is organized as follows: Section 2 presents Scenarios, i* and AspectT. Section 3 deals with the integration process. Section 4 illustrates the use of the method. In section 5 we assess the results from applying the strategy to the EC System. Section 6 concludes and points out to future research.

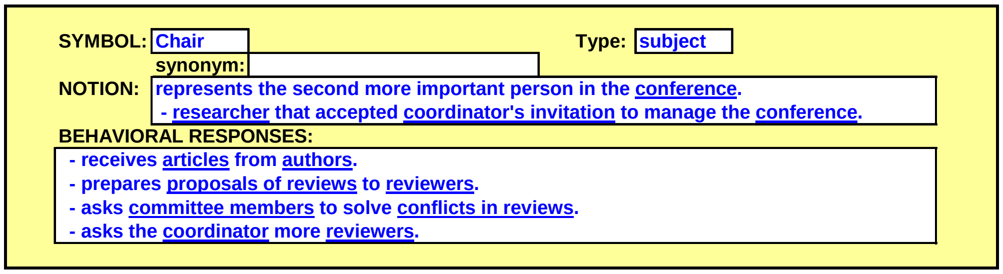
2 Scenarios, i*, and AspectT

In this section we briefly describe each of these techniques.

2.1 Scenarios and the Language Extend Lexicon (LEL)

A scenario is a structured description of one situation [13] that occurs in the real world. Leite's notation for scenarios [13] strongly recommends the construction of scenarios using words or sentences peculiar and most used in the Universe of Discourse (UofD). The UofD includes all the sources of information and all the people related to the software. This recommendation is central to our process and will be dealt at Section 4.

Prior to the construction of scenarios [13], the requirements engineer should elicit the Language Extend Lexicon (LEL). LEL is a representation model of symbols (words and sentences) of the application language. LEL [12] is centered on a very simple idea: "understanding the language of the problem without worrying about understanding the problem". The goal of the LEL is to represent all words or sentences, called symbols, peculiar to the application. Each symbol, or entry, in LEL is identified by a name or names (if there are synonyms) and is represented by two descriptions. The first one, called notion, is the denotation of the symbol, equivalent to a description found in a dictionary. The second one, called behavioral response, is the connotation of the symbol, which describes the contextualization of that symbol in the UofD. The symbols in the LEL are classified in four classes: object, subject, verb and state. Symbols are underlined in LEL; see an example in Figure 1.


SYMBOL: Chair **Type:** subject
synonym:
NOTION: represents the second more important person in the conference.
- researcher that accepted coordinator's invitation to manage the conference.

BEHAVIORAL RESPONSES:
- receives articles from authors.
- prepares proposals of reviews to reviewers.
- asks committee members to solve conflicts in reviews.
- asks the coordinator more reviewers.

Figure 1: A LEL example

Title: Prepare proposal
Goal: Designate **articles** to **reviewers**.
Context:
Geographical Location: WEB
Temporal Location: Right after the **submission deadline**.
Precondition: **Reviewers list** and **articles list** have been prepared.
Constraint:
Resources: Computer, Internet, **reviewers list**, **articles list**
Constraint:
Actors: **Chair** and **reviewers**
Episodes: Select **reviewers** within the same **area** of the **article**.
Separate out **reviewers** of the same **institution** of the **author**.
Make **proposals** to **reviewers**.
Send **proposals** to each **reviewer** giving the **acceptance deadline** .
Constraint: Each **reviewer** can not receive more than 3 **articles**.
Exceptions: If there is at least one **article** without 3 **reviewers**:
(scenario: "Hire more **reviewers**")

Figure 2: Leite's scenarios notation.

While describing symbols the requirements engineer must follow (1) the circularity principle (also called "closure principle") and (2) the minimal vocabulary principle. The circularity principle states that we have to maximize the use of LEL symbols when describing a symbol while the minimal vocabulary principle states that we have to minimize the use of words that are external to the Lexicon. These principles are vitally important in order for LEL to be self-contained and highly connected [12].

Scenarios describe situations which have characteristics: they are concrete and have goals; they involve actors and need resources; they happen in a defined time and place, they may have constrains which may either qualify the scenario or force some impositions. Scenarios are individually independent, interrelated and may have alternative course represented by exceptions. See an example of scenario definition in Figure 2. Figure 3 shows the relationship of components in the Scenario model. Each scenario is identified by one name (title) and could have exceptions that scenario must consider. A scenario satisfies one goal and one context specifies its boundaries. They need resources and involve one or more actors. Scenarios are compound by episodes that detail its behavior.

2.2 i*

The i* Framework [19] models organizational contexts using the strategic relationships among actors. Systems and their environments are described in terms of intentional relationships among strategic actors. Actors are taken to be intentional in therefore, they have goals and beliefs. Actors are strategic since they seek to exploit opportunities and to mitigate vulnerabilities. Actors may be abstract (roles defining responsibilities), concrete (agents – human and non-human individuals or classes with specific capabilities), or other organizational constructs. In models created with i*, each actor can be autonomous as is quoted by agent-oriented soft-

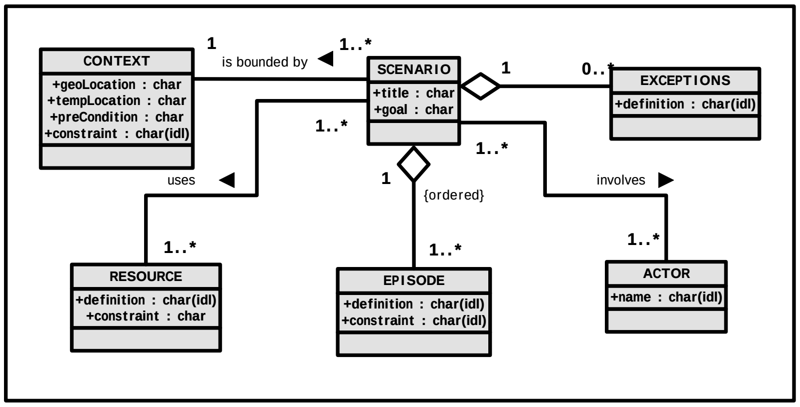


Figure 3: Leite's scenarios – class diagram.

ware technologies [21]. The dependency relationship with actors drives the requirements engineer to elicit during early stages of software development the goals to be modeled (Why do actors have these dependencies?).

External relationships among actors are expressed in the Strategic Dependency (SD) model. The SD model depicts the organizational context of the system as a network of dependency relationships among actors. This network consists of a set of nodes and links where each node represents an actor and each link represents one dependency between two actors. A dependency is a relationship in which one actor (the depender) depends on another actor (the dependee) to achieve a goal, to perform a task, to provide a resource or to achieve a softgoal; reflecting distinct types of freedom allowed by the relationship. Internal relationships among the intentional elements within an actor's reasoning are expressed in the Strategic Rationale (SR) model.

The SR model provides a more detailed level of modeling than the SD model, by looking "inside" actors to model internal intentional elements. The SR model is used to express the rationales of each actor about its intentional links. In the SR model, not only the external dependencies are represented, but internal elements (goals, tasks, resources, and softgoals), are also linked by means of task-decomposition or means-ends relationships, and softgoal contributions.

2.3 Aspect-oriented Approach with AspectT Framework

Multi-agent systems are characterized by agency properties such as autonomy, adaptation and interaction, which are potentially crosscutting [8]. In addition to this, aspect-oriented software development [11] is growing in popularity as the means of providing improved separation of concerns in design and implementation. Since most agency properties can be viewed as crosscutting concerns, the use of an aspect-oriented approach seems to be a good match. We selected AspectT software framework [7] because its main purpose is to support the modeling of agency properties in a modular way.

The AspectT software framework [7] is based on the assumption that an agent is an object with added features and aspect notion is used to enrich objects with agents' properties in a transparent way. In fact, objects can be considered the basic structure for building agents. However, a single agent is a much richer abstraction than an object, and then the AspectT emphasizes separation of agency concerns in order to reason about the agent behavior from different perspectives. In this approach, objects encapsulate the basic functionality of the system (such as resources and beliefs), while aspect-oriented mechanisms are used to modularize agency concerns (such as, interaction, adaptation, autonomy, learning, mobility, and

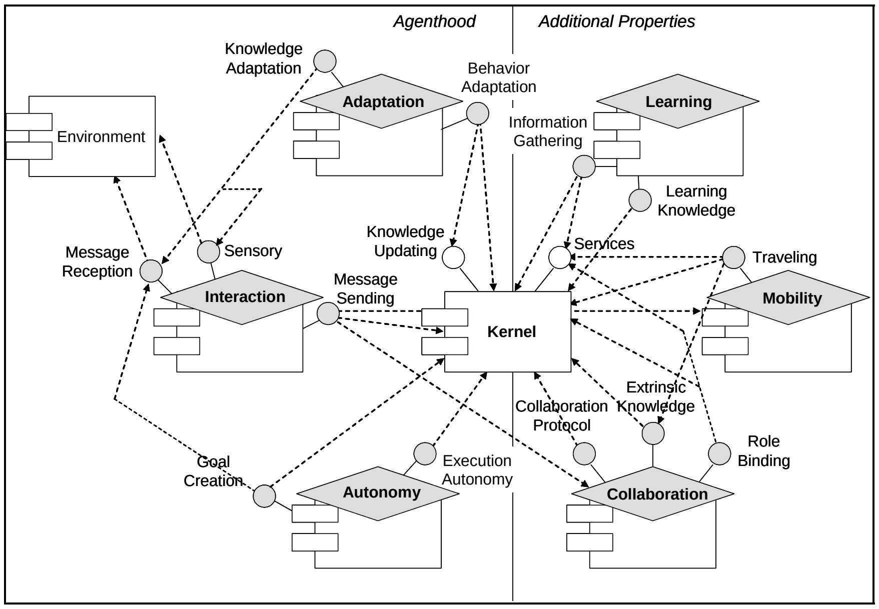


Figure 4: AspectT framework: Kernel and agenthood aspects.

collaboration). Each of the agents' properties should be modeled as an aspect due to their crosscutting nature [8], as showed in Figure 4, and these aspects are responsible for providing the appropriate behavior for an agent's basic property. The AspectT framework is based on the use of AspectJ language [1], which is an extension to Java™ that supports general-purpose aspect-oriented programming.

AspectT software framework is based on a method which has 5 steps for MAS development as follow:

1	Define Agents State and Behavior.
2	Capture and Associate Agency Properties.
3	Create Agent Types.
4	Associate Additional Agency Properties.
5	Define Collaboration and Roles.

Table 1 – The steps of AspectT method for MAS

3 The Proposed Integration Process

In this section we briefly show and describe the integration strategy. Figure 5 provides an SADT actigram [16] that stress the process we are proposing. The first activity prepares scenarios definition, which uses UofD symbols. The second activity uses the Scenarios, according to i* and MAS concepts, to produce intentionally driven models (Strategic Dependency and Strategic Rationale). The third activity implements the software using scenarios and the models, according to the scenarios, the MAS concepts and AspectT. These activities are supported by software tools showed as mechanisms in the SADT (bottom arrow on each box).

3.1 Define Scenarios

The first activity describes scenarios in details for an specific UofD. Scenarios construction must follow a certain technique and has to emphasize MAS concepts. The C&L tool software, which is a management tool for lexicons and scenarios, supports this activity. C&L is an open tool developed by the requirements engineering group at PUC-Rio and is available at [3].

Figure 5 also shows that to Model Intentionality, changes in scenarios are due to proper feedback. Heuristics were created to use the elicited scenarios as front-end to build i* models. These heuristics will be detailed in section 4.

3.2 Model Intentionality

The second activity creates i* models which reflect the "whys" and "what's" that are wanted. External relationships among actors are expressed in the Strategic Dependency (SD) model. Internal relationships among the intentional elements within an actor's reasoning are expressed in the

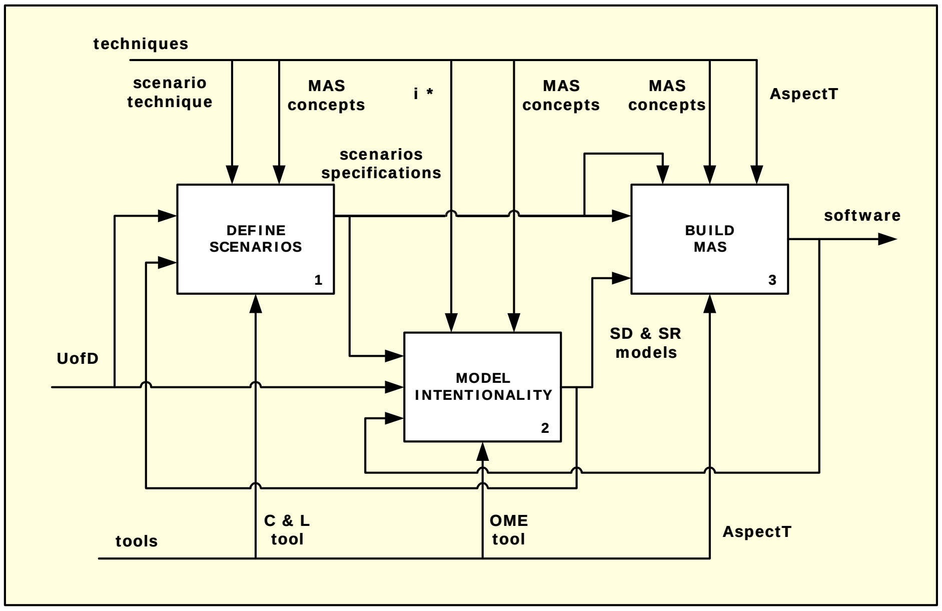


Figure 5: SADT of the development process.

Strategic Rationale (SR) models. The diagram in Figure 5 shows that the current activity must follow i* definitions and MAS concepts. A software tool called OME is used in this activity.

3.3 Build MAS

Finally, in the third activity, the Multi-Agent System is implemented using AspectT. Section 4 presents heuristics to correlate i* elements with AspectT elements as well as on how to use these correlations to get to the final software product.

4 Example: The Expert Committee System[1]

4.1 Define Scenarios

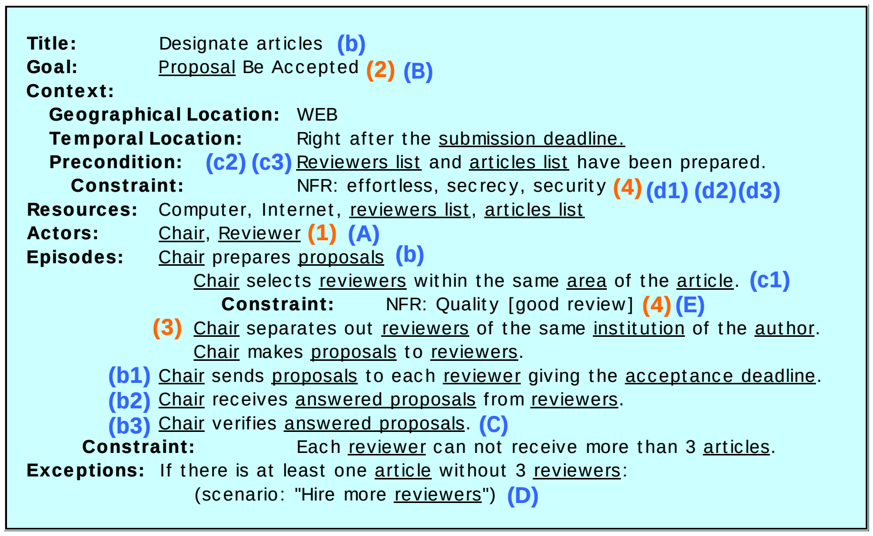
i) Elaborate the LEL.

- Identify symbols (words or sentences) that are peculiar to the social environment. Use one or more technique for fact gathering (e.g.: interviews, observation, document reading) [9]. Classify symbols in LEL as: object, subject, verb, and state. Follow both the circularity principle and the minimal vocabulary principle. We have to identify actors as symbols; they will be defined as subjects in the LEL. In our example we elicited as subjects some actors. The actors elicited were: Author, Chair, Reviewer, Coordinator, and Committee Member. Some symbols elicited were: abstract, acceptance due time, article, author information, author institution, conference, co-author, institution.

ii) Write the scenarios[2].

- Maximize the use of LEL symbols when describing each scenario. Figure 6 portrays the scenario Designate articles. This scenario depicts what is involved by designating articles to be reviewed by reviewers. Note that the underlined words or sentences are LEL symbols.
- Give emphasis to elements which have agency properties, like autonomy, proactiveness, sociability, adaptation, and interaction as well as collaboration, learning, and mobility. It is important to show what was considered when preparing the scenario: (1) Actors in episodes received special attention because they represent the interaction property; (2) Scenario's goal obeys i* description rules; (3) Episodes that do not have relevance regarding MAS properties were grouped into one higher level episode (Chair prepares proposals). (4) Quality attributes were represented with the same syntax as softgoals.

Initial scenarios are in nature mostly non-intentional since assessing the intentionality behind some scenario may prove to be a challenge. However, since we use Leite's notation for scenario [14] and this notation calls for defining a



Title: Designate articles (b)
Goal: Proposal Be Accepted (2) (B)
Context:
Geographical Location: WEB
Temporal Location: Right after the submission deadline.
Precondition: (c2) (c3) Reviewers list and articles list have been prepared.
Constraint: NFR: effortless, secrecy, security (4) (d1) (d2)(d3)
Resources: Computer, Internet, reviewers list, articles list
Actors: Chair, Reviewer (1) (A)
Episodes: Chair prepares proposals (b)
Chair selects reviewers within the same area of the article. (c1)
Constraint: NFR: Quality [good review] (4) (E)
(3) Chair separates out reviewers of the same institution of the author.
Chair makes proposals to reviewers.
(b1) Chair sends proposals to each reviewer giving the acceptance deadline.
(b2) Chair receives answered proposals from reviewers.
(b3) Chair verifies answered proposals. (C)
Constraint: Each reviewer can not receive more than 3 articles.
Exceptions: If there is at least one article without 3 reviewers:
(scenario: "Hire more reviewers") (D)

Figure 6: Scenario definition: Designate articles.

[1] A more detailed description of this example was the aim of a companion paper [15].

[2] For this particular example, 13 scenarios have been built.

generic goal, some intentionality will be already present. We understand that ideally we should be able to elicit and represent as much intentionality as possible in the scenarios but we left that for future work. We added to Leite's notation the compulsory representation of all the actors involved in each episode. Making the actors to be involved in an explicit manner, does facilitate the visualization of dependencies among different actors. Red numbers, in the Figure 6, mean MAS properties applied to scenario's definition and blue letters give the association to i* models.

4.2 Model Intentionality

Table 2 shows the correspondence among the elements from scenarios to i*. A scenario can have 1 or more actors and an actor in a scenario is the same actor in i* modeling. A scenario has 1 goal and a goal in a scenario may result in one or more goals in i* modeling. A scenario can have 1 or more resources and a resource in a scenario is the same resource in i* modeling. A scenario has 1 or more episodes; an episode may be a task or a goal in i*. Moreover an episode may map more than one element in i*. A scenario can have zero or more constraints (modeled as an attribute to 3 different entities (see Figure 3)) and constraint in a scenario may result one or more softgoals in i* modeling. It is important to observe that there is not a correspondence one to one and so the same i* element may appear in more than one scenario and vice versa.

SCENARIOS		I* MODELING	
Element	**correspondence**		**Element**
actor	1	1	actor
goal	1	n	goal
context	1	n/a	n/a
resource	n	n	resource
episode	n	n	task / goal
constraint	n	n	softgoal

Table 2 – From scenarios to i* (addressing elements)

HEURISTICS FOR DERIVING SD MODELS:

i) Pick up actors from scenarios specifications. (Who?) ? See (A) in SD model and in scenario definition.

- An actor represented in one scenario will be mapped to an actor in i* SD model. This is a straight correspondence. However, the requirements engineer may want later to refine the actor representation into a more detailed representation such as roles played, positions occupied or an agent that occupies this position.
- The association of two actors in one episode will be mapped as a dependency in the SD model. Episodes have functional bias thus the engineer should identify the "destination" or the reason for each episode. Who is that episode working for? If another actor is related with the episode then you may have a goal dependency, or a task dependency, or a resource dependency in the SD model. One can also put that episode as a sub-episode; see (3) in scenario description.

ii) Model dependencies between actors. (Why?)

- Use the SD model to express intentional relationships. Using the heuristics presented in Table 2 for addressing elements from scenarios to i*, pick up and represent in SD goals, tasks, resources and softgoals from scenarios (episodes) specifications.
- Although the goal in i* modeling means the intentional agreement between two actors, the scenario goal has a functional conception. Therefore, we can assign intentional goals to scenarios. For example, the scenario Designate articles yields the goal "Proposal Be Accepted" as an intentional agreement of REVIEWER and CHAIR, and in the i* SD model (Figure 8).
- Only one goal dependency can be mapped using the goal from the scenario. See (B) in the SD model and in scenario definition. "Proposal Be Accepted" is a GOAL DEPENDENCY.
- Each resource may be mapped either as a resource dependency or as a task dependency depending on the direction of the dependency and if the actor can perform the task, needed by the other actor.
- Normally a resource represented in the scenario will be mapped to a resource in i* SD model. The designation is direct but the nature of the dependency can be changed. See (C) in the SD model and (c1, c2, and c3) in the scenario definition. "Articles to Review" and "Proposal of Reviews" are RESOURCE DEPENDENCIES that is shown in the same way either in scenarios or in SD model but, as label (D) in SD model and in scenario

definition shows, "Review Articles" is a TASK DEPENDENCY. This decision is based on another scenario, <u>Review articles</u> (not showed) and is indirectly confirmed in this scenario.

- One or more episodes presented in Scenarios may be modeled by one task in the i* model. Note that we may have one scenario modeled by one task when the scenario is simple (i.e.: it involves only one actor). A task dependency represented in i* SD model means that one actor (the depender) needs the other actor (the dependee) to perform the task.
- A constrain represented in the scenario will be mapped to a softgoal dependency in i* SD model. CHAIR hopes REVIEWERS make good review. See (E) in SD model and in scenario definition. "Quality [good review]" is a SOFTGOAL DEPENDENCY.

The SD model shows 3 dependencies: CHAIR depends on REVIEWER to achieve the goal "Proposal Be Accepted", to achieve the softgoal "Quality [good review]" and to perform the task "Review Articles" furthermore; the REVIEWER depends on the CHAIR to get the resource "Articles to review". The SD model illustrates the relationships among several actors. However, due to space limitation in this paper we only focus on the relationship between CHAIR and REVIEWER.

HEURISTICS FOR DERIVING SR MODELS:

MAS modeling process faces 3 additional types of interactions. There are three new interactions (Actor ↔ Agent, Agent ↔ System, and Agent ↔ Agent) comparing with the traditional software modeling (Actor ↔ System) [10] [17]. Jennings [10] calls "environment" the set of all agents and the part of resources that are shared among agents and actors. To manage and control the complexity, and explore the appropriated use of agency properties, we recommend the elaboration of 4 (four) SR models instead of model all agent interactions in one unique diagram. The approach is shown in the sub-sections (iii), (iv), and (v).

MODEL ALL INTERACTIONS ACTOR – ACTOR: This model is a base to create the other three models.

iii) For each dependency between two actors, create one SR model to express how the dependency will be fulfilled by each actor. Show goals, tasks, resources and softgoals like intentional elements inside of the boundary of the actor (dot-dashed circle). Pick up goals, tasks, resources and softgoals from scenario specifications. Connect the elements using intentional links (means-ends, decomposition, and contribution).

- The same dependencies between the actors that appeared in SD model will appear here again connecting the actors; these elements appear outside of the boundaries of the actors, but now, in SR model, the connections will appear connecting the elements inside the actor's boundary. See (B, C, c1, D, E) both in SR model (Figure 9) and in scenario definition. Identify the internal main goal for each actor. One actor can have more than one goal. Often one actor has one or more intermediary goals. The scenarios provide the goals but the engineer should represent the hierarchy inside the SR model in the correct way. Try either to consider or adapt scenario's goal to be the main goal for actors. See (a) "Proposals Be Approved" and "Proposal Be Accepted" in SR model. Find intermediary goals from episodes.
- High level episodes will map tasks and they will be represented as "means-end" connections to the goal. Alternative episodes will be mapped as alternatives in the SR model, if there is more than one way to achieve a goal (means-end link to a goal). See (b) in SR model and in scenario definition.
- Low level episodes will be sub-tasks connected using a decomposition link. See (b1, b2, b3) in SR model and in scenario definition.
- Resources used in an episode will be mapped either as a resource needed by the task (should be connected using a decomposition link) or as a resource dependency when this is an exchange between another actor and an agent. See (c1, c2, c3) in SR model and in scenario definition.
- Quality attributes (NFRs), such as performance, will be softgoals and they should be connected to tasks using a contribution link. However, some softgoals will have a decomposition link to a task. This happens whenever the softgoal is essential to perform the task adequately. See (d1, d2, d3) in SR model and in scenario definition.

MODEL ALL INTERACTIONS ACTOR – AGENT: These models give the interaction of delegations.

iv) Prepare one SR model for each relationship between the actor and the agent which will represent the actor in the environment. We call this type of SR diagram (Figure 10) a "Delegation Model".

- Model the intentional relationships between the actor and his agent. It must be decided which duties the actor wants to delegate to the agent.
- Adapt the same rules showed in (iii) to this situation. For each goal (that the agent will support the actor), use the similar goal of the actor including the word "support" in the name of the agent goal.
- Create tasks for the interactions between them.

MODEL ALL INTERACTIONS AGENT – AGENT: Represent agents' interaction inside the environment.

v) Model the intentional relationships among agents. Use the SR model (Figure 10) to express intentional relationships. Prepare one model for each relationship among agents that will be representing actors in the environment.

- Model the intentional relationships between two agents.
- Adapt the same rules showed in (iii) to this situation.

Applying the heuristics we have: a) actors that appears in scenario Designate articles are Reviewer and Chair. b) From scenario Designate articles we find out the goal: "Proposal Be Approved" (Proposal Be Accepted), resources: Reviewer list, Articles list ("Articles to Review"), Proposals ("Proposal of Reviews"), and acceptance deadline (included in "Proposal of Reviews").

Figure 9 portrays the basic SR model involving the actors Reviewer and Chair. Chair's main task is to "Manage Articles and Reviews", which has one only decomposition: the goal "Articles Be Reviewed". This association, by decomposition, means that the goal is part of the task and only if CHAIR achieves the goal the task should be concluded. Alternatively, the model shows a softgoal "Quality [Good Review]" meaning that CHAIR depends on REVIEWER, it is also represented that CHAIR has a task "Prepare Reviews Standard", to give the quality specifications and directions to reviewers for the task of review articles. The task "Manage Review" shows that CHAIR needs softgoals: "Secrecy, Security and Effortless". These softgoals were elicited from the scenario Designate articles. In the model it is shown that the softgoal "Security" contributes (some -) "negatively" to the softgoal "Effort" because the user will have to enter with a password for this process. Showing the softgoals in the SR model the software engineer has the information about these concerns must be operationalized.

In order to create each SR model we may use one or more scenarios descriptions. For example, to model the actor CHAIR, we used all scenarios in which CHAIR is collaborating and used heuristic (iii). Picking up scenarios in which CHAIR appears we got the goals: From scenario Designate articles we obtained the goal "Proposals Be Approved"; from scenario Review articles we obtained the goal "Reviewed Articles Be Received"; and from scenario Solve conflicts in reviews we obtained the goal "Conflicts Be Solved. In order to simplify the SR model Reviewer and Chair, we put only the goals that had some dependency between REVIEWER and CHAIR. From these scenarios we also obtained the tasks and the resources. From scenario Designate articles we obtained the tasks: "Prepare Proposals", "Send Proposals", "Receive Answered Proposals" and "Verify Answered Proposals" and the resources: "Articles", "Authors of Articles" and "Event Reviewers". We also obtained the resources: "Articles to Review" and "Proposals of Reviews" which appeared as resource dependency in the SD model. Using the previous SD model we linked the dependencies to the REVIEWER decomposition. Because lack of space, we do not present a SR Model showing Reviewer and Reviewer Agent.

4.3 Build MAS

In Table 3 we show the correspondence between i* elements and AspectT framework elements. Although in i* models we do not use roles for actors and agents, they must be implemented as agents playing roles in AspectT.

In the real world each actor can play more than one role as well as in EC system the same intentional actor can play more than one role. E.g., a committee member can be a reviewer too. Agents playing roles are handled by their implementation, and so we created two different intentional actors but in the implementation we created just one agent.

I* modeling	AspectT framework
actor	agent role
agent	agent role
dependency	Message / event
goal	ReactiveGoal
softgoal	Aspect
task	ReactivePlan
resource	Belief

Table 3 – Correspondence table

Dependencies among actors in i* were implemented as messages and events in AspectT. Each dependency found in the SR model resulted in one message in the implementation and when it was necessary to implement a dependency involving "one to more" we used the event element of AspectT.

i* elements: goals, tasks, and resources were implemented directly as goals, plans and beliefs. NFRs, which appear in i* as softgoals, were implemented as aspects in AspectT.

The following heuristics were used to derive elements to the AspectT Framework from i* models, they are also depicted in Figure 7.

a) Actors and Agents in SR models will be mapped as Agent and Roles class in AspectT.
b) Because agents play roles, AspectT Framework permits specializations of roles (e.g.: Author, Reviewer, Chair, and so on). Agents can play one or more roles in this platform. Agents must be linked with the correspondent role.
c) Both dependency between two agents and between an agent and an actor will be mapped using either class Message or class Event (depends on the direction of the dependency). If the element (goal, softgoal, task, or resource) is sent it will be mapped as a class Message. If the element is received it will be mapped as a class Event.
d) A Goal will be mapped to a ReactiveGoal.
e) A Task will be mapped to a ReactivePlan and tasks can be specialized if there are alternatives.
f) A Resource will be mapped as a belief.

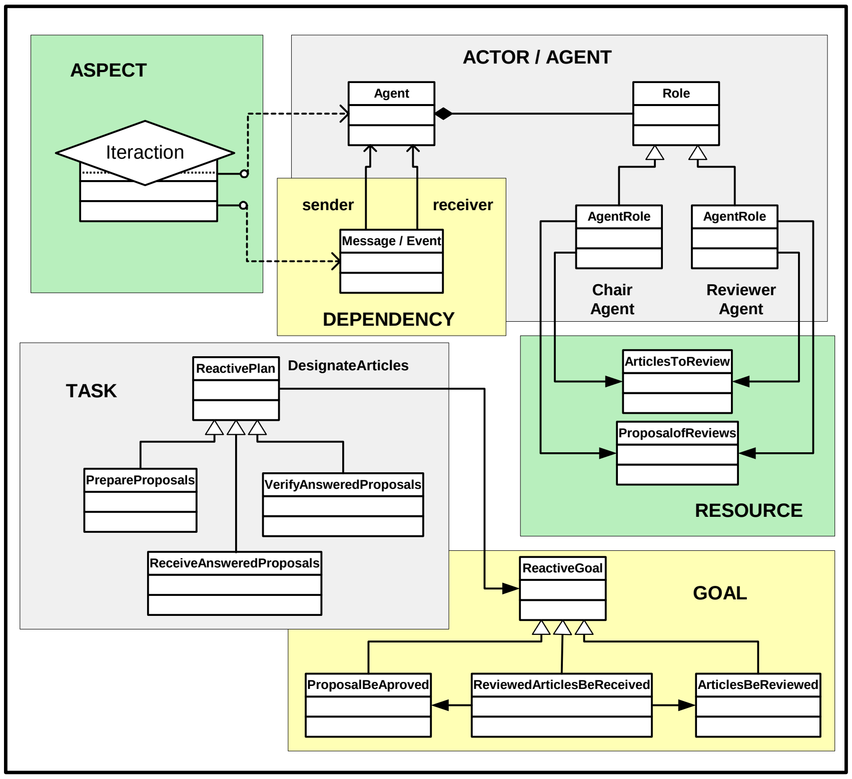


Figure 7: Mapping from i* to AspectT - ChairAgent and ReviewerAgent.

g) After having mapped goals, tasks, and resources, attributes of these elements should be created.
h) A ReactivePlan must be linked with the correspondent ReactiveGoal and each ReactivePlan should be created and linked with the correspondent class of agency property: interaction, adaptation, autonomy, collaboration, learning, and mobility.
i) For each softgoal (NFR) one new aspect must be created the same way as well as agency properties has been created previously into the framework.

Figure 7 shows an example of the mapping from i* to AspectT framework. It shows the mapping of ReviewerAgent elements using the SR model: ReviewerAgent & ChairAgent. During the mapping we do not need SD model.

In order to instantiate AspectT we should use all SR models. For example: to prepare the agent ChairAgent, we used all SR models in which ChairAgent is interacting and used heuristics from a) to i). Picking up SR models in which ChairAgent appears (they were four, because, in EC, Chair is the most important intentional actor, he has four interactions with: Author, Reviewer, Committee Member, and Coordinator). ChairAgent interacts with all other agents. We got the goals: ArticlesBeReceived and CameraReady BePublished from SR model AuthorAgent & ChairAgent; ProposalsBeApproved and ReviewedArticlesBeReceived from SR model ReviewerAgent & ChairAgent; and ConflictsBeSolved from SR model CommitteeAgent & ChairAgent. We created one instance for each goal as ReactiveGoal and also we created the pointers assigning a hierarchy among them. In this case, the goal ReviewedArticlesBeReceived will only be achieved if both ProposalsBeApproved and ArticlesBeReviewed were achieved. For each means-ends connection in SR model we created an instance of the task in the class ReactivePlan and we created the pointer to the correspondent ReactiveGoal (e.g. DesignateArticles had a pointer to ProposalsBeApproved). We created the decomposition of the task as sub-tasks (DesignateArticles has sub-tasks: PrepareProposals, ReceiveAnsweredProposals, and VerifyAnsweredProposals). We created the beliefs of ChairAgent: Articles, AuthorsofArticles, EventReviewers, and ReviewedArticles. At this moment we created attributes (and in case of ReactivePlans and ReactiveGoals the methods were created too) for each class already created. In order to finish dependencies, we created the pointer to aspects, which are outside of the Kernel, in the correspondent property (SendArticlestoReview was linked as "interaction aspect" and we created the message that will transmit the resource ArticlestoReview that the agent had one belief).

The implementation of the EC prototype as MAS can be seen at [6].

5 Observations from applying the approach to the EC case study

As a proof of concept we have applied the approach shown in this paper to the EC example as said before. It was possible to note that using scenarios prior to start modeling the problem with the i* framework helped us to get a better understanding of the elements to be modeled in i*. The heuristics to derive i* models from elicited scenarios were particularly useful, allowing initial versions of the i* to be more consistent. Two of us, Antonio de Padua and Eduardo Magno, spent around 35 hours describing scenarios and 55 hours developing i* models. We also noted that programming became straightforward because the framework AspectT was already implemented, and the heuristics to use the i* models as a front-end to AspectT helped its instantiation. In this sense the AspectT framework showed to be an efficient tool to obtain software using a multi-agent platform. The implementation was performed by one of us, Eduardo Magno, who had prior experience with the framework. He spent 40 hours. Testing and fixing bugs took around 18 hours.

Although we recognize that further experiments should be implemented, e.g. replicating the development of the same example by a different team as well as applying this proposal to larger and more complex problems, we understand that the results we had applying the approach to the EC are tackling an important concern in MAS construction, which is the bridge from definition to implementation.

On the other hand, the use of AspectT framework brings some problems. First we can not implement goal dependency in a straightforward manner. Second, AspectT is a framework that has a steep learning curve.

6 Conclusions and Future Works

In this paper we have shown an approach to integrate the scenarios technique, the i* framework and the AspectT framework. Heuristics are presented to guide developing i* models from elicited scenarios. We also have shown heuristics to obtain AspectT classes from the i* models.

We applied the approach to the EC system as a proof of concept. Our results are encouraging, since going from MAS definition to implementation is a hard problem. Our main achievement is to present a smooth transition, with well defined traces, from situations where agents may intervene with implementation of such agents. Naturally, more experiments are necessary not only to validate our initial results but also to improve our set of heuristics. While carrying out these experiments we will also be evaluating how well the approach scales to more complex problems.

One of the issues to be dealt in more detail related to the traceability links that are derived from our heuristics, in particular regarding MAS evolution.

About the Authors

Antonio de Padua A. Oliveira is currently a PhD candidate in Computer Science at Pontifical Catholic University of Rio de Janeiro (PUC-Rio). He received the BSc degree (1976) in electrical engineer from the Federal University of Rio de Janeiro (UFRJ) and MSc Degree in Computer Science at PUC – Rio in 1993. He is currently a visiting researcher at York University in Toronto. He worked more than 20 years as System Analyst at Brazilian Oil Company - Petrobras. He has been an assistant professor at UERJ (Rio de Janeiro State University) since 1993. His research interests include Requirements Engineering, Multi-Agent Systems, Information Systems and Software Engineering.

Luiz Marcio Cysneiros received the PhD degree from the Pontificia Universidade Catolica do Rio de Janeiro, Brazil, in 2001 and has been involved with requirements engineering since 1996. He is an assistant professor at York University, Toronto. He has held a post doc position at the University of Toronto from April 2001 to July 2002, where he continued his studies on non-functional requirements. He has published papers at several requirements related conferences and journals, including the Requirements Engineering Conference and IEEE Transaction on software Engineering. He also has been presenting tutorial on nonfunctional requirements in many international conferences such as ICSE '02, RE '03, and UML '03. He has an extensive industrial experience, mainly in a computer manufacturer and on health care domain. He is a member of the IEEE Computer Society.

Julio Cesar Sampaio do Prado Leite received the PhD degree at the University of California, Irvine, in 1998. He is an associate professor at Pontificia Universidade Catolica do Rio de Janeiro. He has been program chair and conference chair for several conferences. He is a member of the IFIP W.G. 2.9 on software requirements engineering, member of the IEEE committee on software reuse, and is an associate editor for the Requirements Engineering Journal. He is a founding member of the Brazilian Computer Society, member of the IEEE Computer Society, and of the ACM.

Eduardo M. L. Figueiredo received the BSc degree (2004) in computer science from the Federal University of Ouro Preto (UFOP) and the MSc degree (2006) in computer science from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil. He is currently a PhD student in computer science and a research associate of the Software Engineering Laboratory at PUC-Rio. MSc Figueiredo has participated in projects in cooperation with several academic groups. His current research interests include aspect oriented programming, empirical software engineering, software metrics and tools.

Carlos J. P. de Lucena (M'92) received the BSc degree from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil, in 1965, the MMath degree in computer science from the University of Waterloo, Canada, in 1969, and the PhD degree in computer science from the University of California at Los Angeles in 1974. He has been a full professor in the Departamento de Informatica at PUC-Rio since 1982. His current research interests include software design and formal methods in software engineering. He is a member of the ACM and various other scientific organizations. He is also a member of the editorial board of the International Journal on Formal Aspects of Computing.

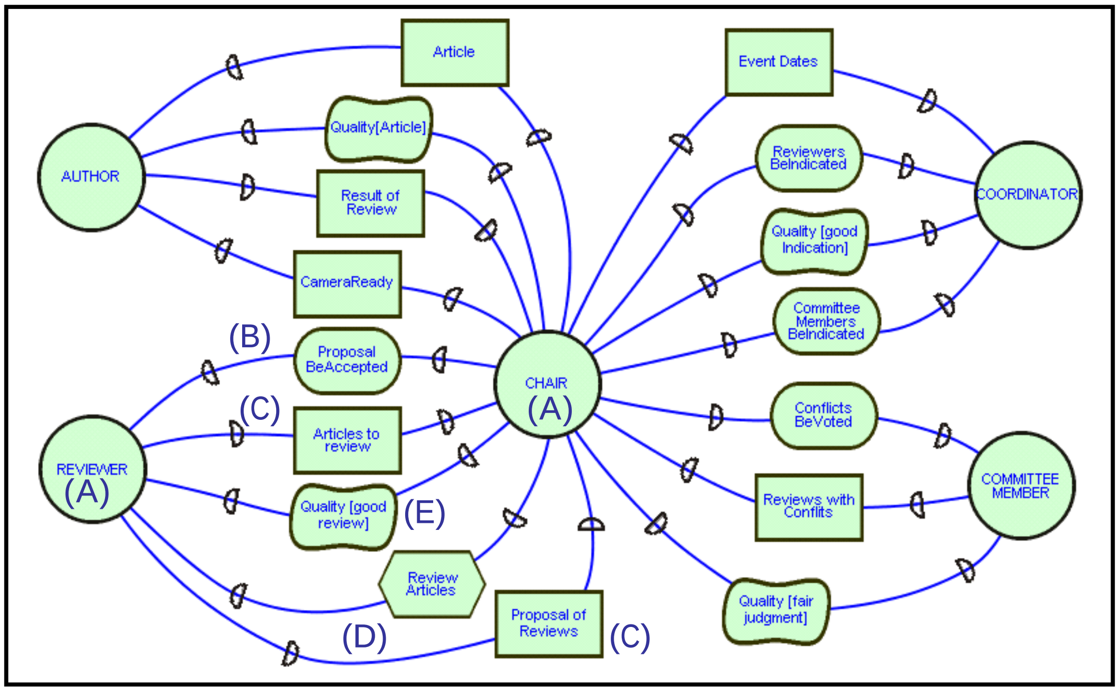


Figure 8: SD model of Expert Committee.

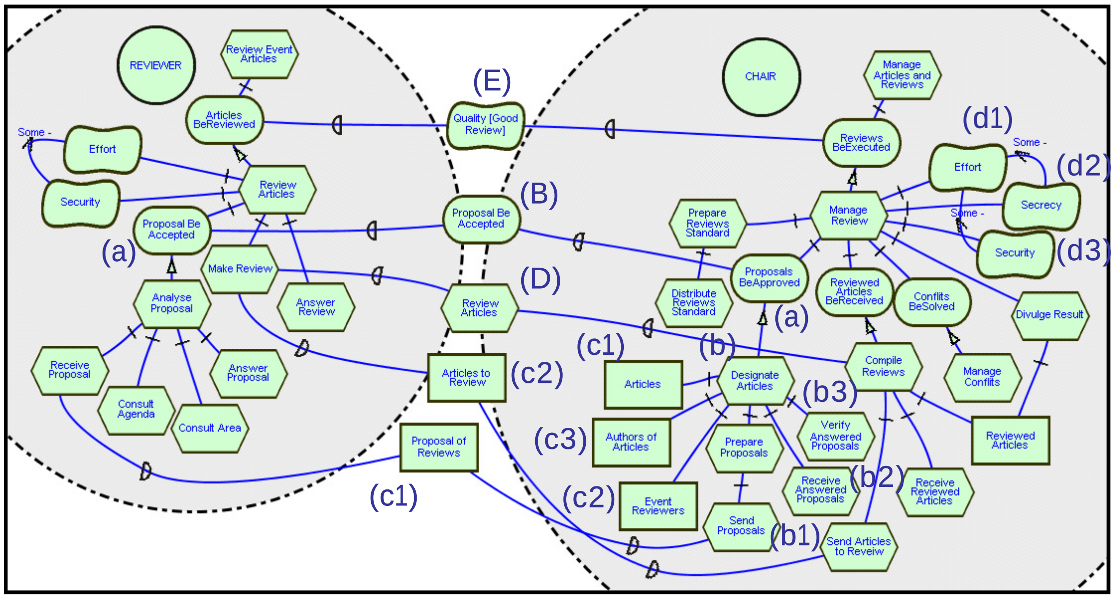


Figure 9: SR model - Reviewer and Chair.

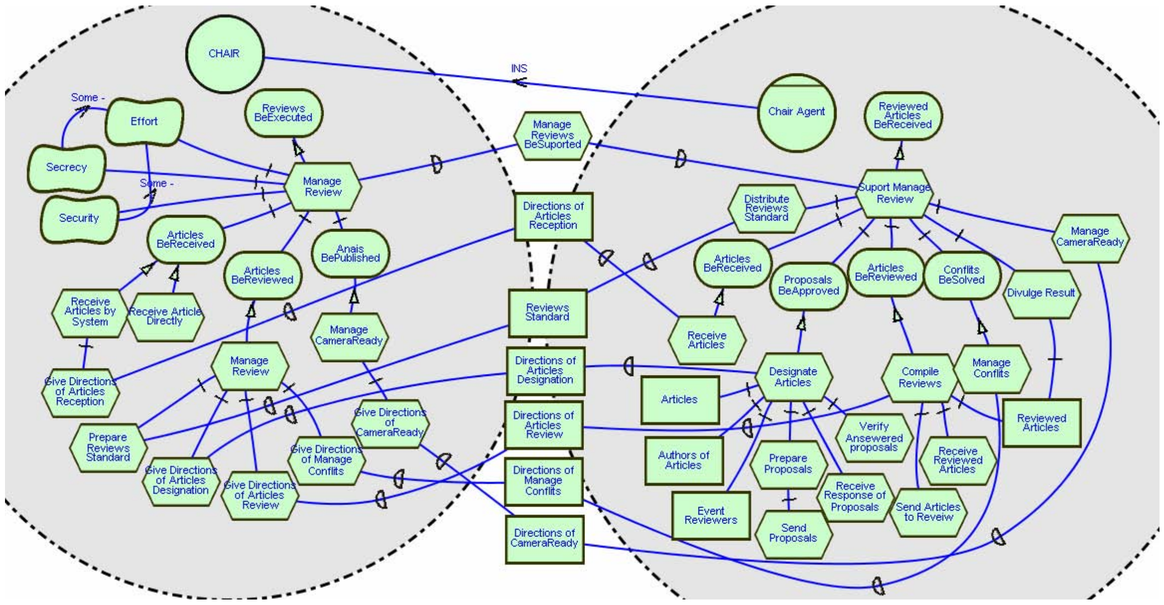


Figure 10: SR model - Chair & ChairAgent.

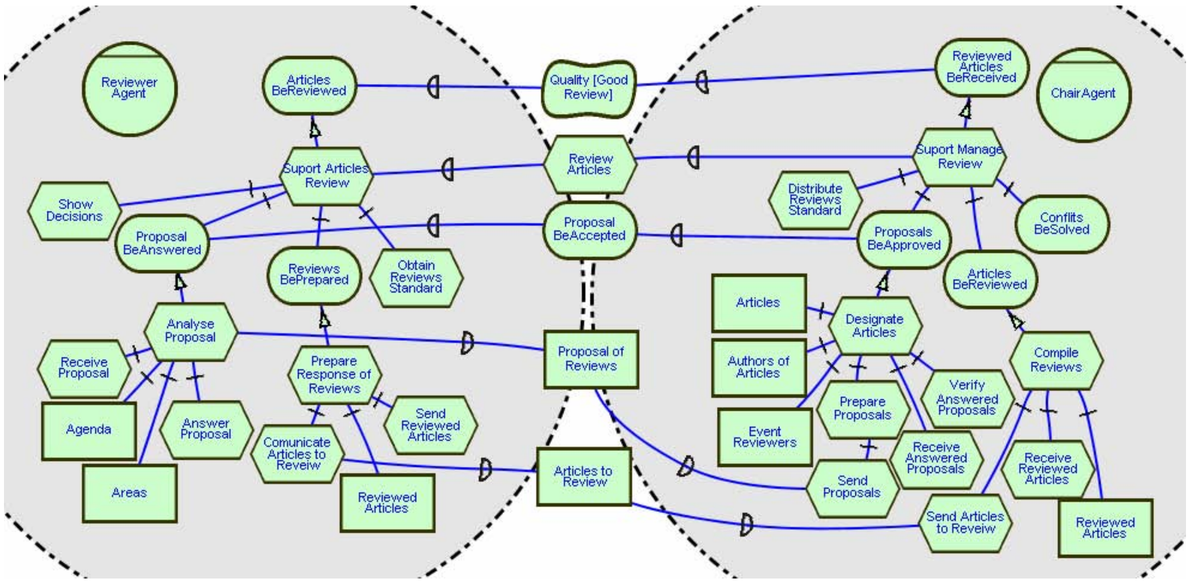


Figure 11: SR model - ReviewerAgent & ChairAgent.

References

[1] AspectJ – The AspectJ Programming Guide. [Online] http://eclipse.org/aspectj/, December 2004.

[2] Castro, J.; Kolp, M.; Mylopoulos, J. (2002) "Towards Requirements-Driven Information Systems Engineering: The Tropos Project." In: The 13th international conference on advanced information systems engineering, Oxford: Elsevier Science Ltd, v.27, n.6. p. 365-389.

[3] CeL Cenarios e Lexico (C&L): http://sl.les.inf.puc-rio.br/cel/ Jul /2006.

[4] Cysneiros, L.M. and Yu, E; Requirements Engineering for Large-Scale Multi-Agent Systems Book chapter in Software Engineering for Large-Scale Multi-Agent Systems – Research Issues and Practical Applications. A. Garcia, C. Lucena, F. Zambonelli, A. Omicini and J. Castro (eds.) LNCS 2603, Springer Verlag. 2003. (Revised and extended version of [SELMAS02])

[5] Deloach, S. et al. Multiagent Systems Engineering. International. In: Journal of Software Engineering and Knowledge Engineering, 11(3):231--258, 2001.

[6] EC prototype: www.teccomm.les.inf.puc-rio.br/emagno/ec/ Jul 2006.

[7] Garcia, A. F.; From Objects to Agents: An Aspect-Oriented Approach. PhD Tese, PUC-Rio, 2004.

[8] Garcia, A. F.; Lucena, C. J. P.; Cowan, D. D.; Agents in Object-Oriented Software Engineering; In: Software, Practice & Experience, Elsevier, vol. 34, Issue 5, pp. 489-521, May 2004.

[9] Goguen, J.A. and Linde, C. - Techiques for Requirements Elicitation, In Proceedings of the First IEEE International Symposium on Requirements Engineering, San Diego, Ca, IEEE Computer Society Press - 1994, pp 152-164.

[10] Jennings, N., "An Agent-Based Approach for Building Complex Software Systems" Communications of the ACM, April 2001, Vol. $$ No 4.

[11] Kiczales, G. et al. Aspect-Oriented Programming. In proceedings of the European Conference on Object-Oriented Programming (ECOOP'97). Finland, 1997, p. 220-242.

[12] Leite, Julio C. S. P.; Franco, Ana P. M. A Client Strategy for Conceptual Model Acquisition, Proceedings of the International Symposium on Requirements Engineering, IEEE Computer Society Press, San Diego (1993), pp. 243-246.

[13] Leite, J.C.S.P., Hadad, G., Doorn, J., Kaplan, G. A Scenario Construction Process - Requirements Engineering Journal: Vol. 5, N. 1, Pags. 38 -- 61, (2000), Springer-Verlag London Limited.

[14] Leite, J. C. S. P.; Doorn, J. H.; Hadad, G. D. S. and Kaplan, G. N.; "Scenario Inspections" - Requirements Engineering Journal 10.1007/s00766-003-0186-9 – Springer Verlag - London Limited – 2004.

[15] Oliveira, A. Padua A.; Cysneiros, L. M. "Defining Strategic Dependency Situations in Requirements Elicitation" The IX Workshop on Requirements Engineering; Rio de Janeiro, Brazil - July/2006.

[16] Ross, D. T.; "Structured Analysis (SA): A Language for Communicating Ideas", IEEE Transactions on Software Engineering Vol. SE-3 NO. 1; 1977.

[17] Silva, V.; Lucena, C. "From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language", In: Sycara, K., Wooldridge, M. (Edts.), Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, 2004.

[18] Wooldridge, Michael, An Introduction to Multi-Agent Systems, John Wiley and Sons Limited: Chichester, 2002, ISBN 047149691X

[19] Yu, E. Modeling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada, 1995, pp. 124.

[20] Yu, E. Agent Orientation as a Modeling Paradigm Wirtschaftsinformatik. 43(2) April 2001. pp. 123-132.

[21] Yu, E., Agent-Oriented Modeling: Software Versus the World In Agent-Oriented Software Engineering AOSE-2001 Workshop Proceedings, Montreal, Canada - May 29th 2001. LNCS 2222.