INTEGRATING EARLY AND LATE-PHASE REQUIREMENTS: A Factory Case Study

Fernanda Maria Ribeiro de Alencar[1] , Jaelson F. Brelaz Castro[2]

[1]UFPE - Universidade Federal de Pernambuco
CT - Departamento de Eletrônica e Sistemas
Rua Acadêmico Hélio Ramos, s/n - Cidade Universitária
Recife - PE - CEP: 50.740-530
E-mail: fmra@di.ufpe.br

[2]UFPE - Universidade Federal de Pernambuco
CCEN - Departamento de Informática
Av. Prof. Luiz Freire, s/n - Cidade Universitária
Recife - PE - CEP: 50.740-540
E-mail: jbc@di.ufpe.br

Abstract

Requirements Engineering (RE) is getting more attention as it has been recognized as a crucial phase in the development of the software system life cycle. Recent works have made a distinction between early-phase RE and later-phase RE. Early-phases RE activities are typically informal and addresses non-functional requirements. The later-phase RE usually focuses on completeness, consistency, and automated verification of requirements. In this paper, we show how early and late requirements specifications can be integrated. For the organization modeling we use the i* technique [1], which allows a better description of the organizational relationships among the various agents of a system as well as an understanding of the rationale of the decisions taken. For the formal functional specification of the requirements, we use Structured Modal Action Logic - MAL [2]. Some guidelines are presented for the integration of the two phases. Throughout the paper, we make use of a mineral water factory as an example, to describe the approach.

Keywords: Requirements Specification, Early and Late Requirements, Organizational Learning, Requirements Formalization.

1. Introduction

Requirements Engineering (RE) is the crucial phase in the life cycle of the development of a software system. It deals not only with technical knowledge but also with organizational, managerial, economic and social issues. The emerging consensus is that a requirement specification should include not only software specifications but also any kind of information describing the context in which the intended system will function. In this way, there is a need for modeling and analysis of stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives.

However, the production of high quality specifications is not easy. Usually the customers do

not exactly know what they want and sometimes the requirements may not reflect the real needs of the customers [3]. It is typical for requirements to be incomplete and/or inconsistent. It is also well known that the high costs of maintenance are related to poor definition and analysis of the requirements. Many studies show that the later a wrong requirement is detected the more expensive it is to correct it.

Among the goals of the Requirements Engineering we can highlight: (a) to propose communication techniques that facilitate the acquisition of information; (b) to develop techniques and tools that result in appropriate and precise specifications of requirements; (c) to consider alternatives in the specification of requirements and (d) to develop executable specifications to help to speed up the production of a prototype.

Recent works have made a distinction between early-phase Requirements Engineering and later-phase Requirements Engineering [1]. Early-phases RE activities are typically informal and addresses organizational or non-functional requirements. The emphasis is on understanding the motivation and rationale that underlie system requirements.

The later-phase RE usually focuses on completeness, consistency, and automated verification of requirements. It is concerned with the production of a requirement document such that the resulting system would be adequately specified and constrained in a contractual setting.

The transition from early (informal) to late (formal) requirements constitutes a conceptualization activity within which a developer might make use of domain knowledge partly expressed in descriptions of the organization, and partly in existing requirements specifications [3], [4]. Unfortunately, few work addresses the integration of early and late requirements [5] and as a result we end up with systems being developed that fail to support critical organizational goals and non-functional requirements. If we intend to produce high quality systems, it is vital to try to bridge this gap.

In this work and we show one approach for the integration of early and late requirements specifications. To model and understand issues of the application domain (the enterprise) we use the i* technique [1],[6], which allows a better description of the organizational relationships among the various agents of a system as well as an understanding of the rationale of the decisions taken. For the (formal) functional specification of the requirements, we use Structured Modal Action Logic - MAL [2].

Throughout the paper, we make use of a mineral water factory as an example, to describe the approach. We fix our attention in the section of filling up large bottles. For large bottles of mineral water, we consider bottles of 10 and 20 liters and for (simple) bottles, we consider bottles of 200 ml and 300 ml. Our intention is to present first the i* framework and then show how to migrate from an organizational model in i* to a functional model in structured MAL.

In Section 2 we describe some related work. Section 3 introduces the language used for the early requirements description, namely the i* technique. Section 4 introduces the language used for late requirements specification, the structured Modal Action Logic - MAL. In Section 5 we provide some means for integrating the description in i* into formal specifications in MAL. Section 6 concludes the paper with a discussion of its contributions.

2. Related Work

Various other organizational modeling techniques and formalization have been proposed in the literature. In organizational modeling area we can find for example KAOS [7] and Bubenko techniques [8]. In KAOS all goals are explicitly modeled and are simplified (reduced) through means-end reasoning until it reaches the agent level of responsibilities. Agents should behave as prescribed, which makes it difficult to analyze strategic relationships and implications. Bubenko [8] emphasizes the need to model organizations and their actors, motivations and reasons. The Organizational Model is composed of five interrelated sub-model, representing areas of knowledge of the organization: 1) Objectives Model (OM); 2) Activities & Usage Model (AUM); 3) Actors Model (AM); 4) Concepts Model (CM); and 5) Information Systems Requirements Model (ISRM).

The dependency concept has also been used. In [9] it is employed for the coordination of organization. In this line, in [10] business rules are statements about the enterprise's way of doing business, searching sentences which deal with limits, responsibilities and rights of the organization entities.

An other important issue is the non-functional aspect and its representation. In [11] we have non-functional aspects being treated during the initial phases of software development. They have proposed a representation that integrates non-functional requirements with a data modeling representation.

In the area of formal methods we can see in [12] the ALBERT framework. It supports the modeling of functional requirements in terms of a collection of agents interacting together in order to provide services necessary for the organization. Each agent is characterized by actions that change or maintain their own state of knowledge about the external world and/or the states of other agents. Such actions are performed by agents in order to discharge contractual obligations expressed in terms of internal and cooperation constraints.

The integration of the organizational model with the formal methods has also been studied by some authors [5], [13], [14]. The integration of ALBERT and i* is presented in [5]. It is worth noting that ALBERT and structured MAL have different modeling power and semantics.

Currently there is a standard for object-oriented development. The Unified Modeling Language - UML [15] and the Object Constraint Language – OCL [16] (a language for capturing semantics). Although the UML would support some concepts expressed by the i* model, others wouldn't be graphically represented, i.e., softgoals in the SD model and positive/negative contributions of these goals in the SR model. Further research will be done in the integration of i* with OCL.

3. The I* Technique

In this section we will review the main concepts of the i* technique [1], [5]. It is a framework, which focuses on the modeling of strategic actor relationships of a richer conceptual model of business processes in their organizational settings. Usually when we try to understand an organization, the information captured by standard models (DFD, ER, Statechart, etc.) is not enough because the majority of these models describe only entities, functions, data flows, states of system. They are not capable of expressing the reasons and "why's" of the process (motivations, intentions and rationales). The ontology of the i* technique [17] caters to some of these advanced concepts. It can be used for: (i) obtaining a better understanding of the Organizational relationships among the various system agents; (ii)

understanding the rationale of the decisions taken; and (iii) illustrating the various characteristics found in the early phases of requirements specification [5]. According to this technique, the participants of the organizational setting are actors with intentional properties, such as, goals, beliefs, abilities and compromises. These actors depend upon each other in order to fulfill their objectives and have their tasks performed. The i* technique consists of two models: The Strategic Dependency Model (SD) and the Strategic Rationale Model (SR). In the sequel we describe the characteristics of these models, further details can be find in [17] or [5].

3.1 The Strategic Dependency

The Strategic Dependency Model (SD) consists of a set of nodes and links connecting them, where nodes represent actors and each link indicates a dependency between two actors. Hence, a model is described in terms of network of dependency relationships among various actors, capturing the motivation and why of activities. We can distinguish, four types of dependencies, three of them related to existing intentions – *goal dependency*, *resource dependency* and *task dependency* – while the fourth is associated with the notion of non-functional requirements, the so called *soft-goal dependency*. In the *goal dependency,* an agent depends on another one to provide the desired condition, and it does not worry about how this condition is achieved. In the *resource dependency,* the agent depends on the availability of physical resource or information. In the *task dependency,* the agent informs the other what (and how) should be done. The *soft-goal dependency* is similar to the *goal dependency*, except that the condition is not precisely defined at the start of the process, i.e., the goals in a sense involves subjective aspects, that gradually are clarified during the development process. This type of dependency provides an important link connecting two important aspects in software engineering: (i) the technical and (ii) managerial side. We still can identify different degrees of dependencies: open, committed and critical [5]. We can distinguish actors as agents, *roles* and *positions*. An agent is an actor with concrete physical manifestations. It is a person or artificial agents (hardware/software). A role is an abstract characterization of the behavior of a social actor within some specialized context, domain or endeavor. A position is a set of roles typically played by one agent. And we can analyze opportunities and vulnerabilities of the chain dependency [17].

In the Figure 1, we have the Strategic Dependency (SD) model of the mineral water factory. There are four main actors *Company, Supplier, Client* and *Worker*. The *Worker* agent may occupy several positions: he/she may be in the 'Main Office', 'Filling up Glass', Filling up Bottles' and 'Filling up Large Bottles'. If we look closer (see Figure 1), we can observe that while in the position of 'Filling up Large Bottles' the *Worker* can be doing several different roles: 'Inspection', 'External Cleaning', 'Washing', 'Rising' and 'Filling up'.

In this case study, we consider strategic aspects of general operation of the factory (especially the process of filling up large bottles). Observing the Strategic Dependency (SD) model, in the figure 1, the concrete objective of the Client is to have his/hers bottles filled with mineral water. This is represented by a goal-dependency '*To have bottle filled up'* that indicates that he/she depends on the agent *Company* to provide the desired service. On the other hand, he/she hopes the water is of good quality, i.e., the water is appropriate to drink. This wish is expressed by the soft-goal dependency 'Safety [Water Cond]'. The *Company*, of course, wishes the satisfaction of its client regarding the quality of its services. That is shown by the

soft-*goal dependency* 'Satisfaction[Service]'. In order to maintain its costs the *Company* wishes the *Client* to pay for its services, as we can see in the *resource-dependency* 'Payment'.

If we consider the relationship between the *Client* and the *Worker* we also find some dependencies. The Client expects to be serviced quickly by the *Worker*. It is the *soft-goal dependency* 'Quickly[Attendance]'.

Fixing our attention on the position 'Filling up Large Bottles' we can observe that the *Client* also depends on the several roles performed by the Worker. We shown four resource-dependencies:

(a) 'Large Empty Bottles' indicates that the actor *'Do Inspection'* expects to receive from the *Client* large empty bottles;

(b) 'Non-suitable Large Empty Bottles' indicates that the *Client* expects to have returned the large bottles rejected by the first inspection of the actor *'Do Inspection'*;

(c) 'Rejected Large Empty Bottles' indicates that the *Client* also expects to have returned the large bottles rejected by the second inspection of the actor *'Do External Cleaning*';

(d) 'Filled Large Bottles' indicates that the *Client* expects to receive the large bottles (that are in good condition) filled up of water at the end of the process.

The soft-goal dependency 'Large Bottles [Good State]' indicates that the Worker, in his/hers role of 'Filling Up Large Bottles', expects that the large bottles provided by the *Client* to be in good state. The task-dependency 'Return Large Bottles' indicates that the *Client* hopes that the task of delivery of the large bottles to be accomplished.

The Worker depends on the *Company* to maintain safe his/hers working conditions. This is represented by the *soft goal-dependency* 'Safety [Work Con]'. He/she also depends on the *Company* for the wages, as expressed by the *resource-dependency* 'Wages'. The *Company* depends on the *Worker* to maintain the quality of the services (*soft goal-dependency* 'Quality [Service]`).

The Supplier and the *Company* also have strategic dependencies. The *Supplier* has the objective to sell the *Company* its products ('To sell products' is a goal-dependency). The *Company* expects that the Supplier has the products for delivery (resource-dependency 'Products'). The Company expects the Supplier to make the delivery of the products as faster as possible (soft goal-dependency 'Quickly [Delivery]').

3.2 The Strategic Rational Model

The second model of the technique **i*** is the Strategic Rationale Model (SR). It is used to: (i) describe the interests, concerns and motivations of participants process; (ii) enable the assessment of the possible alternatives in the definition of the process; and (iii) research in more detail the existing reasons behind the dependencies between the various actors. Nodes and links also compose this model. It includes the previous four types of nodes (present in the SD model): *goal*, *task*, *resource* and *soft-goal*. There are two new types of relationship, *means-end* that suggests that there be other means of achieving the objective (alternatives) and *task-decomposition* that describes what should be done in order to perform a certain task.

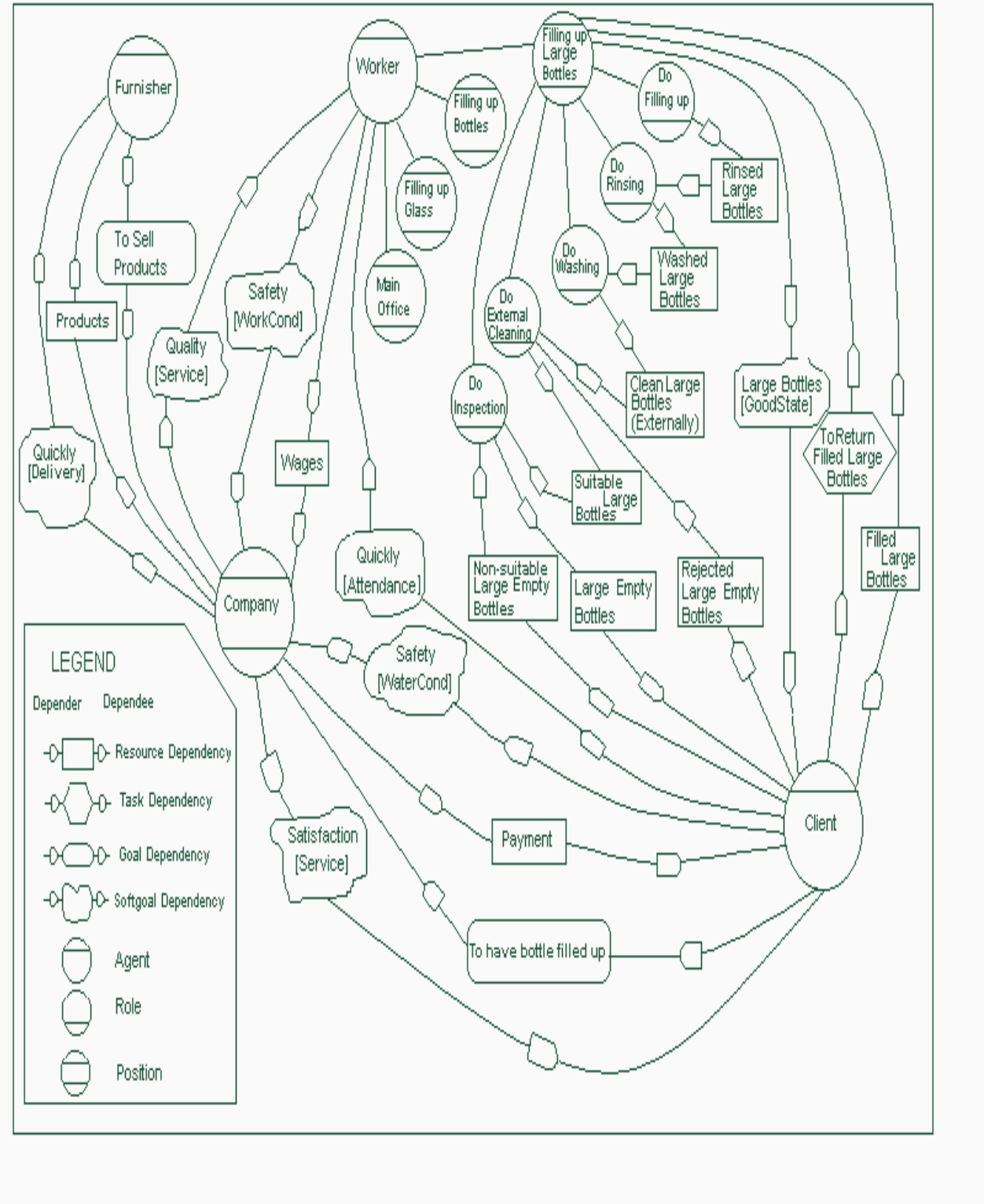

Figure 1 – Strategic Dependency Model of the Mineral Water Factory

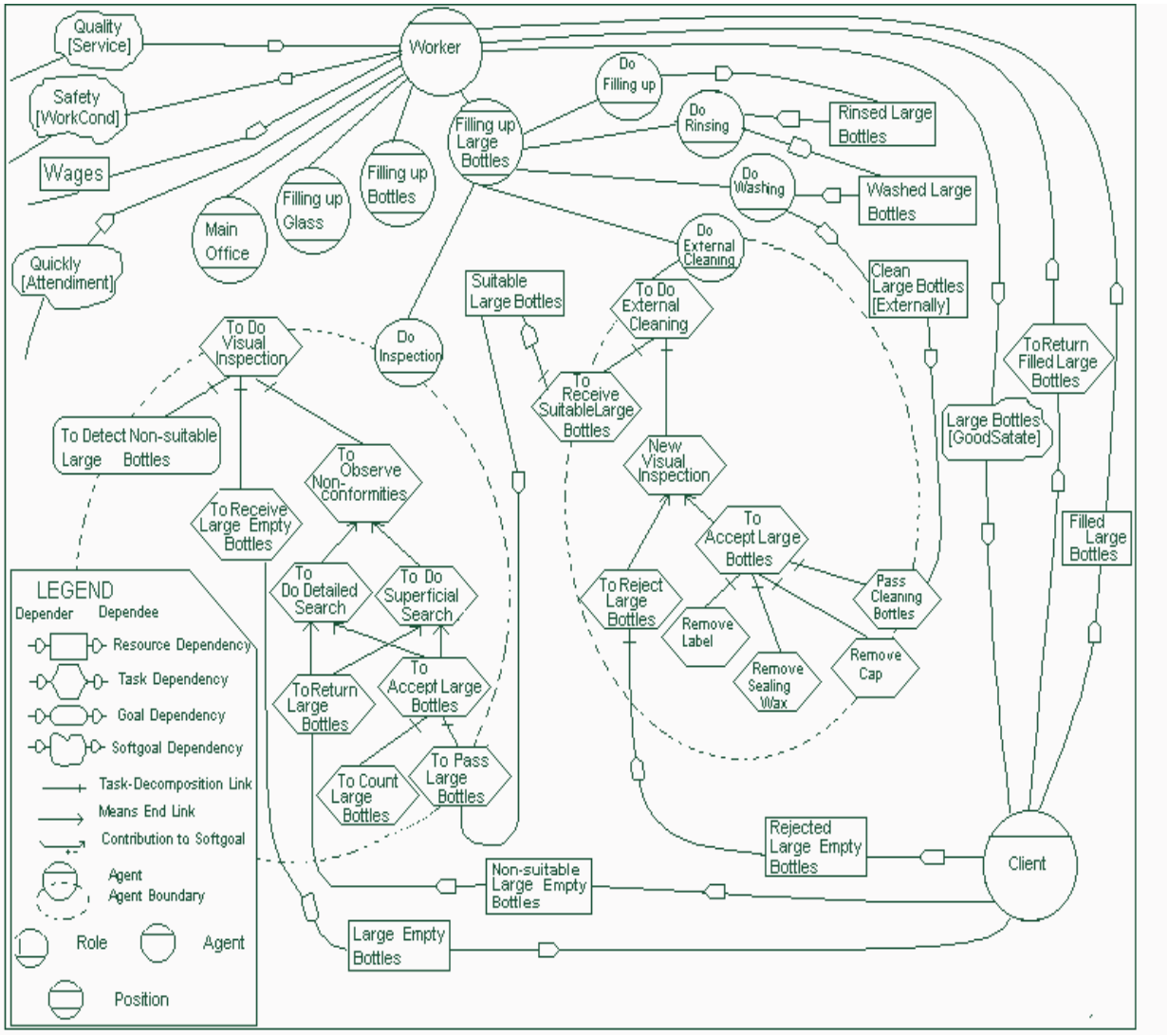

Figure 2 - Strategic Rationale Model of the Mineral Water Factory

In Figure 2 we use de SR notation to detail roles of the Worker agent in the position of 'Filling Up large Bottles'. Due to space limitation we now only comment on the role 'Do Inspection'. This actor is responsible for the task of visual inspection of the large bottles (task-dependency 'To do Visual Inspection'). It can be decomposed in three aspects:

1. The goal is to identify large bottles that do not conform with the standard (goal-dependency 'To detect non-suitable large bottles');
2. To receive the Client's empty bottles (task-dependency 'To receive Empty Large Bottles');
3. To search for non-conformities in the large bottles under inspection (task-dependency 'To observe non-conformities'). There are two alternative for performing this task. If the bottles seems to be all right a *superficial* inspection is performed (task-dependency ´To Do Superficial Search`) and the bottle is passed to another agent for a second test. If the bottle seems to be damaged a *detailed* inspection is executed (task-dependency ´To Do Detailed Search`). The two options are expanded bellow:

The superficial and detailed search consists of either returning or accepting the large bottle under inspection. Hence, one of the following steps is performed:

- The need to return large bottles that are not suitable (task-dependency 'To Return Large bottles') ,
- To accept the large bottle (task-dependency 'To Accept Large bottles'),

If the large bottle is accepted it is then counted and passed to different agent for a second examination and external cleaning. Hence, both tasks are perfomed:

- The need to count the large bottles (task-dependency 'To count large bottles') ,
- To forward the large bottles to another agent for a second test (task dependency ´To Pass Large Bottles`). This task produces the resource-dependency 'Suitable Large Bottles' indicating the counted suitable large bottles are passed to the next actor 'Do External Cleaning',

At this point, we may stop the process of modeling the strategic dependencies of the mineral water factory. We are already capable of understanding some issues of the application domain (the enterprise). We can then move to provide a functional description of system. A technique suitable for the precise specification of the requirements is the Structured Modal Action Logic [2].

4. Structured Modal Action Logic

In this section, we review the main concepts of Structure Modal Action Logic - MAL [2] and show how it can be used in the description of the behavior of objects (agents) of a mineral water factory.

The Modal Actions Logic (MAL) [18] was developed in order to (i) to produce an useful mathematical formalism for the specification of requirements, without the need that the specialist had a great mathematical training, and (ii) to produce a document of the system requirements specification.

Modal Action Logic - MAL is based on Typed First Order Logic. It includes types, variables, logical symbols, predicates, functional symbols, constant symbols, terms, atomic formulas and a number of axioms and inference rules. Structured MAL is an extension of MAL that has added: (a) pre-defined types called “agents” and “actions” that respectively define real world entities and describe processes that the agents can execute; (b) the modality [ ] to capture the effect of the occurrence of actions, i. e. [action a] can be considered as the post-condition or result of an action a that has been completed; (c) deontic operators per (permission) and obl (obligation) that allow the control over what action can be executed by the agents; (d) combinators ; and (e) interval temporal logic of branching linear type. Therefore, a structured MAL specification corresponds to a set of agent (object) descriptions, where the descriptions consist of a set of declarations and axioms that define the behavior of the agents that can interact sharing attributes and actions (label s - shared). Some attributes or actions have only local effects and are labeled l - local.

5. Integrating Organizational Description in i* with Formal Specifications in MAL

In this section, we deal with the question of formalization of requirements expressed through the i* technique. We present some guidelines to help to integrate the i* description (appropriate for the expression of organizational and non-functional requirements) with the structured MAL (used for specifying functional requirements of the system). These heuristics will help us to establish relationships between fragments of the formal specification in MAL and some organizational goal described in the i* models. Many concepts employed in the i* technique can be directly translated into the structured MAL.

In this paper, due to space limitation, we concentrate on the formalization of agent *Worker*, in the role of ´Do Inspection´, when in the position of ´Filling Up large Bottles` (see Figure 2).

Five guidelines are suggested to help in the process of formalization of i * models into structured MAL specifications:

Guideline G1: Related to agents;
Guideline G2: Related to resources;
Guideline G3: Related to tasks;
Guideline G4: Related to relationships;
Guideline G5: Related to goals and soft-goals.

Guideline G1:

Agents in the strategic models can be translated as **Agents** in structured MAL.

According to the model SD in the Figure 1, the mineral water factory is composed of four main **agents**. In structured MAL agents corresponds to **Objects.** In Figure 3**, the object** 'Mineral Water Factory', composed of four objects (Company, Supplier, Worker and Client) which are included in the specification.

```
Object Mineral Water Factory
    ......
include
  Company;
  Supplier;
  Client;
  Worker end
axioms
        ..................
        end

  end Object Mineral Water Factory
```

Figure 3 - The Mineral Water Factory Definition in MAL

Each one of the objects in Figure 3 can be further detailed. For example, the **object** Worker consists of four other **objects** (see figure 4). These objects will receive the same name of the

positions identified in the model SD of the Figure 1, i.e., 'MainOffice', 'FillingUpGlass', 'FillingUpBottles' and 'FillingUpLargeBottles'.

```
Object Worker
   .............
include
  MainOffice;
  FillingUpGlass;
  FillingUpBottles;
  FillingUpLargeBottles
  end
axioms
   ........
  end
  end Object Worker
```

Figure 4 - The Component Worker

As explained previously, in this paper attention will be given only to the specification of the 'FillingUpLargeBottles' object. According to the SD description (Figure1) it includes the following roles: ´Do Inspection`, ´ DoExternalCleaning`, ´DoWashing`, ´DoRinsing` and `DoFillingUp`. In structured MAL each role corresponds to an object to be included (see Figure 5).

```
Object FillingUpLargeBottles
      ..............
include
  DoInspection;
  DoExternalCleaning;
  DoWashing;
  DoRinsing;
  DoFillingUp
  end
axioms
     ..............
  end
  end Object FillingUpLargeBottles
```

Figure 5 - The Component FillingUpLargeBottles of the Component Worker

Observe that in structured MAL no distinction is made among the several types of actors, i. e., if it represents an agent, a position or a role. Any type of **actor** in i* will be mapped as a structured MAL **object**.

Guideline G2:

The ***resources*** will be mapped into ***attributes*** of structured MAL **objects;**

We noticed in our example (see Figure 2) that when the *Worker* is in the role of ´Doing Inspection`, he/she may have to deal with some physical resources. For example, from the *Client*, he/she receives ´LargeEmptyBottles`. Occasionally the worker may have to reject and return some of the ´Non-suitableLargeEmpty Bottles`, or if it seems to be all right, to forward `SuitableLargeBottles` to a different agent for a second inspection. All these resources will be mapped into shared attributes in the agent specification.

As far as structured MAL is concerned the resource 'LargeEmptyBottles' corresponds to attribute 'LargeEmptyBottles' in Object ´DoInspection`. In the Figure 6 we specify the 'DoInspection' component. We can see that the resources are mapped as attributes that may be shared (s) or local (l). Thus the resources of the i* models will be mapped into attributes of the structured MAL objects. Later they may appear as parameters of the object's actions.

Guideline G3:

The **tasks** will be mapped into ***actions*** in structured MAL.

We noticed in our example (see Figure 2) that when the *Worker* is in the role of ´Do Inspection`, he/she can engage in several tasks. The main one is to ´ToDoVisualInspection`. It is decomposed into two tasks: ´ToReceiveLargeEmptyBottles` and ´ToObserveNon-Conformities`. The latter task can be done in two ways: ´ToDoDetailedSearch' or ´ToDoSuperficialSearch` .The result of a search can be negative, in which case it is necessary 'ToReturnLargeBottles' or the result can be positive, in which case the task 'ToAcceptLargeBottles' is performed. When a bottle is accepted two tasks are executed, "ToCountLargeBottles' and 'ToPassLargeBottles' for a second inspection.

All tasks will be mapped into **actions** of **Object** Worker. Those that are internal to the object are labeled **l** (local), while those that involve interaction with other agents are labeled **s** (shared). In Figure 6 you will find the corresponding actions. Those actions that are shared, have as parameters, the shared resources (for example, action **s** ToReturnLargeBottles(Non-suitableLargeEmptyBottles)).

Guideline G4:

The relationships in the strategic models can be translated as **axioms** in structured MAL.

The axioms in MAL specification establish the behavior of objects. Action combinators can be used to describe the (de)composition of actions. The action modality ([]) capture the effect of the occurrence of actions, and deontic operators (permission – per and obligation – obl) are used to define when actions are permitted or obliged to happen. In Figure 6 we present fragments of the axiomatic specification of the ´DoInspection` object.

According to Figure 2, when doing the inspection, the *worker* has to receive empty bottles and then check possible problems with it. In Figure 6, axiom 1 tells us precisely that the occurrence of action ´ToReceiveLargeEmptyBottles` is followed (obligation) by the action ´ToObserveNon-Conformities`. Similarly, Axiom 2 specifies that the occurrence of action ´ToObserveNon-Conformities` is followed (obligation) either by action ´ToDoDetailedSearch` or ´ToDoSuperficialSearch`. Axiom 4 states that obligation to return the bottles to the user if they have failed the inspection.

Object DoInspection

attributes

s LargeEmptyBottles : int;

/This variable is shared and represents the large bottles that were given by the client/

s Non-suitableLargeEmptyBottles : int;

/This variable is shared and represents the large bottles that didn't pass the inspection and therefore are returned to the client/

s SuitableLargeBottles: int;

/This variable is shared and represents the large bottles that passed the first inspection. They are counted and forwarded to the object DoExternalCleaning/

l GoodCondition: bool;

l ToDetectNonSuitableLargeBottles: bool end

/These variable are local and represent the result of the inspection on to the large bottles/

actions

s ToReturnLargeBottles((Non-suitableLargeEmptyBottles);

s ToPassLargeBottles (SuitableLargeBottles);

s ToReceiveLargeEmptyBottles(LargeEmptyBottles);

l ToDoVisualInspection;

l ToObseveNon-Conformities;

l ToDoDetailedSearch;

l ToDoSuperficialSearch;

l ToAcceptLargeBottle;

l ToCountLargeBottles;

/ The actions preceded by **s** may be shared with other objects and actions preceded by **l** are local/

end

axioms

1. **[**ToReceiveLargeEmptyBottles(LargeEmptyBottles)**] obl** (ToObserveNon-Conformities);

/After receiving the large empty bottles the actor will do the visual inspection observing the occurrence of any problem with the bottles/

2. **[**ToObseveNon-Conformitie**] obl (**ToDoDetailedSearch) V **obl** (ToDoSuperficialSearch);

/After observing the non-conformities the actor is obligated to do two kinds of search (detailed or superficial) /

3. **[**ToDoDetailedSearch**] obl** (ToReturnLargeBottles) V **obl** (ToAcceptLargeBottles);

/The result of the detailed search is either to return the large empty bottles to the user or to accept it./

4. **obl** (ToReturnLargeBottles(Non- suitableLargeEmptyBottles))-> GoodCondition = False;

/Here we have the deontic operator (obl) indicating the obligation to return the large bottles to the client if they are not in good condition/

5. **[**ToDoSuperficialSearch**] obl** (ToReturnLargeBottles(Non- suitableLargeEmptyBottles)) V **obl** (ToAcceptLargeBottles);

/The result of the superficial search either to return the large empty bottles to the user or to accept it./

6. **per** ToCountLargeBottles ∧ ToPassLargeBottles(SuitableLargeBottles) -> GoodCondition = True;

/To the counting and passing of large bottles for further examination is permitted only if the large bottle is apparently in good state./

7. **[**ToAcceptLargeBottles**] obl** (ToCountLargeBottles) ∧
obl (ToPassLargeBottles(SuitlableLargeBottles));

/After acceptance, a bottle count is done and it is forwarded for a second inspection./

8. **[**ToDoVisualInspection**] obl** (ToReceiveLargeEmptyBottles) ∧ **obl** (ToObserveNon-Conformities);

/The visual inspection consists of receiving large empty bottles and observing non-conformity and detecting no suitable bottles./

9. **[**ToReturnLargeBottles(Non-suitableLargeEmptyBottles)**]** ToDetectNonSuitableLargeBottles = True;

/After returning the bottles that are not suitable, the goal of detecting inappropriate bottles 'ToDetectNon-suitableLargeBottles' is met/

..... end

end Object DoInspection

Figure 6 – The Component ‘DoInspection’

Axiom 8 in Figure 6, specify that as a result of the occurrence of action ´ToDoVisualInspection` actions ´ToReceiveLargeEmptyBottles` and ´ToObserveNon-Conformities´ are obliged to occur.

Guideline 5: The **Goals** and **Softgoals** in the strategic models can be translated as **predicates** in structured MAL

Goal-dependencies and softgoal-dependencies (See Figure 1 and 2) can initially be mapped as predicate (attributes) of an object specification. For example, in axiom 9 the condition (goal) ´ToDetectNon-SuitableBottles` is met after the occurrence of action 'ToReturnLargeBottles(Non-suitableLargeEmptyBottles)'.

These guidelines are appropriate to define an outline of a (formal) functional requirements specification. Of course, when necessary a more detailed specification can be produced. Remember that some issues have not yet been discussed because they were not considered important (strategic).

In the real world, during the requirements engineering process, the engineer can iterate navigating between the i* models (SD and SR) and the MAL specification, enabling him/her to cope with the impacts of each description. The two levels of description (organizational and formal specification) have enabled us to adopt different concepts of agent at each level, each one appropriate to the kind of modeling and reasoning required at each stage. At the level of organizational relationships, it is necessary a notion of agent that is flexible enough to express the freedom that agents have to violate restrictions and commitments. At the level of formal requirement specification, a prescriptive view is more appropriate than a descriptive one. Due to space limitation we are not dealing with exceptions and other situations.

6. Conclusion

The need for modeling the environment is well recognized in Requirement Engineering. Enterprise and organizational models have long been developed. The need for better precision, completeness and consistency of requirements has led to the proposal of many tools and techniques. However, when developing system that truly fulfill the real needs of an organization it is required to have a deeper knowledge of intentional and strategic aspects of the agents of the system. Many requirements models can not cope with the questioning of the reasons (or why) and end up dealing only with the functions of the system. The understanding of the rationale related to the agents of the system are important, not only to help in the development of a successful system, but also to facilitate the evolution of the system under development.

The i* technique has provided some means for modeling adequately the requirements of a system in the early-phases of activities. It deals also with non-functional aspects that traditionally have not been well represented in the existing conventional models. Thus to model and understand issues of the application domain of the mineral water factory, we used the i* technique [1]. This allows for a better description of the organizational relationships among the various agents of a system as well as an understanding of the rationale of the decisions taken.

On the other hand, the later-phase RE activities usually focus on completeness, consistency, and automated verification of requirements. They are concerned with the functional requirement description. However, for this, another technique is required. In this work we relied on the Structured Modal Action Logic - MAL [2].

Thus the need to integrate the early-phase with the later-phase of the RE activities is evident. In this work we gave some guidelines on how to integrate the descriptive model of the i* technique with the prescriptive model of the structured MAL logic, which enable a more precise specification.

In the process of integration some dependencies found in early-phase are directly translated. But other dependencies are not directly translated in MAL and may need further refinement. This is the case of softgoals, which initially are define as predicates, but whose valuation function may have later to be defined.

The benefits of this translation (from i* to MAL) are numerous and include the possibility of the specification animation that would help the validation of requirements and the potential for formal reasoning of the desired system properties.

Some tools support these techniques. For the i* technique there exists the OME, while for the MAL language, the MULTIVIEW environment allows the generations of MAL specifications, guided by the VSCS Method [19,20]. However, future work is required for the integration of the tools under single environment and the use of other formal language as target language. We are currently considering Z and OCL.

ACKNOWLEDGMENTS. This work was sponsored by CNPq grants Proc. 301052/91-3 and 520674/93-6.

REFERENCES

[1] E. S. K. Yu "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering". Proceedings of IEEE International Symposium on Requirements Engineering - RE97, p.226-235, Jan. 1997.

[2] S. Kent, T. Maibaum, and W. Quirk. "Formally Specifying Temporal Constraints and Error Recovery". Proceedings of IEEE International Symposium on Requirements Engineering - RE93, p.208-215, Jan. 1993.

[3]Berry, Daniel. “The Requirement Iceberg and Various Icepicks Chipping at it”. VIII Simpósio Brasileiro de Engenharia de Software. Curitiba – PR, Brasil. 25-28 Out., 1994.

[4] Loucopoulos, P. and Karakostas, V. "System Requirements Engineering". McGraw-Hill Book Company. 1995.

[5] Eric Yu, P. Bois and J. Mylopoulos. "From Organization Models to System Requirements - A Cooperating Agents Approach". The 3rd International Conference on Cooperative Information Systems - CoopIS-95, Vienna (Austria), May, 2-9, 1995.

[6]Eric Yu and John Mylopoulos. “Capturing Intentions for Business Process Reengineering”. IX Simpósio Brasileiro de Engenharia de Software, Recife – PE - Brasil, pp. 03-21, 03-06 Out., 1995.

[7]A Dardenne, A van Lamsweerde, and S. Fickas. “Goal-directed Requirements Acquisition”. Science of Computer Programming, 20:3-50, 1993.

[8]Janis A Bubenko. “On Concepts and Strategies for Requirements and Information Analysis”. In Information Modeling, p. 125-169. Chartwell-Brant, 1993.

[9]Thomas W. Malone and Kevin Crowston. “The Interdisciplinary Study of Coordination”. Computing Surveys, 26:87-119, Mar. 1994.

[10]María Carmem Leonardi, Julio Cesar S. do Prado Leite e Gustavo Rossi. “Estrategias para la identificación de Reglas de Negocio (in Spanish). XII Simpósio Brasileiro de Engenharia de Software, Maringá – PR - Brasil, pp. 53-67, 13-16 Out., 1998.

[11]Luiz M. Cysneiros e Julio Cesar S. do Prado Leite. “Definindo Requisitos Não-Funcionais (in Portuguese)”. XI Simpósio Brasileiro de Engenharia de Software, Fortaleza - CE - Brasil, pp. 49-64, 13-17 Out., 1997.

[12]E. Dubois, P. Du Bois and M. Petit. “O-O Requirements Analysis: an Agent Perspective”. In O Nierstraz, editor, Proc. Of the 7th European Conference on Object-Oriented Programming - ECOOP’93, p. 458-481, Kaiserslautern, Germany, Jul. 26-30, 1993.

[13]Roel Wieringa and E. Dubois. “Integrating Semi-Formal and Formal Software Specification Techniques”. In Matthias Jarke (Germany) and Dennis Shasha (USA) editors, Information Systems Vol. 23, No. 2/4, pp. 159-178, May/June 1998, Published by Elsevier Science Ltd.

[14] Emanuele Ciapessoni, A Coen-Porisini, E. Crivelli, D. Mandrioli, P. Mirandola, A Morzenti. “From Formal Models to Formally-based Methods: an Industrial Experience”. To appear in ACM Transactions on Software Engineering and Methodologies.

[15]Rumbaugh, J., Jacobson, I. and Booch, G.. “The Unified Modeling Language Reference Manual”. Addison-Wesley Object Technology Series. December, 1998.

[16]Warmer, Jos B. and Kleppe, Anneke G.. “The Object Constraint Language: Precise Modeling with UML”. Addison-Wesley Object Technology Series. March, 1999.

[17] Eric S. K. Yu, “Why Agent-Oriented Requirements Engineering”. REFSQ’97 - Requirements Engineering: Foundation of Software Quality, Barcelona, and June 1997.

[18] A. Finkelstein and C. Potts. “Building Formal Specifications using Structured Common Sense”. In. Proc. of the 4th International Workshop on Software Specification and Design - IWSSD’87, p. 108-113, Monterey, CA, April 3-4, 1987. IEEE, CS Press.

[19] Castro, J. B., Toranzo, M. A. and Gautreau, C.. “Tool Support For Requirements Formalisation”. International Workshop on Multiple Perpective in Software Development. San Francisco, USA. pp. 202-206. Out., 1996.

[20] Castro, J. B. and Toranzo, M. A.. “Tool Software Quality: Multiview Case”. Third International Workshop on Requirements Engineering: Foundation for Software Quality – REFSQ’97. Barcelona, Espanha. pp. 107-118. June 16-17, 1997.