Ontology and Goal Model in Designing BDI Multi-Agent Systems

Patrizia Ribino*, Massimo Cossentino*, Carmelo Lodato*, Salvatore Lopes*, Luca Sabatucci*, and Valeria Seidita†*
*Istituto di Calcolo e Reti ad Alte Prestazioni - Consiglio Nazionale delle Ricerche
Email: {cossentino, lodato, lopes, ribino, sabatucci}@pa.icar.cnr.it
†Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica
Email: valeria.seidita@unipa.it

***Abstract*—Nowadays several methodological approaches exist, each of them tightly tied up with the implementation platform supporting it. In this paper we propose an intermediate step toward the definition of a methodological approach for supporting the JACAMO framework. This paper resumes a previous work, focused on modeling BDI organizations, and we now address the requirements analysis phase. In particular, we propose the use of an ontological model and a goal model for representing requirements and the domain formalization respectively. The two portions of design process are connected by a heuristic process that allows to extract goals from the ontological model. The resulting models are also used for completing each other and for enhancing the problem description that is considered an input to our process. In the paper we use the well-known case study of the conference management system for illustrating the proposed portion of process.**

I. Introduction

It is widely accepted that methodological approaches offer significant advantages for system development. Accordingly, several complete methodological approaches have been proposed. Many of them are tightly tied up with the implementation platform supporting it. Each implementation platform gives support for the abstractions the methodology deals with and normally cannot be fit to methodologies different from the ones it has been created for.

In the past, we created and used AO methodologies such as PASSI [5] and ASPECS [6] that are strictly tied respectively to JADE [2] and Janus[11]. In the first case Jade allows to implement agents that principally are a state machine, whereas the second allows the implementation of holonic structures. A specific platform is suitable for a methodology in the sense that it supports its features, for instance we could not easily implement holonic structures/organizations with plain Jade.

In the latest period we have been experimenting the JaCaMo framework and its feature. JaCaMo is based on the BDI paradigm [15] and it provides an integration of tools and languages for programming the following dimensions: agents (Jason), environment (Cartago), and organisations ($\mathcal{M}$oise) [16]. Our work is focused on the normative and organizational aspects of developing BDI multi-agent systems under both the notational and methodological points of view. As regard the former, in [8] we illustrated a UML profile and a graphical notation for describing MASs based on the Jason metamodel. Whereas, as regard the methodological issues, our aim is to define a complete methodological approach that will include goal oriented analysis, the design of organizations and the design of agents based on the Jason platform; a first part of this work has been faced (see [7]) and a portion of methodology for developing MASs organized in hierarchical structures, implemented with $\mathcal{M}$oise+, has been completed.

The methodology we are creating is based on some cornerstones: *(i)* organizations, *(ii)* requirements expressed by goals and *(iii)* an ontological formalization of problem domain. This paper illustrates how we face goal modeling starting from a problem domain model represented by means of an ontology. Although we are aware that the use of ontologies is not shared unanimously by the software engineering community, we adhere to the trend that asserts ontologies may have a significant role in the model driven engineering and may offer several benefits to a design process (first of all to the requirement analysis)[1][18][3][17].

In such a context, the contribution of this paper is twofold. Firstly, we present two portions of design process able to cover the early requirements analysis phase resulting in problem specifications in terms of goals. Secondly, we propose the use of an ontology as an analysis (i.e: descriptive) model representing the reality of the problem domain in order to address the ambiguities and the incompleteness of the problem specifications. The two proposed activities result in an analysis model composed of an ontology and a goal model. The first one is used as a model for describing the problem domain by a set of meta classes (Concept, Action and Predicate). The second one is used to describe the objectives of the stakeholders involved in the problem and the dependencies among them. Hence, we have named these activities (portions of design process) *Problem Ontology Description* and *Goal Identification* respectively.

The ontological formalization we propose for performing Problem Ontology Description activity is thought to describe the intentional behaviors that typically characterize the class of problems addressed with agent oriented technologies. To this aim, inspired by the FIPA agent ontology [9], we introduce new elements in order to explicitly model intentional behaviors. These elements represent the problem domain entities able to act, according to their desires, in order to change the state of the world. Moreover, the instances of this kind of ontology present recurrent structures (*patterns*) that provide some useful information for identifying the goals that the solution of the problem has to satisfy.

The paper is organized as follows: section II focuses on the problem and sets our objectives, section III describes the portions of design process together with an example and,

finally, in section IV some conclusions are drawn.

II. Motivation and Objectives

The concept of *goal* is widely used in requirements engineering since it allows, among other things, to reduce the gap between analysts' and stakeholders' knowledge. Many definitions exist for goals. In this paper, when talking about goal, we mean a condition or state of the world that the actor wants to achieve[19]. This definition highlights a very interesting connection with the agent oriented paradigm since it recalls the aspects of autonomy and proactiveness, typical of software agents.

In most methodologies (not only agent oriented ones) the analysis phase results in a model of the requirements and in a model of the problem; this latter is constructed in agreement with the used design paradigm and its abstractions. Usually, the problem model includes at least one structural view and one or more dynamic (behavioral) views, uses a domain specific language and has a direct counterpart with the implementation concepts. Some fundamental thoughts for the identification of proper MAS design abstractions have been presented in [12]. Here the *social level* deals with abstractions like organizations, collaborations, communications and other social aspects of the agents' life and the *knowledge level* sees agents as conceived in terms of the goals they have to achieve and the actions they may perform for pursuing them. An agent processes knowledge in order to attain its goals [14].

As a consequence, we think that useful models for AO developing should include (*i*) a model of requirements in terms of goals and (*ii*) a model of the problem in terms of abstractions to be mapped onto the AO paradigm. Such abstractions should come from both the social and knowledge levels: organization, communication, actions, predicates (beliefs) and so on.

Thanks to these considerations we may argue that a good MAS design methodology should include a *goal model*, an *ontology* and a *model of organizations*. The ontology describes the problem in terms of elements of the environment, their significative states, and what it can be done on them to achieve/mutate their states. Such an ontology should include, *concepts* of the domain, *predicates* (used to represent beliefs and significant states of the concepts), *actions* that can be done on concepts and *positions* that may willingly execute actions; such positions may represent human beings, agents or other not-AO systems.

The goal identification is the preliminary activity of the agent goal modeling phase in which analyst makes it explicit the strategic objectives of the system (intended as the sum of the software and its environment). User goals identification is typically conducted in the early phases of requirement analysis. In this phase the core task is to conquer a significant and transferable understanding of the portion of the world where to introduce the software. This knowledge influences subsequent design decisions. It is frequent that agents' behavior derives from the goals and needs of stakeholders. So far, research has mainly focused on the development of methods for modeling and reasoning on goals, for optimizing their achievement and for solving possible conflicts. We think there is room for novel methods for systematically identifying goals from the domain and from the users.

Starting from the hypothesis that the use of ontology for describing the problem domain is fully used by current research [1][18][3][17], which considers ontologies a powerful mean for requirements engineering, and since several AO methodologies use ontologies for different aims, we propose a way for identifying goals from an ontological model of the problem. More specifically, our main contributions are two portions of design process (to be put early in the analysis phase) for *constructing the problem domain ontology* in a way useful for *identifying goals.* The proposed way of creating the domain ontology lets us support intentionality, it is based on FIPA ontology and above all is characterized by some recurrent structures useful for identifying goals.

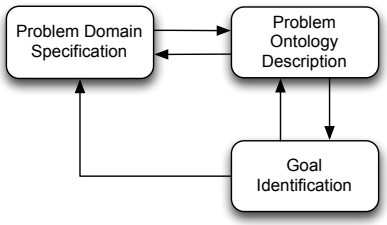

Fig. 1: The Cycle in the Early Requirements Analysis.

Generally, some advantages we found in the use of ontology are listed below:

- it is useful for formalizing knowledge and for disambiguation of terms;
- it helps in better understanding requirements and in eliminating redundancies and ambiguities;
- it simplifies comprehension among stakeholders and is useful for making clear the stakeholders' knowledge;
- it lets the designer create categories for the elements in the domain problem thus allowing to identify the artifacts that will compose the environment.

The portion of process we propose covers the initial activities of the requirement analysis for the methodological approach we are working on [7]. We named these activities respectively Problem Ontology Description and Goal Identification. The former is devoted to produce an in-depth knowledge and a formalization of the problem domain the software has to solve in terms of an ontology. The second one is devoted to produce a goal model describing the objectives of the stakeholders involved in the problem and the dependencies among them.

In the following sections we give some details about how to construct the problem domain ontology starting from a given problem statement, how to use it and how to carry out the goals identification from problem domain ontology.

III. The two proposed portions of design process

We propose an iterative approach to perform an early requirements analysis starting from textual problem specification (see Fig.1). Problem specifications are documents that could be affected by ambiguities since they are susceptible to different interpretations. Moreover, uncertainty in the interpretation of these documents could be caused by the implicit knowledge of domain experts. During knowledge elicitation, in fact, the problem is commonly described by different perspectives and by several stakeholders. This could generate overlapping terms

and redundancies that could be misinterpreted. Moreover, the unconscious tendency of the stakeholder to neglect some implicit knowledge he owns may cause lacks in the problem description. The transition from a textual description to a formal and structured representation of the knowledge (i.e: the problem ontology activity) could make this gap visible. We include the problem ontology description activity in an iterative process in order to refine the domain knowledge and to disambiguate the initial documentation.

We may obtain a better understanding of the problem by deepening the domain knowledge using an ontology. Since we use the problem domain ontology for identifying goals, we expect an improvement in the goal identification. Thus obtaining a more complete list of goals to be considered during the system design. During the goal identification some inconsistencies among goals or deficiencies may also arise. These issues can be faced going back to the problem ontology trying to discover their cause (if any) and to improve the ontology. Otherwise, the origin of these issues could depend on problem specification, as it can be seen in the whole cyclic process of Fig.1.

Throughout this paper, we will use the conference management case study as defined in [20] in order to explain our approach. "*The conference management system is an open multi-agent system supporting the management of international conferences that require the coordination of several individuals and groups. During the submission phase, authors should be notified of paper receipt and given a paper submission number. After the deadline for submissions has passed, the program committee (PC) has to review the papers by either contacting referees and asking them to review a number of the papers, or reviewing them themselves. After the reviews are complete, a decision on accepting or rejecting each paper must be made. After the decisions are made, authors are notified of the decisions and are asked to produce a final version of their paper if it was accepted. Finally, all final copies are collected and printed in the conference proceedings*".

In the following of this section we introduce the proposed activities.

A. The Problem Ontology Description

The *Problem Ontology Description* (POD) activity allows to describe the problem domain elements and their relationships in a formal way. This activity grounds on an ontology used as an analysis (i.e. descriptive) model for representing the reality of problem domains typically addressed by agent oriented technologies. This ontology is described by the Problem Ontology metamodel shown in Fig.2. The Problem Ontology metamodel, we propose, has been inspired by the FIPA (Foundation for Intelligent Physical Agents) standard and ASPECS ontology. Thus, similarly, our meta ontology describes what are the elements of interest in a domain (*Concept*) with their properties (*Predicate*) and how they act in the domain (*Action*) (see the upper part of Fig.2) and it introduces some new elements in order to explicitly model intentional behaviors.

In the following we give a detailed definition of each element and relationship in the Problem Ontology metamodel. *(I)* ***Concept*** is a term usually used in a broad sense to identify "*anything about which something is said*"[4]. We use the term Concept just for representing categories of domain entities. Such categories may either be physical or logical (abstract); *(II)* ***Action*** is defined as follow: "*an action is the cause of an event by an acting concept*" (adapted from [13]). We classified actions as intentional and unintentional [10].

Intentionality implies a kind of consciousness to act, the capability to plan and enact strategies for the achievement of a purpose. Therefore, the entity that performs an Intentional Action should be able to carry out a more or less complex reasoning such as having the ability to acquire and apply knowledge. This means the entity should be endowed with some kind of intelligence. Conversely, an Unintentional Action is an automatic response governed by fixed rules or a particular set of circumstances, in other words a purely reactive action in the sense of automatics control theory. Usually, an action can have different kinds of relations with the other elements of the ontology. An action is related with (i) a concept that performs it, (ii) a concept (the target) on which the action is accomplished causing a change of this concept state, (iii) a concept (the input) required from the action to be performed even without changing the state of the input concept, (iv) a position that is notified or receives the result of the action, (v) a predicate that plays the role of precondition, i.e. a condition under which the action can be performed and (vi) a predicate that plays the role of postcondition, i.e. the condition determined by executing the action. *(III)* ***Position*** is a concept performing both Intentional Actions and Unintentional Actions. Therefore, its actions can be part of a plan to fulfill some purpose. Whilst ***Object*** represents all the concepts that can perform only unintentional actions. Positions and Objects can be both *input* and *target* of an action. *(IV)* a ***Predicate*** is the expression of a property, a state or more generally a clarification to specify a concept.

As concerns relationships, our ontology metamodel supports: *(I)* ***is-a*** that is the relationship between an ontological element and a refined version of it; *(II)* ***part-of*** that is the part-whole relationship, in which ontological elements representing the components of something are associated with the ontological element representing the entire assembly; *(III)* ***association*** that is used in order to express all types of relations different from the previous ones that can occur between two ontological elements. Usually a label is used to specify the peculiarity.

Several manual and semi-automatic methods for building an ontology exist in literature. In the following we propose some guideline for building a model of our Problem Ontology adapted from the ones proposed in literature. We suggest this one as a quick and easy guidelines to follow, but the

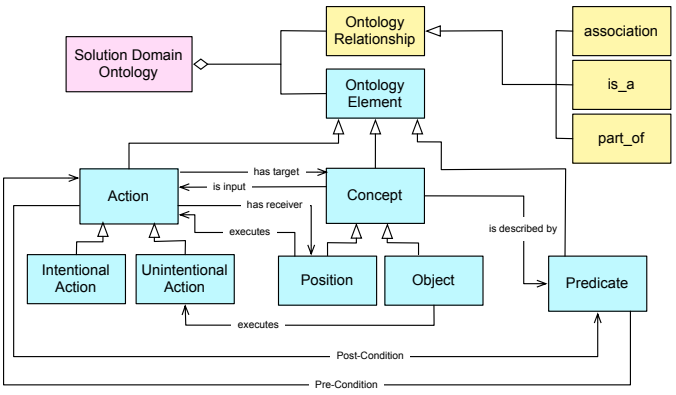

Fig. 2: The Problem Ontology Metamodel

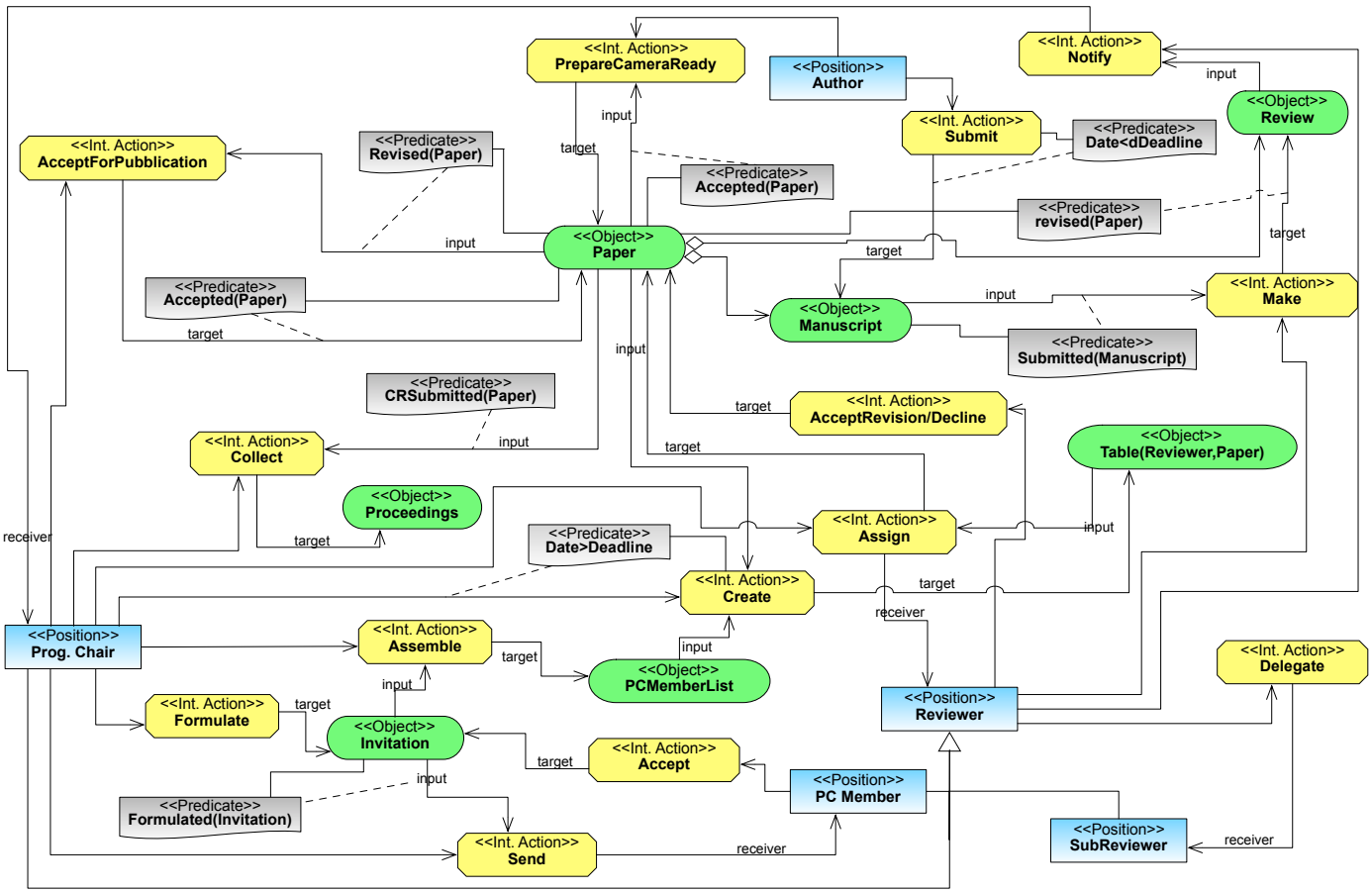

Fig. 3: The Problem Ontology Model for the Conference Management Case Study.

same results, or even better, may be achieved using different approaches.

Guidelines - With the aim of building a model of our Problem Ontology, we assume that a set of informal textual documents describe the problem the analyst is dealing with. Firstly, we extract the nouns from text and we create a list of items grouped in different clusters according to their grammatical function within the sentence. Thus, a noun used as: (i) "subject" followed by a verb is a candidate *Position* if the verb describes an *Action* on the domain; (ii) "adjective" is a candidate *Predicate* of the noun it describes; (iii) "direct object" is a candidate *Object*. Secondly, we identify verbs. In a sentence, a verb may indicate: (i) an action performed by the subject of the sentence thus becoming a candidate *Action*; (ii) a relation among nouns such as aggregation (PART-OF), inheritance (IS-A) or generic association. Finally, we identify the adjective to be candidate as a *Predicate*. Starting from these candidates we try to disambiguate the elements (if any) by asking more information to domain experts or stakeholders. Then we group in the same category the term with the same meaning. These categories will be the elements of the problem domain ontology.

Fig. 3 shows the ontology produced at the end of this activity and related to the before mentioned specification for the Conference Management case study. In this diagram the type of ontology elements is indicated by means of stereotypes and colors. Label on relationships indicates the type of ontological relationship.

B. Goal Identification

The problem domain ontology gives a deeper knowledge on the problem domain, especially when we perform some kind of reasoning on it. Thus moving in our context and reasoning about the elements and the relationships of our ontology, we determined many of the possible semantic combinations that could occur. We consider that the ontology is constructed by following the meta classes in Fig. 2, in so doing it presents repetitive semantic structures (hereafter *patterns*). Reasoning on these patterns, and on the meaning of the involved ontological elements, we deduced some useful information about the goals of the problem. A pattern contains many combination of elements useful for identifying at least one goal. In a pattern, at least one intentional action performed by one position is present. Intentionality implies a kind of consciousness to act, capacities to plan and enact strategies for achieving specific purposes. Consequently the intentional action may provide evidences to identify goals.

We have discovered many elementary and composed patterns. For space concerns, here we present only some of the elementary patterns we have found. We will show all possible patterns in a specific future work. The three patterns we present in this paper (see Fig.4) are elementary patterns we say atomic. These elementary structures may be surely found looking at each action and reasoning on its significance, on the position performing it and on the related input and target concepts.

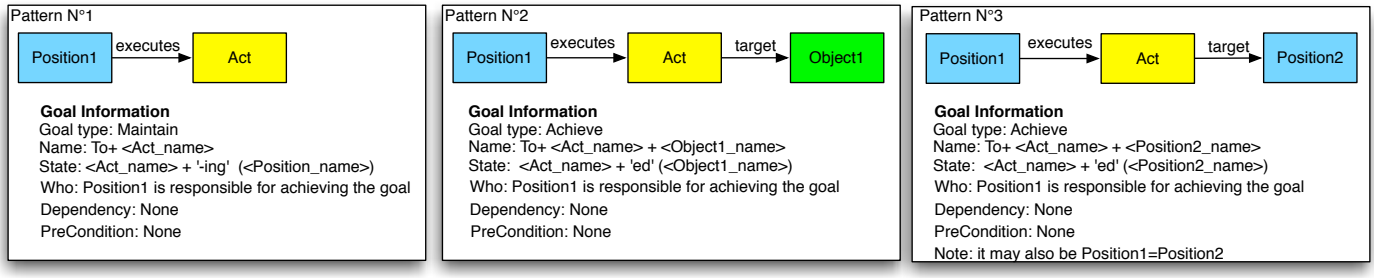

Fig. 4: Patterns Used for Identifying Goals

Each pattern is completed by a set of information useful for completely describing the goal:

- the *Goal Type*. We identified two possible types of resulting goal: the *achieve goal* describes the achievement of a particular state of affair (e.g: win the game) while the *maintain goal* describes the maintenance of some state of affair (maintain temperature below 100 degrees);
- the *Name* of the resulting goal encapsulating the goal semantics. The name is gathered from the specific pattern (see Fig.4);
- the *State* that is the desired condition or state of the world achieved or maintained.
- *Who* is responsible for achieving the goal. It may differ from who has the real interest in achieving that state of the world;
- some kinds of *Dependency* that is a relationship between the current and another goal. It indicates that the achievement/maintenance of a goal depends on the achievement/maintenance of the other goal.

The pattern number 1 presents a Position (Position1) that executes an Action (Act). The pattern number 2 presents a Position (Position1) that executes an Action (Act) on a target Object (Object1). Finally, the pattern number 3 presents a Position (Position1) that executes an Action (Act) on another receiver or target Position (Position2). When pattern 2 occurs in a problem ontology model, we can deduce that the resulting goal type is *achieve* because the effect of the action is to change the current state of the object and to achieve a desired state by the position performing the intentional action. The name of the resulting goal is obtained as combination of the basic form of the verb representing the action along with the infinitive marker "to" and the name of the object. The state is obtained by using the past simple of verb representing the action along with the name of the object, in such a way we indicate that the state is reached. On the contrary, in the pattern 1, no object states are changed by the action so the associated goal type is *maintain* and the -ing form of the verb is used to highlight the progressive (or continuous) aspect of this state. The pattern 3 is similar to the 2, the action is made on a position instead of on an object and the information for describing the goal are the same.

These patterns are the basic configurations we may find but they may be composed with other ones and actions may have inputs or not. Dependencies can be deduced from the composition or extension of patterns and above all from the input to actions. For this reason, we do not have information on dependency for patterns of Fig.4.

The goal information we retrieve are very useful later in the design phase in order to obtain system goals, to identify their dependencies, to identify roles to be assigned to agents, to represent agent beliefs and so on.

Guidelines - In order to perform the goal identification activity, we assume that the problem ontology model has been sketched and, in each pattern, actions may have inputs. In the case there are not inputs the goal does not depend on another goal (see Fig. 4). Therefore, we accomplish the following three steps: Prepare Goals List, Describe Goals, Prepare Goals Diagram. These three steps may be done sequentially or iteratively, in the sense that the analyst may decide to initially identify the whole list of goals, then describe them etc. or (s)he may complete the description and insertion in the diagram of one goal every time.

In preparing the goals list, for each action that appears in the ontology, the analyst has to identify the smallest pattern to which it might belong. When the pattern is identified it may be described so as it is evident in the patterns (Fig. 4).
The resulting work products of these two activities are two text documents. Whereas listing goals is a trivial activity because the analyst has only to scan the ontology and to identify each couple action-position, and it is not important which is the starting element, the description of the identified goal prescribes some reasoning about the mutual position of elements in order to discover dependencies. Indeed, for each action presenting inputs, the analyst has to check if the input is a target of another action, if yes probably there is a dependency between the goals associated to that actions. The dependency is given by the semantic relationships among actions and concepts in a domain ontology. The following example in the conference management system illustrates this point.
During the goals identification some considerations may be done, they lead to reveal some inconsistencies, ambiguities, mistakes or missing elements in the domain ontology.

In order to illustrate how to identify goals, let us consider Fig. 3 and let us start from the *AcceptForPublication* action. This action is performed by the *Program Chair* and the target is the *Paper* that, after the action has been executed, is in the state *Accepted* (the final state may be underlined by the Predicate associated to the object and to the target relation whereas the precondition for action is linked to the input action). The pattern recognized is the second one where the action also presents an input. Thus, the goal may be named *To AcceptForPubblication Paper* and the final state is *Accepted-*

Goal Information		
NAME: [to Submit Manuscript]	STATE: [submitted(Manuscript)]	WHO: Author
GOAL TYPE: Achieve	PRE-CONDITION: [Date<deadline]	DEPENDENCIES: []
NAME: [to PrepareCameraReady Paper]	STATE: [PreparedCameraReady(Paper)]	WHO: Author
GOAL TYPE: Achieve	PRE-CONDITION: accepted(Paper)	DEPENDENCIES: [To AcceptForPublcation Paper]
NAME: [To Collect Proceedings]	STATE: [collected(Proceedings)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [CRSumbitted(Paper)]	DEPENDENCIES:[To PrepareCameraReady Paper]
NAME: [to Assign Reviewer]	STATE: [assigned(Reviewer)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [created (Table(Reviewer,paper))]	DEPENDENCIES: [to Create Table(Reviewer,paper)]
NAME: [to Send To PC Member]	STATE: [sentTo(PCMember)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [formulated(Invitation)]	DEPENDENCIES: [to Formulate Invitation]
NAME: [to Create Table(Reviewer,paper)]	STATE: [created (Table(Reviewer,paper))]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [collected(PCMemberList)] AND [Date>deadline]	DEPENDENCIES: [to Collect PCMemberList]
NAME: [to Collect PCMemberList]	STATE: [collected(PCMemberList)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: [accepted(Invitation)]	DEPENDENCIES: [to Accept Invitation]
NAME: [to AcceptForPublication Paper]	STATE: [acceptedForPublication(Paper)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: Revised(Paper)	DEPENDENCIES: [to Make Review]
NAME: [to Formulate Invitation]	STATE: [formulated(Invitation)]	WHO: Prog.Chair
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES:[]
NAME: [to Accept Invitation]	STATE: [accepted(Invitation)]	WHO: PC Member
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES:[]
NAME: [to Delegate SubReviewer]	STATE: [delegated(SubReviewer)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: []	DEPENDENCIES: []
NAME: [to Notify Prog.Chair]	STATE: [Notified(Prog.Chair)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: [made(Review)]	DEPENDENCIES: [to Make Review]
NAME: [to make Review]	STATE: [made(Review)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION: [submitted(Manuscript)]	DEPENDENCIES: [to Submit Manuscript]
NAME: [to Accept Revision/Decline Paper]	STATE: [acceptedRevision(Paper)]	WHO: Reviewer
GOAL TYPE: Achieve	PRE-CONDITION:[]	DEPENDENCIES: []

Fig. 5: The List of Goals for the Conference Management Case Study

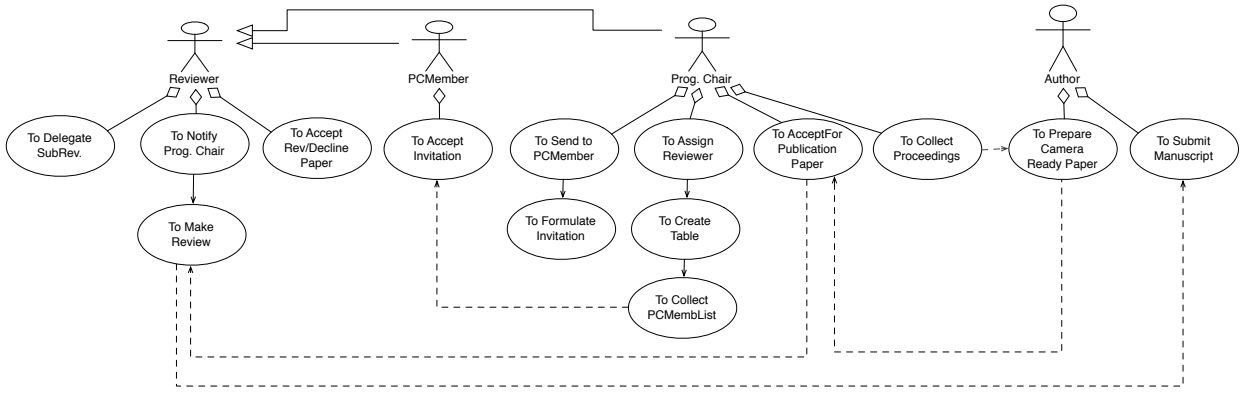

Fig. 6: The Goal Diagram Related to the Ontology in Fig. 3

ForPublication Paper. For pursuing this goal the related action has to have as input a *Paper* in the initial state *Revised*. The *Paper* is in this state after that *Reviewer* executes the *Make* action. This means that the goal *To AcceptForPubblication Paper* depends on the goal associated to the *Make* action; looking at this latter action the pattern number 2 may be identified, hence the related goal is *To Make Review* and the dependency is from *To AcceptForPubblication Paper* to *To Make Review*. Going on in this way the list of goals illustrated in Fig. 5 can be identified and the Goal Diagram in Fig.6 can be drawn. It is worth nothing that, during the first iteration of the Problem Domain Ontology description, the top part of Fig. 3 was in the form reported in Fig. 7, here we missed some predicates and we did not well represented the concept of Paper. We realized the omission while searching the *To AcceptForPubblication Paper* dependency. In fact, we have firstly realized the need for having a predicate RevisedPaper for the object Paper and then we realized that there was no actions devoted to change the state of Paper from submitted to revised. For this reasons we added the Manuscript and the Review as part-of Paper. In this way, we well represented the elements in the domain and then we were able to coherently find goal dependencies and the cycle presented in Fig. 1 was useful for identifying mistakes and omissions.

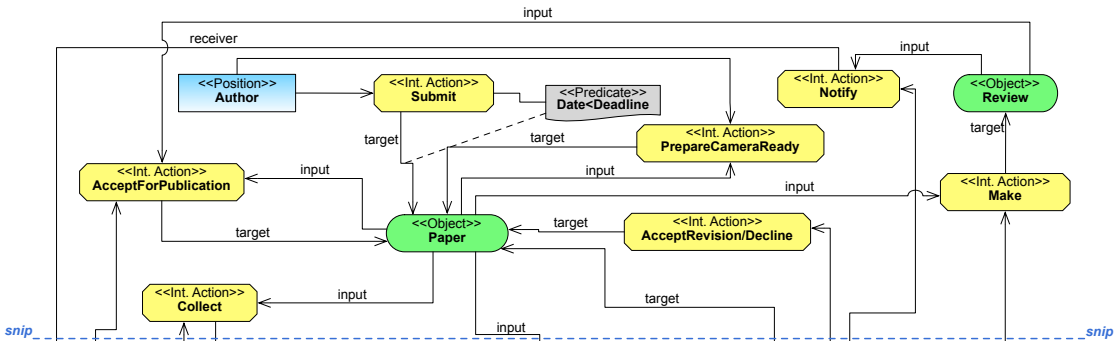

Fig. 7: A Portion of the Initial Problem Ontology Description.

IV. Discussions and Conclusions

The presented work is a part of a larger plan in order to cover the whole agent oriented requirement analysis for developing multi-agent systems for the JACAMO framework. This work addresses how to move from the problem formulation to the problem specification in order to obtain analysis models that can be fruitful employed for designing MASs. We experimented that the use of ontology in an iterative approach may allow to discover inconsistencies and lacks in the problem description and to improve the communication among stakeholders.

Moreover, the identification of goals from the ontological model of the problem domain allows the analyst to correlate goals with the corresponding portion of textual requirements. This double loop allows to act on the requirements after that the goal identification and problem ontology description activities have been performed. For instance, from goals identification it may arise a deficiency in the problem domain description that might thus be refined. Vice-versa a variation of the requirements can influence the goals that derive from them. This closed-chain process is ordered and not chaotic and it may result in a useful contribution to the quality of both the requirements elicitation and the analysis phase.

In addition, our approach allows to transform knowledge from an unstructured form to a structured one. This could result in an extra effort in the case of small size problems where it could be more easy to directly identify goals from the domain description, user interviews, etc. But the proposed approach can be an advantage in the case of large size problems. In fact, when the problem size grows up, the effort spent in managing unstructured information raises more quickly than the effort spent in applying our process.

Finally, the goal identification approach based on patterns we propose can be profitably used by an analyst with a limited expertise in goal identification and it could be automatized applying techniques of pattern recognition on problem ontology models.

References

[1] Uwe Aßmann, Steffen Zschaler, and Gerd Wagner. Ontologies, metamodels, and the model-driven paradigm. *Ontologies for Software Engineering and Software Technology*, pages 249–273, 2006.

[2] F. Bellifemine, A. Poggi, and G. Rimassa. JADE–A FIPA-compliant agent framework. In *Proceedings of PAAM*, volume 99, pages 97–108. Citeseer, 1999.

[3] Verónica Castañeda, Luciana Ballejos, Ma Laura Caliusco, and Ma Rosa Galli. The use of ontologies in requirements engineering. *Global Journal of Researches In Engineering*, 10(6), 2010.

[4] O. Corcho and A. Gómez-Pérez. A roadmap to ontology specification languages. *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*, pages 80–96, 2000.

[5] M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies*, chapter IV, pages 79–106. Idea Group Publishing, Hershey, PA, USA, June 2005.

[6] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. ASPECS: an agent-oriented software process for engineering complex systems. *Autonomous Agents and Multi-Agent Systems*, 20(2):260–304, 2010.

[7] M. Cossentino, C. Lodato, S. Lopes, P. Ribino, V. Seidita, and A. Chella. Towards a design process for modeling MAS organizations. In M. Cossentino, M. Kaisers, K. Tuyls, and G. Weiss, editors, *Multi-Agent Systems*, volume 7541 of *Lecture Notes in Computer Science*, pages 63–79. Springer Berlin Heidelberg, 2012.

[8] Massimo Cossentino, Antonio Chella, Carmelo Lodato, Salvatore Lopes, Patrizia Ribino, and Valeria Seidita. A notation for modeling jason-like bdi agents. *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, 0:12–19, 2012.

[9] Foundation for Intelligent Physical Agents. *FIPA RDF Content Language Specification*, 2001.

[10] H.G. Frankfurt. The problem of action. *American Philosophical Quarterly*, pages 157–162, 1978.

[11] N. Gaud, S. Galland, V. Hilaire, and Koukam A. An organisational platform for holonic and multiagent systems. In *In Proceedings of Programming Multi-Agent Systems (PROMAS) workshop*, 2008.

[12] N.R. Jennings. On agent-based software engineering. In *Artificial Intelligence*, volume 117, pages 277–296, 2000.

[13] E.J. Lowe. *A survey of metaphysics*. Oxford University Press Oxford, 2002.

[14] A. Newell. The knowledge level. *Artificial Intelligence*, 1982.

[15] Anand S Rao, Michael P Georgeff, et al. Bdi agents: From theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995.

[16] A. Ricci, M. Viroli, and A. Omicini. Carta go: A framework for prototyping artifact-based environments in mas. *Environments for Multi-Agent Systems III*, pages 67–86, 2007.

[17] M. Shibaoka, H. Kaiya, and M. Saeki. GOORE: Goal-oriented and ontology driven requirements elicitation method. In *Advances in Conceptual Modeling–Foundations and Applications*, pages 225–234. Springer, 2007.

[18] K. Siegemund, E. J Thomas, Y. Zhao, J. Pan, and U. Assmann. Towards ontology-driven requirements engineering. In *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*, 2011.

[19] Eric Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.

[20] Franco Zambonelli, Nicholas R Jennings, and Michael Wooldridge. Organisational rules as an abstraction for the analysis and design of multi-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(03):303–328, 2001.