Enhancing Requirements to derive Multi-Agent Architectures

Lúcia R. D. Bastos and Jaelson F. B. Castro

Centro de Informática, Universidade Federal de Pernambuco, Av. Prof. Luiz Freire S/N, Recife PE, Brazil
{lrdb, jbc}@cin.ufpe.br

Abstract. Software systems of today are characterized by the increasing size, complexity, distribution and heterogeneity. Understanding and supporting the interaction between software requirements and architectures remains one of the challenging problems in software engineering research. In this paper we present an approach for integration of system requirements and software architectures within the context of the Tropos project, an information system development framework that is requirement-driven in the sense that it adopts concepts used during early requirements analysis. Our framework advocates that a multi agent system corresponds to the organizational structure, in which actors are members of a group in order to perform specific tasks.

1. Introduction

Requirements Engineering and Software Architecture have become established areas of research, education and practice within the software engineering community. Requirements engineering is concerned with identifying the purpose of the system and the context in which it will be used. Requirements are related to concepts such as goals, conflicts, options and agreements [14]. Moreover, systems characteristics and properties (functional and non-functional) are also described in terms of requirements [7]. Software architectures are important because they represent the particular abstraction for understanding the structure of a system [1], [10].

There is a clear relationship between requirements and architectures. Unfortunately, terminology and concepts used for architectural description are quite different from those used for the requirement specification. In this paper we present an approach for integration of system requirements and software architectures within the context of the Tropos project, an information system development framework that is requirement-driven in the sense that it adopts organizational concepts used during early requirements analysis [5], [6]. Our framework proposes that a multi agent system corresponds to an organizational structure, in which actors are members of a group in order to perform specific tasks [2], [3].

This paper is structured as follows. Section 2 presents some basic organizational concepts. Section 3 overviews our approach in context of the Tropos methodology and introduces i* (i-star) requirement models and organizational architectural styles.

Section 4 outlines the definitions of our framework**.** Finally, Section 5 concludes the paper with considerations, related works, contributions and points of further research.

2. Organizational Concepts

The broad range of subjects and disciplines covered by organizational sciences has given rise to several models and theories on organizations. In every organization in the society, people play various roles to implement various functions of the organization [4]. Organizations are social groups that are goal-driven. A set of structured activities is required to achieve their goals [9]. This view of organizational group has a closer alignment to the functioning of real world. In our approach, organizational concepts are described in terms of the member actors and the roles that they play:

- ***Actor*** is an entity with intentional properties, such as, goals, beliefs, abilities and compromises. Actors may be people, software agents or organizational units (group or sub-group). An actor is the entity (e.g., 'university' or 'professor') that plays one or more roles.
  - ***Group*** is an organization unit (e.g., 'university') with a set of organized members being involved in social relationships, pursuing common goals for some period of time with an identifiable domain.
  - **Agent** is a member of a group (e.g., 'professor'), which plays roles. In this work, agents are system entity, situated in the group environment that is capable of flexible autonomous action in order to meet their goal.
- ***Position*** is a social status (or social identity) of a member within of a group (e.g. 'Adjunct professor'.
- ***Role*** is an abstract representation of the behavior of actor(s) that perform similar functions in a group, i.e., a *role* denotes a collection of responsibilities (e.g., 'teacher', 'adviser', etc.). Discharging these responsibilities requires the realization of a set of *role responsibilities*. For example, the 'teacher' role involves the tasks 'to teach' and 'to supervise'.
- ***Responsibility*** identifies the set of task (e.g., 'to teach' or 'to research') necessary to achieve social objectives (goals) of an actor playing a role.
- ***Goal*** is a condition or state of affairs in the world that the stakeholders would like to achieve.
- ***Task*** specifies a particular way of doing something. Tasks can also be seen as the solutions in the target system, which will satisfy the goals. These solutions provide operations, processes, data representations, structuring and constraints to meet the needs stated in the goals.

An important point to note is the distinction between the actor, i.e. the physical organizational entity, and the role, a notion that expresses the responsibility of satisfying certain organizational goal by performing the various tasks within the group. Roles are assigned to actors and summarize a set of skills or capabilities necessary to fulfill a goal. The role is separate from the actor that plays the role. For example, a 'professor' may play multiple roles such as 'teacher', 'department chair', etc.

The extensive use of organizational concepts in multi-agent system design emphasizes their importance for complex domains and implementation [16], [20]. Others approaches are using organizational concepts for modeling social actors in organizational structures [9], social agents in multi-agent system [8], [22], [23], [27], and authority in role-based access control (RBAC) [15]. Roles in specific contexts of action provide abstract specifications of distributed behavioral patterns.

The Tropos methodology is presented in the sequel.

3. Tropos Methodology

The Tropos methodology adopts the view of information systems as social structures that is a collection of social actors, human or software, which act as agents, positions, or roles and have social dependencies among them.

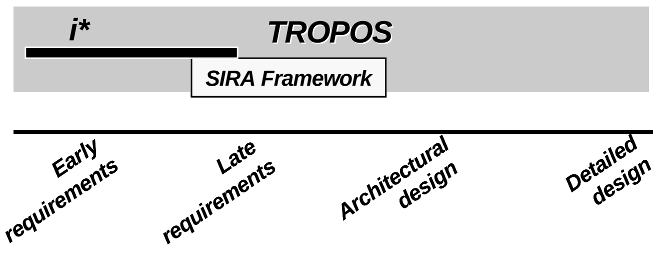

Fig. 1- SIRA Framework in Tropos context

As seen in Fig. 1, the Tropos methodology spans four phases: Early requirements, Late requirements, Architectural design and Detailed design. Tropos methodology does not explicitly cover the correlation between requirement elements and architectural elements. Moreover, the information available in the late requirement models (SD and SR) is not enough to derive architectural information. In order to address this issue, the Systematic Integration between Requirements and Architecture (SIRA) framework provides a set of elements to specify properties of organizational groups as well as to help to derive architectural properties in the context of the Tropos methodology. Further details about Tropos can be found in [5], [6].

In the sequel we present the i* framework which is used for describing requirements, as well as the organizational-inspired architectural catalogue. Both are central pieces of the Tropos approach.

3.1 The Requirement Model

The i* (i-star) framework focuses on the modeling of strategic actor relationships of a richer conceptual model of business processes in their organizational settings [24], [25]. The participants of the organizational setting are actors with intentional properties, such as, goals, beliefs, abilities and compromises. A dependency describes an "agreement" (called dependum) between two actors playing the roles of depender and dependee, respectively. The depender is the dependant actor, and the dependee, the actor who is depended upon.

In Fig. 2, we have the Strategic Dependency model of the e-commerce example. The Media Shop is a store selling and shipping different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes, and the like. To increase market share, Media Shop has decided to open up a B2C retail sales front on the Internet. The system has been named Medi@ and is available on the world-wide-web using communication facilities provided by Telecom Cpy. It also uses financial services supplied by Bank Cpy, which specializes on on-line transactions. Medi@ system is introduced as an actor in this strategic dependency model.

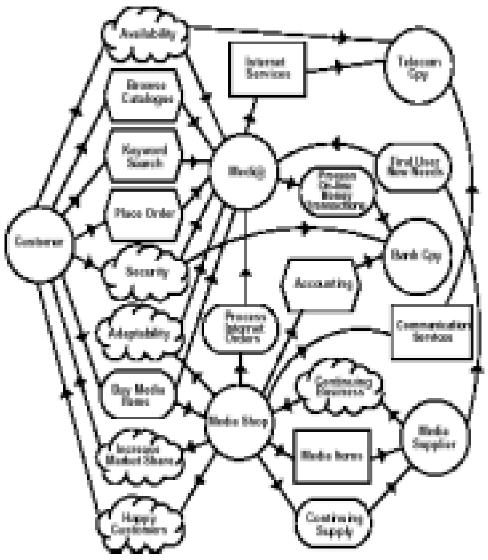

Fig. 2 - The SD model of e-commerce

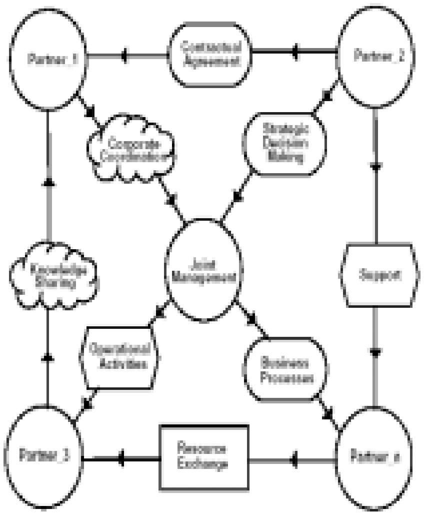

Fig. 3 - The Joint Venture Style

3.2 Architectural Catalogue

Multi-Agent systems (MAS) are being advocated as a next generation model for engineering complex and distributed systems. Currently, there are several classical software architecture styles, such as pipes/filters style, layered structure style [21]. However, MAS can benefit from new organizational architectural style, which are based on concepts and design alternatives coming from research on organization management [17]. These architectural styles (pyramid, joint venture, structure in 5, takeover, arm's length, vertical integration, co-optation, bidding) support the design of cooperative, dynamic and distributed MAS applications [11], [12].

Due to lack of space in this paper we only detail the joint venture style that is a decentralized style based on an agreement between two or more components called principal partners who benefit from operating at a larger scale and reuse the experience and knowledge of their partners. Each principal partner is autonomous on a local dimension and interacts directly with other principal partners to exchange services, data and knowledge. However, the strategic operation and coordination of the joint venture is delegated to a Joint Management actor, who coordinates tasks and manages the sharing of knowledge and resources (see Fig. 3).

The SIRA Framework will be detailed in sequel.

4. SIRA Framework

The SIRA (***S***ystematic ***I***ntegration between ***R***equirements and ***A***rchitecture based on Organizational Concepts) Framework emerges from the understanding that in organizations interactions aim at achieving some desired global goals, and the members are autonomous and heterogeneous. The software system is described from the perspective of an organization [2], [3].

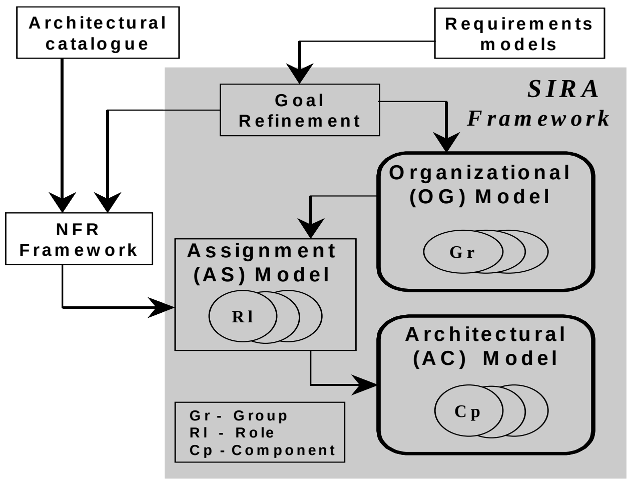

Fig. 4 - The *SIRA* Framework

Our approach is based on three complementary analysis models that separate the concerns of the social organization from those of the architectural organization, as seen in Fig. 4:

- Considering a requirement model as the functionality requested for the system, we identify the main goal. A goal is decomposed in sub-goals or tasks so that each sub-goal requires the cooperation of fewer actors. These actors are organized within groups in the Organizational (OG) Model. The organization of the social system consists of groups, roles and interactions, as intended by the system and its environment.

- Considering each one of the sub-groups, we identify a set of possible architectural style. Groups and Roles in the OG are related to components, based on their similarity with constraint and interfaces of architectural component candidates. The rules and guidelines concerning the assignment of responsibilities to architectural components are describes in the Assignment (AS) Model.
- Finally, given a set of members (group and sub-group) and the selected architectural style, the Architectural (AC) Model describes the architectural configuration result, which is the allocation of group and roles to architectural components.

In the sequel we describe the three model of the SIRA Framework.

4.1 Organizational Model

The SIRA Organizational (OG) model considers a group as a collection of roles whose behavior co-operatively determines the accomplishment of organizational goals. SIRA Group is meant to describe systems as organizational group structure at conceptual level.

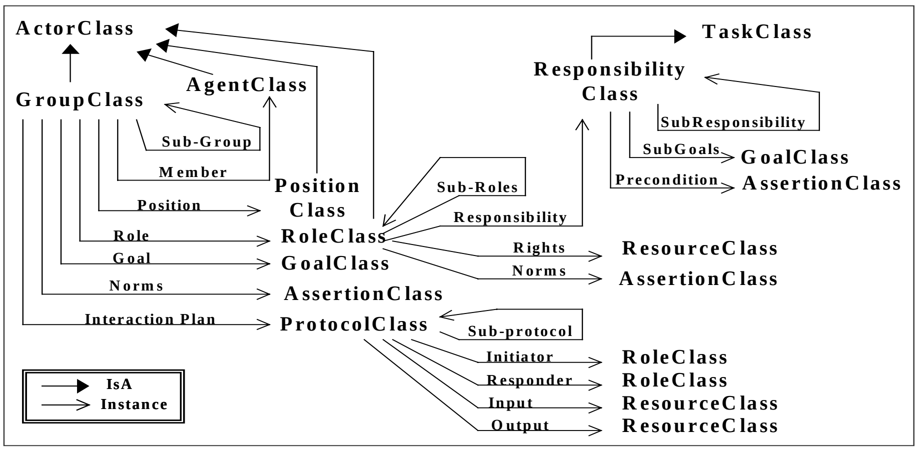

Fig. 5 – Organizational Elements in Telos

The meta-model of the SIRA organizational group is defined at the metaclass level in Telos, as a complementary set of i* (istar) meta-model [24]. Telos is a knowledge representation language described in [18] and has been chosen as the language to formally define the organizational group specifications. As seen in Fig. 5, Group Class is a sub-type of Actor Class from i*. GroupClass can be refined in SubGroup, Member, Positions or Role in order to represent the social structure of an agent society. GroupClass has a set of attributes:

- Member corresponds to a set of agents in the group;

- Positions are the set of social identity defined by the architectural style;
- Roles correspond to the division of labor between the members;
- Goal corresponds to a set of main responsibilities that the group should accomplish;
- Norm corresponds to constraints (or rules) on group behavior and interactions;
- Interaction Plan corresponds to a sequence of interaction protocols that members should perform to fulfill their group goals;

The organizational group structure can be defined in order to represent the social structure of an agent society. Groups are described in terms of their goals, the roles played by their members and norms that apply to the actor tasks. In the e-commerce example, the Medi@ actor provides service for three others actors, identified as *Media Shop*, *Customer* and *Media Supplier*. Consequently, the main responsibilities are to fulfill the goals identified as: “Process Internet Orders”, “Buy Media Items”, and “Find User New Needs” (see Fig.2). Roles are captured in a specific and bounded domain. A role clusters types of behavior into a meaningful unit (role responsibility), which is required to contribute to the group goals. Hence, the e-commerce domain can identify roles such as Buyer, Seller, Order Provider, Delivery or Manager. The Medi@ Group initial specification is shown in Table 1.

Table 1. Group initial specification

Group	Medi@
Goal	Process Internet Order, Buy Media Items and Find Users New Needs
Roles	Buyer, Seller, Order Provider, Delivery and Manager

Each *Group* can be refined in *Sub-Groups* to perform some goal in a particular context.

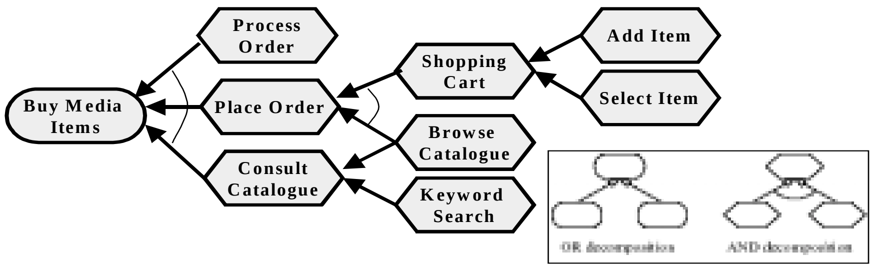

Fig. 6 – A partial goal refinement for ‘Buy Media Items’

Fig. 6 shows the goal refinement with i* means-ends analysis and AND/OR decomposition [5]. In particular means-end analysis aims at identifying tasks, resources and softgoals that provide means for achieving a goal. The goal ‘Buy Media Item’ is achieved through the tasks ‘Consult catalogue’, ‘Place order’ and ‘Process order’. The task ‘Consult catalogue’ is achieved through the tasks ‘Browse catalogue’ or ‘Keyword search’.

The basic idea behind the notion of groups is to provide means to collectively refer to a set of roles performing a collaborative set of tasks. Roles are described in terms of responsibilities, rights and norms (Fig. 4). Role responsibility allocation involves the definition of distribution of task and interactions to perform a full group goal. For example, if we considering the goal 'Buy Media Item' (see Fig.6), a possible set of roles and responsibilities for Medi@ Group is identified as follows:

- *Seller Role* is responsible for handling the internet shopping services and for supplying customers (playing the Buyer Role) with a web interface to keep track of items the customer is buying (ex. Consult catalogue and Place order).
- *Order Provider Role* is responsible for handling the process services that will be executed for a given order (ex. Process Order).
- *Security Manager Role* is responsible for monitoring and controlling the security check services, suck as: customer profiler and security order form.
- *Delivery Provider Role* is responsible for interacting with information system of delivery companies.

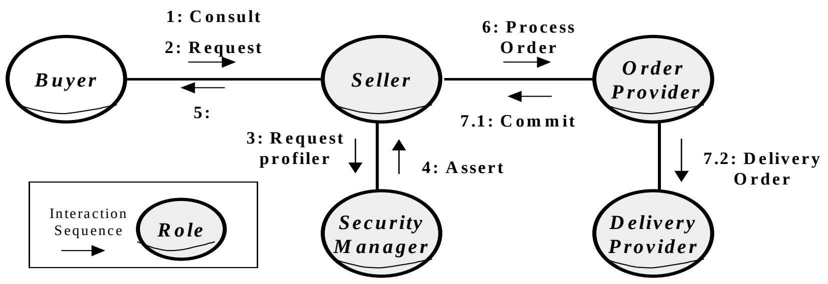

Fig. 7 – Role interaction model and a protocol specification between two roles

Fig. 7 shows the interaction sequence of UML collaboration graph [19] with the roles played to fulfill the group goal 'Buy Media Item' and their interaction protocols. The interaction protocols are expressed within a particular organizational level, i.e., roles are local to Groups or Sub-Groups. Note that UML collaboration diagram interaction semantics provide a precedence relation (i.e., partial order) among interactions. The interaction sequence can be viewed as directed and acyclic graph, with the nodes representing the roles and the arc representing the interaction among them. This representation makes it possible to distinguish between roles that interact and those that do not.

In the sequel we outlines the assignment model.

4.2. Assignment Model

The architecture of a multi-agent system is the pattern in which agents, processes performed by agents, and resource produced by agents make up an organization structure. In order to be able to say something about the relation between architecture and organization, we must to know which are the relevant properties of a multi-agent architecture, we must to select an architectural style, and we must to define which alternatives of responsibility assignment exist for each of those properties or pattern.

4.2.1 Defining Architectural Properties

Architecture of a system can be described as a collection of components together with a description of the interactions between these components [10]. *Multi-agent models* of organization represent a way of looking at software architecture styles. MAS are systems comprised from components, called agents. Besides agents, communication channels are distinguished as basic elements of the organizational structure.

Software architectures have been the focus of considerable research, which has resulted in a collection of well-understood architectural styles and a methodology for evaluating their effectiveness with respect to particular software qualities. However, there are few comparative studies of MAS. Some software quality attributes for multi-agent architectures were identified from a perspective of organizational styles, such as: predictability, security, adaptability, coordinativity, availability, integrity, modularity, aggregability [11], [12].

4.2.2. Selecting an Architectural Style

This work uses the NFR (non-functional requirements) framework [07] to evaluate the architectural styles. The notations includes: +, ++, -, --, to model partial/positive, sufficient/positive, partial/negative and sufficient/negative contributions, respectively. As an example, we compare four architecture styles, including some conventional (Pipes & Filter, Layers) [10] and organizational (Structure in 5, Joint Venture) ones. The following quality constraints to select a business-to-consumer architectural alternative could be stated accordingly [6], [11]: *Security, Adaptability, Availability*.

Table 2. Strengths and weaknesses of 4 architectural styles

	PIPES&FILTERS	*LAYERS*	*S-IN-5*	*JOIN VENT.*
Security	+	+-	+	+
Availability	+-	+	+	+
Adaptability	-	+-	+-	++

Table 2 summarizes strengths and weaknesses of four architecture styles with respect to the software quality attributes of Medi@ application. The layered architecture gives precise indications as to the components expected in a business to consumer system. The pipes-and-filters pattern concentrates on the dynamics of input/output data streams. The organizational patterns (Structure-in-5 and Joint Venture) focus on how to organize components expected in an e-business system as well as on the intentional and social dependencies governing these components. An exhaustive evaluation is difficult to be established at this point. But, considering preliminary results from Table 2, we can argue that the Joint-Venture architectural style is a better solution because it is a more decentralized style.

Once the architectural style has been chosen, it should conform to the application domain functional and non-functional requirements. The Assignment Model assigns actors (sub-group or agents) into particular categories or pattern of behavior (positions in architectural style).

4.2.3 Assigning SIRA Group responsibilities to Architectural Components

In SIRA Framework, actors should work according to the position they occupy in organizational structure. In order to relate actors (agents or sub-groups) and positions (identity of an architectural component), the responsibility assignment can be addressed based on the pattern of relationships in organizational structure. This work relies on two organizational structural views: relational and positional [26]. In the relational view, actor members (or agent) are clustered together based on the strength of the direct relationships with others also referred to as *cohesion*. The positional approach clusters actor members who have similar pattern of relations with others. All those playing a similar role are said to occupy similar structural status positions (see Table 3).

Table 3. Patterns of relationships in organizational structure

	Relational	**Positional**
Clusters based on:	Cohesion	Structural similarity
Belief sharing based on:	Interaction with similar others creates shared beliefs among the clusters members	Playing similar roles creates shared beliefs among those in the same role position

In the e-commerce example, we grouping actors based on structural similarity (or positional) . The set of sub-groups are:

1. Store Front sub-group interacts primarily with Customer (Buyer Role) and provides her with a front-end web application.
2. Back Store sub-group keeps track of all information about customers, products, sales, bills and other strategic important data.
3. Billing Processor sub-group is in charge of the secure management of orders and bills, and other financial data.
4. Manager sub-group manages all of other sub-groups controlling security gaps, availability bottlenecks and adaptability issues.

The Architectural Model will be outlined in next sub-section.

4.3. Architectural Model

A first architectural configuration is obtained from the set of sub-groups assigned to functional requirements, identified by roles and interactions. According to the Joint Venture style (Fig.5), the Medi@ software architecture can be decomposed into some autonomous components: a coordination component named Joint Management, and others components identified as Principal Partner_n, Principal Partner_1, Principal Partner_2..., Secondary Partner_1.

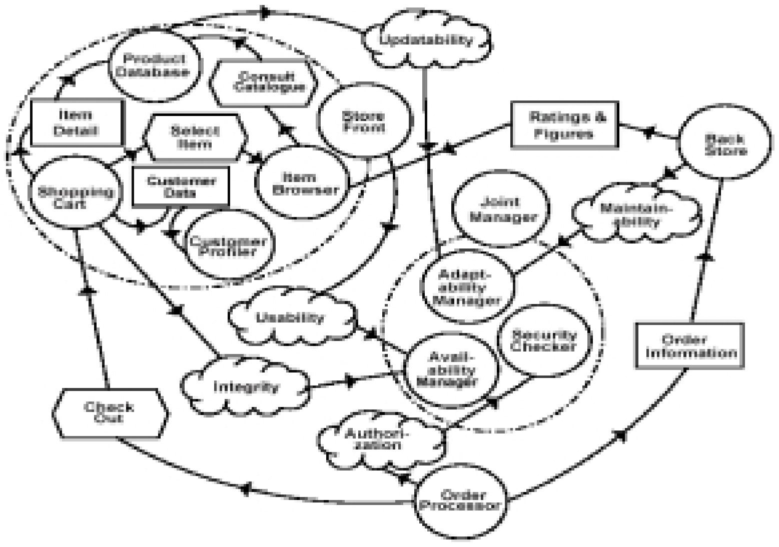

Fig. 8 - Medi@ system architecture model

Figure 8 shows the e-commerce architecture. The Medi@ Group decomposition into Sub-Groups suggest a possible assignment of system responsibilities:

- The Store Front Sub-Group, with input order responsibilities, can be related to Front Store to provide a customer with a usable front-end web application, which includes a web shopping cart and item browsing. Store Front occupies the Partner_1 position.
- The Provider Sub-group, with order-processing responsibilities, can be related to Order Processor to support the processing for a given order initialized in Store Front. Order Processor occupies the Partner_2 position.
- The Manager Sub-group, with managing responsibilities, can be related to Joint Manager to manage controlling security, availability and adaptability. Joint Manager occupies the Joint Management position.
- The Back Store Sub-Group, with support responsibilities, can be related to Back Store to aggregate functions outside the basic flow of operational tasks, which includes the delivery of items. Back Store occupies the Partner_3 position.

5. Final Considerations and Related Work

Software systems demand special care in the requirement and architectural modeling phases. A number of goal-based requirement approaches, most notably KAOS [13] and the NFR framework [07], support the explicit use of the notion of 'goals' to structure system requirements and architecture. Researches in Multi Agent System show that roles can represent system goal and constrain agent behaviors (ex. AALAADIN [8], MaSE [23] and GAIA [27]). In spite of the significant progress accomplished in the areas of requirement specification and architectural description, we still need frameworks, techniques and tools to support the systematic achievements of the architectural objectives in the complex context of the stakeholders' needs.

In this paper we present an approach for integration of requirements and software architectures within the context of the Tropos project. Our approach advocates that a system corresponds to the organizational structure, in which actors are members of a group assigned of roles in order to perform specific tasks. Roles can be used both as an intuitive concept in order to analyze requirements in multi-agent systems as well as a behavioral structure in order to derive coherent software architectures. This is an ongoing research and further investigations are still required to evolve this work. In particular we need to improve the architectural derivation rules. Among the main concepts we have: a) to incorporate more detailed rules to analyze the *Group* structures (group, roles and responsibilities); b) to incorporate more systematic rules to match architectural components from *Sub-Groups and Roles*; and c) to apply the proposal to more real study cases.

6. References

1. Bass, L., Clements, P. and Kazman, R. "Software Architecture in Practice". Addison-Wesley, (1998).
2. Bastos, L.R.D. and Castro, J.F.B.: "Integrating Organizational Requirements and Socio-Intentional Architectural Styles". Proceedings of the Second International Workshop From SofTware Requirements to Architectures (STRAW03), 2003. p.114 – 121. In 25th International Conference on Software Engineering 2003 (ICSE'03). Portland, May 2003.
3. Bastos, L.R.D. and Castro, J.F.B.: "Integration between Organizational Requirements and Architecture". WER03 - VI Workshop em Engenharia de Requisitos. Piracicaba, Brasil, November 2003.
4. Biddle B. J. and Thomas, E. J.: "Role Theory: Concepts and Research". New York: Robert E. Krieger Publishing Company, 1979.
5. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A.: "TROPOS: An Agent-Oriented Software Development Methodology". In Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May 2004.
6. Castro, J., Kolp, M. and Mylopoulos, J.: "Towards Requirements Driven Information Systems Engineering: The Tropos Project". In Information Systems, Vol. 27. Elsevier, Amsterdam, The Netherlands (2002) 365–389.
7. Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: "Non-Functional Requirements in Software Engineering". Kluwer Publishing, 2000.
8. Ferber, J. and O. Gutknecht.: "A meta-model for the analysis and design of organizations in multiagents systems". In Demazeau, Y., ICMAS' 98, pages 128–135, Paris, 1998.
9. Fox, M., Barbuceanu, M., Gruninger, M. and Lin, J.: "An Organization Ontology for Enterprise Modelling". Simulating Organizations: Computational Models of Institutions and Groups, M. Pritula, K. Carley & L. Gasser (Eds), Menlo Park CA: AAAI/MIT Press, pp. 131-152. 1996.

10. Garlan, D. and Shaw, M.: “An introduction to software architecture”. Ambriola & Tortola (eds.), Advances in Software Engineering & Knowledge Engineering, vol. II, World Scientific Pub Co., Singapore, 1993, pp. 1-39.
11. T. T. Do, S. Faulkner and M. Kolp.: “Organizational Multi-Agent Architectures for Information Systems”. In Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003), Angers, France, April 2003.
12. Kolp, M., Giorgini, P. and Mylopoulos, J.: “Organizational Patterns for Early Requirements Analysis”. 15th International Conference on Advanced Information Systems Engineering (CAiSE'03), Velden, Austria. June 2003.
13. Lamsweerde, A. van.: “Goal-Oriented Requirements Engineering: A Guided Tour”. Proceedings RE’01, 5th International Symposium on Requirements Engineering. Toronto, Canadá. August 2001, 249-263.
14. Lamsweerde, A. van.: “Requirements Engineering in the Year 00: A Research Perspective”. 22nd International Conference on Software Engineering. Ireland. June 2000.
15. Lupu, E.C. and Sloman, M.: “Towards a role based framework for distributed systems management”. Journal of Network and Systems Management, vol. 5, no. 1, P.Press, 1997.
16. Mao, X. and Yu, E.: “Organizational and Social Concepts in Agent Oriented Software Engineering”. Agent-Oriented Software Engineering (AOSE) IV, James Odell, P. Giorgini, Jörg Müller, eds, Lecture Notes on Computer Science volume (forthcoming), Springer, Berlin, 2004.
17. Mintzberg, H.: “Structure in Fives: Designing effective organizations”. P. Hall, 1992.
18. Mylopoulos J., Borgida A., Jarke M., and Koubarakis M.: “Telos: Representing Knowledge About Information Systems”. ACM Transactions on Information Systems, 8(4): 325–362, 1990.
19. Odell, J., Parunak H. Van D., Bauer, B.: “Representing Agent Interaction Protocols in UML”, *Agent-Oriented Software Engineering,* Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp. 121–140, 2001.
20. Partsakoulakis, I., and Vouros, G.: “Roles in MAS: Managing the Complexity of Tasks and Environments”. Multi-Agent Systems: An application Science, T. Wagner (eds.), Kluwer Academic, 2004.
21. Perry, D. and Wolf, A.L.: “Foundations for the study of software architecture”. ACM SIGSOFT Software Engineering Notes, 17(4), Oct. 1992, pp. 40-52.
22. Trzebiatowski, G. L. and Münch, I.: “The Role Concept for Agents in Multi-Agent Systems”. Modelling Artificial Societies and Hybrid Organizations. Workshop at KI2001, the Joint German/Austrian Conference on Artificial Intelligence. Vienna, 19-21, 2001.
23. Wood, M.F. and DeLoach, S.A.: “An overview of the multiagent systems engineering methodology”. First international workshop, AOSE 2000 on Agent-oriented software engineering, p.207-221, January 2001, Limerick, Ireland.
24. Yu, E.: “Modeling Strategic Relationships for Process Reengineering”. Ph.D. thesis, Department of Computer Science, University of Toronto, Canada (1995).
25. Yu, Eric and Mylopoulos, John.: “Understanding Why in Software Process Modelling, Analysis, and Design”. Proceedings of 16th International Conference on Software Engineering, May 16-21, 1994, Sorrento, Italy, pp. 159-168.
26. Zack, M. H.: “Researching Organizational system using social network analysis”. Proceedings of the 33rd Hawaii International Conference on System Sciences. Maui, Hawaii, January 2000 (IEEE 2000).
27. Zambonelli, F., Jennings, N.R., and Wooldridge, M.: “Developing Multiagent Systems: The Gaia Methodology”. ACM Transactions on Software Engineering and Methodology, 12(3): pp. 317-370. 2003.