Integration between Organizational Requirements and Architecture

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 50732-970, +1 5581
{{lrdb, jbc}@cin.ufpe.br

Abstract. *Software systems of today are characterized by increasing size, complexity, distribution, heterogeneity, and lifespan. Understanding and supporting the interaction between software requirements and architectures remains one of the challenging problems in software engineering research. To address these challenges we are investigating the relationship between the requirements and software architecture.In this work we show an approach for this integration of systems requirements and software architectures within the context of the Tropos project.*

1. Introduction

Requirements Engineering and Software Architecture have become established areas of research, education and practice within the software engineering community.
The requirements engineering is concerned in identifying the purpose of the system and the context in which it will be used. The software architecture has long been recognised to have a profound impact on the achievement of non-functional goals ("ilities") such as availability, reliability, maintainability, safety, confidentiality, evolvability, and so forth.

There is a clear relationship between requirements and architectures. In spite of this, evolving and elaborating system requirements into a viable software architecture satisfying those requirements is still a difficult task, mainly based on intuition. 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 the integration of systems requirements and software architectures within the context of the Tropos project, an information system development framework that is requirements-driven in the sense that it adopts concepts used during early requirements analysis [3]. This paper is structured as follows. Section 2 presents the Tropos project, including a modeling framework for requirements analysis namely the i* framework, and the organizational-inspired architectural styles. Section 3 emphasizes the existence of conceptual differences between requirements and architecture. Section 4 introduces our approach to integrate

organizational requirements and socio-intentional styles. Section 5 summarizes the related work. Finally, section 6 concludes the paper with final considerations, contributions and points further research.

2. The Tropos Methodology

An information system plays a critical role in the management of an enterprise. The need for modeling the enterprise and organizational environment is well recognized in requirement engineering [1][2]. However, when developing system that truly fulfil the real needs of an organization it is required to have a deeper knowledge of intentional and strategic aspects of the system. Many requirements models cannot cope with the questioning of the reasons (or why) and end up dealing only with the functions of the system. Goals from the organizational model can be used as a starting point when constructing the architectural description. Goals related to functional abilities provide the basis for system requirements, while goals related to business and system qualities provide the basis for non-functional requirements.

The Tropos methodology adopts the view of information systems as social structure [3]. By social structures, we mean a collection of social actors, human or software, which act as agents, positions, or roles and have social dependencies among them. The Tropos adopts concepts offered by the i* organizational modeling framework [2] [4], such as actor, agent, position, role, and social dependency. More details about *Tropos Methodology* can be found in [3].

2.1 Requirements in i* framework

The i* framework focuses on the modeling of strategic actor relationships of a richer conceptual model of business processes in their organizational settings. The i* caters for some of these advanced concepts. It can be used for: (i) obtaining a better understanding of the organizational relationships among the various system actors; (ii) understanding the rationale of the decisions taken; and (iii) illustrating the various characteristics found in the early phases of requirements specification.

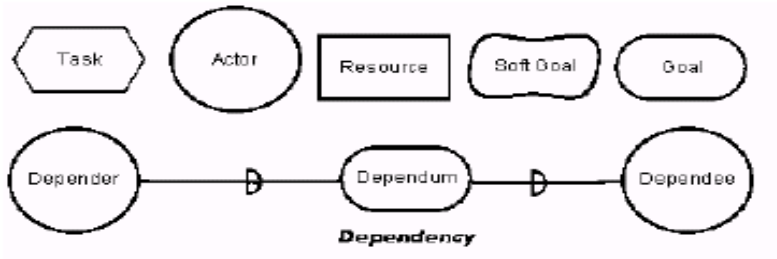

Fig. 1. Elements from i* framework

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. A dependency describes an "agreement" (called *dependu*m) between two actors playing the roles of *depender* and *dependee,* respectively. The *depender* is the depending actor, and the *dependee,* the actor who is depended upon. Dependencies have the form *depender→dependum→dependee*.

The i* technique consists of two models: The Strategic Dependency Model (SD) and the Strategic Rationale Model (SR).

The Strategic Dependency Model (SD) includes a set of nodes and links connecting them, where nodes represent actors and each link indicates a dependency between two actors. There are 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*.

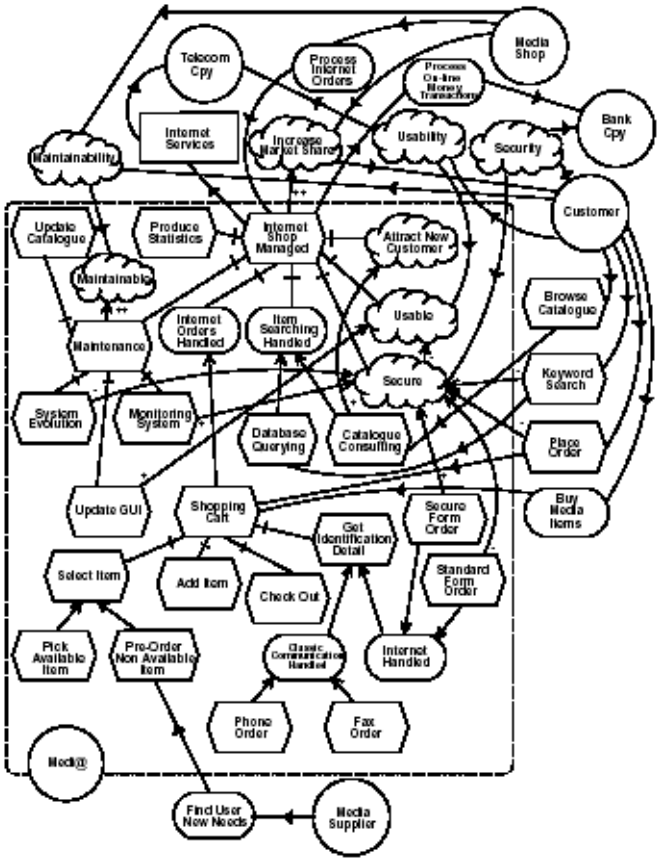

Fig. 2. The SR model of the e-commerce example. The Media Shop is a store selling and shipping different kinds of media items such as books, newspapers, audio CDs.

The second model of the technique i* is the Strategic Rationale Model (SR model). 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. This model includes the previous four types of nodes (present in the SD model): *goa*l, *tas*k, *resource* and *soft-goa*l. There are two new types of relationship, *means-end* that suggests that there may be other means of achieving the objective (alternatives) and *task-decomposition* that describes what should be done in order to perform a certain task

In Fig. 2. the *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. The figure postulates a root task *Internet Shop Managed* providing sufficient support to the softgoal *Increase Market Share*. That task is firstly refined into goals *Internet Order Handled* and *Item Searching Handled*, softgoals *Attract New Customer*, *Secure* and *Usable* and tasks *Produce Statistics* and *Maintenance*. *Internet Order Handled* is achieved through the task *Shopping Cart*, which is decomposed into subtasks: *Select Item*, *Add Item*, Check *Out,* and *Get Identification Detail*. More details can be founded in [3].

In next sub-section we will detail the organizational-inspired architectural styles *Tropos*.

2.2. Socio-Intentional Architectural Styles

System architecture constitutes a relatively small, intellectually manageable model of system structure, which describes how system components work together. *Tropo*s has defined organizational architectural styles [5] [6] to guide the design of the system architecture. These architectural styles *(pyramid, joint venture, structure in 5, takeover, arm's length, vertical integration, co-optation, bidding)* are based on concepts and design alternatives coming from research on organization management [18].

Due to lack of space in this paper we only detail the *joint venture style* which is a decentralized style based on an agreement between two or more 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. Outside the joint venture, *secondary partners* supply services or support tasks for the organization core, as seen in Fig.3.

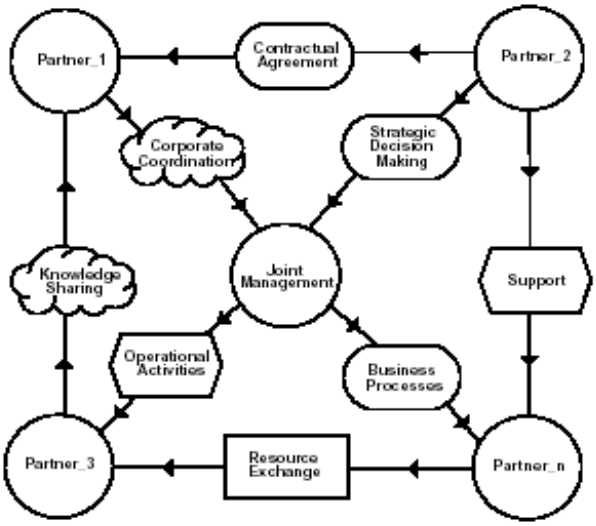

Fig. 3. The joint venture pattern

However, the interconnection of requirements and intentional software architecture is not straightforward. Some problems are presented in next section.

3. The Gap between Requirements and Architectural Description

The requirements are related to concepts such as goals, conflicts, options and agreements. Moreover, systems characteristics and properties (functional and non-functional) are also described in terms of requirements. Indeed, requirements can be simple or complex, necessary or ambiguous, declared concisely or elaborated carefully. On the other hand, the terminology and concepts used for architectural description are quite different from those used for the requirements specification. Architecture includes components, which are the computational element and data elements in a software system. The interactions among components are captured through explicit software connectors.

The architecture models a solution for the problem described in the requirements and provides high-level abstractions for representation of the structure, behavior and main properties of a software system. In addition to specifying the structure and topology of the system, the architecture should show the intended correspondence between the system requirements and elements of the constructed system. It can additionally address system-level properties such as capacity, throughput, consistency, and component compatibility [12].

The inter-dependencies and constraints between requirement elements and architectural elements are thus not well understood and consequently only little guidance is available in bridging the gap among requirements and architecture. The

existence of conceptual differences between what to do (requirements) versus how to do it (architecture, design and code) constitutes a semantic gap.

The following section outlines the basis of our approach that tries to fill this gap.

4. Systematic Integration between Requirements and Architecture (SIRA)

In this research work, we propose a framework to identify and map key architectural elements and the dependencies among those elements, based on the stated system requirements. In our approach, the requirement specification should include not only software specifications but also other kinds of information describing the context and the environment in which the intended system will function.

In this section we present the theoretical foundations and describe some initials guidelines to support the systematic integration of the requirements modeling and architectural design phases of Tropos. Sub-section 4.1 presents the Systematic Integration between Requirements and Architecture Descriptions (SIRA) framework. Sub-section 4.2 outlines the set of complementary information proposed in the SIRA component named ***SIRA-Elements.*** Section 4.3 summarizes the activities of the ***SIRA-Process*** to map architectural information.

4.1 *SIRA* Framework

The *Systematic Integration between Requirements and Architecture (SIRA) framework* provides a set of complementary elements to enhance the requirements analysis of the i* Strategic Dependency - SD and Strategic Rationale – SR models and supplements the information needed to derive a high-level architectural description. This approach has a systematic process to support the identification and the mapping of architectural decision from a given requirements specification. The *SIRA* Framework in Tropos context is showed in Fig. 4.

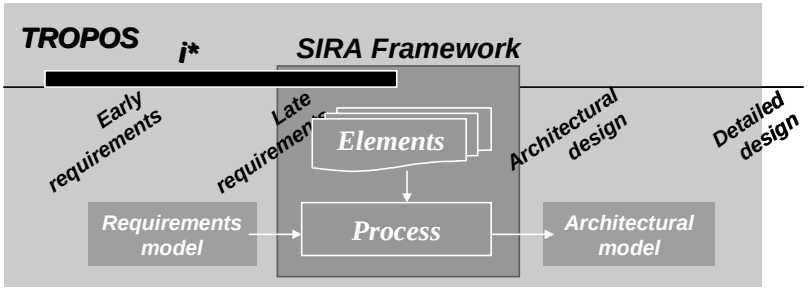

Fig. 4. SIRA Framework in Tropos context

The *SIRA* Framework is composed of SIRA Elements and SIRA Process: *SIRA Elements* includes the definitions of schemas, templates and guidelines. These element models are used to increase the i* requirements models with complementary information; and SIRA Process provides a systematic support to capture and to analyze the SIRA Elements using as input the i* requirements models and architectural catalogue (socio-intentional styles). It generates the *System Group schema* and the i* architectural model, as output.

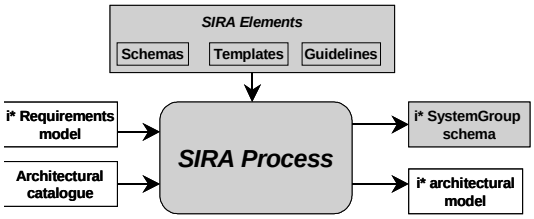

Fig. 5. The *SIRA* Framework.

As showed in Fig.5. the *SIRA Elements* are represented by Schemas, Templates and Guidelines:

- Schemas – Each schema represents, in i* notation, the relationships between requirements elements and provide a systematic decomposition of system actors and sub system components. The SIRA Elements are System Groups, Constraints, Architectural Elements. The output set is named *System Group Schema*;
- Templates – Each template are used to capture and to refine the complementary properties from *SIRA elements* represented in each schema. In this proposal we are presenting two templates examples: *System Group* template (Table 4.) and *System Role* template (Table 3.);
- Guidelines – A set of guidelines are defined to support the mapping from strategic rationale model into *SIRA*-Elements (Groups and sub-groups) and the mapping from SIRA-Elements to architectural elements, in the selected architecture style. Some initial guidelines will be presented in the *SIRA*-Process.

As showed in Fig. 6. the *SIRA Process* has three activities: The first activity (*Analyzing Elements)* takes as input the i* requirements models (together with the Architectural catalogue) to generate the SIRA Element named *System Group Schema*. In the second activity (*Applying NFR)*, the Non-Functional Requirements framework (NFR) is used to select architectural styles, based on non-functional requirements extracted from i* models. In the third activity (*Relating Elements*), the *System Group Schema* are related to the architecture elements and used to generate the i* architectural models.

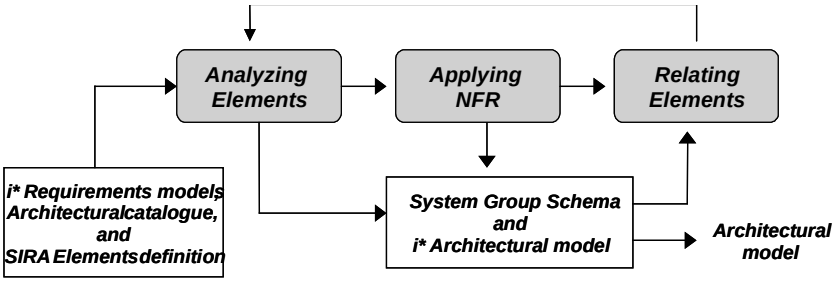

Fig. 6. The *SIRA* Process Activities The *SIRA Process* focuses on a systematic way to support the transition from requirements specification to architecture model.

4.2 The SIRA Elements

The organizational view extracted from Strategic Dependency and Strategic Rationale models is used to capture system related goals. In `i*`, an actor is an active entity that depends on other actors for goals to be fulfilled, softgoals to be achieved, tasks to be performed, and resources to be furnished. The initial set of SIRA Schemas identified to complement the i* requirement models include definitions of:

1. *System Group* - The software elements of the architecture. A *System Group* can be a component or sub-components of a software system. In our case study necessary to support the selling of media items in Internet, like Medi@ actor in (Fig.2). *Each System* Group can be refined into Sub Group to cover some service in a particular context:
   - *System Roles* – Each Sub Group assume a specific behavior to execute a service in the context;
   - *Responsibility* – Services and capabilities assigned to sub groups. Responsibilities are extracted from the set of tasks to be performed and goals to be fulfilled by system actors, like "Place order" or "Buy media items".
2. *Constraints* – Assertions and constraints that apply to the entire system or components. A constraint can be a softgoal, like security or availability;
3. *Architectural Elements* - Elements to represent an architectural model in i*. They are System Actors (Components) and interactions between components.

4.2.1 The System Group

The Role Theory can also be of some use as it is widely applied for enterprise modeling, postulating that individuals occupy positions in an organization [15, 16, 17]. Associated with each position is a set of activities including required interactions that constitute the responsibilities of that position. The organizational model offers a set of abstraction that can influence the division of labor and the coordination mechanisms, and consequently the system responsibilities and task assignments. The organizational structure can defines the roles of various components (actor), their

responsibilities, defined in terms of tasks and goals they have assigned and resources they have been allocated.

In this work we propose that the software system can be seen as an organizational structure, in which *actors* are members of a system structure in order to perform specific tasks. *Software System* is a concept that correspond a group in role theory. A software system (*System Group*) consists of an actor (or set of actors) that plays one or more role (*System Role*).

The *SIRA* framework uses the *i** notation to model the relationship between group, system and roles. In i*, the term *actor* refers generically to any unit (agent, role or position) to which intentional dependencies can be ascribed. An agent can be seen as an actor with concrete and physical manifestations (person or system). A role is an abstract characterization of the behavior of a social actor within some specialized context. A position is a set of roles typically played by one agent. Fig. 7. presents i* notation of agent, role and position.

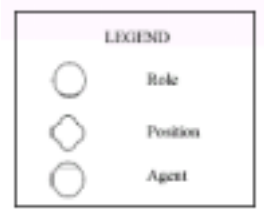

Fig. 7. – i* notation to Role, Positions and Agents

The *System Group* schema is a generic structure defined at a *metalevel* that can be instantiated to model a specific application domain. An example using a generic system of the e-commerce domain is illustrated in the Fig. 8. An identified system actor is represented as a *System Group* position (SystemGroup). A *System Group* can be refined into *Sub-Groups* (SubSystem1…3). A sub-group in a particular position may cover *System Roles* such manager, provider or customer.

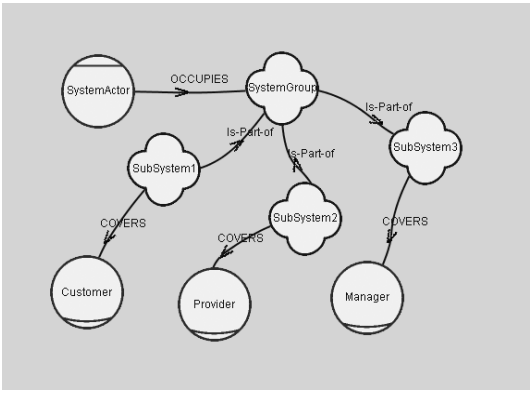

Fig. 8. An e-commerce *System Group* schema

To identify each *System Role* we are using the categorization based on functions (tasks that an user can carry out) and context (target that an user may carry out the tasks) to establish the key role characteristics of the organizational domain [18]. Each domain has a *System Group* (actor) that assumes the functionality required by the organization. Each set of functionality identifies one or more role in the organizational system context. An initial *System Role* categorization is identified from an e-commerce system context, as described in Table 1. .

Table 1. - *System Role* Categorization

Customer/Client	Role to handle goals or tasks to receive a service or product.
Provider	Role to handle goals or tasks to perform a service or deliver a product.
Manager	Role to handle goals or tasks to monitor and control a service or product.

Each role is defined with a set of goals to be fulfilled, i.e., a role is an abstract representation of the responsibilities of a specific actor. In order to identify the *roles,* we propose an initial classification of the responsibilities (tasks and goals) based on the idea of context-based skill aggregation, as showed in Table 2. Some of then can be refined to provide more specific skill, like *basic order input* or *basic order process*.

Table 2. Responsibilities

Basic	To aggregate functions of input, processing and output services
Manager	To aggregate functions of coordination and managing services
Support	To aggregate functions outside the basic flow of operational tasks.

4.3 The Integration Process

In this section we outline the activities of the *SIRA Process*, which focuses on a systematic way to support the transition from requirements specification to architectural model.

4.3.1 Analyzing Elements

This activity covers a requirements analysis using as input the i* requirements model. It consists of guidelines to identify the *SIRA-Elements* and capture architectural relevant information about these elements. As output we have the role schemas and templates to complement the information of requirement elements. This activity includes three initial sub-activities:

Identify Correlation among the i* Strategic Rationale Model and SIRA-Elements – The possible architectural elements are identified from i* Strategic Dependency and Strategic Rationale models;

Identify the System Responsibilities – The organizational context is used to identify the functionality that the system component should provide. Usually this functionality is identified from the main goal dependencies and its task decomposition. Hence, the system responsibilities should be classified according to the Table 2.

Identify System Roles – The system functionality should be aggregated and categorized into *System Roles* according to the Table 1. **Table 1.** Each set of responsibilities can be related to each organizational role of a specific domain (like e-commerce domain). The i* system actor can be refined into system component to address each *System Role*.

Some guidelines and templates are provided to support these activities. The Medi@ actor is used as example, as seen in Fig.2.

Guideline 1: (*i*) actor can be identified as a System Actor (software) to be analyzed.* The Medi@ system is an e-commerce information software system that supports the business in question, i.e, the commerce of media items in the Internet platform. The first system component identified could be the Medi@ actor.

Guideline 2: *(i*) goal dependency can be identified as main goal dependency.* The main goal of Medi@ actor is to provide service for others two actor, identified as *Media Shop* and *Customer*. The main dependencies are identified and the main goals to be fulfilled are "Process Internet Orders" and "Buy Media Items".

Guideline 3: (*i*) goals and tasks* assignment *can be identified.* For instance, the task (responsibility) assignment that fulfills the two main goals are: The goal identified as "Process Internet Orders" can be associated to the task Internet Shop Managed. The goal identified as "Buy Media Items" can have the following tasks: Shopping Cart, Place Order and Browse Catalogue (see Fig.2.).

Guideline 4: The i* tasks can be identified by responsibility type (see Table 2)

- The *basic* responsibilities include tasks to fulfill the main organizational goals. These tasks include inputs for production and transformation into outputs. An example is the task "Shopping Cart";
- The *support* responsibilities include tasks that fulfill organizational goals such as "Security". These tasks include process standardization. An example is "Secure Form Order";
- The *manager* responsibilities include tasks that fulfill the organizational goals such as "Increase Market Share". These tasks should monitor and analyze the goals fulfillment. An example is "Internet Shop Managed".

Guideline 5: (i*) System actor can be identified as a System Group position.

Guideline 6: (i*) System Group position can be refined into Sub-Group. The system role categorization is used to identify each sub system that plays the role. In our case study, a possible assignment of roles and responsibilities for an e-commerce software system can be identified following the system role schema for e-commerce domain (as seen in Fig.8).:

Guideline 7: (i*) Sub-Group position can cover one (or more) System Role.

1. *The Sub-Group SubSystem1 covers Customer Interface Role* to handler user interface services. These include basic input and output responsibilities, like "Shopping Cart";

2. *The Sub-Group SubSystem2 covers Provider Order Role*, to handler order processing services. These include basic processing responsibilities, like "Orders Handler";
3. *The Sub-Group SubSystem3 covers Manage*r Order *Role* with control system functions. These include managing and controlling responsibilities, like "Internet Shop Managed".

Table 3. System Role template for Media@. It shows the partial template definition of a System Role named Customer Interface Role, categorized as Customer, that has basic input/output responsibilities to handler user interface services to fulfill an goal identified as Buy Media Item.

Type: System Role Name: Customer Interface Category: Customer Description: Handler user interface services to input and output customer order data. Responsibility Type: basic input/output Goal: Buy Media Item

Table 4. Main component templates for Media – The table shows the partial template definition of a main component with complementary architectural definitions. The *Name* attribute is the i* specification from which the element (actor) was derived. In our example "Medi@" actor is a system component. The System *Roles* attribute is an initial list of responsibility assignment (tasks and goals). *Composed of* attribute identifies the sub-components that implement the component.

Type: System Group Name: Medi@ System Roles: Customer Interface; Provider Order and Manager Order Architectural Pattern: {will be defined in phase 2} Composed of: {sub components – will be defined in phase 3} ...

The *Architectural Pattern* attribute identifies the architectural style and will be defined in the next sub-section.

4.3.2 Applying NFR Framework

An important task during architectural design is to select among alternative architectural styles using as criteria the desired qualities (NFR) identified in the previous phase (Late Requirements). They will guide the selection process of the appropriate architectural style.

As an example, we compare four architectures styles, including some conventional (Pipes & Filter, Layers) [12] and organizational (Structure in 5, Joint Venture) ones. Table 5. summarizes strengths and weaknesses of the 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 but also on the intentional and social dependencies governing these components. An exhaustive evaluation is difficult to be established at that point. But, considering preliminary results from Table 5. , we can argue that the organizational architectural style (*Joint-Venture* or *Structure in 5*) better fit systems and applications that need open and cooperative components like the e-commerce example.

Table 5. strengths and weaknesses of four architectures

	Pipes & Filters	Layers	S-in-5	Joint Venture
Security	+	+-	+	+
Availability	+-	+	+	+
Adaptability	-	+-	+-	++

The evaluation results in contribution relationships from the social structures to the quality attributes, labeled "+", "++", "-", "--" that mean respectively *partially satisfie*d, *satisfie*d, *partially denied* and *denie*d.. The analysis involves refining these qualities, represented as softgoals, to sub-goals that are more specific and more precise and then evaluating alternative architectural styles against them, After this non-functional analysis, the Joint Venture style of the Tropos socio-intentional catalogue is selected as the best architecture candidate to be applied in Medi@ example. The joint venture style was introduced in sub-section 2.2 (see Fig.3).

More details about the selection and non-functional requirement decomposition process can be found in [11].

4.3.3 Relating Elements

This activity consist of applying the i* architectural extension (templates and guidelines) to support the relationship between the i* requirements elements and architectural elements. This relationship can be identified by functionality and related to the SubSystem roles in *Analyzing Elements activity*.

Guideline 9: The identified System Sub-Groups can be mapped to a system components in the selected style.

In *SIRA* framework, the *System Group* schema can be used to define the responsibilities and behavior (System Role) of possible architectural components. It also can relate each SubSystem (Sub-Group) of the application domain to the functionality of each architectural component. For instance, the Medi@ actor is a *System Group* of an e-commerce domain. The Medi@ System Group decomposition into SubGroup has some identified role (System Role). Each role suggests a possible

assignment of responsibilities for each architectural components of the joint venture style:

- The Customer Interface role with input responsibilities can be related to *Store Front* to supply a customer with a usable front-end web application for supplying a web shopping cart and item browse.
- Provider Order role with order-processing responsibilities can be related to *Order Processor* to provide the processing for a given order initialized in Store Front.
- Manager Order role with managing responsibilities can be related to *Joint Manager* to manage controlling security, availability and adaptability.
- The component *Back Store* assumes support responsibilities to produce statistical analyses and historical charts.

Hence, a possible refinement of the Medi@ system architecture model is shown in Fig.10. Each component system is generated from a System Group, as seen in previous activity phase (Relating Elements). The Medi@ system model is generated with three components that assumes the principal partners positions (Store Front, Billing Processor and Back Store), and component to coordinates tasks (Joint Management). Each component is represented as i* actor.

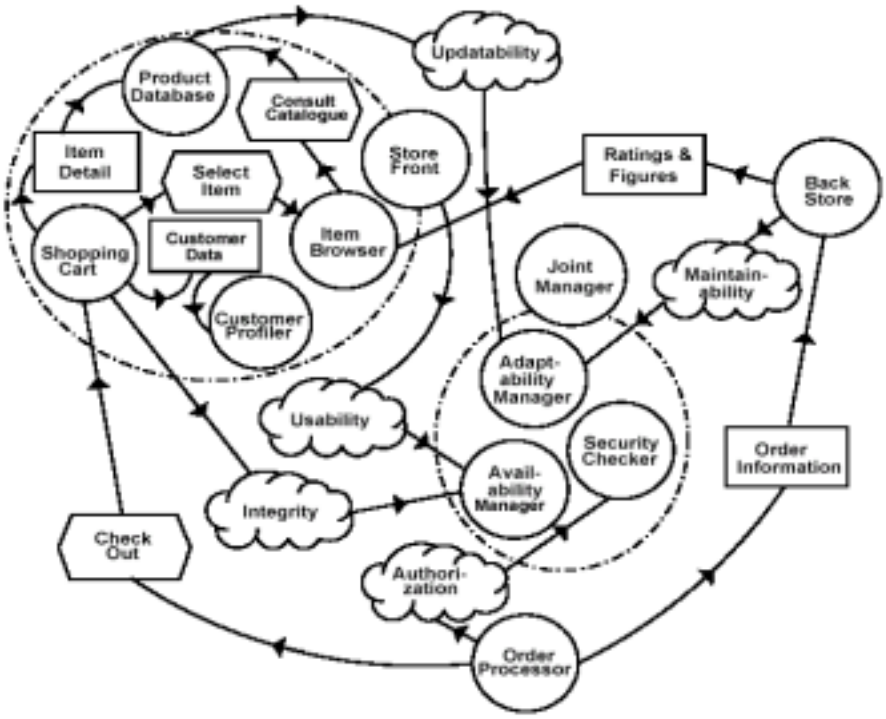

Fig. 9. Medi@ system architecture model

5. Related Works

The software systems of today are characterized by increasing size, complexity, distribution, heterogeneity, and lifespan. These systems demand special cares in the requirements modeling and in the architectural model in the early phases of the analysis. This relationship between requirements and architectures has been received a growing attention more recently [13][14]. A number of goal-based requirements approaches, most notably KAOS [8] [9] and the NFR framework [11], have proposed the explicit use of the notion of 'goals' to structure system requirements and architecture. The CBSP approach [7] [10] explores the relationships between software requirements and architectures, and proposes a technique to reconciling mismatches between requirements terminology and concepts with those of architectures. But these approaches do not establish an explicit relation between elements of the problem domain and architectural components in solution domain.

6. Final Considerations

In spite of the significant progress accomplished in the areas of requirement specification and architectural description, we still need frameworks, techniques and tools for the systematic support in the systematic achievements of the architectural objectives in the complex context of the stakeholders' needs. Our research focuses on finding a systematic process to support the transition from requirements specification to architectural description. In doing so we hope to achieve means to show that a given software architecture satisfies a group of functional and non-functional requirements.

The main contribution of our integration approach is to assure that the architecture components represent or are associated to the main organizational requirements (goals and tasks) which in turn will be fulfilled by a software system. Additionally, this approach also makes it possible to improve the requirement traceability along the whole software development process. It emphasizes the organizational environment (actors, goals and tasks) and hopefully helps to reduce the gap among requirement specification and architectural models.

Further work is still required to evolve this proposal. In particular, we need to improve the complementary information (schemas, templates and guidelines) to augment the requirement specification and to derive architectural Information.

References

[1] Yu, E.: "Modeling Strategic Relationships for Process Reengineering". Ph.D. thesis, Department of Computer Science, University of Toronto, Canada (1995).

2 Yu, E., and Mylopoulos, J.: "Modeling Organizational Issues for Enterprise Integration". ICEIMT'97 - International Conference on Enterprise Integration and Modeling Technology. Turin, Italy. October 1997.

3 Castro, J., Kolp, M., Mylopoulos, J.: “Towards Requirements Driven Information Systems Engineering: The Tropos Project”. In Information Systems, Vol. 27. Elsevier, Amsterdam, The Netherlands (2002) 365–389.
4 Yu, E.: ‘Agent Orientation as a Modelling Paradigm”. Wirtschaftsinformatik. 43(2) April 2001. pp. 123-132.
5 Kolp, M., Castro, J., Mylopoulos, J.: “A social organization perspective on software architectures”. In Proc. of the 1st Int. Workshop From Software Requirements to Architectures. STRAW’01, Toronto, Canada (2001) 5–12.
6 M. Kolp, P. Giorgini and J. Mylopoulos. “Organizational Patterns for Early Requirements Analysis”. 15th International Conference on Advanced Information Systems Engineering (CAiSE'03), Velden, Austria. To appear, June 2003.
7 Grünbacher, P., Egyed, A. and Medvidovic, N.: “Reconciling Software Requirements and Architecture: The CBSP Approach”. Proceedings RE’01, 5th International Symposium on Requirements Engineering. Toronto, Canada. August 2001.
8 Lamsweerde, A. van.: “Requirements Engineering in the Year 00: A Research Perspective”. 22nd Proceedings of International Conference on Software Engineering, Limerick, Ireland. Jun. 2000.
9 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.
10 Nuseibeh, B.: “Weaving the Software Development Process between Requirements and Architectures”. First International Workshop From Software Requirements to Architectures (STRAW'01). May, 2001.
11 Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J.: “Non-Functional Requirements in Software Engineering”. Kluwer Publishing, 2000.
12 Shaw, M.: “Abstraction for Software Architecture and Tools to Support Them”. IEEE Transactions on Software Engineering, 21(4): pp.314-335, April 1995.
13 STRAW’01. Proceedings of First International Workshop From Software Requirements to Architectures (STRAW’01), 2001. http://www.cin.ufpe.br/~straw01.
[14] STRAW’03. Second International Workshop From Software Requirements to Architectures (STRAW'03). May 9,2003. Portland, Oregon, USA. http://se.uwaterloo.ca/~straw03/.
15 Biddle B. J. and Thomas, E. J.: “*Role Theory: Concepts and Research”*. New York: Robert E. Krieger Publishing Company, 1979.
16 Crook, R., Ince, D. and Nuseibeh, B.: “Towards an Analytical Role Modeling Framework for Security Requirements”. In proceedings of Eighth International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ'02. September 2002. Essen, Germany.
17 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.
18 Mintzberg, H.: “Structure in Fives: Designing effective organizations”. Prentice Hall, 1992.