Organizational Patterns for Early Requirements Analysis

Manuel Kolp[1], Paolo Giorgini[2], and John Mylopoulos[3]

[1]IAG - Information Systems Research Unit - University of Louvain, 1 Place des Doyens, B-1348 Louvain-La-Neuve, Belgium, tel.: 32-10 47 83 95, kolp@isys.ucl.ac.be
[2]Department of Information and Communication Technology - University of Trento 4 via Sommarive, I-38100, Trento, Italy, tel.: 39-0461-88 2052, paolo.giorgini@dit.unitn.it
[3]Department of Computer Science - University of Toronto, 40 St George Street, M5S 2E4, Toronto, Canada, tel.: 1-416-978 5180, jm@cs.toronto.edu

Abstract. Early requirements analysis is concerned with modeling and understanding the organizational context within which a software system will eventually function. This paper proposes organizational patterns motivated by organizational theories intended to facilitate the construction of organizational models. These patterns are defined from real world organizational settings, modeled in *i** and formalized using the Formal Tropos language. Additionally, the paper evaluates the proposed patterns using desirable qualities such as coordinability and predictability.

1 Introduction

Modeling the organizational and intentional context within which a software system will eventually operate has been recognized as an important element of the requirements engineering process (e.g., [1, 5, 22]). Such models are founded on primitive concepts such as those of *actor* and *goal*. This paper focuses on the definition of a set of organizational patterns that can be used as building blocks for constructing such models. Our proposal is based on concepts adopted from organization theory and strategic alliances literature. Throughout the paper, we use *i** [22] as the modeling framework in terms of which the proposed patterns are presented and accounted for.

The research reported in this paper is being conducted within the context of the *Tropos* project, whose aim is to construct and validate a software development methodology for agent-based software systems. The methodology adopts ideas from multi-agent system technologies, mostly to define the implementation phase of our methodology. It also adopts ideas from Requirements Engineering, where actors and goals have been used heavily for early requirements analysis. The project is founded on that actors and goals are used as fundamental concepts for modeling and analysis during *all phases of software development*, not just early requirements, or implementation. More details about *Tropos* can be found in [3]. The present work

J. Eder and M. Missikoff (Eds.): CAiSE 2003, LNCS 2681, pp. 617–632, 2003.

continues the research in progress about social abstractions for the *Tropos* methodology. In [14], we have detailed a social ontology for *Tropos* to consider information systems as social structures all along the development life cycle. In [13, 15, 10], we have described how to use this *Tropos* social ontology to design multi-agent systems architectures. As a matter of fact, multi-agent systems can be considered structured societies of coordinated autonomous agents. In the present paper, we emphasize now the use of organizational patterns based on organization theory an strategic alliances for early requirements analysis, with the concern of modeling the organizational setting for a system-to-be in terms of abstractions that could better match its operational environment (e.g., an enterprise, a corporate alliance, …)

The paper is organized as follows. Section 2 describes organizational and strategic alliance theories, focusing on the internal and external structure of an organization. Section 3 details two organizational patterns – the structure-in-5 and the joint venture – based on real world examples of organizations. These two patterns are modeled in terms of social and intentional concepts using the *i** framework and a formal specification language (Formal Tropos) founded on *i**. Section 4 identifies a set of desirable qualities for comparing and evaluating these patterns. Finally, Section 5 summarizes the contributions of the paper and overviews related work.

2 Structuring Organizations

Since the origins of civilization, people have been designing, participating in, and sharing the burdens and rewards of organizations. The early organizations were primarily military or governmental in nature. In the *Art of War*, Sun Tzu describes the need for hierarchical structure, communications, and strategy. In the *Politics*, Aristotle wrote of governmental administration and its association with culture. To the would-be-leader, Machiavelli advocated in the *Prince* power over morality. The roots of organizational theories, then, can be traced to antiquity, including thinkers from around the world who studied alternative organizational structures. Such structures consist of stakeholders – individuals, groups, physical or social systems – that coordinate and interact with each other to achieve common goals. Today, organizational structures are primarily studied by two disciplines: *Organization Theory* (e.g., [17, 18, 19, 21]), that describes the internal structure of an organization, and *Strategic Alliances* (e.g., [6]), that model the external collaborations of independent organizations who have agreed to pursue a set of shared business goals.

2.1 Organization Theory

"An organization is a consciously coordinated social entity, with a relatively identifiable boundary, that functions on a relatively continuous basis to achieve a common goal or a set of goals" [21]. Organization theory is the discipline that studies both *structure* and *design* in such social entities. Structural issues deal with descriptive aspects while design issues address prescriptive ones. Organization theory, as far back as Adam Smith, describes how practical organizations are actually structured, offers suggestions on how new ones can be constructed, and how old ones

can change to improve effectiveness. To this end, schools of organization theory have proposed patterns to try to find and formalize recurring organizational structures and behaviors.

For instance, the structure-in-5 pattern [17] consists of the typical strategic and logistic components generally present in many organizations. At the base level, the Operational Core takes care of basic tasks – the input, processing, output and direct support procedures – associated with running the organization. At the top lies the Apex, composed of executive actors. Below it, sit the Technostructure, Middle Line and Support components, which are responsible for control/standardization, management, and logistics, respectively. The Technostructure component carries out the tasks of standardizing the behavior of other components. Additionally, it is responsible for applying analytical procedures that help the organization to adapt to the environment. Actors joining the apex to the operational core make up the Middle Line. The Support component assists the operational core for non-operational services that are outside the basic flow of operational tasks and procedures. We describe and model examples of structures-in-5 in Section 3. Other proposed patterns are, for example, the matrix, the pyramid, and the lattice (see e.g. [18]). For further information about the patterns we are working on, see [13].

2.2 Strategic Alliances

A strategic alliance links specific facets of the businesses of two or more organizations. At its core, this structure is a trading partnership that enhances the effectiveness of the competitive strategies of the participant organizations by providing for the mutually beneficial trade of technologies, skills, or products based upon them. An alliance can take a variety of forms, ranging from arm's-length contracts to joint ventures, from multinational corporations to university spin-offs, from franchises to equity arrangements (see e.g. [6]. Varied interpretations of the term exist, but a strategic alliance can be defined as possessing simultaneously the following three necessary and sufficient characteristics:

- The two or more organizations that unite to pursue a set of agreed upon goals remain independent subsequent to the formation of the alliance.
- The partner organizations share the benefits of the alliances and control over the performance of assigned tasks.
- The partner organizations contribute on a continuing basis in one or more key strategic areas, e.g., technology, products, and so forth.

For instance, the joint venture pattern involves agreement between two or more intra-industry partners to obtain the benefits of larger scale, partial investment and lower maintenance costs. A specific joint management actor coordinates tasks and manages the sharing of resources between partner actors. Each partner can manage and control itself on a local dimension and interact directly with other partners to exchange resources, such as data and knowledge. However, the strategic operation and coordination of such an organization are only ensured by the joint management actor in which the original actors possess participation equity. We describe and

model examples of joint ventures in Section 3. For further information about the patterns we are working on, see [13].

3 Modeling Organizational Patterns

We have overviewed our organizational patterns in [13]. To model and formalize two of them in more detail, we describe in this section four case studies. The first two examples – FoodCo and Agate Ltd – will be used to illustrate and define formally the structure-in-5, a pattern adopted from organization theory; the others – Airbus and Eurocopter – serve the same purpose for the joint-venture pattern used in strategic alliances.

3.1 Structure-in 5 Pattern

We describe first two case studies from [2]. The presented organizations are modeled in terms of the structure-in-5 pattern. We then formalize the pattern.

FoodCo. FoodCo is a food enterprise based in the East Anglian region of the UK that produces a range of perishable foods for major UK supermarket chains. Its products line ranges from extended to pre-packed vegetables and salads, includes a wide range of sauces, pickles, sandwich toppings, and almost anything made of vegetable that can be sold in jars. There are one farm with a market garden and three factories on the site as well as two warehouses.

The structure of the organization follows the structure-in-5. A *Board* of eight directors forms the *strategic apex*. It is responsible for defining the *general strategy* of the organization: five different chief managers (administration & finance, marketing, planning, operation, and distribution) are required to apply the different aspects of that general strategy in the coordination of the work in the area of their competence: *Policy and Budget* for *Planning and Administration/Finance*, *Production Management* for *Operation,* and *Customer Relationship Management* for *Marketing and Distribution*.

Operation groups production managers and, typically, *coordinates* all managerial aspects of the production. To this end, it relies on *Planning and Administration/ Finances* for dealing with *Planning and Control* aspects of the production and on *Marketing and Distribution* for *Delivery & Sales Logistics*. The *Planning and Administration/Finances* departments constitute the *technostructure* that implements work procedures and policy, management control, planning and budget of the enterprise. This includes the financial strategy, the general administration and human resources management.

The *support* involves the *Marketing and Distribution* staff. *Marketing* coordinates the *customer relationship management* (market study, sales, …), while *Distribution* controls the work at the warehouse, and pick-up & dispatch activities.

Finally, the *operational core* groups line workers, factory and farm foremen that are under the direct supervision of production managers (*middle line*).

Figure 1 models the FoodCo structure-in-5 using the *i** strategic dependency model.

*i** is a modeling framework for early requirements analysis [22], founded on notions such as *actor*, *agent, role, position, goal, softgoal*, *task*, *resource, belief* and different kinds of social *dependency* between actors. Its strategic dependency model describes the network of social dependencies among actors. It is a graph, where each node represents an *actor*, and each link between two actors indicates that one actor depends on another for something in order that the former may attain some goal. A dependency describes an "agreement" (called *dependum*) between two actors: the *depender* and the *dependee*. The *depender* is the depending actor, and the *dependee,* the actor who is depended upon. The type of the dependency describes the nature of the agreement. *Goal* dependencies are used to represent delegation of responsibility for fulfilling a goal; *softgoal* dependencies are similar to goal dependencies, but their fulfillment cannot be defined precisely (for instance, the appreciation is subjective, or the fulfillment can occur only to a given extent); *task* dependencies are used in situations where the dependee is required to perform a given activity; and *resource* dependencies require the dependee to provide a resource to the depender. As shown in Figure 1, actors are represented as circles; dependums – goals, softgoals, tasks and resources – are respectively represented as ovals, clouds, hexagons and rectangles; and dependencies have the form *depender* → *dependum* → *dependee*. We also use later the notion of role (circle with a double line) allowing us to model the same actor assuming different roles

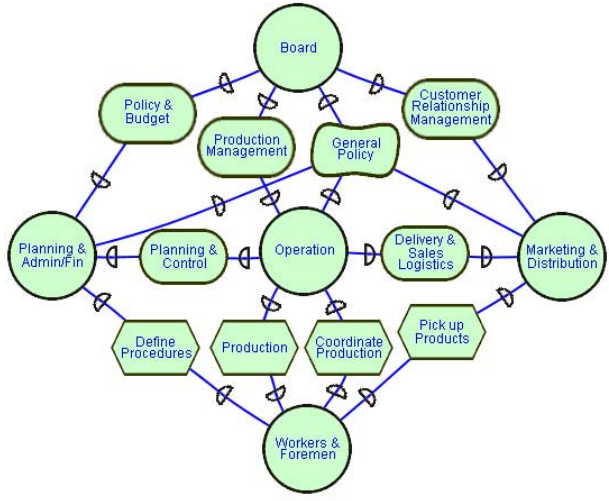

Fig. 1. FoodCo in Structure-in-5

Agate. Agate Ltd is an advertising agency in Birmingham, UK that employs more than fifty staff as described in Figure 2.

Direction
1 Campaign Director
1 Creative Director
1 Administrative Director
1 Finance Director

Edition
2 Editors
4 Copy writers

IT
1 IT manager
1 Network administrator
1 System administrator
1 Analyst
1 Computer technician

Administration
1 Office manager
3 Direction assistants
4 Manager Secretaries
2 Receptionists
2 Clerks/typists
1 Filing clerk

Accounts Edition
1 Accountant manager
1 Credit controller
2 Accounts clerks
2 Purchasing assistants

Campaigns Management
2 Campaign managers
3 Campaign marketers
1 Editor in Chief
1 Creative Manager

Graphics
6 Graphic designers
2 Photographers

Documentation
1 Media librarian
1 Resource librarian
1 Knowledge worker

Fig. 2. Organization of Agate Ltd

The *Direction* – four directors responsible for the main aspects of Agate's *Global Strategy* (advertising campaigns, creative activities, administration and finances) – forms the *strategic apex*. The *middle line* composed of the *Campaigns Management* staff is in charge of *finding* and *coordinating* advertising campaigns (marketing, sales, edition, graphics, budget, …) supported in these tasks by the *Administration and Accounts* and *IT and Documentation* departments.

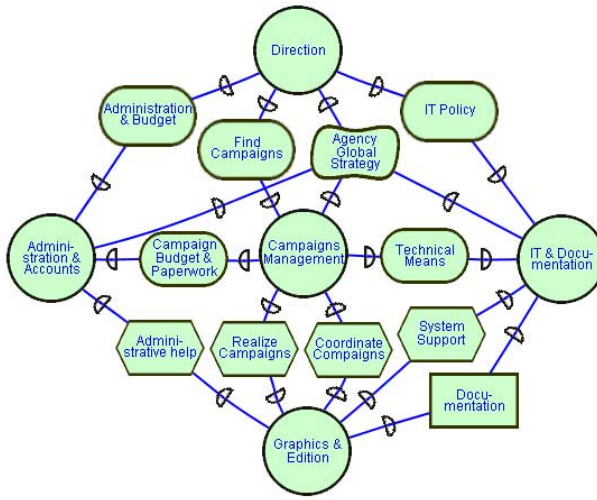

Fig. 3. Agate in Structure-in-5

The *Administration and Accounts* constitutes the *technostructure* handling administrative tasks and policy, paperwork, purchases and budgets. The *support* is constituted of the *IT and Documentation* departments. It defines the *IT policy* of Agate, provides *technical means* required for campaigns and ensures *system support*

as well as information retrieval services (*documentation* resources). The *operational core* includes the *Graphics and Edition* staff in charge of the creative and artistic aspects of *realizing campaigns*: texts, photographs, drawings, layout, design, logos,…

Figure 3 models the structure-in-5 organization of Agate Ltd.

Figure 4 generalizes the structure-in-5 pattern explored in Figures 2 and 3. The pattern must be composed of five actors. Each of them assumes the responsibilities described in Section 2.

Dependencies between the *Strategic Apex* as depender and the *Technostructure*, *Middle Line* and *Support* as dependees must be goal dependencies. A softgoal dependency models the strategic dependence of the *Technostructure*, *Middle Line* and *Support* on the *Strategic Apex*. Relationships between the *Middle Line* and *Technostructure* and *Support* must be of type goal dependencies. The *Operational Core* relies on the *Technostructure* and *Support* through task and resource dependencies. Only task dependencies are permitted between the *Middle Line* (as depender or dependee) and the *Operational Core* (as dependee or depender).

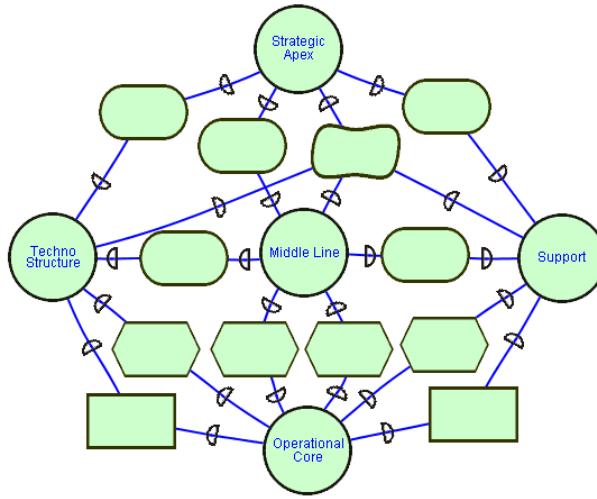

Fig. 4. The Structure-in-5 Pattern

To specify the structure and formal properties of the pattern, we use *Formal Tropos* [8] which offers the primitive concepts of *i** augmented with a rich specification language inspired by KAOS [5]. Formal Tropos offers a textual notation for *i** models and allows one to describe dynamic constraints among the different elements of the specification in a first order linear-time temporal logic. Moreover, *Formal Tropos* has a precise semantics which makes specifications amenable to formal analysis. Basically, *Formal Tropos* conceives three main types of classes: *actor, dependency, and entity*. The attributes of a *Formal Tropos* class denote relationships among different objects being modeled.

In order to express conditions about Strategic Dependency models, such as for instance our organizational patterns, we extend *Formal Tropos* with metaclasses. In particular, we have:

```
Metaclasses
  Actor := Actor name [attributes][creation-properties]
           [invar-properties][actor-goal]
    With subclasses:
      Agent (with attributes occupies: Position, play: Role)
      Position (with attribute cover: Role)
      Role
  Dependency:= Dependency name type mode Depender name Dependee name
               [attributes][creation-properties][invar-properties]
               [fulfill-properties]
  Entity:= Entity name [attribute] [creation-properties]
           [invar-properties]

Classes: Classes are instances of Metaclasses.
```

Part of the Structure-in-5 pattern specification is in the following:

```
Actor StrategicApex
       SoftGoal StrategicManagement
Actor MiddleLine
       Goal ManagementControl
       Task OperationCoordination
Actor Support
       Goal PolicyDefinition
       Goal Logistics
```

The following structural (global) properties must be satisfied by the pattern:

Only one instance of the Strategic Apex

$\forall$ sa1,sa2: StrategicApex $\rightarrow$ (sa1=sa2)

Only softgoal dependencies between the Stategic Apex as dependee and the Middle Line, the Technostructure and the Support as dependers

$\forall$ sa: StrategicApex, ml: Technostructure $\vee$ Middle_Agency $\vee$ Support, dep: Dependency
((dep.dependee=sa $\wedge$ dep.depender=ml) $\rightarrow$ (dep.type=softgoal))

The previous softgoal dependency is fulfilled if and only if all the goal dependencies between the Middle Agency, the Technostructure and the Support as dependers and the Stategic Apex as dependee have been achieved sometimes in the past

$\forall$ sa: StrategicApex, ml: MiddleLine, dep1: Dependency
((dep1.type=softgoal $\wedge$ dep1.dependee=sa $\wedge$ dep1.depender=ml) $\wedge$
($\forall$ dep2: Dependency (dep2.type=goal $\wedge$ (dep2.depender=sa $\wedge$
dep2.dependee = ml $\wedge$ $\blacklozenge$Fulfilled(dep2)))) $\rightarrow$ Fulfilled(dep1))

Only task dependencies between the Middle Line and the Operational Core

$\forall$ml: MiddleLine, oc: OperationalCore, dep: Dependency
((dep.depender=ml $\wedge$ dep.dependee=oc) $\vee$
(dep.depender=oc $\wedge$ dep.dependee=ml)) $\rightarrow$ (dep.type = task))

Only resource or task dependencies between the Technostructure and the Operational Core

$\forall$ts: Technostructure, oc: OperationalCore, dep : Dependency
((dep.depender=ts $\wedge$ dep.dependee=oc) $\rightarrow$
(dep.type=task $\vee$ dep.type=resource))

Only resource or task dependencies between the Support and the Operational Core

```
∀ sp: Support, oc: OperationalCore, dep: Dependency
   ((dep.depender=sp ∧ dep.dependee=oc) →
    (dep.type=task ∨ dep.type=ressource )
```

3.2 Joint-Venture Pattern

We describe here two case studies from [6]. The presented organizations are modeled following the joint venture structure. We then formalize it as an organizational pattern.

Airbus. The Airbus Industrie joint venture coordinates collaborative activities between European aeronautic manufacturers to built and market airbus aircrafts. The joint venture involves four partners: Aerospatiale (France), DASA (Daimler-Benz Aerospace, Germany), British Aerospace (UK) and CASA (Construcciones Aeronauticas SA, Spain). Research, development and production tasks have been distributed among the partners, avoiding any duplication. Aerospatiale is mainly responsible for developing and manufacturing the cockpit of the aircraft and for system integration. DASA develops and manufactures the fuselage, British Aerospace the wings and CASA the tail unit. Final assembly is carried out in Toulouse (France) by Aerospatiale. Unlike production, commercial and decisional activities have not been split between partners. All strategy, marketing, sales and after-sales operations are entrusted to the Airbus Industrie joint venture, which is the only interface with external stakeholders such as customers. To buy an Airbus, or to maintain their fleet, customer airlines could not approach one or other of the partner firms directly, but has to deal with Airbus Industrie. Airbus Industrie, which is a real manufacturing company, defines the alliance's product policy and elaborates the specifications of each model of aircraft to be launched. Airbus defends the point of view and interests of the alliance as a whole, even against the partner companies themselves when the individual goals of the latter conflict with the collective goals of the alliance.

Figure 5 models the organization of the Airbus Industrie joint venture using the *i** strategic dependency model. Airbus assumes two roles: Airbus Industrie and Airbus Joint Venture. *Airbus Industrie* deals with demands from customers, *Customer* depends on it to receive airbus aircrafts or maintenance services. The *Airbus Joint Venture* role ensures the interface for the four partners (*CASA, Aerospatiale*, *British Aerospace* and *DASA*) with *Airbus Industrie* defining Airbus strategic policy, managing conflicts between the four Airbus partners, defending the interests of the whole alliance and defining new aircrafts specifications. *Airbus Joint Venture* coordinates the four partners ensuring that each of them assumes a specific task in the building of Airbus aircrafts: wings building for *British Aerospace*, tail unit building for *CASA*, cockpit building and aircraft assembling for *Aerospace* and fuselage building for *DASA*. Since Aerospatiale assumes two different tasks, it is modeled as two roles: *Aerospatiale Manufacturing* and *Aerospatiale Assembling*. *Aerospatiale Assembling* depends on each of the four partners to receive the parts of the planes.

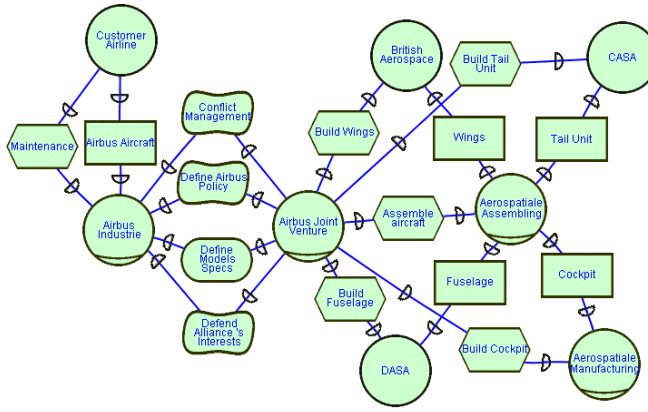

Fig. 5. The Airbus Industrie Joint Venture

Eurocopter. In 1992, Aerospatiale and DASA decided to merge all their helicopter activities within a joint venture Eurocopter. Marketing, sales, R&D, management and production strategies, policies and staff were reorganized and merged immediately; all the helicopter models, irrespective of their origin, were marketed under the Eurocopter name. Eurocopter has inherited helicopter manufacturing and engineering facilities, two in France (La Courneuve and Marignane), one in Germany (Ottobrunn). For political and social reasons, each of them has been specialized rather than closed down to group production together at a single site. The Marignane plant manufactures large helicopters, Ottobrunn produces small helicopters and La Courneuve concentrates on the manufacture of some complex components requiring a specific expertise, such as rotors and blades.

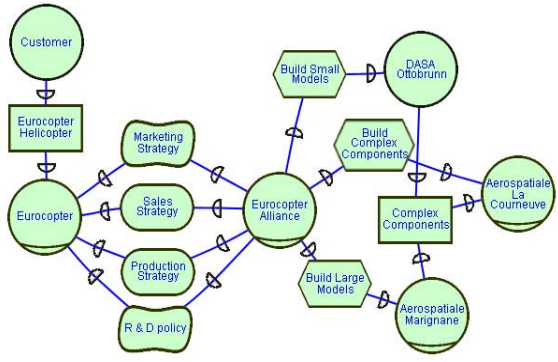

Fig. 6. The Eurocopter Joint Venture

Figure 6 models the organization of the Eurocopter joint venture in *i**. As in the Airbus joint venture, Eurocopter assumes two roles. The *Eurocopter* role handles helicopter orders from customers who depend on it to obtain the machines. It also defines marketing, sales, production and R & D strategies and policy. The *Eurocopter joint venture* role coordinates the manufacturing operations of the two partners – DASA and Aerospatiale – and depends on them for the production of small

helicopters (*DASA Ottobrunn*), large ones (La Courneuve) and complex components (Marignane) such as rotors and blades. Since Aerospatiale assumes two different responsibilities, it is considered two roles: *Aerospatiale Marignane* and *Aerospatiale La Courneuve*. *DASA Ottobrunn* and *Aerospatiale Marignane* depends on *La Courneuve* to be supplied with complex helicopter parts.

Figure 7 generalizes the joint venture model explored in Figures 5 and 6. Partners depend on each other for providing and receiving resources. Operation coordination is ensured by the joint manager actor which depends on partners for the accomplishment of these assigned tasks. The joint manager actor must assume two roles: a private interface role to coordinate partners of the alliance and a public interface role to take strategic decisions, define policy for the private interface and represents the interests of the whole partnership with respect to external stakeholders.

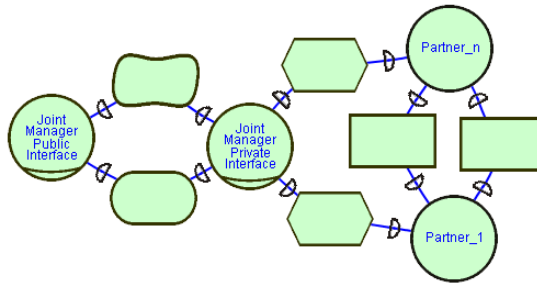

Fig. 7. The Joint Venture Pattern

Part of the Joint Venture pattern specification is in the following:

```
Role JointManagerPrivateInterface
     Goal CoordinatePatterns
Role JointManagerPublicInterface
     Goal TakeStrategicDecision
     SoftGoal RepresentPartnershipInterests
Actor Partner
```

The following structural (global) properties must be satisfied:

Only one instance of the joint manager

```
∀ jmpri1, jmpri2: JointManagerPrivateInterface (jmpri1=jmpri2)
```

Only resource dependencies between partners

```
∀ p1, p2:  Partner, dep: Dependency
 (((dep.depender=p1 ∧ dep.dependee=p2) ∨
  (dep.depender=p2 ∧ dep.dependee=p1)) → (dep.type=resource))
```

Only task dependencies between partners and joint manager, with joint manager as depender

```
∀ jmpri: JointManagerPrivateInterface, p:Partner, dep:Dependency
  ((dep.dependee=p ∧ dep.depender=jmpri) → dep.type=task)
```

Only goal or softgoal dependencies between the joint manager roles

```
∀ jmpri:JointManagerPrivInterf,  jmpui: JointManagerPubInterf,
  dep: Dependency((dep.depender=jmpri ∧ dep.dependee=jmpui) →
    (dep.type=goal ∨ dep.type=softgoal))
```

Partners only have relationships with other partners or the joint manager private interface

```
∀ dep: Dependency,  p1: Partner
 ((dep.depender=p1 ∨ dep.dependee=p1) →
  ((∃p2: Partner(p1≠p2 ∧ (dep.depender=p2 ∨ dep.dependee=p2))) ∨
   (∃jmpi: JointManagerPrivInterf
    ((dep.depender=jmpi ∨ dep.dependee= jmpi))))
```

The joint manager private interface only has relationships with the joint manager public interface or partners

```
∀ dep: Dependency,  jmpi: JointManagerPrivInterf
 ((dep.depender=jpmi ∨ dep.dependee=jpmi) →
  ((∃p: Partner((dep.depender=p∨ dep.dependee=p))) ∨
   (∃jmpui: JointManagerPubInterf (
    (dep.depender=jmpui ∨ dep.dependee= jmpui))))
```

4 Evaluation

Patterns can be compared and evaluated with quality attributes [20]. For instance, the following qualities seem particularly relevant for organizational structures [10]:

Coordinativity. Actors must be able to coordinate with other actors to achieve a common purpose or simply their local goals.

Predictability. Actors can have a high degree of autonomy in the way they undertake action and communication in their domains. It can be then difficult to predict individual characteristics as part of determining the behavior of the system-at-large.

Fallibility-Tolerance. A failure of one actor does not necessarily imply a failure of the whole structure. The structure then needs to check the completeness and the accuracy of data, information and transactions. To prevent failure, different actors can, for instance, implement replicated capabilities.

Adaptability. Actors must be able to adapt to changes in their environment. They may allow changes to the component's communication protocol, dynamic introduction of a new kind of component previously unknown or manipulations of existing actors.

The ***structure-in-5*** improves *coordinativity* among actors by differentiating the data hierarchy - the support actor – from the control hierarchy - supported by the operational core, technostructure, middle agency and strategic apex. The existence of three different levels of abstraction (1 – Operational Core; 2 – Technostructure, Middle Line and Support; 3 – Strategic Apex) addresses the need for managing *predictability*. Besides, higher levels are more abstract than lower levels: lower levels

only involve resources and task dependencies while higher ones propose intentional (goals and softgoals) relationships. Checks and control mechanisms can be integrated at different levels of abstraction assuming redundancy from different perspectives and increase considerably *failability-tolerance*. Since the structure-in-5 separates data and control hierarchies, integrity of these two hierarchies can also be verified independently. The structure-in-5 separates independently the typical components of an organization, isolating them from each other and allowing then dynamic *adaptability*. But since it is restricted to no more than 5 major components, more refinement has to take place inside the components.

The ***joint venture*** supports *coordinativity* in the sense that each partner actor interacts via the joint manager for strategic decisions. Partners indicate their interest, and the joint manager either returns them the strategic information immediately or mediates the request to some other partners. However, since partners are usually heterogeneous, it could be a drawback to define a common interaction background. The central position and role of the joint manager is a means for resolving conflicts and preventing *unpredictability*. Through its joint manager, the joint-venture proposes a central communication controller. It is less clear how the joint venture style addresses *fallibility-tolerance*, notably *reliability*. However, exceptions, supervision, and monitoring can improve its overall score with respect to these qualities. Manipulation of partners can be done easily to *adapt* the structure by registering new ones to the joint manager. However, since partners can also exchange resources directly with each other, existing dependencies should be updated as well. The joint manager cannot be removed due to its central position.

Table 1 summarizes the strengths and weaknesses of the reviewed styles.

Table 1. Strengths and Weaknesses of some Organizational Patterns

	Coord.	Predict.	Failab-Tol.	Adapt.
S-in-5	+	+	++	+-
Joint-Vent.	+-	+	+-	+-

A more precise and systematic analysis of these quality attributes can be done with goal-oriented frameworks such as KAOS [5] or the NFR framework [4]. In the NFR framework, qualities are represented as *softgoals*. Analyzing them amounts to a means-ends decomposition of softgoals into more fine-grained subgoals. Each pattern contributes positively/negatively to some of the identified subgoals. The overall evaluation of a pattern with respect to a quality is arrived at by propagating contributions from bottom towards the top of a softgoal dependency graph. A partial example of such a graph is shown in Figure 8.

The analysis resulting in a softgoal dependency graph is intended to make explicit the space of alternatives for fulfilling a top-level attribute. The organizational patterns are represented as operationalized attributes (saying, roughly, "fulfilled by the pattern *structure-in-5 / joint-venture*").

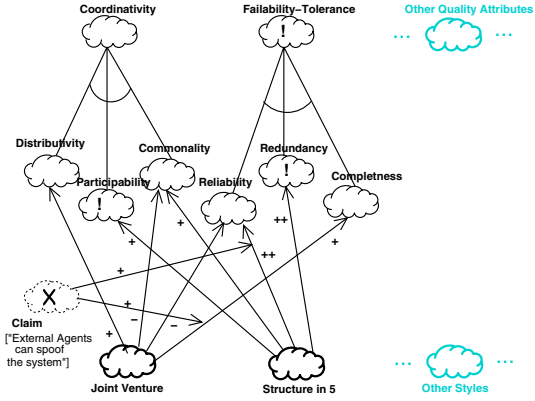

Fig. 8. Partial Evaluation for Organizational Styles

The evaluation process is defined in terms of contribution relationships from softgoals to softgoals, labeled "+", "++", "-", "--" that mean respectively *partially satisfied*, *satisfied*, *partially denied* and *denied*. Design rationale is represented by claims drawn as dashed clouds. Such features make it possible for domain characteristics such as priorities to be considered and properly reflected in the decision-making process. Exclamation marks are used to mark priority attributes while a check-mark "✓ " indicates a fulfilled softgoal and a cross " ✕" labels a denied one.

Relationships types (AND, OR, ++, +, -, and --) between quality attributes are formalized to offer a tractable proof procedure. To each quality attribute we associate two different variables: S for satisfiability and D for deniability. These variables can assume three possible values: *null* (–) , *partial* (*p*), and *total* (*t*). For instance, when S=*t*, an attribute is totally satisfied, when S=*p* it is partially satisfied, and when S=– there is no evidence to say something about its satisfiability (analogously for D).

S and D are not required to be logically exclusive since there may be contradictory contributions, e.g., for a particular pattern, a softgoal is satisfied and partially denied at the same time. Table 2 shows propagation rules for ++, +, -, and -- relationships with respect to satisfiability (S). Notice that the null value does not produce any effect in the propagation. A dual table is given for the deniability and the partial deniability.

Table 2. Propagation rules for Satisfiability

S	**++**	**+**	**-**	**--**
t	**S**=*t*	**S**=*p*	**D**=*p*	**D**=*t*
p	**S**=*p*	**S**=*p*	**D**=*p*	**D**=*p*

Under the assumption that $- < p < t$, we use min-value and max-value functions respectively for AND and OR relationships. The basic algorithm for the labels propagation is presented in Figure 9.

Initially, all the nodes are initialized with the available evidence, a null value is assigned to the nodes for which we do not have evidence. At each step the value of

the two variables S and D of each node is calculated using the nodes' value of the previous step. The final value for D and S is given by the maximum value of all contributions of the incoming relations. The algorithm terminates when an iteration adds no new values to any the variables of any node of the graph. The use of maximum value function guarantees the termination of the algorithm. Further details about this propagation algorithm are presented in [11].

```
1   Initialize NODES'
2   do
3   NODES ← NODES'
4  foreach node n_i
5    foreach incoming relation A_ij
6      D_j ← ComputeD(A_ij)
7      S_j ← ComputeS(A_ij)
8   n_i.D' ← Max_j(D_j)
9   n_i.S' ← Max_j(S_j)
10  while(NODES≠NODES')
```

Fig. 9. Basic propagation algorithm

5 Conclusions

Modelers need to rely on patterns, styles, and idioms, to build their models, whatever the purpose. We argue that, as with other phases of software development, early requirements analysis can be facilitated by the adoption of organizational patterns. This paper focuses on two such patterns and studies them in detail, through examples, a formalization using Formal Tropos, and an evaluation with respect to desirable attributes.

There have been many proposals for software patterns since the original work on design patterns [9]. Some of this work focuses on requirements patterns. For example, [16] proposes a set of requirements patterns for embedded software systems. These patterns are represented in UML and cover both structural and behavioral aspects of a requirements specification. Along similar lines, [7] proposes some general patterns in UML. In both cases, the focus is on late requirements, and the modeling language used is UML. On a different path, [12] proposes a systematic approach for evaluating design patterns with respect to non-functional requirements (e.g., security, performance, reliability). Our approach differs from this work primarily in the fact that our proposal is founded on ideas from Organization Theory and Strategic Alliances literature. In [13, 15, 10], we have already described organizational patterns but to be used for designing multi-agent system architectures. Considering real world organizations as a metaphor, systems involving many software actors, such as multi-agent systems could benefit from the same organizational models. In the present paper, we have focused on patterns for modeling organizational settings, rather than software systems and emphasized the need for organizational abstractions to better match the operational environment of the system-to-be during early requirements analysis.

References

[1] A. I. Anton, "Goal-Based Requirements Analysis", *Proceedings of the 2nd Int. Conf. On Requirements Analysis, ICRE'96*, 1996, pp.136–144.

[2] S. Bennett, S. McRobb, and R. Farmer. *Object-Oriented Systems Analysis and Design – using UML.* McGraw Hill, 1999.

[3] J. Castro, M. Kolp and J. Mylopoulos. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project". In *Information Systems* (27), Elsevier, Amsterdam, The Netherlands, 2002.

[4] L. K. Chung, B. A. Nixon, E. Yu and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, 2000.

[5] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal–directed Requirements Acquisition", *Science of Computer Programming, 20*, 1993, pp. 3–50.

[6] P. Dussauge and B. Garrette, *Cooperative Strategy: Competing Successfully Through Strategic Alliances*, Wiley and Sons, 1999.

[7] Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.

[8] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. "Model Checking Early Requirements Specification in Tropos". In *Proc. of the 5th Int. Symposium on Requirements Engineering, RE'01*, Toronto, Canada, Aug. 2001.

[9] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of reusable Object-Oriented Software*, Addison-Wesley, 1995.

[10] P. Giorgini, M. Kolp, and J. Mylopoulos. "Multi-Agent and Software Architecture: A Comparative Case Study". In *Proceedings of the 3rd International Workshop on Agent Software Engineering* (AOSE'02), Bologna, Italy, July 2002.

[11] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. In Proceedings of the 21st International Conference on Conceptual Modeling (ER02), LNCS 2503 Springer Verlag. Tampere, Finland, October, 2002,

[12] D. Gross and E. Yu, "From Non-Functional Requirements to Design Through Patterns", *Requirements Engineering 6(1),* 18–36, 2002.

[13] M. Kolp, P. Giorgini and J. Mylopoulos. "A Goal-Based Organizational Perspective on Multi-Agents Architectures". In *Proc. of the 8th Int. Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages* (ATAL2001), Seattle, USA, August 2001.

[14] M. Kolp, P. Giorgini, and J. Mylopoulos. "Information Systems Development through Social Structures". In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering* (SEKE'02), Ishia, Italy, July 2002.

[15] M. Kolp, P. Giorgini, and J. Mylopoulos. "Organizational Multi-Agent Architecture: A Mobile Robot Example". In *Proceedings of the 1st International Conference on Autonomous Agent and Multi Agent Systems* (AAMAS'02), Bologna, Italy, July 2002.

[16] Konrad, S., and Cheng, B., "Requirements Patterns for Embedded Systems", *Proceedings of the Tenth IEEE Joint International Requirements Engineering Conference* (RE'02), Essen, September 2002.

[17] H. Mintzberg, *Structure in fives:designing effective organizations*, Prentice-Hall, 1992

[18] J. Morabito, I. Sack and A. Bhate. *Organization Modeling : Innovative Architectures for the 21st Century*, Upper Saddle River, N.J., Prentice Hall PTR, 1999.

[19] W. R. Scott. *Organizations: rational, natural, and open systems*, Prentice Hall, 1998.

[20] Shaw, M., and Garlan, D. *Software Architecture: Perspectives on an Emerging Discipline*, Upper Saddle River, N.J., Prentice Hall, 1996.

[21] M.Y. Yoshino and U. Srinivasa Rangan. *Strategic alliances: an entrepreneurial approach to globalization*, Boston, Mass., Harvard Business School Press, 1995.

[22] E. Yu. Modeling *Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.