

ANALYSIS STYLES FOR REQUIREMENTS ENGINEERING: AN ORGANIZATIONAL PERSPECTIVE

MANUEL KOLP* and T. TUNG DO

*University of Louvain,
ISYS — Information Systems Research Unit,
Place des Doyens, 1, 1348, Louvain-La-Neuve, Belgium
E-mail: {kolp, do}@isys.ucl.ac.be*

STÉPHANE FAULKNER

*University of Namur,
Department of Management Sciences,
Rempart de la Vierge, 8, 5000 Namur, Belgium
E-mail: Stephane.Faulkner@fundp.ac.be*

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

Keywords: Organizational styles, i^* , Tropos, requirements engineering, organizational modeling.

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, 6, 40]). Such models are founded on primitive concepts such as those of actor and goal. This chapter focuses on the definition of a set of organizational styles 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.

The research reported in this chapter is being conducted within the context of the Tropos project [3, 4, 30], 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

*Corresponding author.

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 [4]. The present work continues the research in progress about social abstractions for the Tropos methodology. In [23], we have detailed a social ontology for Tropos to consider information systems with social structures all along the development life cycle. In [14,22,24], we have described how to use this Tropos social ontology to design multi-agent systems architectures, notably for e-business applications [7]. As a matter of fact, multi-agent systems can be considered structured societies of coordinated autonomous agents. In the present chapter, which is an extended and revised version of [25], we emphasize the use of organizational styles based on organization theory and 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, . . .) Throughout the chapter, we use i^* [40] as the modeling framework in terms of which the proposed styles are presented and accounted for.

The chapter 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 styles — the structure-in-5 and the joint venture — based on real world examples of organizations. These styles are modeled in terms of social and intentional concepts using the i^* framework and the Formal Tropos specification language. Section 4 identifies a set of desirable non-functional requirements for evaluating these styles and presents a framework to select a style with respect to these identified requirements. Section 5 overviews the *Tropos* methodology. Finally, Sec. 6 summarizes the contributions of the chapter 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. [28, 32, 39]), that describes the structure and design of an organization and *Strategic Alliances* (e.g. [10, 17, 29, 33]), that model the strategic

collaborations of independent organizational stakeholders who have agreed to pursue a set of agreed upon business goals.

Both disciplines aim to identify and study organizational patterns. These are not just modeling abstractions or structures, rather they can be seen, felt, handled, and operated upon. They have a manifest form and lie in the objective domain of reality as part of the concrete world. A pattern is however not solely a set of execution behaviors. Rather, it exists in various forms at every stage of crystallization (e.g. specification), and at every level of granularity in the organization. The more manifest is its representation, the more the pattern emerges and becomes recognizable, whether at a high or low level of granularity.

At the lowest level of granularity, we find *information patterns* and *service patterns* that represent the “nitty-gritty” of business that an organization must deal with on a day-to-day basis. When we move to an upper level, we find *business patterns* — the mix of products and markets that flows from organizational styles. The highest level of granularity is the *organizational styles* that addresses the mix of socio-technical context and organizational constructs: they are manifestation of organization invariants, layers of organizational constructs, organization molecules, and complex arrangements of molecules, the collection of which constitutes organizational structures.

Many organizational styles are fully formed patterns with definite characteristics as the ones we present in the rest of this section. In contrast, many other organizational styles are not very explicit, that is, not easily specified, operationalized, and measured. Michael Porter’s generic strategies [31] are examples of such patterns. Each strategy type is characterized by general properties that distinguish one strategy from another. For the most part, however, the distinguishing characteristics of each style are only partially described in terms of an organization’s architecture.

In this chapter, we are interested to identify and use, for requirements engineering, organizational styles that have already been well-understood and precisely defined in organizational theories. Our purpose is not to categorize them exhaustively nor to study them on a managerial point of view. The following sections will thus only insist on styles that have been found, due to their nature, interesting candidates also considering the fact that they have been studied in great detail in the organizational literature and presented as fully formed patterns.

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” [29]. Organization theory is the discipline that studies both structure and design in such social entities. Structure deals with the descriptive aspects while design refers to the prescriptive aspects of a social entity. Organization theory 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, since Adam Smith, schools of organization theory have proposed models and patterns to try to find and formalize recurring organizational structures and behaviors.

In the following, we briefly present organizational styles identified in organization theory. The structure-in-5 will be studied in detail in Sec. 3.

The Structure-in-5. An organization can be considered an aggregate of five sub-structures, as proposed by Mintzberg [28]. At the base level sits the *Operational Core* which carries out the basic tasks and procedures directly linked to the production of products and services (acquisition of inputs, transformation of inputs into outputs, distribution of outputs). At the top lies the *Strategic Apex* which makes executive decisions ensuring that the organization fulfils its mission in an effective way and defines the overall strategy of the organization in its environment. The *Middle Line* establishes a hierarchy of authority between the Strategic Apex and the Operational Core. It consists of managers responsible for supervising and coordinating the activities of the Operational Core. The *Technostructure* and the *Support* are separated from the main line of authority and influence the operating core only indirectly. The Technostructure serves the organization by making the work of others more effective, typically by standardizing work processes, outputs, and skills. It is also in charge of applying analytical procedures to adapt the organization to its operational environment. The Support provides specialized services, at various levels of the hierarchy, outside the basic operating work flow (e.g. legal counsel, R&D, payroll, cafeteria). We describe and model examples of structures-in-5 in Sec. 3.

The pyramid style is the well-know hierarchical authority structure. Actors at lower levels depend on those at higher levels. The crucial mechanism is the direct supervision from the Apex. Managers and supervisors at intermediate levels only route strategic decisions and authority from the Apex to the operating (low) level. They can coordinate behaviors or take decisions by their own, but only at a local level.

The chain of values merges, backward or forward, several actors engaged in achieving or realizing related goals or tasks at different stages of a supply or production process. Participants who act as intermediaries, add value at each step of the chain. For instance, for the domain of goods distribution, providers are expected to supply quality products, wholesalers are responsible for ensuring their massive exposure, while retailers take care of the direct delivery to the consumers.

The matrix proposes a multiple command structure: vertical and horizontal channels of information and authority operate simultaneously. The principle of the unity of command is set aside, and competing bases of authority are allowed to jointly govern the work flow. The vertical lines are typically those of functional departments that operate as “home bases” for all participants, the horizontal lines represents project groups or geographical arenas where managers combine and coordinate the services of the functional specialists around particular projects or areas.

The bidding style involves competitiveness mechanisms, and actors behave as if they were taking part in an auction. An auctioneer actor runs the show, advertises

the auction issued by the auction issuer, receives bids from bidder actors and ensures communication and feedback with the auction issuer who is responsible for issuing the bidding.

2.2. *Strategic alliances*

A strategic alliance links specific facets 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. 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.

In the following, we briefly present organizational styles identified in Strategic Alliances. The joint venture will be studied in details in Sec. 3.

The joint venture style 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, and its actors on a global dimension, are only ensured by the joint management actor in which the original actors possess equity participations. We describe and model examples of joint ventures in Sec. 3.

The arm's-length style implies agreements between independent and competitive, but partner actors. Partners keep their autonomy and independence but act and put their resources and knowledge together to accomplish precise common goals. No authority is lost, or delegated from one collaborator to another.

The hierarchical contracting style identifies coordinating mechanisms that combine arm's-length agreement features with aspects of pyramidal authority. Coordination mechanisms developed for arm's-length (independent) characteristics involve a variety of negotiators, mediators and observers at different levels handling conditional clauses to monitor and manage possible contingencies, negotiate

and resolve conflicts and finally deliberate and take decisions. Hierarchical relationships, from the executive apex to the arm's-length contractors restrict autonomy and underlie a cooperative venture between the parties.

The co-optation style involves the incorporation of representatives of external systems into the decision-making or advisory structure and behavior of an initiating organization. By co-opting representatives of external systems, organizations are, in effect, trading confidentiality and authority for resource, knowledge assets and support. The initiating system has to come to terms with the contractors for what is being done on its behalf; and each co-opted actor has to reconcile and adjust its own views with the policy of the system it has to communicate.

3. Modeling Organizational Styles

We will define an organizational style as a metaclass of organizational structures offering a set of design parameters to coordinate the assignment of organizational objectives and processes, thereby affecting how the organization itself functions. Design parameters include, among others, goal and task assignments, standardization, supervision and control dependencies and strategy definitions.

This section describes two of the organizational styles presented in Sec. 2: the structure-in-5 and the joint-venture.

3.1. *Structure-in-5*

To detail and specify the structure-in-5 as an organizational style, this section presents three case studies: Agate [2], Volvo Trucks Corporation [27] and GMT [16]. They will serve to propose a model and a semi-formal specification of the structure-in-5.

Agate is an advertising agency located in England that employs about seventy-five staff, as detailed in Table 1.

Table 1.

<i>Direction</i>	<i>Edition</i>	<i>IT</i>
1 Campaigns Director	3 Editors	1 IT manager
1 Creative Director	7 Copywriters	2 Network administrator
1 Administrative Director		2 System administrator
1 Finance Director	<i>Documentation</i>	2 Analyst
	2 Media librarian	2 Computer technician
<i>Campaigns Management</i>	2 Resource librarian	
3 Campaign managers	1 Knowledge worker	<i>Accounts</i>
4 Campaign marketers		2 Accountant manager
2 Editor-in-Chief	<i>Administration</i>	1 Credit controller
2 Creative Manager	4 Direction assistants	2 Accounts clerks
	6 Manager Secretaries	2 Purchasing assistants
<i>Graphics</i>	2 Receptionists	
10 Graphic designers	2 Clerks/typists	
4 Photographers	1 Filing clerk	

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, . . .). It is supported in these tasks by the *Administration and Accounts* and *IT and Documentation* departments. The *Administration and Accounts* constitutes the *Technostructure* handling administrative tasks and policy, paperwork, purchases and budgets. The *Support* groups the *IT and Documentation* departments. It defines the *IT policy* of Agate, provides *technical means* required for the management of campaigns, and ensures services for *system support* as well as information retrieval (*documentation* resources). The *Operational Core* includes the *Graphics and Edition* staff in charge of the creative and artistic aspects of *realizing campaign* (texts, photographs, drawings, layout, design, logos).

Figure 1 models Agate in structure-in-5 using the *i** strategic dependency model. *i** is a modeling framework for organizational modeling [40], which offers goal- and actor-based notions such as *actor*, *agent*, *role*, *position*, *goal*, *softgoal*, *task*,

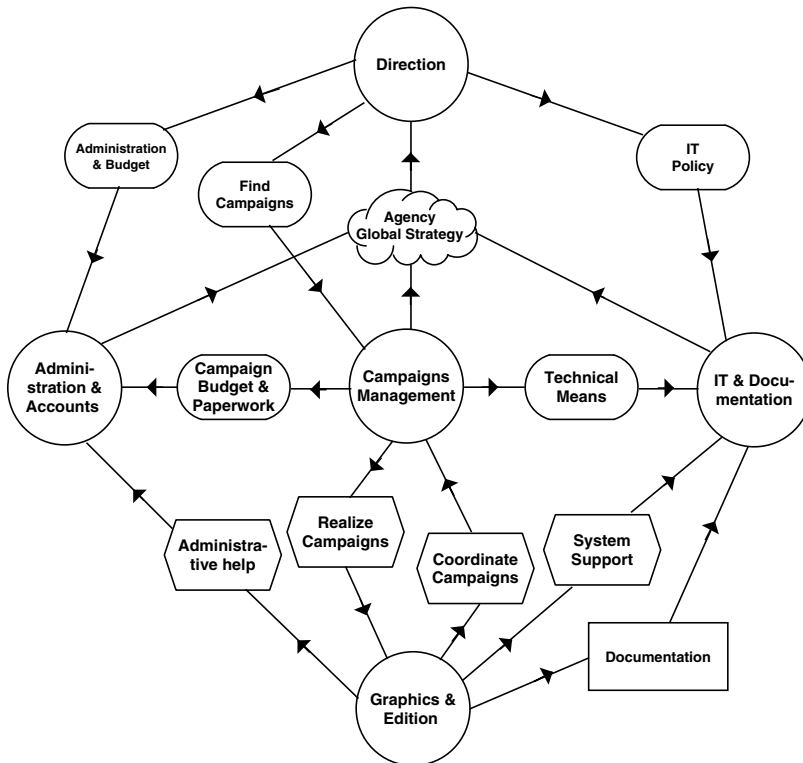


Fig. 1. Agate as a Structure-in-5.

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 the other for some goal to be attained. A dependency describes an “agreement” (called *dependum*) between two actors: the *dependor* and the *dependee*. The *dependor* 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 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 fulfillment is obtained 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 dependor. As shown in Fig. 1, actors are represented as circles; dependums — goals, softgoals, tasks and resources — are represented as ovals, clouds, hexagons and rectangles; respectively, and dependencies have the form *dependor* → *dependum* → *dependee*.

Volvo Trucks Corporation (VTC) is a subsidiary company of AB Volvo, the automobile manufacturer. The VTC distributive network is segmented into eight commercial divisions responsible for the production and commercialization of trucks. These divisions take charge of the coordination of assemblage factories attached to a geographical zone and supervise dealers’ networks. Dealers are responsible for the elaboration of sales local systems that correspond to customers’ needs. Commercial divisions coordinate sales at the national or international level.

The product development division is in charge of the design of new trucks and components models. It also provides dealers with technological training.

The marketing communication is under the supervision of a single entity. The objective is double: to increase the coherence of the sale supports and promotion in Europe; and to improve the efficiency and the profitability of these supports.

The audit division constitutes an independent control entity that has for objective to define administrative procedures for dealers and supervise their implementation. Finally, the financial division that collaborates with the audit division provides the financial forecasts required by the executive committee. It also determines the budget of commercial divisions.

Figure 2 models the VTC structure-in-5 using the i^* strategic dependency model. The executive committee composed of three administrators responsible for the main aspects of VTC’s *General Strategy* form the *Strategic Apex*. The *Middle Line* composed of the different *Commercial Divisions* implements the *Distributive Network Management*. It coordinates the *Operational Core* (i.e., *Dealers* and *Assemblage* networks). *R&D* and *Information Divisions* constitute the *Technostructure*. *R&D* is in charge of the *Design* of the new models and the *Training* for the *Operational Core*. The *Information* department defines VTC *Communication Policy* as well as the public *Image* of the firm. It also provides the *Operational Core* with *Sales* and *Promotion* support. The *Support* groups the *Audit* and *Financial*

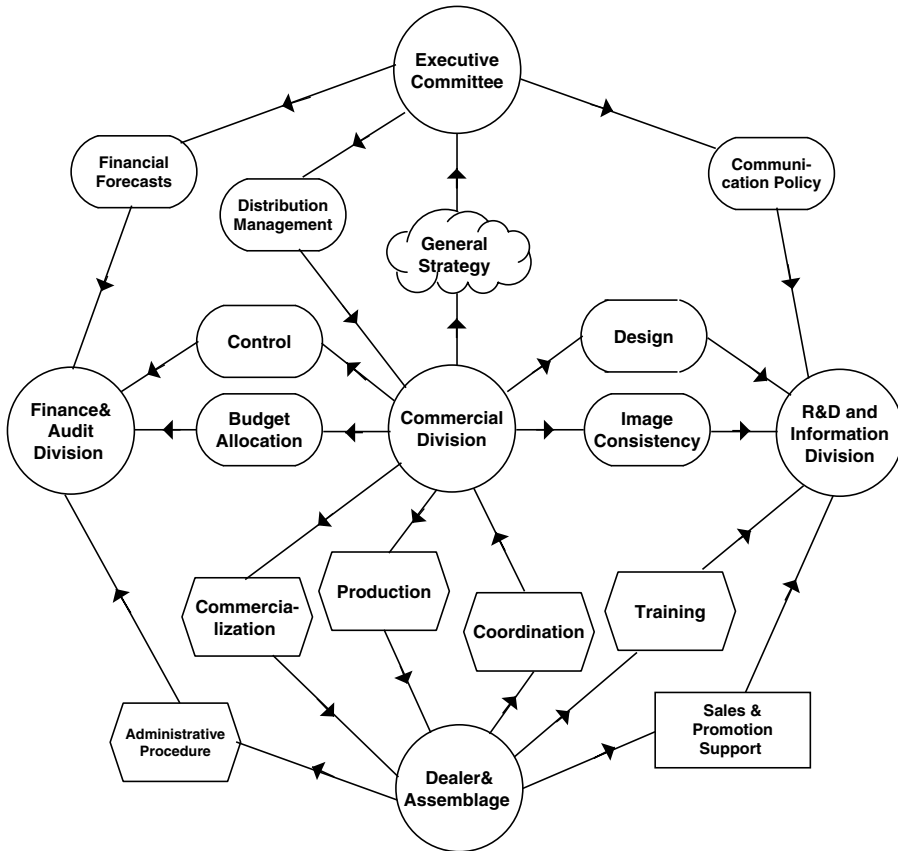


Fig. 2. Volvo trucks corporation in structure-in-5.

Divisions. The *Audit Division* defines the *Administrative Procedures* and controls how these are implemented. The *Financial Division* is in charge of the *Financial Forecasts* and defines the *Budget Allocations* for *Commercial Divisions*.

GMT is a company specialized in telecom services in Belgium composed of 50 employees. Its lines of products and services range from phones & fax, conferencing, line solutions, internet & e-business, mobile solutions, and voice & data management. As shown in Fig. 3, the structure of the commercial organization follows the structure-in-5. An *Executive Committee* constitutes the *Strategic Apex*. It is responsible for defining the *general strategy* of the organization. Five chief managers (*finances, operations, divisions management, marketing, and R&D*) apply the specific aspects of the *general strategy* in the area of their competence: *Finances & Operations* is in charge of *Budget and Sales Planning & Control*, *Divisions Management* is responsible for *Implementing Sales Strategy*, and *Marketing and R&D* define *Sales Policy* and *Technological Policy*.

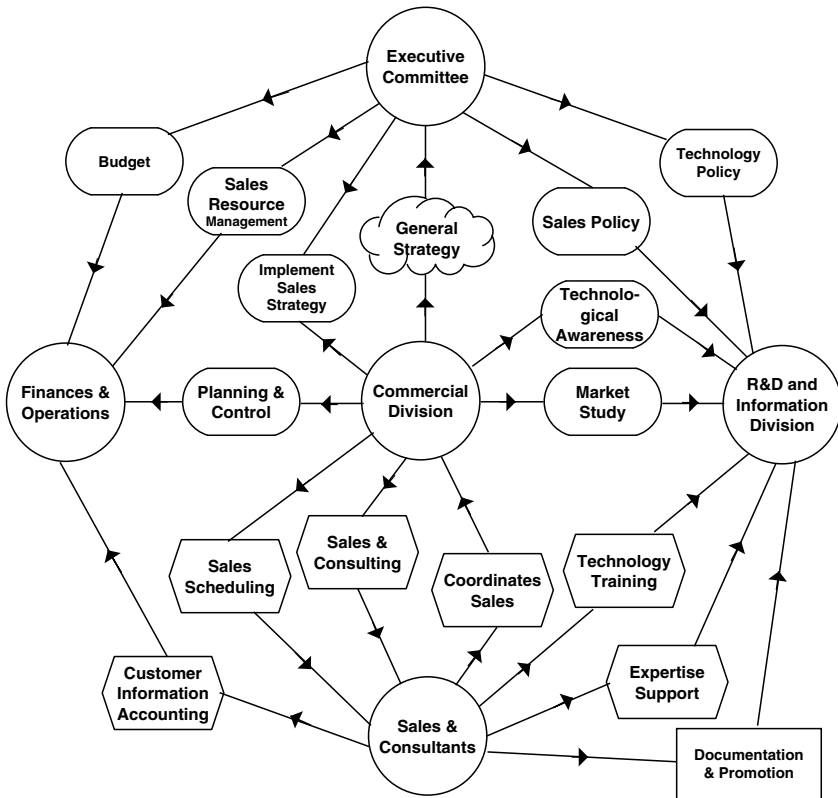


Fig. 3. GMT's sales organization as a structure-in-5.

The *Divisions Management* groups managers that coordinate all managerial aspects of product and service sales. It relies on *Finance & Operations* for handling *Planning* and *Control* of products and services, it depends on *Marketing* for accurate *Market Studies* and on R&D for *Technological Awareness*.

The *Finances & Operations* departments constitute the *technostructure* in charge of management *control* (financial and quality audit) and sales *planning* including *scheduling* and *resource management*.

The *Support* involves the staff of *Marketing* and *R&D*. Both departments jointly define and support the *Sales Policy*. The *Marketing* department coordinates *Market Studies* (customer positionment and segmentation, pricing, sales incentive, ...) and provides the *Operational Core* with *Documentation* and *Promotion* services. The *R&D* staff is responsible for defining the technological policy such as *technological awareness services*. It also assists *Sales people* and *Consultants* with *Expertise Support* and *Technology Training*.

Finally, the *Operational Core* groups the *Sales people* and *Line consultants* under the supervision and coordination of *Divisions Managers*. They are in charge of selling products and services to actual and potential customers.

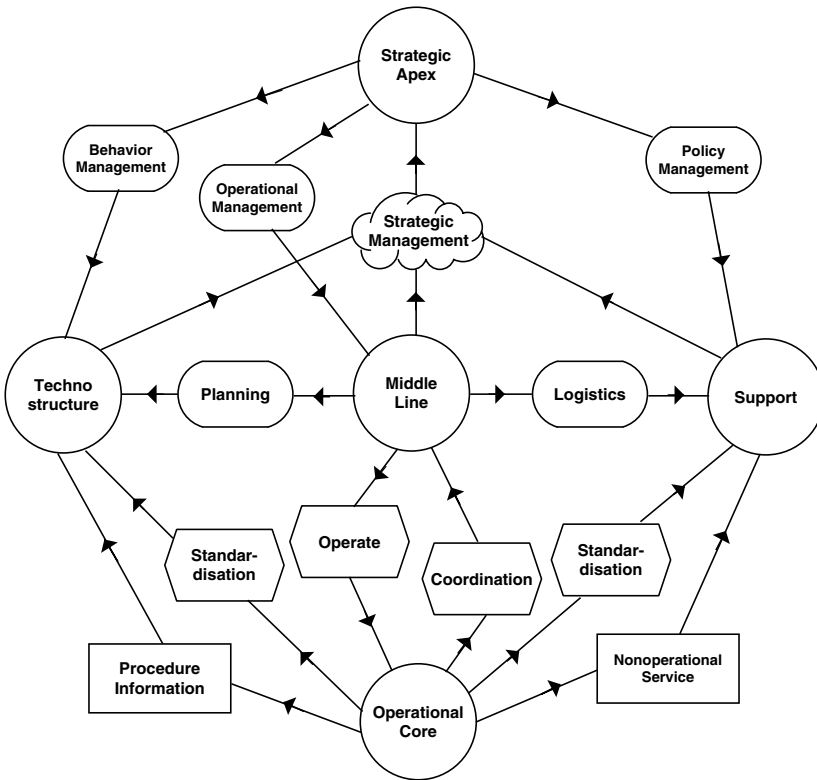


Fig. 4. The structure-in-5 style.

Figure 4 abstracts the structures explored in the case studies of Figs. 1–3 as a structure-in-5 style composed of five actors. The case studies also suggested a number of constraints to supplement the basic style:

- the dependencies between the *Strategic Apex* as depender and the *Technostructure*, *Middle Line* and *Support* as dependees must be of type goal;
- a softgoal dependency models the strategic dependence of the *Technostructure*, *Middle Line* and *Support* on the *Strategic Apex*;
- the relationships between the *Middle Line* and *Technostructure* and *Support* must be of goal dependencies;
- the *Operational Core* relies on the *Technostructure* and *Support* through task and resource dependencies; and
- only task dependencies are permitted between the *Middle Line* (as depender or dependee) and the *Operational Core* (as dependee or depender).

To specify the formal properties of the style, we use *Formal Tropos* [12], which extends the primitives of i^* with a formal language comparable to that of KAOS [6]. Constraints on i^* specifications are thus formalized in a first-order linear-time

temporal logic. *Formal Tropos* provides three basic types of metaclasses: *actor*, *dependency*, and *entity* [14]. The attributes of a *Formal Tropos* class denote relationships among different objects being modeled.

Metaclasses

Actor := **Actor** name [attributes] [creation-properties] [invar-properties][actor-goal]

With subclasses:

Agent(with attributes occupies: Position, play: Role)

Position(with attributes 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]

Actor-Goal := (**Goal|Softgoal**) name mode **FulFillment**(actor-fulfill-property)

Classes: Classes are instances of Metaclasses.

In Formal Tropos, constraints on the lifetime of the (meta)class instances are given in a first-order linear-time temporal logic (see [12] for more details). Special predicates can appear in the temporal logic formulas: predicate *JustCreated*(x) holds in a state if element x exists in this state but not in the previous one; predicate *Fulfilled*(x) holds if x has been fulfilled; and predicate *JustFulfilled*(x) holds if *Fulfilled*(x) holds in this state, but not in the previous one.

In the following, we only present some specifications for the *Strategic Management* and *Operational Management* dependencies.

Actor StrategicApex

Actor MiddleLine

Actor Support

Actor Technostructure

Actor OperationalCore

Dependency StrategicManagement

Type SoftGoal

Depender te: Technostructure, ml: MiddleLine, su: Support

Dependee sa: StrategicApex

Invariant

$\forall dep : Dependency (JustCreated(dep) \rightarrow Consistent(self, dep))$

$\forall ag : Actor - Goal (JustCreated(ag) \rightarrow Consistent(self, ag))$

Fulfillment

$\forall dep : Dependency (dep.type = goal \wedge dep.depender = sa \wedge (dep.dependee = te \vee dep.dependee = ml \vee dep.dependee = su)) \wedge Fulfilled(self) \rightarrow \blacklozenge Fulfilled(dep)$

[Invariant properties specify, respectively, that the strategic management softgoal must be consistent with any other dependency of the organization and with any other goal of the actors in the organization. The predicate *Consistent* depends on the particular organization we are considering and it is specified in terms of goals' properties to be satisfied. The fulfillment of the dependency necessarily implies that the goal dependencies between the Middle Line, the Technostructure, and

the Support as dependees, and the Strategic Apex as depender have been achieved some time in the past]

Dependency OperationalManagement

Type Goal

Mode achieve

Depender sa: StrategicApex

Dependee ml: MiddleLine

Invariant

$Consistent(self, StrategicManagement)$

$\exists c : Coordination (c.type = task \wedge c.dependee = ml \wedge c.depender =$
 $OperationalCore \wedge ImplementedBy(self, c))$

Fulfillment

$\forall ts : Technostructure, dep : Dependency (dep.type = goal \wedge$
 $dep.depender = ml \wedge dep.dependee = ts) \wedge Fulfilled(self))$
 $\rightarrow \blacklozenge Fulfilled(dep)$

[The fulfillment of the Operational management goal implies that all goal dependencies between the Middle Line as depender and the Technostructure as dependee have been achieved some time in the past. Invariant properties specifies that Operational Management goal has to be consistent with Starategic Management softgoal and that there exists a coordination task (a task dependency between MiddleLine and Operational Core) that implement (ImplementedBy) the OperationalManagaemnt goal.]

In addition, the following structural (global) properties must be satisfied for the structure-in-5 style:

- $\forall inst1, inst2 : StrategicApex \rightarrow inst1 = inst2$

[There is a single instance of the Strategic Apex (the same constraint also holds for the Middle Line, the Technostructure, the Support and the Operational Core)]

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

[Only softgoal dependencies are permitted between the Strategic Apex as dependee and the Technostructure, the Middle Line, and the Support as dependers]

- $\forall sa : StrategicApex, te : Technostructure, ml : MiddleLine,$
 $su : Support, dep : Dependency :$
 $(dep.depender = sa \wedge (dep.dependee = te \vee dep.dependee =$
 $ml \vee dep.dependee = su) \rightarrow dep.type = goal)$

[Only goal dependencies are permitted between the Technostructure, the Middle Line, and the Support as dependee, and the Statgic Apex as depender]

- $\forall su : Support, ml : MiddleLine, dep : Dependency$
 $((dep.dependee = su \wedge dep.depender = ml) \rightarrow dep.type = goal)$

[Only task dependencies are permitted between the Middle Line and the Operational Core]

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

[Only resource or task dependencies are permitted between the Technostructure and the Operational Core (the same constraint also holds for the Support)]

- $\forall a : Actor, ml : MiddleLine,$
 $(\exists dep : Dependency(dep.depender = a \wedge dep.dependee =$
 $ml) \vee (dep.dependee = a \wedge dep.depender = ml) \rightarrow$
 $((\exists sa : StrategicApex(a = sa)) \vee (\exists su : Support(a = su)) \vee$
 $(\exists te : Technostructure(a = te)) \vee (\exists op : OperationalCore$
 $(a = op))$

[No dependency is permitted between an external actor and the Middle Line (the same constraint also holds for the Operational Core)]

This specification can be used to establish that a certain i^* model does constitute an instance of the structure-in-5 style. For example, the i^* model of Fig. 1 can be shown to be such an instance, in which the actors are instances of the structure-in-5 actor classes (e.g. *Direction* and *IT&Documentation* are instances of the *Strategic Apex* and the *Support*, respectively), dependencies are instances of structure-in-5 dependencies classes (e.g. *Agency Global Strategy* is an instance the *Strategic Management*), and all above global properties are enforced (e.g. since there are only two task dependencies between *Campaigns Management* and *Graphics&Edition*, the fourth property holds).

3.2. Joint venture

We describe here three alliances — Airbus and Eurocopter [10] and a more detailed one, Carsid [20] — that will serve to model the joint venture structure as an organizational style and propose a semi-formal specification.

Airbus. The Airbus Industrie joint venture coordinates collaborative activities between European aeronautic manufacturers to build and market airbus aircrafts. The joint venture involves four partners: British Aerospace (UK), Aerospatiale (France), DASA (Daimler-Chrysler Aerospace, Germany) 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 new 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 enter into 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 (represented as a circle with a curved line): 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 different parts of the planes.

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.

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.

Carsid(Carolo-Sidérurgie) is a joint venture that has recently arisen from the global concentration movement in the steel industry. The alliance, physically located in the steel basin of Charleroi in Belgium, has been formed by the steel companies *Duferco* (Italy), *Usinor* (France) — that also partially owns *Cockerill-Sambre* (Belgium) through the *Arcelor* group — and *Sogepa* (Belgium), a public investment

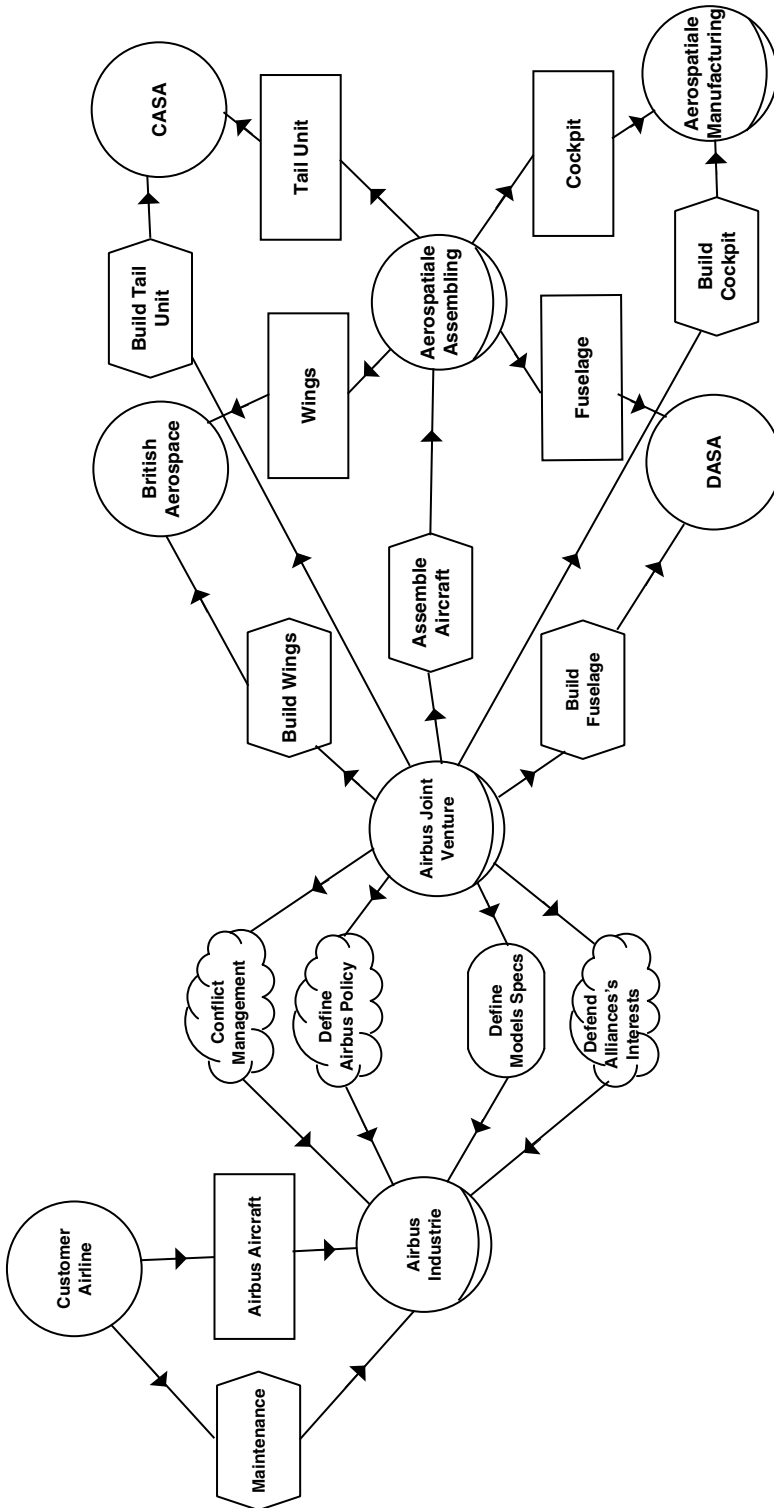


Fig. 5. The airbus industrie joint venture.

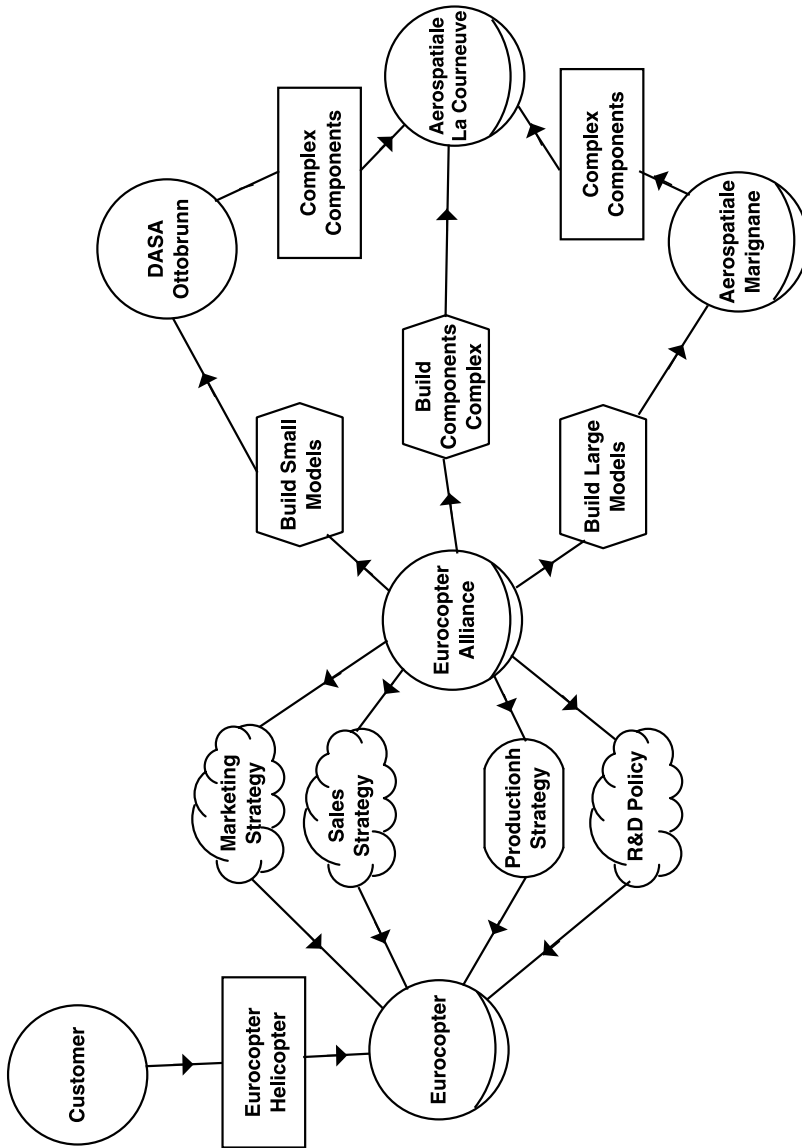


Fig. 6. The Eurocopter joint venture.

company, representing the Walloon Region Government. Usinor has also brought its subsidiary Carlam in the alliance.

Roughly speaking, the aim of a steel manufacturing company like CARSID is to extract iron from the ore and to turn it into semi-finished steel products. Several steps compose the transformation process, and each step is generally assumed by a specific metallurgic plant:

- *Sintering Plant.* Sintering is the preparation of the iron ore for the blast furnace. The minerals are crushed and calibrated to form a sinter charge.
- *Coking Plant.* Coal is distilled (i.e., heated in an air-impoverished environment in order to prevent combustion) to produce coke.
- *Blast Furnace.* Coke is used as a combustion agent and as a reducing agent to remove the oxygen from the sinter charge. The coke and sinter charge are loaded together into the blast furnace to produce cast iron.
- *Steel Making Plant.* Different steps (desulphuration, oxidation, steel adjustment, cooling, ...) are necessary to turn cast iron into steel slabs and billets. First, elements other than iron are removed to give molten steel. Then supplementary elements (titanium, niobium, vanadium, ...) are added to make a more robust alloy. Finally, the result — finished steel — is solidified to produce slabs and billets.
- *Rolling Mill.* The manufacture of semi-finished products involves a process known as hot rolling. Hot-rolled products are of two categories: flat (plates, coiled sheets, sheeting, strips, ...) produced from steel slabs and long (wire, bars, rails, beams, girders, ...) produced from steel billets.

Figure 7 models the organization of the Carsid joint venture in i^* . Carsid assumes two roles Carsid S. A. (“Société Anonyme” i.e., “Ltd”) and Carsid Joint Venture.

Carsid S. A. is the legal and contractual interface of the joint venture. It handles the sales of *steel semi-finished products* (bars, plates, rails, sheets, etc. but also slabs, billets) and *co-products* (coke that does not meet blast furnace requirements, rich gases from the different plants, godroon, naphtalin, etc.) to external *industries* such as vehicle (automobile, train, boat, ...) manufacturers, foundries, gas companies, building companies, ... It is also in charge of the *proper environment policy*, a strategic aspect for steelworks that are polluting plants. Most importantly, Carsid has been set up with the help of the Walloon Region to guarantee *job security* for about two thousands workers in the basin of Charleroi. Indeed, the steel industry in general and the Walloon metallurgical basins in particular are sectors in difficulty with high unemployment rates. As a corrolar, the joint venture is committed to *improve regional economy* and maintain work in the region. Carsid has then been contractually obliged to plan *maintenance investment* (e.g. blast furnace repairing, renovation of coke oven batteries, ...) and develop *production plans* involving regional sub-contractors and suppliers. Since steelmaking is a hard and dangerous work sector, Carsid is legally committed to respect, develop and promote *accident prevention standards*.

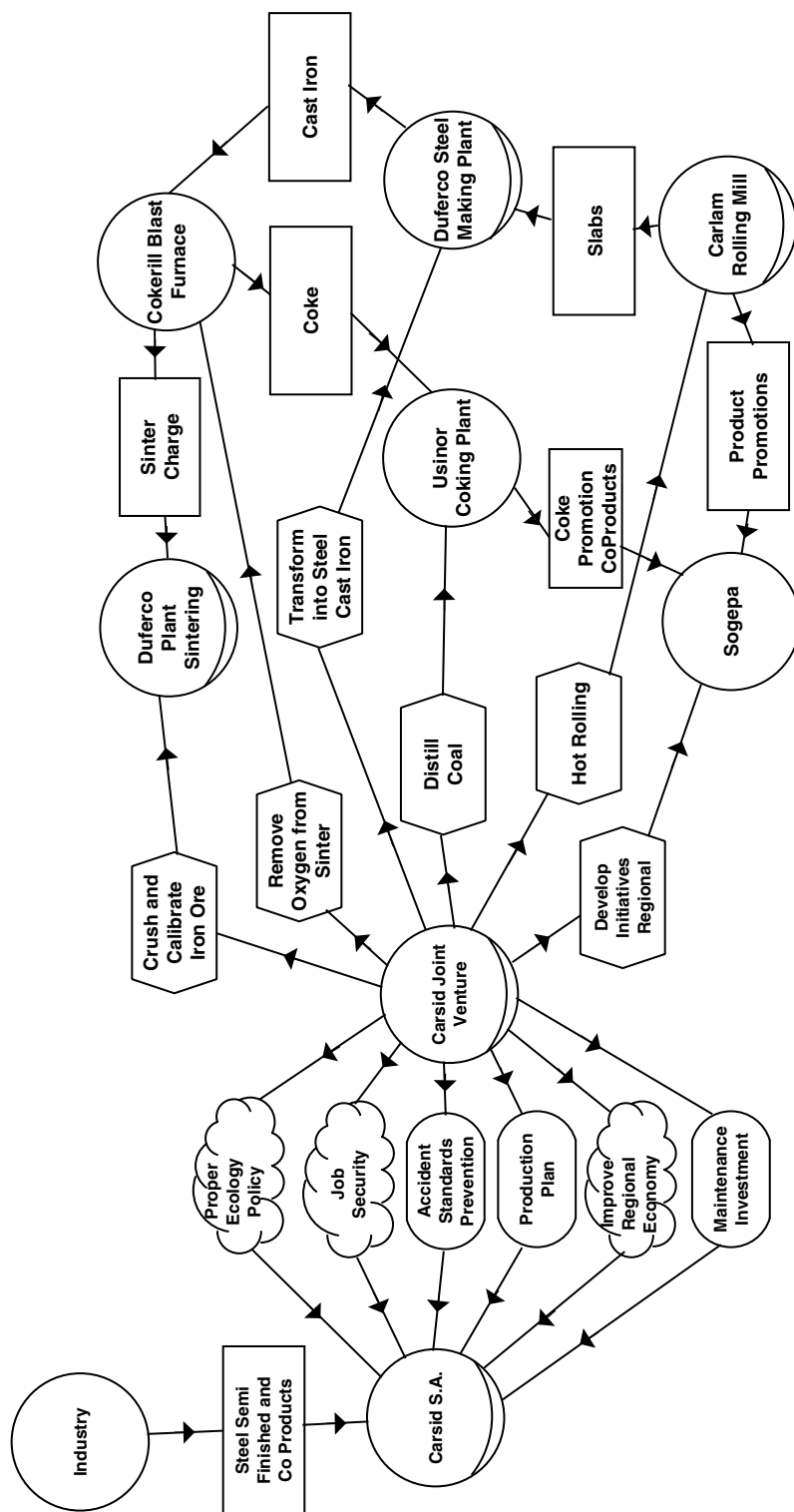


Fig. 7. The Carsid joint venture.

The *Carsid joint venture* itself coordinates the steel manufacturing process. The sintering phase to *prepare iron ore* is the responsibility of *Duferco Sintering Plant* while *Usinor Coking Plant*, *distills coal* to turn it into *coke*. The *sinter charge* and *coke* are used by *Cokerill Blast Furnace* to produce *cast iron* by *removing oxygen from sinter*. *Duferco Steel Making Plant* transforms *cast iron* into *steel* to produce slabs and billets for *Carlam Rolling Mill* in charge of the *hot rolling* tasks. *Sogepa*, the public partner, has the responsibility to *develop regional initiative* to promote *Carsid* activities, particularly in the *Walloon Region* and in *Belgium*.

Figure 8 abstracts the joint venture structures explored in the case studies of Figs. 5–7. The case studies suggest a number of constraints to supplement the basic style:

- 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.
- Individual partners can be decomposed in turn using another style.

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

Role JointManagerPrivateInterface
Goal CoordinateStyles

Role JointManagerPublicInterface
Goal TakeStrategicDecision
SoftGoal RepresentPartnershipInterests

Actor Partner

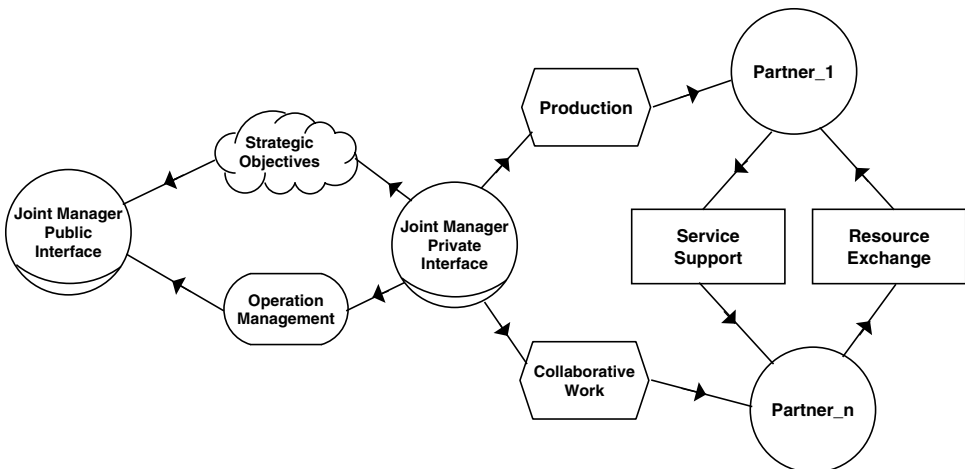


Fig. 8. The joint venture style.

and the following structural (global) properties must be satisfied:

- $\forall jmpri1, jmpri2 : JointManagerPrivateInterface$
 $(jmpri1 = jmpri2)$
[Only one instance of the joint manager]
- $\forall p1, p2 : Partner, dep : Dependency$
 $((dep.depender = p1 \wedge dep.dependee = p2) \vee (dep.depender = p2 \wedge dep.dependee = p1)) \rightarrow (dep.type = resource)$
[Only resource dependencies between partners]
- $\forall jmpri : JointManagerPrivateInterface, p : Partner,$
 $dep : Dependency((dep.dependee = p \wedge dep.depender = jmpri)$
 $\rightarrow dep.type = task)$
[Only task dependencies between partners and the joint manager, with the joint manager as depender]
- $\forall jmpri : JointManagerPrivateInterface,$
 $jmpui : JointManagerPublicInterface, dep : Dependency$
 $((dep.depender = jmpri \wedge dep.dependee = jmpui)$
 $\rightarrow (dep.type = goal \vee dep.type = softgoal))$
[Only goal or softgoal dependencies between the joint manager roles]
- $\forall dep : Dependency, p1 : Partner$
 $((dep.depender = p1 \vee dep.dependee = p1) \rightarrow$
 $((\exists p2 : Partner(p1 \neq p2$
 $\wedge (dep.depender = p2 \vee dep.dependee = p2))$
 $\vee (\exists jmpri : JointManagerPrivateInterface$
 $((dep.depender = jmpri \vee dep.dependee = jmpri))))$
[Partners only have relationships with other partners or the joint manager private interface]
- $\forall dep : Dependency, jmpri : JointManagerPrivateInterface$
 $((dep.depender = jmpri \vee dep.dependee = jmpri) \rightarrow$
 $((\exists p : Partner((dep.depender = p \vee dep.dependee = p))) \vee$
 $(\exists jmpui : JointManagerPublicInterface$
 $((dep.depender = jmpui \vee dep.dependee = jmpui))))$
[The joint manager private interface only has relationships with the joint manager public interface or partners]

4. Evaluation

Styles can be compared and evaluated with quality attributes [34], also called non-functional requirements [5]. For instance, the requirements seem particularly relevant for organizational structures [7, 22]:

Predictability [37]. Actors can have a high degree of autonomy [38] in the way that 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. Generally, predictability is in contrast with the actors capabilities to be adaptive and responsive: actors must be predictable enough to anticipate and plan actions while being responsive and adaptive to unexpected situations.

Security. Actors are often able to identify their own data and knowledge sources and they may undertake additional actions based on these sources [37]. Strategies for

verifying authenticity for these data sources by individual actors are an important concern in the evaluation of overall system quality since, in addition to possibly misleading information acquired by actors, there is the danger of hostile external entities spoofing the system to acquire information accorded to trusted domain actors.

Adaptability. Actors may be required to adapt to modifications in their environment. They may include changes to the component's communication protocol or possibly the dynamic introduction of a new kind of component previously unknown or the manipulations of existing actors.

Generally, adaptability depends on the capabilities of the single actors to learn and predict the changes of the environments in which they act [36], and also their capability to make diagnosis [19], that is being able to detect and determine the causes of a fault based on its symptoms. However, successful organization environments tend to balance the degree of reactivity and predictability of the single actors with their capabilities to be adaptive.

Coordinability. Actors are not particularly useful unless they are able to coordinate with other agents. Coordination is generally [21] used to distribute expertise, resources or information among the actors (actors may have different capabilities, specialized knowledge, different sources of information, resources, responsibilities, limitations, charges for services, etc.), solve interdependencies between actors' actions (interdependence occur when goal undertaken by individual actors are related), meet global constraints (when the solution being developed by a group of actors must satisfy certain conditions if it is to be deemed successful), and to make the system efficient (even when individuals can function independently, thereby obviating the need for coordination, information discovered by one actor can be of sufficient use to another actor that both actors can solve the problem twice as fast).

Coordination can be realized in two ways:

- **Cooperativity.** Actors must be able to coordinate with other entities to achieve a common purpose or simply their local goals. Cooperation can either be communicative in that the actors communicate (the intentional sending and receiving of signals) with each other in order to cooperate or it can be non-communicative [9]. In the latter case, actors coordinate their cooperative activity by each observing and reacting to the behaviour of the other. In deliberative organizations, actors jointly plan their actions so as to cooperate with each other.
- **Competitivity.** Deliberative negotiating organization [9] are like deliberative one, except that they have an added dose of competition. The success of one actors implies the failure of others.

Availability. Actors that offer services to other actors must implicitly or explicitly guard against the interruption of offered services.

Fallibility-Tolerance. A failure of one actor does not necessarily imply a failure of the whole organization. The organization then needs to check the completeness

and the accuracy of information and knowledge transactions and workflows. To prevent failure, different actors can have similar or replicated capabilities and refer to more than one actor for a specific behavior.

Modularity [35] increases efficiency of service execution, reduces interaction overhead and usually enables high flexibility. On the other hand, it implies constraints on inter-organization communication.

Aggregability. Some actors are parts of other actors. They surrender to the control of the composite entity. This control results in an efficient workflow execution and low interaction overhead, however prevents the organization to benefit from flexibility.

As an illustration, we evaluate the styles with respect to coordinativity, predictability, fallibility-tolerance and adaptability. The evaluation can be done in a similar way for the other non-functional requirements.

- 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 line 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 fallibility-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 2. summarizes the strengths and weaknesses of the reviewed styles:

	Structure-in-5	Joint-Venture
Coordinativity	++	+-
Predictability	+	+
Fallibility-Tolerance	++	+-
Adaptability	+-	+-

To cope with non-functional requirements and select the style for the organizational setting, we go through a means-ends analysis using the non-functional requirements (NFRs) framework [5].^a We refine the identified requirements to sub-requirements that are more precise and evaluates alternative organizational styles against them, as shown in Fig. 9. The analysis is intended to make explicit the space of alternatives for fulfilling the top-level requirements. The styles are represented as operationalized requirements (saying, roughly, “model the organizational setting of the system with the *pyramid*, *structure-in-5*, *joint venture*, *arm’s-length* . . . style”).

The evaluation results in contribution relationships from the styles to the non-functional requirements, labeled “+”, “++”, “-”, “--”. Design rationale is represented by claims drawn as dashed clouds. They make it possible for domain characteristics (such as priorities) to be considered and properly reflected into the decision making process, e.g. to provide reasons for selecting or rejecting possible solutions (+, -). Exclamation marks (! and !!) are used to mark priority requirements while a check-mark “√” indicates an accepted requirements and a cross “X” labels a denied requirement.

Relationships types (AND, OR, ++, +, -, and --) between NFRs are formalized to offer a tractable proof procedure. AND/OR relationships corresponds to the classical AND/OR decomposition relationships: if requirement R_0 is AND-decomposed (respectively, OR-decomposed) into R_1, R_2, \dots, R_n then all (at least one) of the requirements must be satisfied for the requirement R_0 to be satisfied. So, for instance, in Fig. 9, Coordinativity is AND-decomposed into Distributivity, Participability, and Commonality. Relationships “+” and “-” model respectively a situation where an requirement contributes positively or negatively towards the satisfaction of another one. For instance, in Fig. 9, Joint Venture contributes positively to the satisfaction of Distributivity and negatively to the Reliability. In addition, relationships “++” and “--” model a situation where the satisfaction of a requirement implies the satisfaction or denial of another goal. In Fig. 9, for instance, the

^aIn the NFR framework, non-functional requirements are represented as softgoals (cloudy shapes).

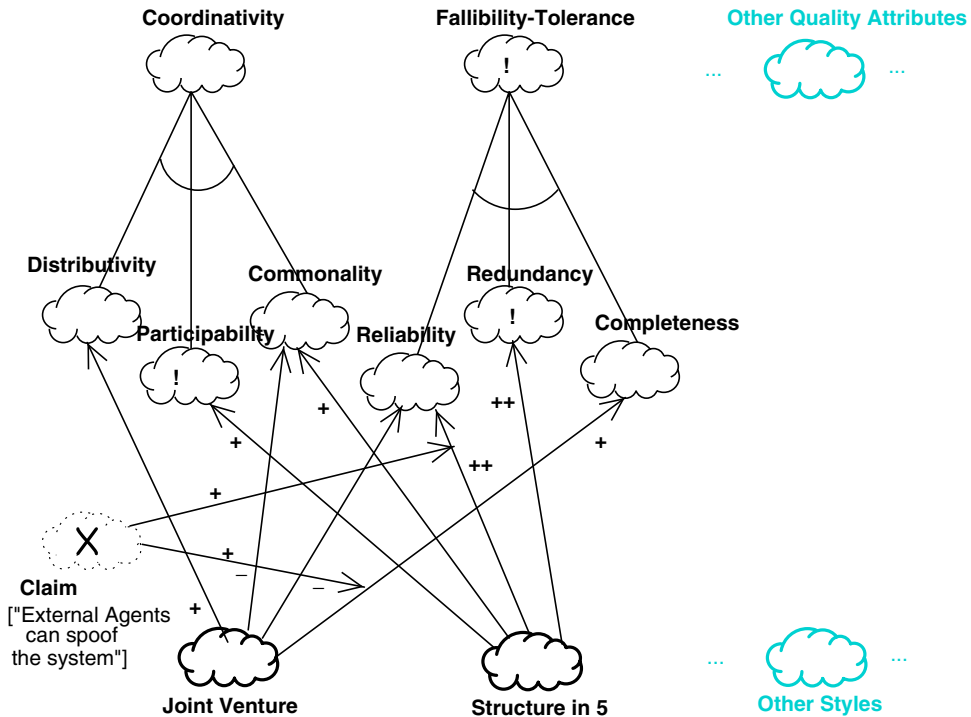


Fig. 9. Partial evaluation for organizational styles.

satisfaction of Structure-in-5 implies the satisfaction of requirements Reliability and Redundancy.

The analysis for selecting an organizational setting that meets the requirements of the system to build is based on propagation algorithms presented in [15]. Basically, the idea is to assign a set of initial labels for some requirements of the graph, about their satisfiability and deniability, and see how this assignment leads to the labels propagation for other requirements. In particular, we adopt from [15] both qualitative and a numerical axiomatization for goal (requirements) modeling primitives and label propagation algorithms that are shown to be sound and complete with respect to their respective axiomatization. In the following, a brief description of the qualitative algorithm.

To each requirement R , we associate two variables $Sat(R), Den(R)$ ranging in $\{F, P, N\}$ (full, partial, none) such that $F > P > N$, representing the current evidence of satisfiability and deniability of the requirement R . For example, $Sat(R_i) \geq P$ states there is at least a partial evidence that R_i is satisfiable. Starting from assigning an initial set of input values for $Sat(R_i), Den(R_i)$ to (a subset of) the requirements in the graph, we propagate the values through the propagation rules of Table 3. Propagation rules for AND (respectively OR) relationship are min-value

Table 3.

	$G_2 \overset{+}{\mapsto} G_1$	$G_2 \overset{-}{\mapsto} G_1$	$G_2 \overset{++}{\mapsto} G_1$	$G_2 \overset{--}{\mapsto} G_1$
$Sat(G_1)$	$\min\{Sat(G_2), P\}$	N	$Sat(G_2)$	N
$Den(G_1)$	N	$\min\{Sat(G_2), P\}$	N	$Sat(G_2)$

```

1   Current=Initial;
2   do
3       Old=Current;
4       for each  $R_i$  do
5           Current[i] = Update_label(i, Old);
6       until not (Current==Old);
7       return Current;
8   for each  $Rel_j$  s.t. target( $Rel_j$ ) ==  $R_i$  do
9        $sat_{ij}$  = Apply_Rules_Sat(i,  $Rel_j$ , Old);
10       $den_{ij}$  = Apply_Rules_Den( $R_i$ ,  $Rel_j$ , Old);
11  return ( max(max_j(sat_ij), Old[i].sat),
12          max(max_j(den_ij), Old[i].den) );

```

Fig. 10. Schema of the label propagation algorithm.

function for satisfiability (max-value function) and max-value function (min-value function) for deniability. A dual table is given for deniability propagation.

The schema of the algorithm is described in Fig. 10. *Initial*, *Current* and *Old* are arrays of pairs $\langle Sat(R_i), Den(R_i) \rangle$, one for each R_i of the graph, representing respectively the initial, current and previous labeling status of the graph.

The array *Current* is first initialized to the initial values *Initial* given in input by the user. At each step, for every requirement R_i , $\langle Sat(R_i), Den(R_i) \rangle$ is updated by propagating the values of the previous step. This is done until a fixpoint is reached, that is, no updating is mode possible ($Current == Old$).

The updating of $\langle Sat(R_i), Den(R_i) \rangle$ works as follows. For each relation Rel_i incoming in G_i , the satisfiability and deniability values sat_{ij} and den_{ij} derived from the old values of the source requirements are computed by applying the rules of Table 3. Then, it is returned the maximum value between those computed and the old values.

5. A Requirements-Driven Methodology

This research is conducted in the context of the *early requirements* phase of *Tropos* [3, 4, 30], a software development methodology for building multi-agent systems which is founded on the concepts of actor and goal.

The *Tropos* methodology adopts ideas from multi-agent systems technologies, mostly to define the detailed design and implementation phase, and ideas from requirements engineering, where agents/actors and goals have been used heavily for

early requirements analysis [6,40]. In particular, the *Tropos* project adopts Eric Yu's i^* model which offers actors (agents, roles, or positions), goals, and actor dependencies as primitive concepts for modelling an application during early requirements analysis. The key assumption which distinguishes *Tropos* from other methodologies is that actors and goals are used as fundamental concepts for analysis and design during *all phases of software development*, not just requirements analysis. That means that, in the light of this chapter, *Tropos* describes in terms of the same concepts and styles the organizational environment within which a system will eventually operate, as well as the system itself. *Tropos* spans four phases of software development:

- Early requirements, concerned with the understanding of a problem by studying an organizational setting; the output is an organizational model which includes relevant actors, their goals and dependencies.
- Late requirements, in which the system-to-be is described within its operational environment, along with relevant functions and qualities.
- Architectural design, in which the system's global architecture is defined in terms of subsystems, interconnected through data, control and dependencies.
- Detailed design, in which behaviour of each architectural component is defined in further detail.

6. Conclusion

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 styles. This chapter focuses on two such styles and studies them in detail, through examples, a formalization using Formal Tropos, and an evaluation with respect to desirable requirements. There have been many proposals for software patterns (e.g. [8]) since the original work on design patterns [13]. Some of this work focuses on requirements patterns. For example, [26] 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, [11] 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, [18] 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. We have already described organizational styles but to be used for designing multi-agent system architectures [14,22,24] and e-business systems [7]. 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 chapter, we have focused on styles 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 the early requirements analysis.

References

1. A. I. Anton, "Goal-based requirements analysis", *Proceedings of the 2nd International Conference on Requirements Analysis, ICRE'96* (1996) 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, "A requirements-driven development methodology", *Proceedings of the 13th International Conference on Advanced Information Systems Engineering, CAiSE'01*, Interlaken, Switzerland (June 2001) 108–123.
4. J. Castro, M. Kolp and J. Mylopoulos, "Towards requirements-driven information systems engineering: The Tropos project", *Information Systems*, 2002.
5. L. K. Chung, B. Nixon, E. Yu and J. Mylopoulos, *Non-Functional Requirements in Software Engineering* (Kluwer Publishing, 2000).
6. A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal-directed requirements acquisition", *Science of Computer Programming* **20**, no. 1–2 (1993) 3–50.
7. T. T. Do, S. Faulkner and M. Kolp, "Organizational multi-agent architectures for information systems", *Proceedings of the 5th International Conference on Enterprise Information Systems, ICEIS'03*, Angers, France, April 2003.
8. T. T. Do, M. Kolp and A. Pirotte, "Social patterns for designing multi-agent systems", *Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, SEKE'03*, San Francisco, USA, July 2003.
9. J. E. Doran, S. Franklin, N. R. Jennings and T. J. Norman, "On cooperation in multi-agent systems", *Knowledge Engineering Review* **12**, no. 3 (1997) 309–314.
10. P. Dussauge and B. Garrette, *Cooperative Strategy: Competing Successfully Through Strategic Alliances* (Wiley and Sons, 1999).
11. M. Fowler, *Analysis Patterns: Reusable Object Models* (Addison-Wesley, 1997).
12. A. Fuxman, M. Pistore, J. Mylopoulos and P. Traverso, "Model checking early requirements specification in Tropos", *Proceedings of the 5th International Symposium on Requirements Engineering, RE'01*, Toronto, Canada, August 2001.
13. E. Gamma, R. Helm, J. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley, 1995).
14. P. Giorgini, M. Kolp and J. Mylopoulos, "Multi-agent and software architecture: A comparative case study", *Proceedings of the 3rd International Workshop on Agent Software Engineering, AOSE'02*, Bologna, Italy, July 2002.
15. P. Giorgini, J. Mylopoulos, E. Nicchiarelli and R. Sebastiani, "Reasoning with goal models", *Proceedings of the 21st International Conference on Conceptual Modeling (ER 2002)*, Tampere, Finland, October 2002.
16. GMT, Gmt consulting group, 2002, <http://www.gmtgroup.com/>.
17. B. Gomes-Casseres, *The Alliance Revolution: The New Shape of Business Rivalry* (Harvard University Press, 1996).
18. D. Gross and E. Yu, "From non-functional requirements to design through patterns", *Requirements Engineering* **6**, no. 1 (2002) 18–36.

19. B. Horling, V. Lesser, R. Vincent, A. Bazzan and P. Xuan, "Diagnosis as an integral part of multi-agent adaptability", Technical Report UM-CS-1999-003, University of Massachusetts (1999).
20. M. Ibarz, M. Kolp, F. Fouss and A. Pirotte, "Steel production datawarehouse re-engineering", Technical Report IAG Working paper 85/03, IAG-ISYS Information Systems Research Unit, Catholic University of Louvain, Belgium (February 2003), <http://www.iag.ucl.ac.be/wp/>.
21. N. R. Jennings, "Coordination techniques for distributed artificial intelligence", eds. G. M. P. O'Hare and N. R. Jennings, *Foundations of Distributed Artificial Intelligence* (Wiley, 1996) 187-210.
22. M. Kolp, P. Giorgini and J. Mylopoulos, "A goal-based organizational perspective on multi-agents architectures", *Proceedings of the 8th International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages, ATAL'01*, Seattle, USA, August 2001.
23. M. Kolp, P. Giorgini and J. Mylopoulos, "Information systems development through social structures", *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE'02*, Ischia, Italy, July 2002.
24. M. Kolp, P. Giorgini and J. Mylopoulos, "Organizational multi-agent architecture: A mobile robot example", *Proceedings of the 1st International Conference on Autonomous Agent and Multi Agent Systems, AAMAS'02*, Bologna, Italy, July 2002.
25. M. Kolp, P. Giorgini and J. Mylopoulos, "Organizational patterns for early requirements analysis", *Proceedings of the 15th International Conference on Advanced Information Systems, CAiSE'03*, Velden, Austria, June 2003.
26. S. Konrad and B. Cheng, "Requirements patterns for embedded systems", *Proceedings of the 10th IEEE Joint International Requirements Engineering Conference, RE'02*, Essen, Germany, September 2002.
27. J. J. Lambin, *Strategic Marketing: A European Approach* (McGraw-Hill, 1993).
28. H. Mintzberg, *Structure in Fives: Designing Effective Organizations* (Prentice-Hall, 1992).
29. J. Morabito, I. Sack and A. Bhate, *Organization Modeling: Innovative Architectures for the 21st Century* (Prentice-Hall, 1999).
30. A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini and J. Mylopoulos, "A knowledge level software engineering methodology for agent oriented programming", *Proceedings of the 5th International Conference on Autonomous Agents, Agents'01*, Montreal, Canada, May 2001.
31. M. E. Porter, *Competitive Advantage: Creating and Sustaining Superior Performance* (The Free Press, New York, 1985).
32. W. R. Scott, *Organizations: Rational, Natural, and Open Systems* (Prentice-Hall, 1998).
33. L. Segil, *Intelligent Business Alliances: How to Profit Using Today's Most Important Strategic Tool* (Times Business, 1996).
34. M. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline* (Prentice-Hall, 1996).
35. O. Shehory, "Architectural properties of multi-agent systems", Technical Report CMU-RI-TR-98-28, Carnegie Mellon University (1998).
36. G. Weiss (ed.), *Learning in DAI Systems* (Springer-Verlag, 1997).
37. S. G. Woods and M. Barbacci, "Architectural evaluation of collaborative agent-based systems", Technical Report SEI-99-TR-025, SEI, Carnegie Mellon University, Pittsburgh, USA (1999).

38. M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice", *Knowledge Engineering Review* **2**, no. 10 (1995).
39. M. Y. Yoshino and U. Srinivasa Rangan, *Strategic Alliances: An Entrepreneurial Approach to Globalization* (Harvard Business School Press, 1995).
40. E. Yu, "Modelling strategic relationships for process reengineering", PhD Thesis, University of Toronto, Department of Computer Science (1995).