ORGANIZATIONAL MULTI-AGENT ARCHITECTURES FOR INFORMATION SYSTEMS

T. Tung Do, Stéphane Faulkner, Manuel Kolp
IAG- School of Management, ISYS- Information Systems Research Unit, University of Louvain, 1 Place des Doyens, Belgium
Email: do@isys.ucl.ac.be, faulkner@isys.ucl.ac.be, kolp@isys.ucl.ac.be

Keywords: Multi-Agent Systems, Organizational Styles, System Design, Architectural Patterns

Abstract: A Multi-Agent System (MAS) architecture is an organization of coordinated autonomous agents that interact in order to achieve particular, possibly common goals. Considering real-world organizations as an analogy, this paper proposes MAS architectural patterns for information systems which adopt concepts from organizational theories. The patterns are modeled using the *i** framework which offers the notions of actor, goal and actor dependency, specified in *Formal Tropos* and evaluated with respect to a set of software quality attributes, such as predictability or adaptability. We conduct a comparison of organizational and conventional architectures using an e-business information system case study.

1 INTRODUCTION

The explosive growth of application areas such as electronic commerce, enterprise resource planning and mobile computing has profoundly changed our views on information systems engineering. Information systems must now be based on open architectures that continuously evolve to accommodate new components and meet new requirements. These new requirements, in turn, call for new concepts and techniques for engineering and managing information systems. For these reasons – and more – Multi-Agent Systems (MAS) are gaining popularity over traditional systems, including object-oriented ones. They do provide for an open, evolving architecture which can change at run-time to exploit the services of new agents, or replace under-performing ones.

MAS architectures can be considered organizations (see e.g., [Kol01]) composed of *autonomous* and *proactive* agents that interact and cooperate with each other to achieve common or private goals. Since the fundamental concepts of multi-agent systems are intentional and social, rather than implementation-oriented, we turn to theories which study social and intentional structures for motivation and insights to define and describe multi-agent architectures. But, what kind of social theory should we turn to? There are theories that study group psychology, communities and social networks. Such theories study social and intentional structure as an emergent property of a social context. Instead, we are interested in socio-intentional structures that emerge from a design process. For this, we turn to organization theory and strategic alliances for guidance. Organizational styles from organization theory describe the internal structure and design of an organization, while strategic alliances model the cooperation of independent organizational actors that pursue shared goals.

This paper presents work on the development of a set of organizational architectural patterns for multi-agent systems motivated by these theories. Our organizational patterns are modeled using the strategic dependency model of *i** [Yu95] and specified in *Formal Tropos* [Fux01]. To illustrate these patterns, we use a case study comparing organizational with conventional software architectural styles for e-business systems.

Sections 2 and 3 describe organizational patterns we have identified from organization theory and strategic alliances. Section 4 offers some specification in *Formal Tropos* used to formalize organizational concepts and patterns. Section 5 evaluates the use of our organizational patterns: we introduce an e-business example, identify relevant agent software qualities and compare conventional architectures and organizational ones with respect to identified software qualities. Finally, Section 6 summarizes the results and points to further work.

2 ORGANIZATION THEORY

Organization theory (e.g., [Min92, Sco98]) studies the structure and design of organizations. It

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, Organization Theory proposes organization styles to help organization analysts. In the following, we focus on one of these styles, namely, Mintzberg's structure-in-5. For further information about other organizational patterns we are working on, see [Kol01].

2.1 Structure-in-5

An organization can be considered an aggregate of five sub-structures, as described by Mintzberg [Min92]: the *Operational Core*, the *Strategic Apex*, the *Middle Line*, the *Technostructure* and the *Support*. 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 operational 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).

To model and formalize the structure-in-5 as an organizational pattern, we first analyze a case study of an organization on which the pattern can be applied.

2.2 A Case Study: Volvo Trucks

Volvo Trucks Corporation (VTC) is a subsidiary company of AB Volvo, the automobile manufacturer [Lam93]. The VTC distributive network is segmented into eight commercial divisions responsible for the production and the commercialization of trucks. These divisions take in charge 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.

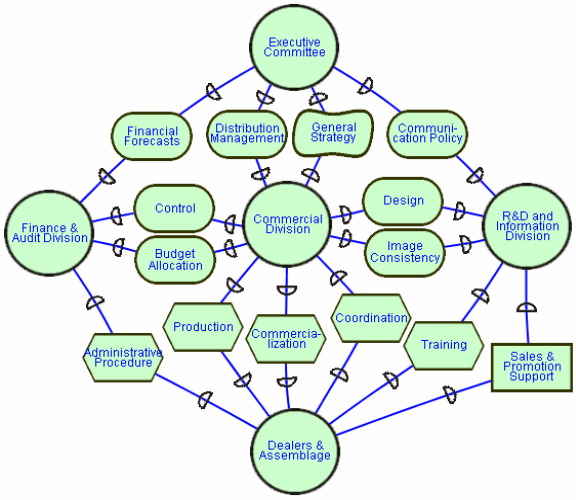

Figure 1. Volvo Trucks Corporation in Structure-in-5

Figure 1 models the VTC structure-in-5 using the *i** strategic dependency model. *i** is a framework for organization modeling [Yu95], which offers goal- and actor-based notions such as *actor*, *agent, role, position, goal, softgoal*, *task*, *resource, belief* and different kinds of social *dependency* between 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 *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 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 when the dependee is required to perform a given activity; and *resource* dependencies require the dependee to provide a resource to the depender.

As show in Figure 1, actors are represented as circles; dependums – goals, softgoals, tasks and resources – are respectively represented as ovals, clouds, hexagons and rectangles; dependencies have the form *depender* → *dependum* → *dependee*. In Figure 1, 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 Division*s 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 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 Division*s.

2.3 The Structure-In-5 as an Organizational Pattern

Figure 2 abstracts the structure explored in Figure 1 as our Structure-in-5 pattern composed of five actors.

It also suggests a number of constraints:

- 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 type goal dependency;
- 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).

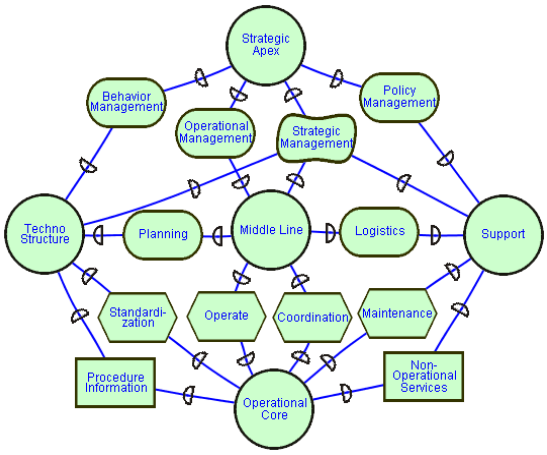

Figure 2. The Structure-in-5 Pattern

3 STRATEGIC ALLIANCES

A strategic alliance links specific facets of the businesses of two or more organizations [Dus99]. 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. 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 alliance 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;

In this paper, we will focus on the joint venture pattern. For further information about other strategic alliances patterns we are working on, see [Kol01].

3.1 Joint Venture

The joint venture pattern involves agreement between two or more intra-industry partners to obtain the benefits of larger scale, shared 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 interacts 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 the sole responsibility of the joint management actor in which the original actors possess equity participations.

3.2 A Case Study: AirBus

The Airbus Industrie joint venture coordinates collaborative activities between European aeronautic manufacturers to build and market Airbus aircrafts [Dus99]. 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 the partner firms directly, but have to deal with Airbus Industrie. Airbus Industrie defines the alliance's product policy and elaborates the specifications of each new aircraft model. 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 3 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 requests 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 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.

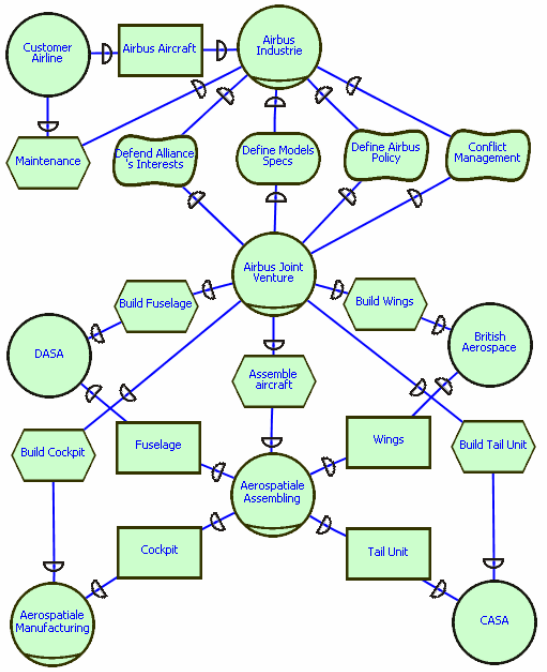

Figure 3: The Airbus Joint Venture

3.3 The Joint-Venture as an Organizational Pattern

Figure 4 abstracts the structures explored in Figure 3 as our Joint-Venture pattern. It also suggested a number of constraints:

- 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, represent the interests of the whole partnership with respect to external stakeholders and ensure communication with the external actors.

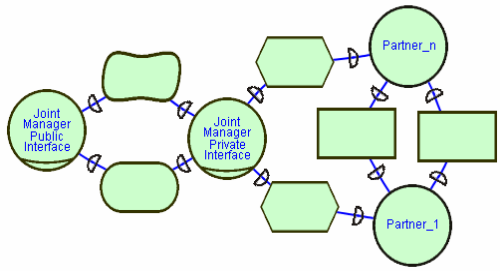

Figure 4. The Joint Venture Pattern

4 FORMALIZING PATTERNS

To specify the formal properties of the patterns, we use *Formal Tropos* [Fux01] that extends the primitives of *i** with a formal language. We augment *Formal Tropos* with specific multi-level-instantiation (i.e., an instance of a class can be in turn a class and so on) that we call *Meta Tropos*. The *Meta Tropos* specification gives us the ability to formalize *i** concepts and models, as well as our organizational patterns. The advantage of this approach is that we define our patterns at a meta level once for all, and any application that conforms with these patterns will be considered an instance of these patterns.

i* Concepts. Due to lack of space, we only present the Actor concept and Strategic Dependency Diagram model. Others i* concepts (e.g., Goal, SoftGoal, Task, Entity, Dependum, …) or models (e.g. Strategic Rational Diagram, Non Functional Requirements Framework, ...) can be similarly specified.

In the following, the user-defined type *Formula* is used to refer to any formula in temporal logic. *Class* is a meta-concept and any other concept will be considered an instance of *Class* defined by the keyword *ISOF*. These concepts could be in turn considered classes of others concepts.

In *i**, an actor is an active entity that depends on other actors for goals to be fulfilled, softgoals to be achieved, tasks to be performed, and resources to be furnished. Each actor is defined by an *ActorName*, sets of *ActorAttributes*, *ActorGoals*, *ActorCreation* and *ActorInvariant* properties. Such properties define conditions that must hold, respectively, at the creation and during the life of each instance of an actor.

```
Actor
    ISOF Class
    Attribute
        ActorName : String
        ActorAttributes : Class
        ActorCreation : Formula
        ActorInvariants : Formula
        ActorGoal : Goal
End
```

The *i* strategic dependency model* is a scheme describing the network of relationships among actors and involves *actors* who have (goal, softgoal, task, resource) *dependencies* among each other. The *DependumGoal*, *DependumSoftGoal*, *DependumTask*, *DependumResource* describe the dependencies of type Goal, SoftGoal, Task, Resource respectively.

```
StrategicDependencyScheme
    ISOF Class
    Attribute
        Actors : Actor
        GoalDependencies : DependumGoal
        SoftgoalDependencies : DependumSoftGoal
        TaskDependencies : DependumTask
        ResourceDependencies : DependumResource
End
```

Patterns. Organizational patterns will be considered instances of the *StrategicDependencyScheme* briefly formalized above. We illustrate our purpose with the Structure-in-5 pattern modeled in Figure 2. The attribute *Actors* of *StrategicDependencyScheme* is instantiated to five attributes *StrategicApex, MiddleLine, TechnoStructure, Support, OperationalCore* corresponding to the five *i* actors* identified in the Stucture-In-5 pattern. The same can be said for attributes specifying goal, softgoal, task and resource dependencies.

```
StructureIn5Pattern
    ISOF StrategicDependencyScheme
    Part, ExclusivePart, DependentPart
    Actors
        StrategicApex
        MiddleLine
        TechnoStructure
        Support
        OperationalCore
```

```
    GoalDependencies
        OperationalManagement
        BehaviorManagement
        PolicyManagement
    SoftgoalDependencies
        StrategicManagement
    TaskDependencies
        Standardization
        Operate
        Coordination
        Maintenance
    ResourceDependencies
       NonOperationalServices
       ProcedureInformation
End
```

Each component (actor, goal, softgoal, task, resource) of this StructureIn5Pattern is formalized in *Meta Tropos*. As an example, we present some specification for the *OperationalManagement* goal.

```
OperationalManagement
        ISOF      DependumGoal
        GoalName : 'OperationalManagement'
        Mode : 'Achieve'
        Depender          sa        StrategicApex
        Dependee          ml        MiddleLine
        GoalInvariant :
        Consistent(self, StrategicManagement)
        ∀om: self ∃>=1 co: Coordination
              (co.dependee : MiddleLine ∧
              co.depender :  OperationalCore  ∧
              ImplementedBy(om, co))
        ∀ plandep: DependumGoal
           (plandep.depender = MiddleLine ∧
           plandep.dependee  =  TechnoStructure)  ∧
           Fulfilled(self)  → ♦ Fulfilled(plandep)
End
```

[The Operational Management goal is fulfilled only if all goal dependencies between the Middle Line as depender and the Technostructure as dependee have been achieved some time in the past. The Operational Management goal has to be consistent with Strategic Management softgoal.There exists a coordination task (a task dependency between MiddleLine and Operational Core) that implement (ImplementedBy) the Operational Management goal.]

5 A B2C CASE STUDY

This section overviews the use of the Structure-in-5 and joint-venture patterns and compares them to some conventional architectures with respect to software quality attributes. We illustrate the comparison with the design of an architecture for a business-to-consumer (B2C) application called *E-Media*. First, we describe some strategic quality attributes for designing e-business systems, then we present classical and organizational architectures and finally we propose an evaluation.

E-Media is a business-to-consumer system allowing on-line customers to buy different kinds of media items such as books, newspapers, magazines, audio CDs, videotapes and the like on the Internet. Customers can search the on-line store by either browsing the catalogue or using a search engine to query the database. *E-Media* also allows to process on-line orders, bills and delivery invoices, and keeps track of all web information of strategic importance for statistical analysis.

5.1 Qualities for E-Business Systems

Software quality attributes (i.e., non functional requirements describing how well the system accomplishes its functions) relevant for MAS have been studied in [Kol01]. These are among others: predictability, security, availability, adaptability, cooperativity, competitivity, failability-tolerance, modularity and aggregability. Three of them have been identified particularly strategic for e-business systems [Kol01] as:

Security. Clients, exposed to the internet are, like servers, at risk in web applications. It is possible for web browsers and application servers to download or upload content and programs that could open up the client system to crackers and automated agents. JavaScript, Java applets, ActiveX controls, and plug-ins represent a certain risk to the system and the information it manages. Equally important are the procedures checking the consistency of data transactions.

Adaptability deals with the way the system can be designed using generic mechanisms to allow web pages to be dynamically changed. It also concerns the catalogue update for inventory consistency.

Availability. Network communication may not be very reliable causing sporadic loss of the server. There are data integrity concerns with the capability of the e-business system to do what needs to be done, as quickly and efficiently as possible in particular with the ability of the system to respond in time to client requests for its services.

5.2 Classical Architectures

For sample classical solutions, we examine two major conventional architectures – the pipes and filters and the layered architectures [Sha96] – we have applied to the *E-Media* application:

Pipes-and-Filters Architecture. Each component has a set of inputs and a set of outputs. Components

read data streams on their inputs and produce data streams on their outputs, delivering a complete instance of the result in a standard order. This is usually accomplished by applying a local transformation to the input streams and computing incrementally so that output begins before input is consumed. The components are termed "filters". The connectors serve as conduits for the streams, transmitting outputs of one filter to inputs of another. They are termed "pipes".

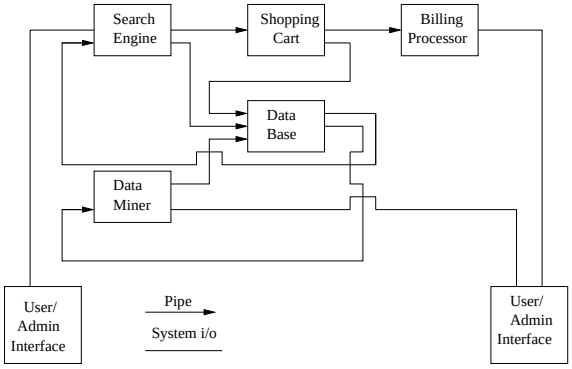

Figure 5. E-Media Pipes and Filters

Figure 5 illustrates a classical pipes-and-filters architecture for *E-Media*. Five filters compose the system: the *Search Engine*, the *Data Base*, the *Shopping Cart*, the *Billing Processor* and the *Data Miner*. Each filter processes the data and sends it to the next filter. The *Search Engine* takes queries from the user and transforms them into representations that can be understood by the *DataBase*. Queries are executed on the *Data Base* and information on selected products is sent back to the *Search Engine* for consultation by the user. The *Shopping Cart* allows users to choose and buy selected products. Information on purchased products is sent to the *Billing Processor* that handles the financial transactions. Information on purchased and selected products is stored in the *DataBase* for statistical analyses. The *Data Miner* monitors queries executed on the *DataBase* and produces business reports and forecasts.

Layered Architecture. A layered system is organized hierarchically. Each layer provides services to one or many layers above it and serves as a client to one or many layers below. The architecture of our *E-Media* application could be composed of five layers. At the lowest level, resides the *DataBase*. The second level is concerned with *searching* through the catalogue. The third and fourth levels deal with e-shopping activities: the *Shopping Cart* (level 3) allows users to select a set of products and the *Billing Processor* (level 4) handles financial transactions. The top layer (level 5) provides the *User Interface*.

5.3 Organizational Architectures

We are developing and testing organizational architectures for e-business systems prototypes based on [Con00] and using a typical Apache/SSL/MySQL/PHP configuration. Currently, we are testing our structure-in-5 and joint-venture patterns that are architectures working with abstractions reminiscent of those encountered in the pipes-and-filters and layered architectures presented above.

Structure-in-5. Figure 6 suggests a possible assignment of system responsibilities for *E-Media* following the structure-in-5 pattern. It is decomposed into five principal components *Store Front*, *Coordinator*, *Billing Processor*, *Back Store* and *Decision Maker*. *Store Front* serves as the *Operational Core*. It interacts primarily with Customer and provides her with a usable front-end web application for consulting and shopping media items. *Back Store* constitutes the *Support* component. It manages the product database and communicates to the *Store Front* information on products selected by the user. It *stores* and *backs up* all web information from the *Store Front* about customers, products, sales, orders and bills to produce *statistical information* to the *Coordinator*. It provides the *Decision Maker* with *strategic information* (analyses, historical charts and sales reports).

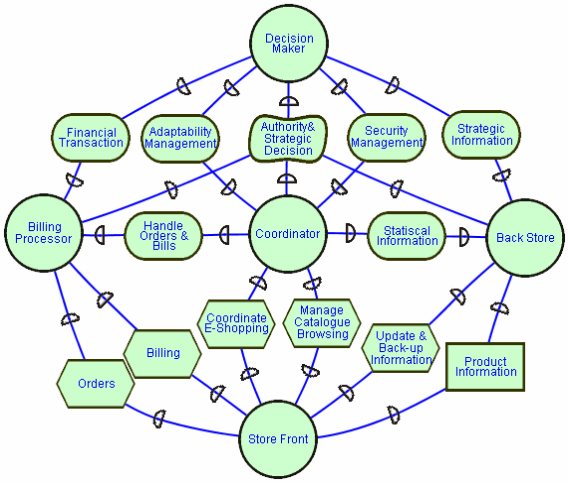

Figure 6. The E-Media Architecture in Structure-in-5

The *Billing Processor* is in charge of handling orders and bills for the *Coordinator* and implementing the corresponding procedures for the *Store Front*. It also ensures the secure management of financial transactions for the *Decision Maker*. As the *Middle Line*, the *Coordinator* assumes the central position of the architecture. It ensures the coordination of *e-shopping* services provided by the *Operational Core* including the management of conflicts between itself, the *Billing Processor*, the *Back Store* and the *Store Front*. To this end, it also handles and implements strategies to manage and prevent *security* gaps and *adaptability* issues. The *Decision Maker* assumes the *Strategic Apex* role. To this end, it defines the *Strategic Behavior* of the architecture ensuring that objectives and responsibilities delegated to the *Billing Processor*, *Coordinator* and *Back Store* are consistent with that global functionality.

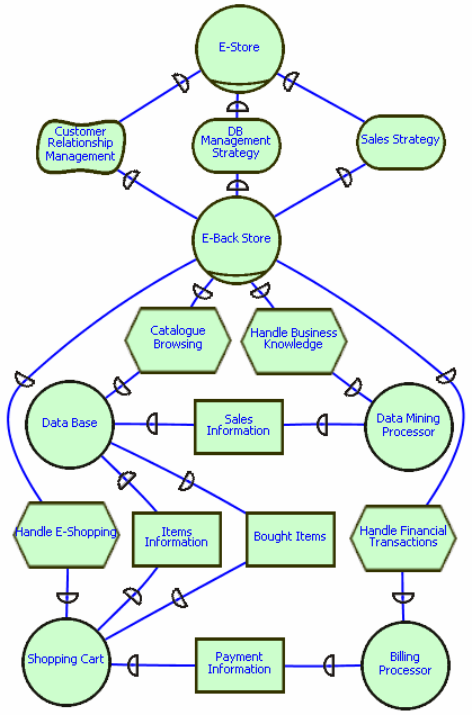

Figure 7. The E-Media Architecture in Joint Venture

Joint venture. Following the pattern depicted in Figure 4, the *E-Media* Joint Venture architecture (Figure 7) is organized around a joint manager assuming two roles: the *E-store* role defines the *Customer Relationship Management* and the operational strategies of *E-Media*, i.e., *Sales* and *DB Management Strategy*. The *Back Store* deals with coordinativity supervising the other actors: a *Data Mining Processor* handling *Business Knowledge* processes, a *DataBase* storing the on-line catalogue and allowing *Catalogue Browsing*, a *Billing Processor* managing all *Financial Transactions* and a *Shopping Cart* implementing *E-Shopping* activities.
Each of these four last actors also interacts directly with each other to exchange data, information and knowledge: the *Shopping Cart* needs data and information about selected products from the *DataBase* and provides *Billing Processor* with financial information about purchased products and on-line customers. The *Data Mining Processor* gets sales data and information from the *Data Base* to produce business knowledge, i.e., historical charts sales reports and business forecasts.

5.4 Evaluation

In this section, we evaluate the styles – pipes-and-filters, layered, structure-in-5 and joint venture – described in Sections 5.2 and 5.3 with respect to the three agent software quality attributes (Security, Adaptability and Availability) identified in Section 5.1

Adaptability. In the *pipes-and-filters* architecture, system components are filters connected to each other through pipes ensuring output/input streaming between filters. Consequently, a component can be modified or replaced if and only if it keeps outputing a data stream acceptable as input for the filters it connects. In the *layered architecture*, the interdependencies between layers prevent the addition of new components or deletion of existing ones. The fragile relationships between the layers can become more difficult to decipher with change. The *structure-in-5* separates independently each typical component of the *E-Media* architecture isolating them from each other and allowing dynamic manipulation. In the *joint venture*, manipulation of partner components can be done easily by registering new components to the joint manager. However, since partners can also communicate directly with each other, existing dependencies should be updated as well.

Security. The simplicity of the *pipes-and-filters* pattern makes verification of components and behavior easy and reduces the chances to get hostile external entities creeping into the system. In the *layered architecture*, security could be served by incorporating many checks and balances at different levels into the system. The drawback is that control commands and transactions may often need to skip intermediate layers to check the system behavior. In the *structure-in-5*, checks and control mechanisms

can be integrated at different abstractions levels assuming redundancy from different perspectives. Contrary to the layered architecture, checks and controls are not restricted to adjacent levels. Besides, since the structure-in-5 permits to separate process (*Store Front*, *Billing Processor* and *Back Store*) from control (*Decision Maker* and *Monitor*), security and consistency of these two hierarchies can also be verified independently. The *jointure venture*, through its joint manager, proposes a central message server/controller. Exception mechanism, wiretapping supervising or monitoring can be supported by the joint manager to guarantee non-failability, reliability and completeness.

Availability. The *pipes-and-filters* architecture can only manage availability issues through limited feed-back iterations. Such control loops can reduce bottlenecks and data jams at each turn but are quite limited. Unfortunately, if more complex steps are needed, the architecture offers no framework for delegating them to separate agent components. In the *layered architecture*, the existence of abstraction layers addresses the need for managing availability. What is unpredictable and contributes to bottlenecks and data jams at the lowest level become clear with the added knowledge in the higher levels. The *structure-in-5* architecture prevents availability problems by differentiating process from control. Besides, contrary to the layered architecture, 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. In the *joint venture*, the central position and role of the joint manager is a means for resolving conflicts between components and prevent availability issues. Through its joint manager, the architecture proposes a central message server/controller. Exception mechanisms, wiretapping supervising or monitoring can be centrally supported by to guarantee non-failability, reliability and completeness.

Table 1 summarizes strengths and weaknesses of the four architectures with respect to the software quality attributes. The layered architecture gives precise indications as to the components expected in a business to consumer system. The pipes-and-filters pattern concentrates on the dynamics of input/output data streams. The organizational patterns (Structure-in-5 and Joint Venture) focus on how to organize components expected in an e-business system but also on the intentional and social dependencies governing these components. An exhaustive evaluation is difficult to be established at that point. But, considering preliminary results from Table 1, we can argue that the Structure-in-5 and the Joint-Venture, since they are patterns governed by organizational characteristics, fit better systems and applications that need open and cooperative components like our B2C example.

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

Table 1. Strengths and Weaknesses of Patterns

6 CONCLUSION

We are working towards the definition of a collection of specific architectural patterns for designing multi-agent information systems. Since multi-agent systems can be viewed as organizational structures of agents that interact to achieve their goals, we propose to use human organizations as a metaphor to suggest generic styles for designing agent systems, with a preference for organizational design theories over social emergence theories.

To this end, the paper has proposed architectural patterns for MAS inspired from organization theory and strategic alliances.

We have proposed an e-business case study to evaluate and compare our organizational patterns with conventional ones with respect to software qualities that are relevant to e-business systems.

The organizational patterns should eventually constitute an architectural macro level. At a micro level we will focus on the notion of social design patterns. Many design patterns can be incorporated into system architecture, such as those identified [Gam95]. For agent inherent characteristics, patterns for distributed, and open architectures like the broker, matchmaker, embassy, mediator, wrapper, mediator are more appropriate.

REFERENCES

[Cas02] J. Castro, M. Kolp and J. Mylopoulos. “Towards A Requirements-Driven Development Methodology: The Tropos Project,” In *Information Systems*, Elsevier, 2002.

[Con00] J. Conallen, *Building Web Applications with*

UML, The Addison-Wesley Object Technology Series, Addison-Wesley, 2000.

[Dus99] P. Dussauge and B. Garrette, *Cooperative Strategy: Competing Successfully Through Strategic Alliances*, Wiley and Sons, 1999.

[Fux01] A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. "Model Checking Early Requirements Specification in Tropos". In *Proceedings of the 5th International Symposium on Requirements Engineering, RE'01*, Toronto, Canada, Aug. 2001.

[Gam95] E. Gamma., R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, 1995

[Kol01] M. Kolp, P. Giorgini, and J. Mylopoulos. "An Organizational Perspective on Multi-agent Architectures". In *Proceedings of the Eighth International Workshop on Agent Theories, architectures, and languages*, *ATAL'01*, Seattle, USA, August 2001.

[Lam93] J.J. Lambin, Strategic marketing: a European approach, McGraw-Hill, 1993

[Min92] H. Mintzberg, *Structure in fives: designing effective organizations*, Prentice-Hall, 1992.

[Sco98] W. R. Scott. *Organizations: rational, natural, and open systems*, Prentice Hall, 1998.

[Sha96] Shaw, M., and Garlan, D. *Software Architecture: Perspectives on an Emerging Discipline*, Prentice Hall, 1996.

[Yu95] E. Yu. Modeling *Strategic Relationships for Process Reengineering*, Ph.D. thesis, Department of Computer Science, University of Toronto, Canada, 1995.