The Structure-in-5 as an Agent Architectural Pattern

Tung T. Do  Stéphane Faulkner  Manuel Kolp
IAG- School of Management, ISYS- Information Systems Research Unit,
University of Louvain, 1 Place des Doyens, Belgium, tel.: +32 10 47 83 95
{do, faulkner, kolp@isys.ucl.ac.be}

Abstract

The structure-in-5 is a model from organization theory used to describe the internal structure of an organization. Since multi-agent systems (MAS) can be structured as organizations of agents, this paper adopts and experiments the structure-in-5 for the design of MAS architectures. We describe the structure-in-5 as an organizational pattern, model it in terms of social and intentional concepts using the i* organizational modeling framework, and give some semi-formal specification using the Formal Tropos language. The paper also revisits and formalizes, in social and intentional terms, conventional architectural elements commonly used to describe system architectures. The structure-in-5 is applied in the design of the architecture of an e-business example. Part of the architecture is expressed in terms of the revisited architectural elements.

1. Introduction

An architectural pattern constitutes an intellectually manageable abstraction of system structure that describes how system components interact and work together. System architectural design has been the aim of proliferating research during the last fifteen years that has produced well-established architectural patterns such as pipes-and-filters, control loop, event-based, partitioning, layers, … [Gar93].

Architectures for MAS can be designed as organizations of agents that coordinate with each other to pursue a set of agreed upon objectives. Taking real-world organizations as a metaphor, architectural patterns for MAS can be based on models from organization theory as described in [Fux01a, Kol01, Gio02].

The structure-in-5 is a well-understood idiom from organization theory detailing the internal structure of an organization. In the paper, we propose to use it to design MAS architectures. We first describe the structure-in-5 as an organizational pattern through the analysis of two real world organizations. We then model it using the i* organizational modeling framework [Yu95] and gives some semi-formal specifications in Formal Tropos language [Fux01]. Finally we apply it to the design of an e-business architecture. In addition, we relate (part of) the e-business architecture to conventional architectural elements revisited in terms of social and intentional primitives.

This research has been conducted within the context of the Tropos project [Cas02, Per01]. Tropos adopts ideas from MAS technologies and requirements engineering, where agents/actors and intentions are used for early requirements analysis [Ant96, Dar93, Yu95]. Tropos is intended as a seamless methodology tailored to describe both the organizational environment of a system and the system itself in terms of the same concepts. In particular, Tropos is founded on the i* modeling framework which offers actors (agents, roles, or
positions), goals, and actor dependencies as primitive concepts for modeling an application during early requirements analysis.

The paper is structured as follows. Section 2 introduces some basic notions from organization theory and it describes the structure-in-5 through the overview of two business organizations. It then models and give a semi-formal specification of a version of structure-in-5 with i* and Formal Tropos, respectively. Section 3 introduces the main lower-level elements composing a system architecture. It then models and formalizes them also in terms of social and intentional concepts. Section 4 describes the application of the structure-in-5 to the design an e-business architecture and expresses part of it in terms of the software components analyzed in Section 3. Finally, Section 5 summarizes the contributions and points to further work.

2. The Structure-in-5

An organization is a social entity with a clear boundary consisting of various types of stakeholders (individuals, physical or social systems) that coordinate on a continuous basis to pursue a set of agreed upon local and global goals [Yos95]. Organization theory (e.g., [Min92, Mor99, Sco98, Yos95]) is the discipline that studies both structure and design for such social entities. To this end, since ancient times, schools of organization theorists have proposed patterns such as the structure-in-5, the matrix, the chain of value and the like to define recurring organizational structures and behaviors. In the following, we will focus on Mintzberg’s structure-in-5. For further information about other organizational patterns we are working on, see [Fux01, Kol01, Gio02].

The structure of an organization can be considered an aggregate of five sub-structures, as described by Mintzberg [Min92]. 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 separate 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).

To model and formalize the structure-in-5 as an organizational pattern, we first analyze two case studies of organizations on which the pattern can be applied: Agate Ltd, an advertising agency and the commercial structure of GMT, a company specialized in telecommunication services.

**Agate.** Agate Ltd is an advertising agency located in Birmingham, UK, that employs about fifty staff, as detailed in Figure 1 [Ben99]. 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,
Figure 1. Organization of Agate Ltd

Figure 2. Agate in Structure-in-5

Figure 2 models the Agate structure-in-5 using the $i^*$ strategic dependency model. $i^*$ is a modeling framework for early requirements analysis [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. 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 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 in situations where the dependee is required to perform a given activity; and resource dependencies require the dependee to provide a resource to the depender. As shown in Figure 2, 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 depender → dependum → dependee.

GMT is a company specialized in telecom services in Belgium. Its lines of products and services range from phones & fax, conferencing, line solutions, internet & e-business, mobile solutions, and voice & data management. 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.

Figure 3. The Commercial Structure of GMT as 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 positioning 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.

Figure 4 abstracts the structures explored in the case studies of Figures 2 and 3 as a Structure-in-5 pattern composed of five actors. The case studies also suggested a number of constraints, whose generality remains to be explored, to supplement the basic pattern:

- 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
- only task dependencies are permitted between the *Middle Line* (as depender or dependee) and the *Operational Core* (as dependee or depender).

![Figure 4. The Structure-in-5 Pattern](image-url)
To specify the formal properties of the pattern, we use Formal Tropos [Fux01], which extends the primitives of $i^*$ with a formal language similar to that of KAOS [Dar93]. 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 [Gio92]. The attributes of a Formal Tropos class denote relationships among different objects being modeled.

In the following, we only present some semi-formal specification for the Strategic Management and Operational Management dependencies. We are currently working on the formalization of the other dependencies.

**Metaclasses**

$\text{Actor} := \text{Actor name [attributes] [creation-properties] [invar-properties] [actor-goal]}

$\text{Dependency} := \text{Dependency name Type name Mode name Depender name Dependee name [attributes] [creation-properties] [invar-properties] [fulfill-properties]}

$\text{Entity} := \text{Entity name [attribute] [creation-properties] [invar-properties]}

$\text{Actor-Goals} := (\text{Goal|Softgoal}) \text{name mode Fulfillment (actor-fulfill-property)}$

**Classes:** Classes are instances of Metaclasses.

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

**Actor** StrategicApex

**Actor** MiddleLine

**Actor** Support

**Actor** Technostructure

**Dependency** Strategic Management

<table>
<thead>
<tr>
<th>Type</th>
<th>SoftGoal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Achieve</td>
</tr>
<tr>
<td>Depender</td>
<td>Technostructure te, MiddleLine ml, Support su</td>
</tr>
<tr>
<td>Dependee</td>
<td>StrategicApex sa</td>
</tr>
</tbody>
</table>

**Invariant** $\text{cond1} \land \text{cond2} \land \text{cond3}$

$\text{cond1: The Strategic management softgoal must be consistent with all changes of the organizational environment}$

$\text{cond2: The Strategic management softgoal takes precedence over dependers' decisions}$

$\text{cond3: Fulfilled(self) } \rightarrow$

$[ \forall \text{dep: Dependency (dep.type = goal } \land \text{dep.depender = sa} \land (\text{dep.dependee = te } \lor \text{dep.dependee = ml } \lor \text{dep.dependee = su}) \rightarrow \Diamond \text{Fulfilled(plandep)} ]$

$[\text{The Strategic management softgoal is fulfilled only if the goal dependencies between the Middle Line, the Technostructure, and the Support as dependees, and the Strategic Apex as depender have been achieved some time in the past}]$
Dependency Operational Management

Type Goal
Mode achieve
Depender StrategicApex sa
Dependee MiddleLine ml
Invariant cond1 ∧ cond2 ∧ cond3

cond1: All goals of type Operational management must be consistent with the Strategic Management softgoal

cond2 : ∃₃ co: Coordination (co.type = task ∧ co.dependee = ml ∧ co.depender = OperationalCore ∧ ImplementedBy(self, co))

[ImplementedBy (self,co) : verifies that the coordination task co is used to implement the Operational Management goal]

cond3: Fulfilled(self) → [ ∃ plandep: Dependency (plandep.type = goal ∧ plandep.depender = ml ∧ plandep.dependee = Technostructure) ]

[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]

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

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)

∀ inst1, inst2 : StrategicApex → inst1 = inst2

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

∀ sa : StrategicApex, te: Technostructure, ma: Middle_Agency, su: Support, dep : Dependency: [(dep.dependee = sa ∧ (dep.depender = te ∨ dep.depender = ma ∨ dep.depender = su)) → dep.type = softgoal]

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

∀ sa : StrategicApex, te: Technostructure, ma: Middle_Agency, su: Support, dep : Dependency: [(dep.depender = sa ∧ (dep.dependee = te ∨ dep.dependee = ma ∨ dep.dependee = su)) → dep.type = goal]

Only goal dependencies are permitted between the Middle Agency and the Support (the same constraint also holds for the Technostructure)

∀ su : Support, ml: MiddleLine, dep : Dependency:
[ (dep.dependee = su ∧ dep.depender = ml) → dep.type = goal ]
Only task dependencies are permitted between the Middle Agency and the Operational Core

\[ \forall \text{ml: MiddleLine, oc: OperationalCore, dep: Dependency} : \\
(\text{dep.dependee = ml } \land \text{dep.depender = oc}) \lor (\text{dep.dependee = oc } \land \text{dep.depender = ml}) \rightarrow \text{dep.type = task} \]

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

\[ \forall \text{te: Technostructure, oc: OperationalCore, dep: Dependency} : \\
\text{dep.dependee = te } \land \text{dep.depender = oc} \rightarrow \text{dep.type = task} \lor \text{dep.type = resource} \]

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

\[ \forall \text{a: Actor, sa: StrategicApex, ml: MiddleLine, te: Technostructure, su: Support, oc: OperationalCore,} \\
\exists \text{dep: Dependency} : [ (\text{dep.depender = ea } \land \text{dep.dependee = ml}) \lor (\text{dep.dependee = ea } \land \text{dep.depender = ml}) \\
\rightarrow \text{a.type = sa } \lor \text{a.type = te } \lor \text{a.type = su } \lor \text{a.type = oc} ] \]

3 System Architectural Components

In addition to patterns and constraints on these patterns, a system architecture involves the description of elements from which systems are built and interactions among those elements. These elements composing a system architecture are: element (component and connector), interface, port, library, instance, iport, configuration, architecture, event, and operation [Lic00].

Figure 5 shows a social and intentional meta-model of these architectural elements in terms of \(i^*\) diagrams. We have previously described the strategic dependency model. \(i^*\) provides a second model, the strategic rationale model allowing to determine, through a means-ends analysis, how goals (including softgoals) can actually be decomposed and fulfilled through the contributions of other actors. A strategic rationale model is a graph with four types of nodes -- goal, task, resource, and softgoal -- and two types of links -- means-ends links and process-decomposition links. For example, to fulfill the goal Architecture Design of Architecture, the strategic rationale analysis postulates a task Build Architecture through which it can be achieved. Tasks are partially ordered sequences of steps intended to accomplish some goal. Tasks can be decomposed into goals and/or subtasks, whose collective fulfillment completes the task. In the figure, Build Architecture is decomposed into four sub-tasks. These decompositions also allow to identify actors that can accomplish a goal, carry out a task, or deliver some resource needed by another actor. For instance, to be able to Build Composite, the Architecture depends on the Configuration to be provided with Composite Input.
An Element is either a Component or a Connector. Components are the computational elements of the architecture bound together by connectors. An Interface primarily defines a set of Ports that ensure connection points through which an element interacts with other elements. It also records element-type information. A Port represents a template for an architectural connection point. A Library models the collection of elements and interfaces defined within an architecture. It also maintains the relationship between an element and the interface through which it interacts. An Instance represents an element that has been instantiated from the template definition of a component or a connector. An IPort (Instance Port) models the port of an Instance. Instantiating an element has the additional effect of instantiating its port templates. This is necessary for distinguishing between the ports of multiple instances of the same basic element. A Configuration is an interconnected set of component and connector instances. The main objective of the configuration (Bind Instances) is to connect component and connector instances to form a more complex construct that will act like a single instance. The goal is fulfilled through a task of the same name, further decomposed into two sub-tasks: Instantiate Element (make an instance by instantiating its corresponding element) and Connect Instances. An Architecture models the full set of design information defined within an architectural specification. It maintains the set of all of the configurations that have been defined and implements configurations as elements (Build Composite task). An Architecture has to find the right library, and add elements, interfaces and configurations to the library. An event is a basic unit of element communication from one point in the architecture to another through a connection. Components interact by initiating and observing events. Ports, as the templates for connection points, handle events. An Operation represents an “executable” command. The behavior of an element is specified as a set of Operations that engage in a set of events in the context of a port.
Part of Figure 5 specification in *Formal Tropos* follows:

**Actor** Connector

**Goal** ConnectComponent

**Actor** Component

**Goal** ProcessComputation

**Actor** Element

**Attribute constant** handle : {Operation}

**Invariant** IsAComponent(self) XOR IsAConnector(self)

[An element is either a component or a connector]

**Task** ProcessOperations(e: Element, op: Operation)

**PRE**

(\(\text{op} \in \text{e.handle}\)) \(\land\) \(\text{NeedToBeProcessed(e, op)}\)

[An operation \(\text{op}\) belongs to the set of operations that an element \(\text{e}\) can handle; \(\text{op}\) needs to be processed by \(\text{e}\); \(\text{op}\) has not yet been processed by \(\text{e}\)]

**POST** Processed(e, op)

[op is processed by \(\text{e}\)]

**Entity** Operation

**Attribute constant** InvokedBy : {Event}

[Set of events that can invoke Operation]

**Entity** Event

**Actor** Port

**Attribute constant** ObservedEvents : {Event} [Set of events a port can observe]

InitiatedEvents : {Event} [Set of events a port can initiate]

Pt: PortType [type of a port]

**Goal** HandleEvent

**Invariant**

\(\forall p_1 : \text{Port}, \exists p_2: \text{Port} \ (p_1 \neq p_2) \land \text{port_map (p_1, p_2)}\)

\(\forall e: \text{ElementType}, \exists p: \text{PortType} \ \text{export_map (e, p)}\)

[port_map checks the legal combinations of port types that may interact through an architectural connection]

[export_map checks the legal port types for a particular element type]

**Task** InitiateEvent (e : event)

**Actor** Port p

**PRE**

(e \(\in\) p.InitiatedEvents) \(\land\) \(\text{NeedToBeInitiated (p, e)}\) \(\land\) not \(\text{InitiatedBy (p, e)}\)

[An event \(e\) belongs to the set of event that can be initiated by \(p\); \(e\) needs to be initiated; \(e\) has not yet been initiated by \(p\)]

**POST** InitiatedBy (p, e) \[e is initiated by \(p\)]

**Task** ObserveEvent

**Actor** Port p

**PRE**

\(\exists q: \text{event} \ \exists q: \text{port (InitiatedBy (q, e) \(\land\) e \(\in\) p.ObservedEvents)}\)

\(\land\) not \(\text{ObservedBy (p, e)}\)

[There exists an event \(e\) initiated by some port \(q\); \(e\) belongs to the set of events that can be observed by \(p\), and \(e\) has not yet been observed by \(p\)]

**POST** ObservedBy(p, e)

[e is observed by \(p\)]

[For each element, if one event is observed by its port, the operation corresponding to this event will be eventually invoked]
4 An E-business Example

This section overviews a typical e-commerce application. We apply the structure-in-5 pattern defined in Section 2 to design the architecture and interpret part of it in terms of the architectural concepts revisited in Section 3.

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, ... on the Internet. Customers can search the on-line store by either browsing the catalogue or querying the database. An on-line search engine allows customers with particular items in mind to search title, author/artist and description fields through keywords or full-text search.

Figure 6 suggests a possible assignment of system responsibilities for E-Media. The architecture follows 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 relevant product information. 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). 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 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.

In the following, we further detail Store Front. This actor is in charge of catalogue browsing and item database searching. It provides on-line customers with detailed information about media items. It is also responsible for supplying a customer with a web shopping cart to keep track of items the customer is buying when using E-Media. Finally, Store Front also initializes the kind of processing that will be done (by Billing Processor) for a given order.

As shown in Figure 7, to accommodate the responsibilities of the Store Front, Operational Core of our structure-in-5 architecture, we decompose the actor into smaller concepts corresponding to architectural components and connectors.

![Figure 7. The Store Front Actor in Terms of Components and Connectors](image)

The Customer Profiler, Catalogue Browser, Interface, Search Engine, Query Displayer and Shopping Cart are architectural components. The Customer Profiler tracks customer data, produces client profiles (personal data), verifies the customer’s login and password (grant access), and records the customer Interface Preferences. The Catalogue Browser manages catalogue navigation to provide the on-line customer with product information. The Interface provides customers with different forms of information retrieval (boolean, keyword, full text, indexed list, etc.). The Search Engine handles the search in database and gives the Query Result that will be further formatted by Query Displayer. The Shopping Cart obtains the Personal Data from Customer Profiler and selected items from Query Displayer.
The Search Engine Wrapper and Browser Wrapper are connectors that mediate the interactions between the Interface and the Search Engine, and the Search Engine and the Query Displayer respectively.

Each component and connector introduced in Figure 7 has its own interface and iports associated with. Figure 8 shows the Search Engine Wrapper connector with its interface and iports.

- **IPort-In (boolean)** is responsible for observing the BooleanConfirmedQuery event that will be initiated by the Interface actor of Figure 7. The same can be defined for other IPort-Ins.
- **Iport-Out (SQL)** is in charge of initiating the SQLTranslatedQuery event that will be further observed by SearchEngine actor of Figure 7.

![Figure 8. The Search Engine Wrapper Actor in Terms of Interface and IPorts](image)

5. Conclusions

Analysts and designers use idioms to structure models and architectures. Multi-agent systems can be described and formalized as organizations of agents that interact to achieve a set of upon agreed intentions. We are working towards the definition of a collection of architectural patterns for multi-agent systems. To this end, the paper focuses on the structure-in-5 and proposes to adapt it for multi-agent architectural design. The structure-in-5 is a well-understood organizational pattern used by organization theorists to describe the structure of real-world organizations. We model the pattern in term of intentional and social primitives from case studies describing real world organizations and propose a semi-formal specification for it. We interpret and formalize in the same intentional and social way the lower-level conventional architectural elements involving (software) components, ports, connectors, interfaces, libraries and configurations. We describe a typical e-business example and apply the structure-in-5 to design its system architecture. Part of it is expressed in terms of the conventional architectural elements we have socially reinterpreted.

Future research directions will extend and formalize precisely the catalogue of organizational patterns and define the sense in which a particular architecture is an instance of such a pattern. We also propose to compare and contrast them with classical software architectural patterns proposed in the literature (pipes-and-filters, layers, event-based, …) using system qualities a multi-agent architecture can or must support.

The organizational patterns should eventually constitute an architectural macro level. At a micro level we will focus on the notion of social agent patterns such as the broker, matchmaker, embassy, mediator, wrapper, mediator [Hay99, Woo99]. They will detail how goals and dependencies identified in an organizational pattern can be refined and achieved.
References


