Organizational Patterns for Early Requirements Analysis

Manuel Kolp
Dept. of Computer Science
University of Toronto
10 King's College Road
Toronto M5S3G4, Canada
mkolp@cs.toronto.edu

Jaelson Castro
Centro de Informática
Universidade Federal de Pernambuco
Av. Prof. Luiz Freire S/N
Recife PE, Brazil 50732-970
jbc@cin.ufpe.br

John Mylopoulos
Dept. of Computer Science
University of Toronto
10 King's College Road
Toronto M5S3G4, Canada
jm@cs.toronto.edu

Abstract

Early requirements analysis is concerned with modeling and understanding the organizational context within which a software system will operate. Such organizational models can describe either the status quo or a desired new status. It is convenient to build such models by deploying organizational patterns which describe often-used organizational structures. The paper proposes a catalogue of patterns which adopt concepts from organization theory and strategic alliances literature. The patterns are modeled using the i framework which offers the notions of actor, goal and actor dependency and specified in Telos. Each proposed pattern is evaluated with respect to a set of quality attributes, such as predictability, adaptability and openness. We illustrate the use of our proposed patterns with a business-to-business example modeling alternative organizational settings. This research has been conducted within the context of a comprehensive software development methodology called Tropos.*

1. Introduction

Modeling the organizational and intentional context within which a software system will eventually operate has been recognized as an important part of the requirements engineering process (e.g., [Ant96, Bub93, Dar93, Yu93]). Such models are founded on primitive concepts such as those of actor (agent, position or role) and goal. In this paper we focus on the problem of defining a set of organizational patterns which can be used as building blocks for constructing such models. Our proposal is based on concepts adopted from organizational theory and strategic alliances literature. Throughout, we use *i** [Yu95] as the modeling framework in terms of which proposed patterns are presented and accounted for.

Section 2 describes some of our organizational patterns and models them using *i** and specify them in Telos. Section 3 introduces a set of desirable quality attributes for comparing organizational models, and offers an evaluation of each pattern with respect to these qualities using the non-functional requirements (NFR) framework [Chu00]. Section 4 describes a business-to-business organizational setting using the proposed patterns, while Section 5 sketches the Tropos project within which this research has been conducted. Finally, Section 6 summarizes the contributions of the paper and points to directions for further research.

2. Organizational Patterns

Organizational theory [Min92, Mor99, Sco98] and strategic alliances [Gom96, Seg96, Yos95] study alternative patterns for (business) organizations. These patterns are used to model the coordination of business stakeholders (individuals, organizations or systems) to achieve common goals. We adopt (some of) these patterns for early requirements analysis, using the strategic dependency model of *i** and specify them in Telos [Myl90].

A strategic dependency model is a graph, where each node represents an actor (an agent, position, or role within an organization) and each link between two actors indicates that one actor depends on another for a goal to be fulfilled, a task to be carried out, or a resource to be made available. In addition, we call the depending actor of a dependency the *depender* and the actor who is depended upon the *dependee*. The object around which the dependency centers (goal, task or resource) is called the *dependum*. By depending on another actor for a dependum, an actor is able to achieve goals that it is otherwise unable to achieve, or not as easily or as well. At

the same time, the depender becomes vulnerable. If the dependee fails to deliver the dependum, the depender would be adversely affected in its ability to achieve its goals.

The model distinguishes among four types of dependencies -- goal-, task-, resource-, and softgoal-dependency -- based on the type of freedom that is allowed in the relationship between depender and dependee. Softgoals are distinguished from goals because they do not have a formal definition, and are amenable to a different (more qualitative) kind of analysis [Chu00].

For instance, in Figure 1 representing the structure-in-5 pattern, the coordination, middle agency and support actors depend on the apex for strategic management. Since the goal "*Strategic Management"* does not have a formal, it is represented as a softgoal (cloudy shape). The middle agency actor depends on the coordination and support actors respectively through goal dependencies "*Control*" and "*Logistics*" represented as oval-shaped icons. The operational core actor is related to the coordination and support actors respectively through the "*Standardize*" task dependency and the "*Non-operational service*" resource dependency.

The **structure-in-5** pattern (Figure 1) consists of the typical strategic and logistic components generally found in many organizations. At the base level one finds the operational core where the basic tasks and operations -- the input, processing, output and direct support procedures associated with running the system -- are carried out. At the top of the organization lies the apex composed of strategic executive actors. Below it sit the logistics, control/standardization and management components respectively support, coordination and middle agency.

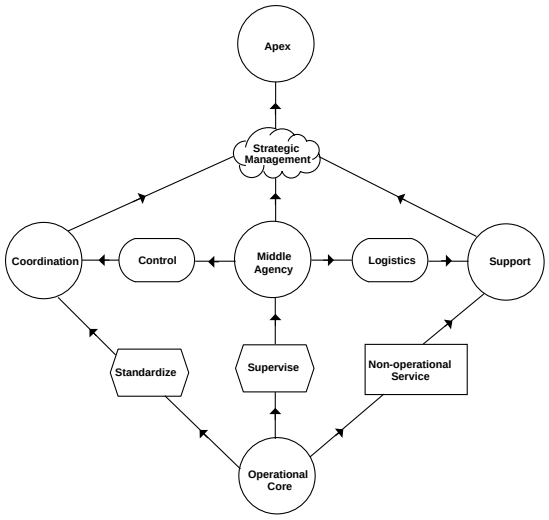

Figure 1 : Structure-in-5

The support component assists the operational core for non-operational services that are outside the basic flow of operational tasks and procedures. The coordination component carries out the tasks of standardizing the behavior of other components, in addition to applying analytical procedures to help the system adapt to its environment. Actors who join the strategic apex to the operational core make up the middle agency.

Figure 2 specifies the structure-in-5 pattern in Telos [Myl90][1]. Telos is a language intended for modeling requirements, design, implementation and design decisions for software systems. It provides features to describe metaconcepts that can be used to represent the knowledge relevant to a variety of worlds – subject, usage, system, development worlds - related to a software system. Our organizational patterns are formulated as Telos metaconcepts, primarily based on the aggregation semantics for Telos presented in [Mot93].

The structure-in-5 pattern is then a metaclass - *StructureIn5MetaClass* - aggregation of five (*part*) metaclasses: *ApexMetaClass*, *CoordinationMetaClass*, *MiddleAgencyMetaClass*, *SupportMetaClass* and *OperationalCoreMetaClass*, one for each actor composing the structure-in 5 pattern depicted in Figure 1. Each of these five components exclusively belongs (*exclusivePart*) to the composite (*StructureIn5MetaClass*) and their existence depend (*dependentPart*) on the existence of the composite. A structure-in-5 specific to an application domain will be defined as a Telos class, instance of *StructureIn5MetaClass* (See Section 4 for an example with the joint venture pattern). Similarly each structure-in-5 component specific to a particular application domain will be defined as a class, instance of one of the five *StructureIn5Metaclass* components.

```
TELL CLASS StructureIn5MetaClass
  IN Class WITH /*Class is here used as a MetaMetaClass*/
      attribute
          name: String
      part, exclusivePart, dependentPart
          ApexMetaClass: Class
          CoordinationMetaClass: Class
          MiddleAgencyMetaClass: Class
          SupportMetaClass: Class
          OperationalCoreMetaClass: Class
END StructureIn5MetaClass
```

Figure 2 : Structure in 5 in TELOS

Figure 3 formulates in Telos one of these five Structure-in-5 components: the Coordination actor. Dependencies are described following Telos specifications for *i** models [Yu95]. The Coordination

[1] A syntactic variant of Telos is used in the paper for simpler presentation.

actor is a metaclass, *CoordinationMetaclass*. According to Figure 1, the Coordination actor is the dependee of a task dependency *StandardizeTask* and a goal dependency *ControlGoal*, and the depender of a softgoal dependency *StrategicManagementSoftGoal*.

```
TELL CLASS CoordinationMetaclass
  IN Class WITH /*Class is here used as a MetaMetaClass*/
      attribute
          name: String
      taskDepended
          s:StandardizeTask
              WITH depender
                      OperationalCoreMetaClass: Class
              END
      goalDepended
          c:ControlGoal
              WITH depender
                      MiddleAgencyMetaClass: Class
              END
      softgoalDepender
          s:StrategicManagementSoftGoal
              WITH dependee
                      ApexMetaClass: Class
              END
END CoordinationMetaclass
```

Figure 3 : Structure-in-5 Coordination actor in TELOS

The **joint venture** pattern (Figure 4) involves agreement between two or more partners to obtain the benefits of larger scale, partial investment and lower maintenance costs. Through, the delegation of authority to a specific joint management actor that coordinates tasks and operations and manages sharing of knowledge and resources, these actors pursue joint objectives and common purpose. Each partner can manage and control itself on a local dimension and interact directly with other principal partners to exchange, provide and receive services, data and knowledge. However, the strategic operation and coordination of such a system and system actors on a global dimension are only ensured by the joint management actor.

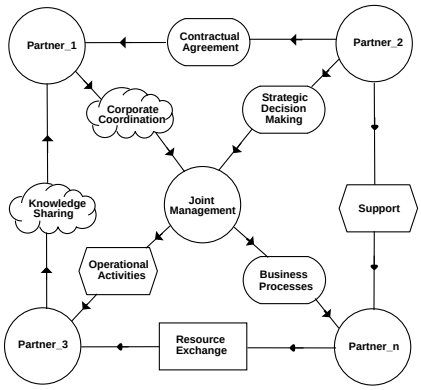

Figure 4 : Joint Venture

Figure 4 describes the pattern in Telos. The composite metaclass *JointVentureMetaClass* is composed of an exclusive and dependent *JointManagementMetaClass*, and an exclusive and independent (i.e., having existence irrespectively of the composite) *PartnerMetaClass*. In Section 4, an instance of this pattern, specific to the media industry application domain, will be modeled (Figure 12) and specified as a Telos aggregation at the class level (Figure 13).

```
TELL CLASS JointVentureMetaClass
  IN Class WITH /*Class is here used as a MetaMetaClass*/
      attribute
          name: String
      part, exclusivePart, dependentPart
          JointManagementMetaClass: Class
      part, exclusivePart /*i.e. exclusive and independent part*/
          PartnerMetaClass: Class
END JointVentureMetaClass
```

Figure 5: Joint Venture in TELOS

The **bidding** pattern involves competitivity mechanisms and actors needed to run an auction. The auctioneer actor runs the show, advertises the auction issued by the auction issuer, receives bids from bidder actors and ensure communication and feedback with the auction issuer. The auctioneer might be a system actor that merely organizes and operates the auction and its mechanisms. It can also be one of the bidders (for example selling an item which all other bidders are interested in buying). The auction issuer is responsible for issuing the bidding. This style implies fast response time and adjustability for the system.

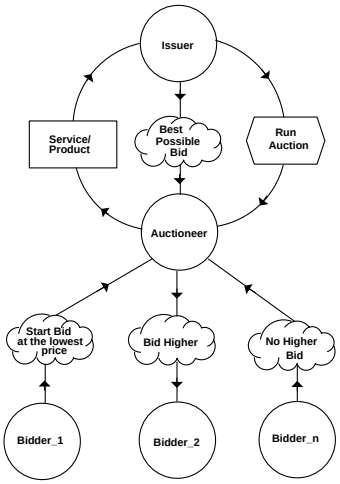

Figure 6: Bidding

The **vertical integration** pattern merges, backward or forward, one or more system actors engaged in related tasks but at different stages of a production process. A

merger can be used to synchronize and control interactions between each of the participant actors. Vertical integrations take place between exchange partners, organizations that are symbiotically related.

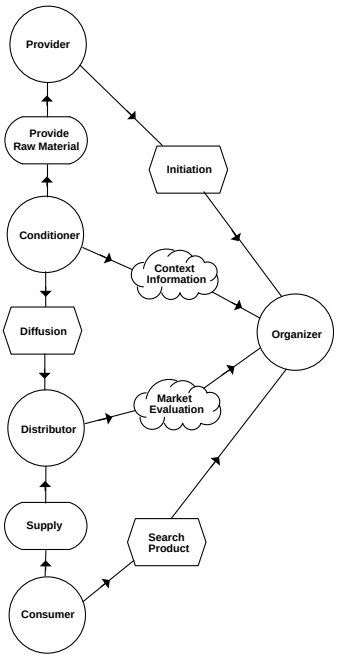

Figure 7 : Vertical Integration

Other organizational patterns are presented in [Kol01]. We briefly describe them here:

The **flat structure** has no fixed structure and no control of one actor over another is assumed.

The **pyramid** is the well-known hierarchical authority structure exercised within organizational boundaries. The crucial mechanism is direct supervision from the apex.

The **takeover** involves the total delegation of authority and management from two or more partners to a single collective takeover actor.

The **arm's-length** implies agreements between independent and competitive but partner actors.

The **hierarchical contracting** identifies coordinating mechanisms that combine arm's-length agreement features with aspects associated with pyramidal authority.

The **co-optation** involves the incorporation of representatives of external systems into the decision-making or advisory structure and behavior of an initiating organization.

3. Organizational Patterns Evaluation

The organizational patterns defined in the previous section can be evaluated and compared using the following quality attributes identified in [Kol01]:

Predictability (1). Actors can have a high degree of autonomy depending on the way they undertake action and communication in their domains. It can be then sometimes difficult to predict individual actor characteristics as part of determining the behavior of an organization at large.

Security (2). Actors are often able to identify their own information and knowledge sources and they may undertake additional actions based on these sources. Strategies for verifying consistency for these sources by individual actors are an important concern in the evaluation of overall organization quality. Indeed, in addition to possibly misleading information and knowledge acquired by individual actors, there is the possibility of external actors acquiring information and knowledge accorded to trusted domain actors.

Adaptability (3). Actors may be required to adapt to modifications in their organization operational environment. They may include changes to the organization's information and knowledge flows or possibly the dynamic introduction of a new actor or the evolution of existing actors.

Coordinability. Actors are not particularly useful unless they are able to coordinate with other actors. This can be realized in two different antagonist ways: **Cooperativity (4)** and **Competitivity (5)**.

Availability (6). Actors that offer services to other actors must implicitly or explicitly guard against the interruption of offered services.

Other quality attributes presented in [Kol01] are **Failability-Tolerance (7)**, **Modularity (8)**, **Aggregability (9)** and **Openess (10)**.

To cope with these quality attributes, the requirements analysts goes through a goal analysis refining them to attributes that are more precise and evaluates alternative organizational patterns against them, as shown in Figure 8. The analysis is intended to make explicit the space of alternatives for fulfilling the top-level quality attributes. Quality attributes are modeled as clouds, organizational patterns are represented as operationalized quality attributes (saying, roughly, "make the organizational operational environment of the software system *pyramid, structure-in-5, co-optation, joint venture, arm's-length*-based, …").

The evaluation results in contribution relationships from the organizational patterns to the quality attributes. "+", "++", "-", "--" respectively model partial/positive, sufficient/positive, partial/negative and sufficient/negative contributions [Chu00]. Design rationale is represented by claim intentions 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 quality attributes. A check-mark "✔" indicates an accepted attribute while a cross "✘" labels a denied attribute.

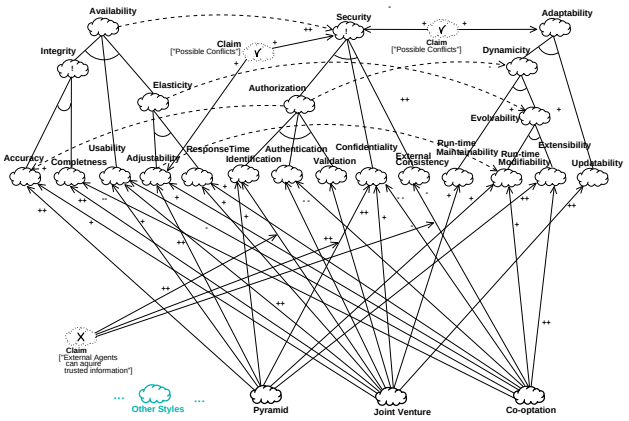

Figure 8: Partial Evaluation for Organizational Patterns

Figure 8 presents a partial evaluation of organizational patterns. It refines top-level quality attributes *Adaptability*, *Security* and *Availability*. *Adaptability* is AND-decomposed into more precise attributes *Dynamicity* and *Updatability*. *Dynamicity* deals with the way the operational environment can be designed to be dynamically and easily changed. *Updatability* is strategically important for the viability of the system, and the competitive changing context of the organizational environment itself. Comparable analysis are carried out in turn for newly identified quality attributes as well as for *Security* and *Availability*. More specific quality attributes are identified in the figure: *Integrity* (*Accuracy*, *Completeness*), *Usability*, *Response Time*, *Maintainability*, *Updatability*, *Confidentiality*, *Authorization* (*Identification*, *Authentication*, *Validation*). Table 1 summarizes the correlation catalogue for the organizational patterns and top-level quality attributes we have considered.

	1	2	3	4	5	6	7	8	9	10
Struct-5	+	+		+	-	+		++	++	
Joint-Vent	+	+	++	+	-	++		+	++	++
Bid	--	--	++	-	++	-	--	++		++
Vert Integr	+	+	-	+	-	+	--	--	--	--
Flat	--	--	-			+	+	++	-	+
Pyramid	++	++	+	++	-	+	--	-		+
Takeover	++	++	-	++	--	+		+	+	-
Arm's-Lgth	-	--	+	-	++	--	++	+		-
Hierch Ctr			+	+	+	+		+	+	+
Coopt	-	-	++	++	+	--	-	--		+

Table 1 : Correlation Catalogue

4. A business-to-Business Example

To illustrate the use our patterns, we consider a business-to-business setting describing the media industry. *Media Shop* is a specialized store selling and shipping different kinds of media items such as books, newspapers, audio CDs, videotapes, and the like. To increase its market shares, *Super Market* is also interested to sell and ship medias items. *Media Library* and *Renting Media* are respectively non-profit and commercial businesses specialized in renting medias. All these retailer actors are supplied with the latest releases by *Independent Distributor* and *Corporate Distributor*. At the production level, *Editor* is specialized in the book business and deals with the newspaper and magazine industry, *Film Studio* makes movies while *Record Label* works in the music industry and *Games Design* creates video and computer games. These actors hire *authors* to write, design, compose and make respectively books and articles, games, music, and movies.

All these actors have agreed to develop *Medi@*, an internet-based information system supporting business-to-business capabilities to facilitate and improve business interactions, reduce costs and delays of traditional information and communication means. Final customers will also be able to use *Medi@* to browse the catalogue, query the item database and order on-line items.

During early requirements analysis, the analyst captures the intentions of stakeholders. These are modeled as goals which, through some form of a goal-oriented analysis, eventually leads to the functional and non-functional requirements of the system-to-be [Dar93]. In *i**, early requirements are assumed to involve social actors who depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. The *i** framework [Yu95] includes the *strategic dependency model* for describing the relationships among actors, as well as the *strategic rationale model* for supporting the reasoning that each actor goes through concerning its relationships with other actors.

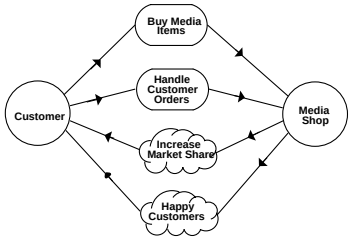

Figure 9: Dependency Model for *Customers* and *Media Shop*

We have already described the strategic dependency model in Section 2. We start the organizational modeling by analysing the business-to-consumer aspects of the setting. The two main B2C stakeholders are *Customer* and *Media Shop*. As shown in Figure 9, the customer has one relevant goal *Buy Media Items*, while the media store has goals *Handle Customer* and softgoals *Orders*, *Happy Customers*, and *Increase Market Share*.

The *i** strategic rationale model determines through a means-ends analysis how these identified goals (including softgoals) can actually be 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. It captures the relationship between the goals of each actor and the dependencies through which the actor expects these dependencies to be fulfilled.

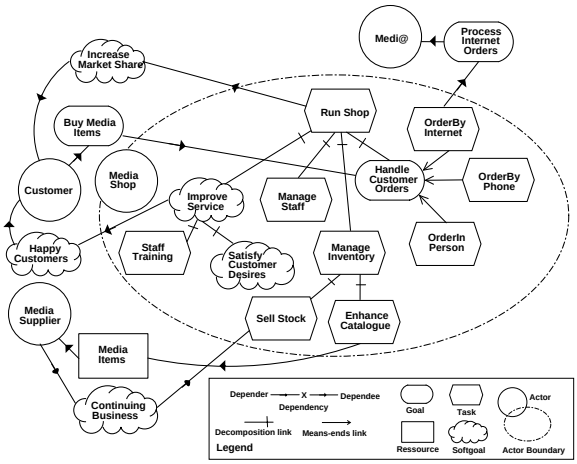

Figure 10: Means-ends analysis for the softgoal *Increase Market Share*

Figure 10 focuses on one of the (soft)goal *Increase Market Share* identified in Figure 9. The analysis postulates a task *Run Shop* (represented in terms of a hexagonal icon) through which it can be fulfilled. Tasks are partially ordered sequences of steps intended to accomplish some (soft)goal. Tasks can be decomposed into goals and/or subtasks, whose collective fulfillment completes the task. In the figure, *Run Shop* is decomposed into goal *Handle Customer Orders*, tasks *Manage Staff* and *Manage Inventory*, and softgoal *Improve Service* which together accomplish the top-level task. As shown in the figure, subgoals and subtasks can be specified more precisely through refinement (see [Cas01]).

Early requirements analysis is 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 inter-dependencies. We consider the system one or more actors contributing to the fulfillment of stakeholder goals, along with other actors from the system's operational environment. For our example, the *Medi@* system, already identified in Figure 10, is introduced as an actor in the strategic dependency model depicted in Figure 13.

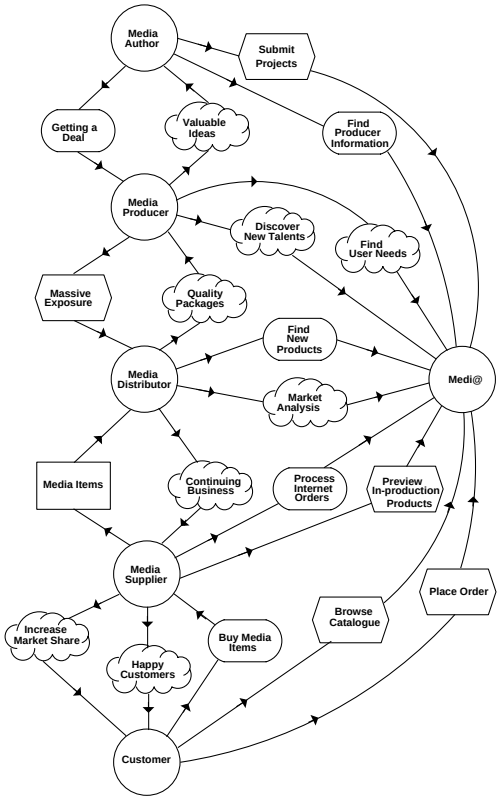

Figure 11 : Vertical Integration and Media Industry

The traditional setting of the media industry described above could be represented as an instantiation of the vertical integration pattern as shown in Figure 11. The media life cycle (from author to consumer) is a sequence composed by actors *Media Author*, *Media Producer* (Generic actor for *Editor*, *Movie Studio*, *Record Label*, *Games Design*), *Media Distributor* (*Independent Distributor* and *Corporate Distributor*), *Media Supplier* (*Media Shop*, *Super Market*, *Media Library* and *Renting Media*) and *Customer* relying on each other: *Media Author* and *Media Producer* negotiate artistic deals, *Media Distributor* ensures massive exposure of media items while *Media Supplier* interacts with *Customer*. Each of these actors use the *Medi@* actor system for specific needs and goals modeled by dependencies: *Media Author* needs to find information about *Media Producer* to submit works, ideas and projects, *Media Producer* would like to find talents and get on-line reports identifying new needs, *Media Distributor* wants to enlarge her markets with on-line distribution and conduct e-market analysis, *Media Supplier* would like to be provided with e-commerce facilities and preview in-production works; finally, *Customer* would like to consult products, information about the media industry actors and place on-line orders.

More precise organizational settings can be modeled using the patterns defined in Section 2. For instance if the

actor *Media Producer* identified in Figure 11 is driven by attributes cooperativity, adapatability and aggregability, it could be organized, according to Table 1, as a joint venture as shown in Figure 12.

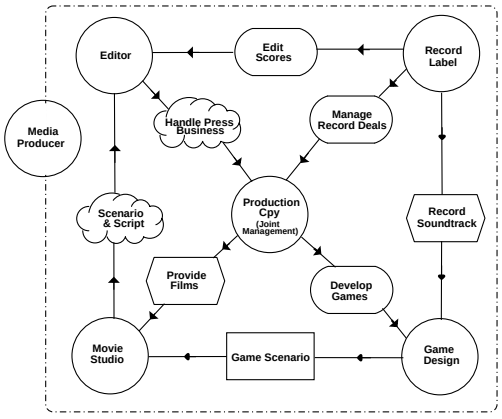

Figure 12 : Joint Venture and Media Producer

Each partner actor composing *Media Producer* is specialized in one or several specific media production areas: *Editor* handles press business and contributes to provide film scenarios and script ideas, *Movie Studio* makes films and video clips for *Record Label* which handles record deals and records soundtracks for *Movie Studio* and *Game Design*. This last actor develops games and relies on *Movie Studio* for game plots. Each actor is responsible for its own business management. The corporate strategic management, however, is assumed by *Production Cpy*, the joint management actor.

```
TELL CLASS MediaProducerClass
      IN JointVentureMetaClass WITH
      attribute
          name: String
      part, exclusivePart, dependentPart
          ProductionCpyClass : JointManagementMetaClass
      part, exclusivePart /*i.e. exclusive and independent part*/
          EditorClass: PartnerMetaClass
          RecordLabelClass: PartnerMetaClass
          MovieStudioClass: PartnerMetaClass
          GameDesignClass: PartnerMetaClass
END MediaProducerClass
```

Figure 13: Media Producer Joint Venture in TELOS

Figure 13 formulates this media producer joint venture in Telos. *MediaProducerClass* is a Telos class, instance of the *JointVentureMetaclass* specified in Figure 5. This aggregation is composed of an exclusive and dependent part *ProductionCpyClass* instance of *JointManagementMetaClass* and four exclusive and independent parts *EditorClass*, *RecordLabelClass*, *MovieStudioClass* and *GameDesignClass*, instances of *PartnerMetaClass*.

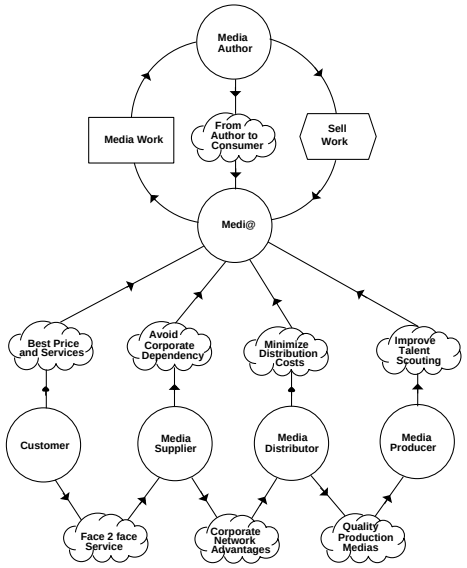

Figure 13 : Bidding Pattern and Media Industry

Finally, we consider an alternative for the organizational of the media industry. Indeed, these days, the traditional view and philosophy of the media industry is changing with the coming of new media formats, protocols and technologies. The vertical integration depicted in Figure 11 might be no longer relevant. To adapt to non-predictable, open and competitive marketplaces, a credible choice could be the bidding pattern as shown in Figure 13. *Customer*, *Media Supplier*, *Media Distributor* and *Media Producer* are on the same level, and no longer in a vertical dependency. Each of them try to get the best deal and propose to *Media Author* concurrent "bids". The system is acting as the only intermediate assuming that works and projects from authors are available on-line.

Customer, *Media Supplier*, *Media Distributor* and *Media Producer* can still rely on each other through softgals dependencies modeling traditional "non-electronic" advantages: face-to-face, discussions, personalized services, …

5. A Requirements-Driven Methodology

This research is conducted in the context of Tropos [Cas01], a software system development methodology which is founded on the concepts of *actor* and *goal*. Tropos is intended as a seamless methodology which describes in terms of the same concepts the organizational environment within which a system will eventually operate, as well as the system itself. The proposed methodology supersedes traditional development techniques, such as structured and object-oriented ones in the sense that it is tailored to systems that will operate

within an organizational context and is founded on concepts used during early requirements analysis. To this end, we adopt the concepts offered by *i** [Yu95], a modeling framework offering concepts like *actor*, *agent*, *position* and *role*, as well as social *dependencies* among actors, including *goal*, *softgoal*, *task* and *resource* ones.
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, where the system-to-be is described within its operational environment, along with relevant functions and qualities.
- Architectural design, where the system's global architecture is defined in terms of subsystems, interconnected through data, control and dependencies.
- Detailed design, where behaviour of each architectural component is defined in further detail.

6. Conclusion

Modellers need to rely on informal styles, patterns, or idioms, to build their models, whatever the purpose. We argue that, as with other phases of software development, early requirements analysis can be facilitated through the use of organizational patterns. This paper proposes a catalogue of patterns which adopt models and concepts from organization theory and strategic alliances literature. These patterns are modeled with *i**, specified as Telos metaconcepts and evaluated with respect to a number of relevant qualities using the NFR framework. The paper also illustrates the use of these patterns with a case study.

Our organizational approach complements well proposals for multi-agent systems which are becoming increasingly important for business software. Considering real world organizations as a metaphor, systems involving many software actors, such as MAS could benefit from the same organizational models.

Future research directions include formalizing the patterns that have been identified, as well as formalizing the sense in which a particular model is an instance of such a pattern. We also propose to compare and contrast these patterns to software architectural styles proposed in the literature.

7. Acknowledgements

The Tropos project has been partially funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and Communications and Information Technology Ontario (CITO), a Centre of Excellence funded by the province of Ontario.

8. References

[Ant96] A. I. Anton, " Goal-Based Requirements Analysis", *Proc. 2nd Int. Conf. On Requirements Analysis (ICRE'96)*, 1996, pp.136-144.

[Bub93] J. A. Bubenko, "Next Generation Information Systems: an Organizational Perspective", *Proc. of the International Workshop on Development of Intelligent Information Systems*, Niagara-on-the-Lake, Ontario, Canada, April 1991, pp. 22-31.

[Cas01] J. Castro, M. Kolp and J. Mylopoulos. "A Requirements-Driven Development Methodology," To appear in *Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering (CAiSE'01)*, Interlaken, Switzerland, June 2001.

[Chu00] L. K. Chung, B. A. Nixon, E. Yu and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*, Kluwer Publishing, 2000.

[Dar93] A. Dardenne, A. van Lamsweerde,and S. Fickas, "Goal–directed Requirements Acquisition", *Science of Computer Programming, 20*, 1993, pp. 3-50.

[Gom96] B. Gomes-Casseres. *The alliance revolution : the new shape of business rivalry*, Cambridge, Mass., Harvard University Press, 1996.

[Kol01] M. Kolp, and J. Mylopoulos, *Architectural Styles for Information Systems: An Organizational Perspective*, Dept of Computer Science, University of Toronto, Tropos Working Paper, January 2001. http://www.cs.toronto.edu/~mkolp/tropos.

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

[Mot93] R. Motschnig-Pitrik, "The Semantics of PartsVersus Aggregates in Data/Knowledge Modeling", Proc. of the 5th Int. Conference on Advanced Information Systems Engineering (CAiSE'93), Paris, June 1993, pp 352-372.

[Mor99] J. Morabito, I. Sack and A. Bhate. *Organization modeling : innovative architectures for the 21st century*, Upper Saddle River, N.J., Prentice Hall PTR, 1999.

[Myl90] J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis: *"Telos: Representing Knowledge About Information Systems"* in *ACM Trans. Info. Sys.*, 8 (4), October 1990, pp. 325 – 362.

[Sco98] W. R. Scott. *Organizations : rational, natural, and open systems*, Upper Saddle River, N.J., Prentice Hall, 1998.

[Seg96] L. Segil. *Intelligent business alliances : how to profit using today's most important strategic tool*, New York, 1996.

[Yos95] M.Y. Yoshino and U. Srinivasa Rangan. *Strategic alliances : an entrepreneurial approach to globalization*, Boston, Mass., Harvard Business School Press, 1995.

[Yu93] E. Yu, "Modeling Organizations for Information Systems Requirements Engineering", *Proceedings of the First IEEE International Symposium on Requirements Engineering*, San Jose, USA, January 1993, pp. 34-41.

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