

Supporting Social Organization Modelling in Cooperative Work Using Patterns

José Luis Isla Montes¹, Francisco Luis Gutiérrez Vela², and Miguel Gea Megías²

¹ Dpt. Computer Languages and Systems, University of Cádiz,
Facultad de Ciencias del Trabajo, Av. Duque de Nájera,
6 dup., 11002 Cádiz, Spain
jose.luis.isla@uca.es

² Dpt. Computer Languages and Systems, University of Granada,
E.T.S.I. Informática, c/ Daniel Saucedo Aranda, s/n,
18071 Granada, Spain
{fgutierr, mgea}@ugr.es

Abstract. A key aspect for the development of CSCW systems is the previous study of the social organization of the members that participate in the collaborative process. Organizations have static and dynamic aspects that are relevant to identify in order to predict the group behaviour, such as changes in member roles. Modelling the organizational structure facilitates the precise description of the responsibilities of each member and the dependences among them, guiding the software analysis and design. This paper proposes the definition and application of organizational patterns to improve the organization modelling. This technique is incorporated in AMENITIES, a complete methodology, developed in our research group, for analysis and design of cooperative systems.

1 Introduction

Systems for cooperative work are inherently complex and their development requires specific methods and modelling techniques with capacity to accurately specify their requirements. We consider that a key aspect for the development of a cooperative system is to know how the members of the cooperative group are organized to achieve the common goals.

Organizational structures are based on roles, guiding user responsibilities and relationships with other participants. Thus, this organizational structure may change in time for several reasons (responsibilities are modified, dependencies are created or overridden, new goals are set, etc.), therefore the system is evolving continuously.

We define a social structure as a collection of actors responsible for carrying out group tasks and a set of social dependencies among them.

Our approach starts from an analysis of cooperative systems as a social structure [1] which evolve in time (for example, actors can assume different roles depending on their capabilities, responsibilities and dependences can be modified because of new work strategies, etc.). This approach is used in the AMENITIES methodology

[6,7], which has been developed in our research group for analysis and design of cooperative systems.

Different contributions related to social structures modelling have been proposed [2,3], most of them have been used for the representation of MAS (Multi-Agent Systems) [4]. These models focus on the static architecture of the system, considering agents as structural elements within a complex organization. Nevertheless, for the specification of social organizations in information systems it is also very important to reflect the dynamic nature of the organization as well as its architecture.

Conceptual/analysis patterns [11] are a valuable technique to facilitate the conceptual modelling of a system. In this work we present how it is possible to define and reuse common organizational structures, including static and dynamic properties, as organizational patterns [10] in AMENITIES. Thus, we can improve modelling decision and make specifications faster, more comprehensible and easier to maintain.

In the following section we present a conceptual model to define an organizational structure and its relationships. Next, in section 3, we briefly discuss the use of AMENITIES methodology to model organizations. In section 4, we show how the methodology allows us to represent general organizational structures (organizational patterns) that facilitate the modelling process. In section 5 we present a template for the uniform pattern description and we use this to describe a case study in section 6. Finally, conclusions and future research are presented.

2 A Conceptual Model of Organization

Group modelling techniques are based on concepts related to user (role, activity, task, etc.) enriched with descriptions of social organization aspects. In order to model a real organization it is necessary to consider static and dynamic aspects. Static issues are the structure of the organization, their dependencies, etc. Dynamic aspects should cover temporal changes in responsibilities or composition, laws imposed, reaction to certain events, etc.

Figure 1 shows a conceptual model (using a UML class diagram) which allows the social organization of a system to be described. This figure reflects the most important elements that appear in any organization and it is similar to those which have traditionally been used in collaborative systems modelling [5].

The conceptual model shows an *organization* mainly composed of *actors*. The actor concept includes *users* and *organizational units*. An example of organizational unit is the group concept. A group is defined as a set of users who temporarily take part in common tasks. Some of these organizations are stable in time while others are highly dynamic.

At any time, an actor plays a *role* in the system. Playing a role implies the possibility or capability to perform *activities* associated with such a role.

Relationships between roles can appear. In this way, we can model associations of a different nature, for example, the possibility of an actor passing from one role to another. *Organizational dependences* also appear between organizational units

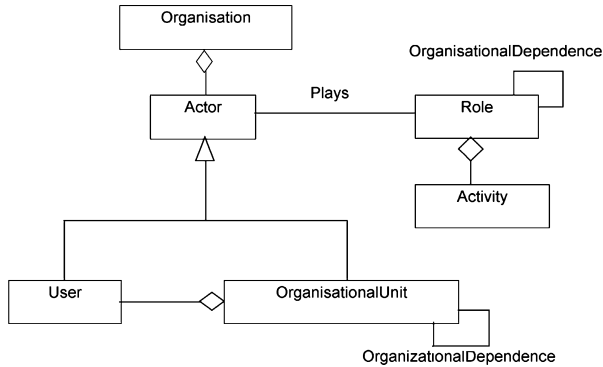


Fig. 1. Conceptual Model of Organization

for structural dependence modelling, for example, the inclusion of an organization into another.

3 AMENITIES Methodology: The Organizational View

AMENITIES [6,7] (A METHodology for aNalysis and desIgn of cooperaTive systEmS) is a methodology, developed in our research group, based on user behaviour and task models for analysis, design and development of cooperative systems. It uses a UML-based notation, called COMO-UML [8], adding several notational elements to capture concepts of a higher level, as for example group, role, actors, organization, etc. In order to model dynamic aspects of the organizational structure of a system AMENITIES introduces two kinds of constraints:

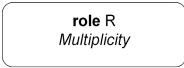
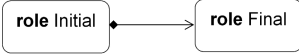
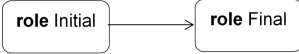

- Law: It defines a constraint imposed by the organization in the group structure. The laws are imposed by the environment or by higher organizations,
- Capability: It defines the ability that an actor or group may acquire within the system. This capability may be linked to cognitive aspects (learning), skills (being an expert in), or features (characteristics or attributes).

In this way, participants could acquire new capabilities, apply new work strategies, etc. In all cases, it is necessary to satisfy the laws which govern the general system behaviour.

The methodology provides different system views (organization, cognition, interaction and information view) which constitute the AMENITIES Cooperative Model. The purpose is to give a description of a system independently of its implementation, providing a better understanding of the problem domain.

In this paper we focus on the organizational view [8] to model group structure and behaviour. The organizational view uses an extension of UML state machine diagrams to represent the organization according to the different roles that the actors could carry out in the system. The set of activities related with each role are described in the cognitive view [8]. Table 1 briefly describes some of the notation elements used in the organization view.

Table 1. COMO-UML notation elements for organizational modelling

Symbol	Semantic
	Role. <i>R</i> is a role that a number of actors, limited by <i>Multiplicity</i> , can play at a given moment in an organization. It is a state belonging to a state machine which represents the dynamism of an organization.
	Additive Transition. An actor who is playing an <i>Initial</i> role may also carry out the <i>Final</i> role. If this transition is labelled with a constraint (law or capability) it must be fulfilled.
	Transition of change. An actor who is playing an <i>Initial</i> role abandons it to adopt a <i>Final</i> role. If this transition is labelled with a constraint it must be fulfilled.
	Decision Box. This diagram determines, through restrictions labelling its outgoing transitions, the different alternatives with respect to the roles to be played. When various alternatives become true, the system or the actor is responsible for choosing the alternative.

4 Organizational Patterns Modelling

From the introduction in Software Engineering [9], patterns have become a valuable instrument for the description and reuse of the empiric knowledge used throughout the different phases which make up the software life cycle. Nonetheless, most effort has focused on the use of patterns during the design phase of software.

We consider that the decisions taken during the early stages of requirements analysis and conceptual modelling have a decisive influence on the final product and the remaining stages of its life cycle. The use of patterns (called analysis or conceptual patterns [11]) in these initial stages has a crucial importance, their use improves decision-making and the specification is faster, more comprehensible and easier to maintain. Therefore the modelling of the organizational structure and behaviour can benefit from the systematic use of specific conceptual patterns (organizational patterns) within a development methodology [13,14].

Different studies dealing with organizations [2,12] have proposed general social structures which often govern these complex systems. For example, organization styles such as structure-in-5, joint venture, vertical integration, pyramid, etc. These structures are suitable to model the whole organization focusing on the distribution of their components (organizational units or individuals) in order to obtain common goals. Nevertheless other social structures (of finer grain), such as broker, mediator, embassy, etc., can often appear within organizations.

Our intention is to encapsulate these organizational structures in the form of organizational patterns with the aim of reusing them to facilitate the modelling of the organization view. They provide a common vocabulary that improves the communi-

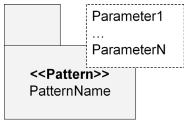
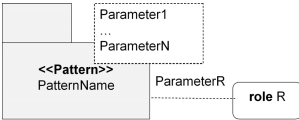
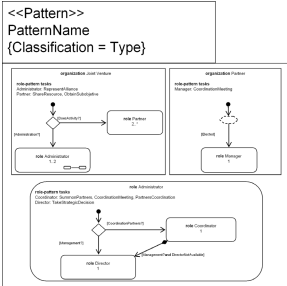

cation and discussion of these organizational structures. In addition, the models are easier to comprehend and maintain.

Some interesting works [10] have been done about organizational patterns focusing on organizations that build (or use, or administer) computer software, but they are not oriented to facilitate organization modelling and lack a specific notation.

In our case, we have defined a complete UML profile [15] to model software patterns in general. Therefore, we use this profile, together with COMO-UML notation, to model organizational patterns.

To understand the case study in section 6, table 2 details the notation elements that we will use:

Table 2. Notation for patterns

Symbol	Semantic
	<p>External View. A parameterized package represents a pattern. The parameters specify which elements of the pattern will be bound to particular elements in a model.</p>
	<p>Binding. A binding consists of connecting each one of these particular elements with the pattern symbol by means a dotted line labelled with the corresponding parameter.</p>
	<p>Pattern Definition. The models that represent the pattern are defined inside a UML-package indicating the pattern’s name and its classification.</p>
	<p>Pseudo-Element. With a dotted hexagon we represent the uncertainty about part of a diagram.</p>

5 A Template for the Uniform Description of Patterns

In order to provide the necessary information that allows us to compare, learn and apply patterns, we use a structured template. This template is divided into different sections:

Name: It should be significant and reflect its essence in few words.

Alias: Another name for this pattern.

Classification: According to some previously established taxonomy.

View: AMENITIES Cooperative Model view where the pattern can be used.

Problem: What is the scenario that we need to describe?

Context: In what situations can you apply it? How to recognize these situations? It shows the preconditions under which the problem and its solution can happen.

Participants: Description of the elements that take part in the pattern definition and their responsibilities.

Solution: A model which describes the participants, structure and behaviour, using COMO-UML notation. It can include variants.

Explanation: Description of the proposed solution.

Example: Application to a real case.

Related Patterns: Other related patterns belonging to the same catalogue. For example, patterns that can be applied (before or later), alternatives, etc.

6 A Case Study

In order to apply the template and the notation defined in this paper, we describe an organization pattern called *Joint Venture*.

A Joint Venture is a typical organization in business companies, where several companies (partners) form a strategic alliance to achieve a common goal which is difficult to obtain separately. In this way, each partner can increase his benefits (cost, product viability, maintenance, etc.). Nevertheless, this organizational structure often appears in different contexts using a different scale.

Name: *Joint Venture*

Alias: Unknown

Classification: Organization

View: Organizational

Problem: It describes an organization of actors (*partners*), where each one has a specialized task for a common goal. Each partner shares his resources and capabilities to achieve large-scale goals, so the advantages for each member are increased (minimisation of cost investment, maintenance reduction, increase of benefits, shared resources, etc.) which each one alone could not obtain by itself.

Context:

- The common goal can be broken down into several sub-objectives.
- Each partner is responsible for some of these sub-objectives.

Participants:

- *Partner* (role)
 - They perform the needed tasks to achieve some of the assigned sub-objectives (*ObtainSubobjective* task)
 - They share their resources with other partners (*ShareResource* task)
- *Administrator* (role)
 - He is responsible for the external relationships of the coalition (*RepresentAlliance* task)
- *Administrator::Director* (role)
 - He chooses the best strategy for the coalition (*TakeStrategicDecision* task)

- *Administrator::Coordinator* (role)
 - o He is responsible for scheduling meetings and communications for alliance partners (*SummonPartners* task)
 - o He performs coordination meeting with partners (*CoordinationMeeting* task)
 - o He decides coordination tasks (*PartnersCoordination* task)
- *Partner::Manager* (role)
 - o He is the member in those meetings where the coalition is requested (*CoordinationMeeting* task)

Solution: See Figure 2.

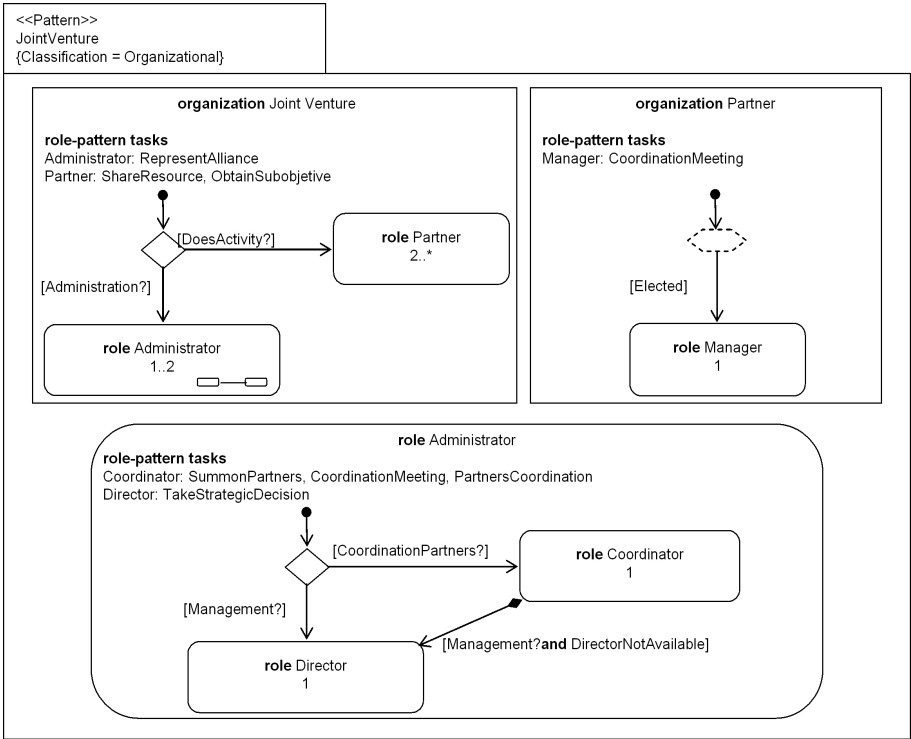


Fig. 2. Solution to the *Joint Venture* Problem

Explanation: When an actor has the necessary capability to carry out a task (e.g. manufacturing one of the pieces of an aeroplane) he will play the *Partner* role. It is important to observe the role multiplicity, indicating that it must have at least two partners in the coalition.

The *role-pattern tasks* section specifies the essential tasks that each actor should perform in the context of the pattern. A role may also perform other kinds of activities, but the essential tasks must be reflected here.

The *Partner* role must carry out at least the *ShareResource* task (the partners must be able to share resources with each other) and the *ObtainSubobjective* task (each

partner must accomplish a specific part of the organization's final goal, for example, to manufacture some of the final product elements). Moreover, as is shown in the partner organization diagram, an actor who carries out the *Manager* role is an actor who has been elected by others actors of the organization. The *Manager* is responsible for holding meetings with the *Coordinator* when it is necessary (*CoordinationMeeting* task). This is a common task for *Coordinator* and *Manager* role.

When an actor achieves administration capability in the *Join Venture*, then he/she can act as *Administrator* (note that only one or two actors can take part in this role). In this situation, an actor must perform at least the *RepresentAlliance* task, assuming responsibility for the external relations of the alliance. If this actor can also achieve the capability of coordinating *partners*, then he plays the *Coordinator* role (only one actor takes part in this role) and therefore, he will have to meet the *managers* of the partner organizations when necessary (*SummonPartners* and *CoordinationMeeting* tasks) as well as performing coordination among partners (*PartnersCoordination* task).

In this organization, an actor who has capabilities to manage the strategy of the alliance is responsible for the *Director* role whose main function is to take strategic decisions for the alliance (*TakeStrategicDecision* task).

In the above diagram we also describe, using an additive transition, the situation in which the actor plays the *Coordinator* role as well as the *Director* role. This situation happens when the *Coordinator* has management capability and the *Director* is not available (i.e. the *Coordinator* acts as a substitute of the *Director*).

Example: A real example of this kind of organization is the Airbus company, which is the coordinator among different partners for manufacturing and selling aircraft: Aerospatiale (it develops and builds the cockpit), DASA (the fuselage), British Aerospace (the wings), CASA (the tail including horizontal and vertical stabilizers) and finally the overall assembly is performed in Aerospatiale.

Another example is the organizational structure to carry out a large-scale design project (see diagram below). In this case, the project (*DesignProject* organization) is divided into design sub-projects and each one is offered to different design groups (*DesignSubProjectGroup* role) which should have capability to carry out the sub-project tasks (*[SubProjectTask?]*).

The coordination process is usually carried out by a manager (*SubProjectsCoordinator* role) who communicates with the different managers of sub-projects (*SubProjectManager* role).

Finally, there is an agent who is responsible for leading the project (*ProjectDirector* role). We can observe, in the *DesignSubProjectGroup* organization, the existence of other necessary roles in this particular organization which are not bound to the pattern.

Figure 3 shows how the Joint Venture Pattern facilitates the modelling and description of the design project organization.

Related Patterns: In order to achieve a common goal through several sequential stages it is possible to use the *Production Line Pattern* [14].

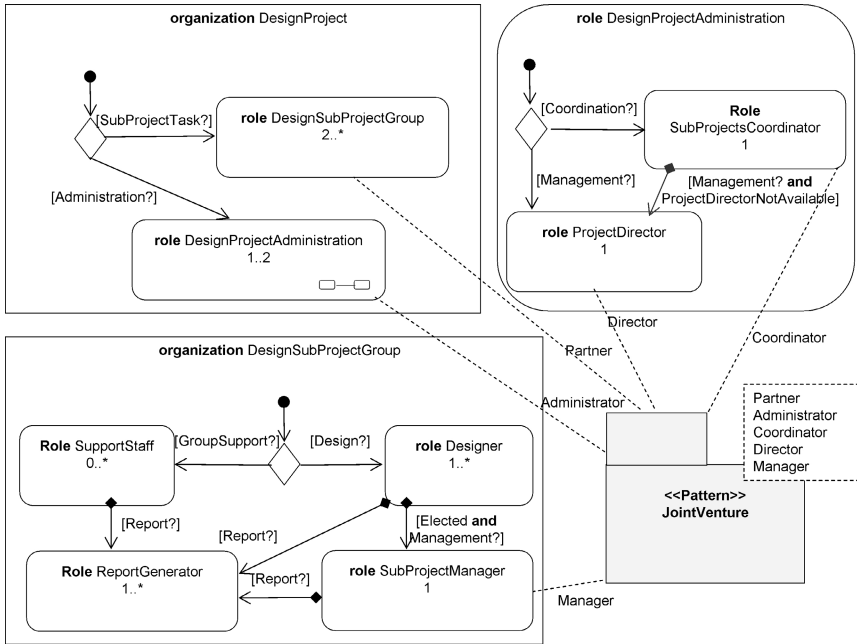


Fig. 3. Modelling and Description of The Design Project Organization

7 Conclusions and Future Work

We highlight the importance of a group-centred methodology to improve the development of CSCW systems, for example those supporting the cooperative design. Therefore, we present AMENITIES, a group-centred methodology for analysis and design of cooperative systems.

An important step is the social organization modelling of the members that participate in the collaborative process. We have shown that AMENITIES is suitable to model static as well as dynamic aspects of an organizational structure.

We propose the definition and application of organizational patterns in AMENITIES, improving organization modelling decision and making specifications faster, more comprehensible and easier to maintain. We introduce a template for the uniform description of patterns and we present a specific notation for pattern modelling. As an example, we describe an organizational pattern and apply it to a particular case.

At this time we are working on the construction of a catalogue of organizational patterns for a future pattern language which could integrate other authors' patterns. Moreover, we are exploring the specification and use of other types of patterns within the remaining views (cognition, interaction and information view) of the AMENITIES Cooperative Model [8].

Acknowledgement

This research is supported by the AMENITIES Project (CICYT TIN2004-08000-C03-02).

References

1. Bubenko J.A.: Next Generation Information Systems: an Organizational Perspective. Proceedings of the Intl. Workshop on Development of Intelligent Information Systems, Niagara-on-the-Lake, Canada, (1991) 22-31
2. Fuxman, A., Giorgini, P., Kolp, M. and Mylopoulos J.: Information systems as social structures. Proceedings of the 2nd International Conference on Formal Ontology in Information Systems, FOIS'01, Ogunquit, USA, (2001) 10-21
3. Kolp M., Giorgini P., Mylopoulos J.: Information systems development through Social Structures. Proceedings of the 14th international Conference on Software Engineering and Knowledge Engineering, Italy, (2002) 183-190
4. Ferber J., Gutknecht O.: A Meta-Model for the Analysis and Design of Organization in MAS. Proc of Int. Conf. in Multi-Agent Systems, IEEE Computer Society, (1998)
5. Van Welie M., Van Der Veer, G.C., Eliëns, A.: An Ontology for Task World Models. Design, Specification and Verification of Interactive System'98, Springer Computer Science, (1998)
6. Garrido, J.L., Gea, M.: Modelling Dynamic Group Behaviours. In: Johnson, C. (ed.), Interactive System - Design Specification and Verification, LNCS 2220, Springer, (2001) 128-143
7. Garrido, J.L., Gea, M., Gutierrez F.L., Padilla, N.: Designing Cooperative Systems for Human Collaboration. In: Dieng, R.; Giboin, A. (Eds), Designing Cooperative Systems: The Use Of Theories and Models, IOS press, Netherlands, (2000)
8. Garrido, J.L.: AMENITIES: Una metodología para el desarrollo de sistemas cooperativos basada en modelos de comportamiento y tareas, PhD Thesis, University of Granada, (2003)
9. Gamma, E., Helm, R. Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software, Reading, MA, Addison Wesley, (1995)
10. Coplien, J.O., Harrison, N.B.: Organizational Patterns of Agile Software Development, Prentice Hall, <http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns>, (2004)
11. Fowler, M.: Analysis Patterns: Reusable Object Models. In: Booch, G., Jacobson, I. and Rumbaugh, J. (eds.), Object Technology Series, Reading, MA, Addison-Wesley Publishing Company, (1997)
12. Mintzberg, H.: Structure in fives: designing effective organizations, Englewood Cliffs, N.J., Prentice-Hall, (1992)
13. Isla, J.L., Gutiérrez, F.L., Gea, M.: Descripción de Patrones de Organización y su Modelado con AMENITIES. In: Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC'04), Madrid, (2004) 3-14
14. Isla, J.L., Gutiérrez, F.L., Gea, M.: Patrones de Organización. Integración en un Proceso de Desarrollo Centrado en el Grupo. In: Lorés y Navarro (eds.): Actas del Congreso Internacional Interacción 2004, Lleida, (2004) 172-179
15. Isla, J.L., Gutiérrez, F.L., Paderewski, P.: Un Profile para el Modelado de Patrones de Software. In Toval y Hernández (eds.): Actas de las X Jornadas de Ingeniería del Software y Bases de Datos (JISBD05), Thomson Paraninfo, Granada, (2005) 265-270