

# Towards the Organizational Engineering Pattern Language<sup>©</sup>

## Resources and Roles Based Patterns: The CONTACT, PERSON, ORGANIZATIONAL UNIT and ORGANIZATION Patterns

ALBERTO RODRIGUES SILVA

Instituto Superior Técnico / INESC-ID

(alberto.silva@acm.org)

**Abstract.** In this paper we argue the need and motivation for an *organizational engineering pattern language* focused on the identification of the basic *entities*, or *constructors*, that would support the modelling, monitoring, simulation, and eventual execution of organizations. The main motivation for those constructors would be the development of a new class of information systems that would allow managers to better design, analyse, simulate and run their own organizations and involved business: something like a “organizational cockpit” or a “business-oriented CASE tool.

We identify the following minimum set of organizational constructors: *resources*; *roles*; *activities and business processes*; *strategy and politics*; *time*; and *space*. Inspired by these constructors we present in this paper a set of organizational patterns, aligned mainly with the *resources* and *roles* constructor, namely: CONTACT, PERSON, ORGANIZATIONAL UNIT and ORGANIZATION. Other patterns, such as BUSINESS PROCESS, INFORMATION SYSTEM or VISION, MISSION AND GOAL are just identified and would be described in future work.

## 1 Introduction

A “pattern” describes a kind of a problem that occurs in some context, and also describes a reasonable solution for that problem, in a way that this solution can be applied systematically in different situations (Alexander, 1977). The original application of Alexander’s patterns was in architecture and civil engineering; the software engineering community later on adopted this concept, especially as object-oriented design patterns (Gamma et. al., 1994), and more recently as analysis and business patterns (Fowler, 1996; Penker & Eriksson, 2000; Adams et al., 2001), which is the main background for this work.

Patterns are about *proven* solutions, not new or unique ones. Patterns are about beauty, elegance, knowledge reutilization, soundness and architecture. Patterns represent years of application development, observation and experience. To find a solution is simple. However, to find the *right* solution is usually very hard: you have to understand the problem, the forces affecting the problem and the tradeoffs and consequences of applying a specific solution.

There are currently many events and work concerning mainly software (analysis and design) patterns, such as Pattern Languages of Programming conferences (e.g., PLoP, EuroPLoP, KoalaPLoP, MensorePLoP, SugarLoafPLoP or ChiliPLoP (Hillside Group, n.d)) or the Software Patterns Series from Addison Wesley (Vlissides, 1996-2003). (The interested reader can start reading the “*The Pattern Almanac*” by Linda Rising (Rising, 1999) to get a broad vision of the most relevant work already done.)

---

<sup>©</sup>Proceedings of the EuroPLoP’2003. Copyright © 2003, ALBERTO RODRIGUES DA SILVA. All right reserved.

In spite of the huge number of patterns identified, predominantly from the object-oriented software engineering area, there is yet an effective opportunity to bring this approach to the information systems area, specifically to describe high level data models and business processes in a more interesting and effective way compared to the recurrent very high-level and conceptual approaches (Zachman, 1987; Wurman, 1997; Sharp & McDermott, 2001).

Traditionally, disparate subjects such as management, economy, sociology, history or even psychology study organizations. However, and due to the increasing emergence and importance of information systems in organizations, they have been also studied following the engineering approach, based on whom we called “*organizational engineering*” (Bider & Khomyakov, 1998; Malone, 1999; Eriksson & Penker, 2000). Organizational engineering is a relatively recent knowledge domain that intent to understand and improve organizations’ structure and behaviour, and also to promote the alignment of their information systems with their respective business goals.

We claim in this paper that an “*organizational engineering pattern language*” is need. This pattern language should provide a common set of concepts in order the design, understanding and re-engineering of organizations, as well as their associated information systems would be performed efficiently and with better results compared to the current situation. Nevertheless, it is not the aim of this paper to propose “*the*” organizational engineering pattern language. We just want to identify and present a relevant, but necessarily incomplete, “set of organizational patterns”, meaning that this is an open research area, and so, other patterns should naturally be proposed by different authors in the future.

There are other initiatives that can be apparently associated with this work. However, they are quite different and with different focus. For instance, the forthcoming “*Common Pattern Language of Organizational Patterns*” (Berczuk et al., --) presents a deep discussion and vast number of organizational patterns but mainly related to processes, people and organizations focused on the *software development engineering*, while the pattern language introduced in this paper is focused in the *business and organizational engineering*.

Our work is definitely closest to initiatives such as Fowler’s “*Analysis Patterns*” (Fowler, 1996; Fowler, 2003), Silverston’s “*Data Models*” (Silverston, 2001), or particularly the Open Information Model, from MDC, and the Organizational Structure Facility, from OMG.

The *Open Information Model (OIM)* (Meta Data Coalition, 2000) is a set of meta data specifications to facilitate sharing and reuse between tools and systems. The OIM consists of over 200 types and 100 relationships, described in UML and organized in an easy-to-use and easy-to-extend subject areas, which include: (1) analysis and design; (2) component description; (3) database and data warehousing; (4) business engineering; (5) and knowledge management. OIM was originally promoted by Microsoft through the MDC (Metadata Coalition) consortium and was supported particularly by Microsoft SQL Server and OLAP analysis services. Principally interesting from this paper point of view, is the OIM’s Business Engineering Metamodel (BEM) that addresses process and organization design among other features. The proposed BUSINESS PROCESS and VISION, MISSION AND GOAL patterns (to be analysed in a future paper) were significantly based on the OIM’s BEM concepts.

On the other hand, the *Organizational Structure Facility (OSF)* (OMG, 2001) is a OMG specification that provides a structural modeling of organizational elements. OSF is very tight with the OMG’s specifications, particularly with CORBA and MOF. Consequently and in spite its flexibility and robustness, it becomes hard to understand and to apply in real scenarios. Nevertheless, an implementation of the entire “Organizational Structure Facility” should allow the creation, destruction, manipulation, and query capabilities for organizational entities and organizational structures that define the hierarchical relationships between those entities.

This paper has four sections. Section 2 (“Pattern Template”) describes briefly the pattern template that is used for defining the organizational patterns. Section 3 (“A Set of Organizational Engineering Patterns: Resources and Roles Based Patterns”) is the main part of this paper, where the CONTACT, PERSON, ORGANIZATIONALUNIT and ORGANIZATION organizational patterns are described. Finally, Section 4 (“Conclusions”) wraps up with conclusions and observations for future work.

## 2 Pattern Template

This section presents the pattern template used along the paper. The template contains the headings that follow.

**Name:** Identification of the pattern using an easy-to-read and expressive name.

**Context:** Description of one or more situations in which the pattern is applicable, as well as the description of how the pattern relates with the other patterns in the language. In particular, we will depicted the context/pattern relationships, as suggested in Figure 1, with the significance that “*pattern-A contains (or uses) pattern-B means that pattern-B helps to complete pattern-A, and pattern-A is in the context of pattern-B*”.

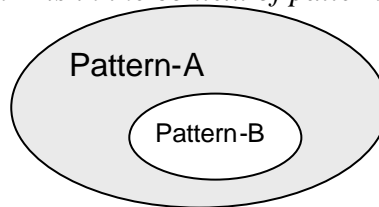


Figure 1: The context/pattern relationships.

**Problem:** Definition of the problem to be solved by the pattern.

**Forces:** Description of the key factors that may influence the decision of when should the pattern be applied.

**Solution:** Description of the solution provide by the pattern. In order to clarify the proposed solution we use UML diagrams (Booch et al, 1999) as well as corresponding relational data models (Ramakrishnan, Gehrke, 2002) represented in a compact-format schema. It should be noticed that the focus of these patterns are essentially structural, meaning that their goal is to discuss concepts arrangements in order to provide consistent and suitable data models.

Note: For the sake of simplicity, we detail data models according relational schemas represented in the compact-format. For example, the following two schemas

```
AddressPerson(AddressID, Name, ZipCode, City, CountryFK)
Country(CountryID, Name)
```

have the corresponding meaning:

- 1) ADDRESSPERSON and COUNTRY are table (or relationship) names.
- 2) ADDRESSID, NAME, ZIPCODE, CITY and COUNTRYID are column (or attribute) names.
- 3) ADDRESSID and COUNTRYID are primary-key columns (or attributes); are underlined.
- 4) COUNTRYFK is a foreign-key column (or attribute); are dashed underlined.

and can be represented in SQL according the following schemas:

```
CREATE TABLE Country (
  CountryID INT NOT NULL,
  Name VARCHAR(100) NOT NULL,
  CONSTRAINT Country_PK PRIMARY KEY (CountryID)
)
```

```

CREATE TABLE AddressPerson (
  AddressID INT NOT NULL,
  Name VARCHAR(100) NOT NULL,
  ZipCode VARCHAR(10),
  City VARCHAR(50),
  CountryFK INT,
  CONSTRAINT AddressPerson_PK PRIMARY KEY(AddressID),
  CONSTRAINT AddressPerson_FK FOREIGN KEY (CountryFK) REFERENCES Country(CountryID)
)

```

**Related Work:** Indication of bibliographic references and or patterns that inspired the specific pattern, in case they exist. Also, discussion of alternative proposals and types of information systems in which the pattern can be applied.

### 3 A Set of Organizational Engineering Patterns: Resources and Roles Based Patterns

When we look for the minimum set of organizational constructors that can allow the modelling, monitoring, simulation, and eventual execution of organizations, we reach the following ones as suggested in the Figure 2: *resources* (e.g., people, money, rooms, computers; software components), *roles* (e.g., manager, worker, programmer, seller, buyer), *activities & business processes*; and *politics & strategy*. Also, the *time* and the *space* can be other relevant constructors if we need to support the lifetime and dynamic of organizations.

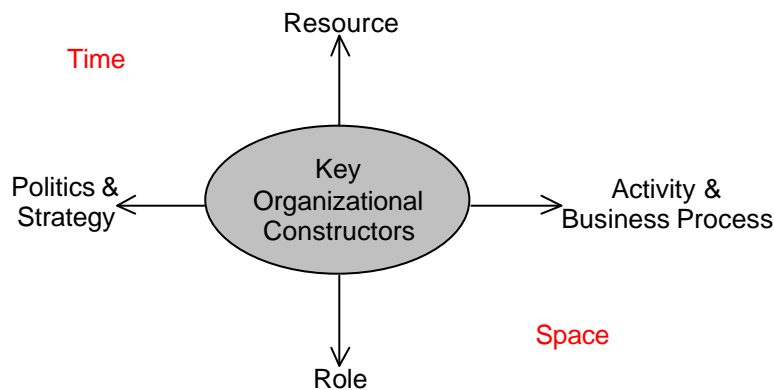


Figure 2: The minimum set of organizational constructors.

This paper is principally focused on the *role* constructor, mainly around the person and organization roles and their respective relationships.

The *resource* constructor is also very important and infrastructural because is the common concept for a lot of other uses in the organizational context. The resources identification and characterization is important in different moments of organizations' life. Particularly in those moments that is essential to decide and to make changes. For instance, when it is need (1) to evaluate an organizations wealth in order to buy or sell it; or (2) to evaluate the number and quality of human resources in order to proceed to employment or unemployment actions; or even (3) to help deciding some investment plan.

To better classify organizations' resources we propose a hierarchy of types based on two distinct notions: tangible and intangible resources, as suggested in Figure 3. *Tangible resources* correspond to concrete things that can be felt by touch and that are easy to identify and evaluate. Examples are (1) people, such as owners, shareholders, users, employees, managers, and external agents that will be involved in the organization in any way; (2) facilities and equipments, such as buildings, cars, computers, telephones, and commodities infrastructures such as electricity, water, and telecommunications; (3) software components,

such as operating systems, applicational and databases servers, specific-domain applications, software libraries; (4) materials, from pencils, paper, notebooks, toner, cartridges, and so on; and (5) money, which is a translation of all of the above into the language of accounting.

On the other hand, *intangible resources* correspond to abstract concepts that are much more hard to identify, such as (1) information and knowledge about everything, from customers, sales, orders, competitors, funding initiatives, resources, costs, projects, and so on; (2) organizational units, services that may be charged on a per use basis, mentoring, sales, marketing, technical support; and (3) information systems, a set of resources and activities which aim is to support the organization’s business processes as well as its information needs.

(Of course, we can identify other classification schemas; such as classify resources as physical (e.g., people) and no-physical resources; or as concrete and abstract resources. Nevertheless, all these classification schemas are reasonable similar and just present minor differences and variations.)

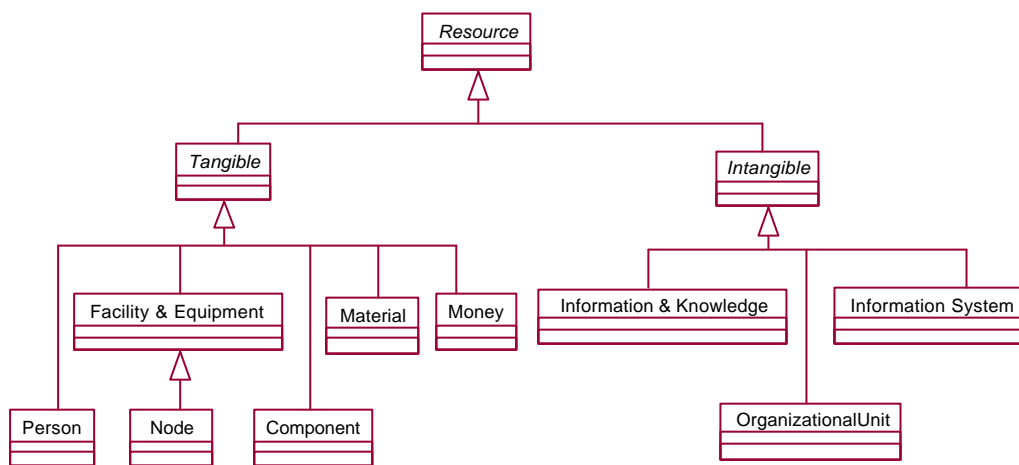


Figure 3: The hierarchy of organizational resources.

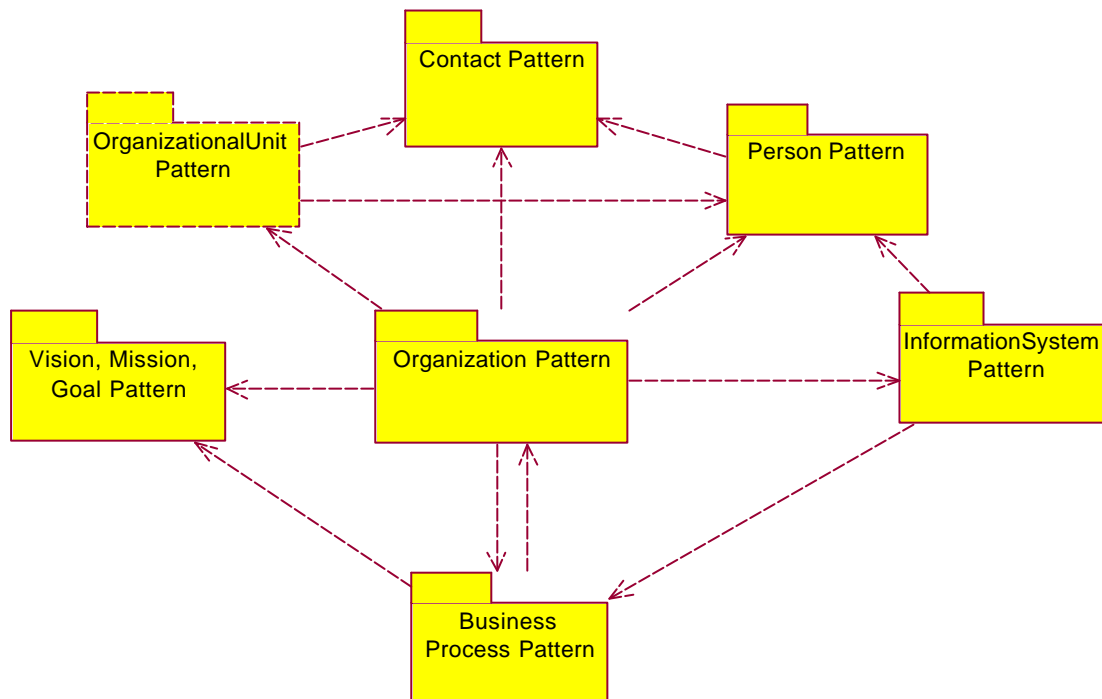
Some of the identified resources are particularly analyse in this paper, namely: person (see the PERSON pattern), organizational unit (see the ORGANIZATIONUNIT pattern), while others such as node, component and information system would be analysed in future work.

Figure 4 shows, through an UML package diagram, the big picture of the proposed organizational patterns, as well as their main relationships. A dashed arrow from pattern A to pattern B means “pattern-A uses pattern-B”.

In this paper (and future papers) we analyse and discuss these patterns according the following sequence: (1) CONTACT, (2) PERSON, (3) ORGANIZATIONALUNIT, (4) ORGANIZATION, (5) VISION, MISSION AND GOAL, (6) BUSINESS PROCESS, and (7) INFORMATION SYSTEM patterns. However, only CONTACT, PERSON, ORGANIZATIONALUNIT and ORGANIZATION patterns are analysed in this paper. The others would be present in the future papers. Nevertheless, for the sake of the reader curiosity and interest, those patterns involve the following considerations.

- The VISION, MISSION AND GOAL pattern is strongly influenced by the OIM’s Business Engineering Model and states clearly the definition of necessary concepts such as business strategy or business analysis activities. Among others are relevant the following concepts: vision, mission, goals and impacts. This pattern addresses naturally the *politics & strategic* constructor.

- The BUSINESS PROCESS is a simple although very important organizational pattern. Business process is a central concept in the organizational engineering activity. Essentially, it allows showing the critical and relevant behaviour of organizations, either inside (i.e., intra) or even outside (i.e., inter) organizations. A business process can be view as an extension of the UML metatype StateActivity, with a set of specific particularities. This pattern addresses the *activity & business process* constructor.
- Finally, the INFORMATION SYSTEM is an ample and complex pattern that has multiple relationships with the other patterns and concepts (e.g., people, organizations, business processes) but also introduces new ones, more related with information system architectures and technologies, concepts like nodes (i.e., computational platforms), software components, execution components, data components, or users and related permissions. This pattern can involve many organizational constructors.



*Figure 4: The big picture of the organizational patterns.*

## CONTACT Pattern

---

### Context

Persons have to manage and to deal with contacts regarding their personal and professional life. A contact usually involves a list of locations, electronic addresses and a set of distinct telephone numbers.

Contacts are usually supported by a disparate number of software applications varying from PIM (personal information managers) to human resource and payroll systems, from directory servers (e.g., LDAP/X.500 directory) to CRM or ERP systems.

Contacts can be kept in distinct systems: from XML files (e.g., in personal PDA or mobile phones) to large directory or databases systems (e.g., in enterprise intranet systems).

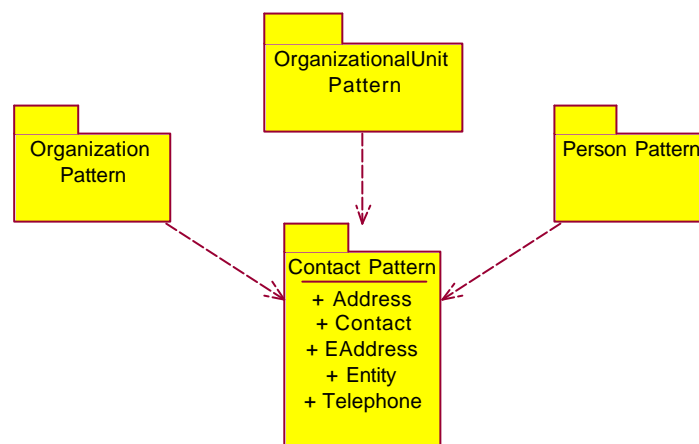


Figure 5: The CONTACT pattern and the other patterns in the language.

**Context/pattern relationships :** Figure 5 shows that the CONTACT hasn't any dependency, but helps to complete (at least) the PERSON, ORGANIZATIONALUNIT and ORGANIZATION patterns.

### Problems

Knowing that *persons, organizations or even organizational units can have contacts*, how do you capture and represent this information in an organizational context? How do you represent contact information, knowing that *a contact can aggregate more than a location, an electronic address or a telephone number*? How do you organize contacts if you would like to *allow contacts sharing among different entities* as well as you would like to *minimize changes impact* without violate ownership properties?

### Forces

A contact can have different types of fields, which number can vary dynamically. For instance, we can describe a contact with just one electronic address or with a variable number of electronic addresses, locations and telephone numbers. Hence, it is not possible to design contacts for a predefined and static number of fields.

A contact can be associated to different kind of entities, such as people, organizations or even organizational units. This means that both entities should share and aggregate the same structure of contacts in order to avoid proliferation of contacts types.

Usually, in simple or personal contexts, contacts belong to just one person: the contacts' owner. However, in business contexts, allow sharing contacts among different entities can provide better integration and minimize the impact of changes (for instance, the change of an organization's telephone number should not require the change of all its employees' telephone numbers). Still, in this situation, where a contact is shared among different entities, it is important to state clearly the specific contact's owner.

## Solution

**Define a Contact as an aggregation of distinct parts (e.g., addresses, electronic-addresses and telephones) and define a generic business Entity that can be the owner or just have access to contacts.**

A flexible solution to this problem is to define a contact has an aggregation of a dynamic set of specific fields (such as addresses, electronic addresses and telephone numbers) as it is shown in Figure 6.

There are three main aspects to be considered too. First, because people, organizations or organizational units can have contacts, they are defined as specializations of the high level and abstract "entity" concept, which is represented by the Entity class.

Second, because an entity can have a variety of contacts, they are defined separately. We allow the dynamic association of common address (i.e., the Address class), telephones (i.e., the Telephone class) and electronic addresses (i.e., the EAddress class), which have a corresponding type attribute (respectively AddressType, PhoneType and EAddressType) in order to offer a convenient classification feature.

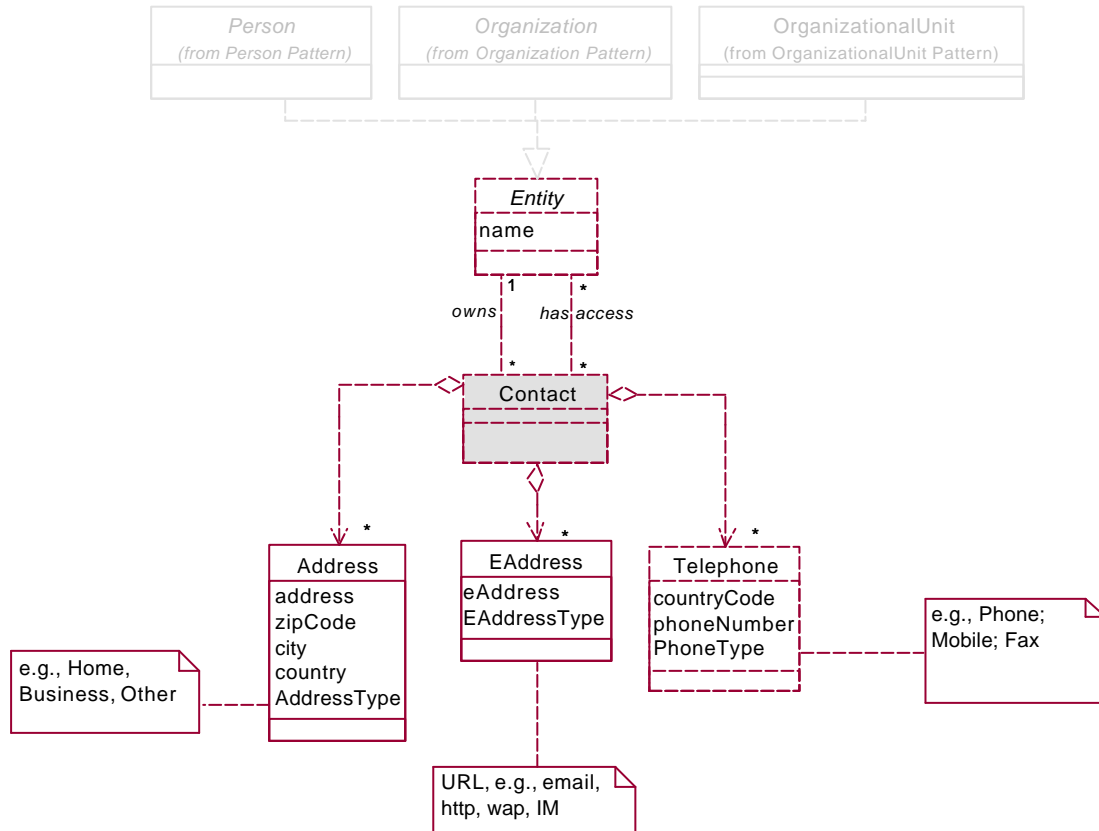


Figure 6: CONTACT pattern, the structural definition.

Third, in order to minimize changes impact, we allow sharing a contact among distinct entities. For example, in a business context, changing the business office contact (e.g., the phone number) should not require to change the contacts of all associated employees. Nevertheless, for the sake of manageability, there should be just one owner for each defined contact. This means that even though a contact can be accessed by different entities (through the *has-access* relationship), just one of them has the right to change it (through the *owns* relationship).

Figure 7 shows the generic collaboration regarding the CONTACT pattern application. [Note: A generic collaboration is represented in UML as a dashed ellipse with a set of dependency relationships (i.e. dashed arrows) directed to the involved participants. This high-level representation can be mapped directly to an object diagram, which should conform with the respective UML class diagram (Booch et al., 1999).]

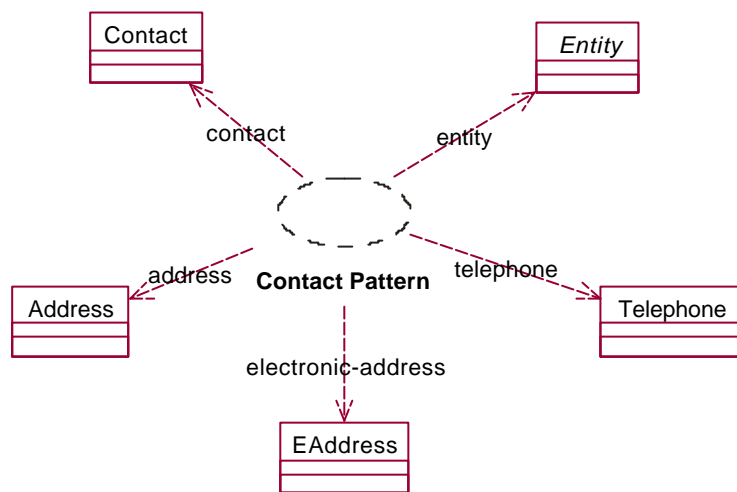


Figure 7: CONTACT pattern, the generic collaboration.

Figure 8 shows an application example of the CONTACT pattern: the author's university contact. The left and right side of the figure are semantically equivalent. The left side shows the information through a UML object diagram, while the right side shows the same information through a specific collaboration diagram, where is more evident the name of the pattern (represented in UML as a dashed ellipse) and its respective participants.

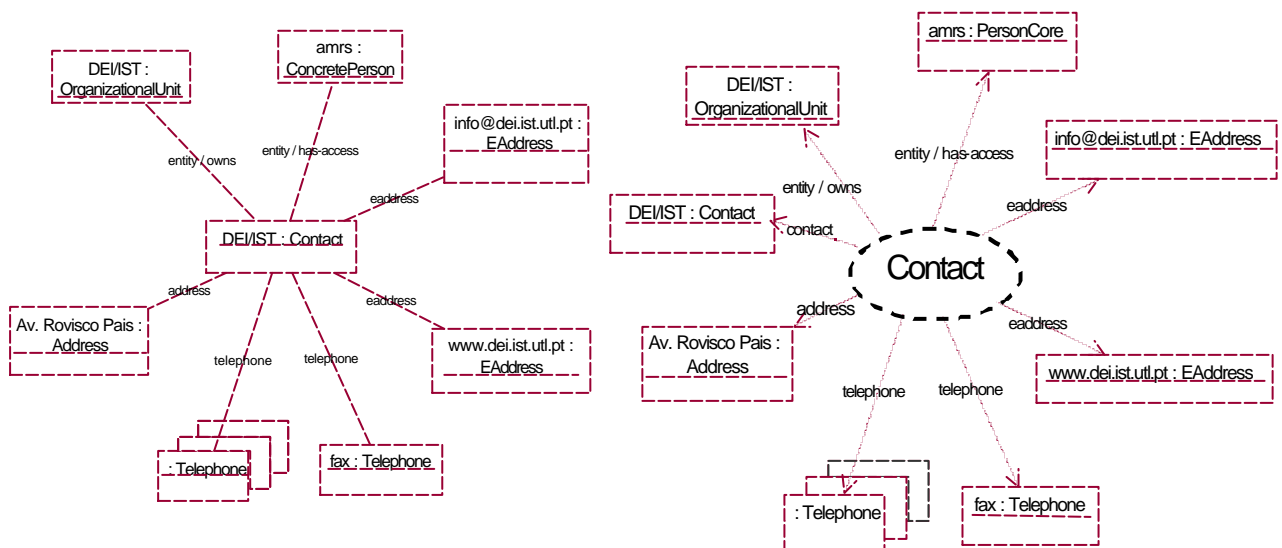


Figure 8: CONTACT pattern, application to the DEI/IST university case.

The next box shows the CONTACT pattern's relational schemas specified in the compact format.

### ***Data Model for the CONTACT Pattern***

```
//  
// Entity and Contacts  
Entity(EntityID, Name, EntityType, ...)  
Contact(ContactID, Name, EntityOwnerFK ...)  
ContactEntityAccess(ContactEntityAccessID, ContactFK, EntityFK)  
// Normal addresses  
Address(AddressID, Address, ZipCode, City, CountryFK, AddressTypeFK, ContactFK, notes, ...)  
AddressType(AddressType, Name) // e.g., home, business, other  
Country(CountryID, Name)  
// Electronic addresses  
EAddress (EAddressID, Address, EAddressTypeFK, ContactFK, notes, ...)  
EAddressType(EAddressTypeID, Name) // e.g., email, web, im, wap  
// Telephones  
Telephone(TelephoneID, CountryCode, PhoneNumber, PhoneTypeFK, ContactFK, notes, ...)  
PhoneType(PhoneTypeID, Name) // e.g., phone, mobile, fax, phone+fax  
//
```

### **Related Work**

The CONTACT pattern is used (or can be used) in several types of information systems such as personal information manager (PIM) systems (e.g., Microsoft Outlook, Elefante or Chandler), human resource and payroll systems, customer relationship managers, application service provider (ASP) based systems, or global identity infrastructures (e.g., Microsoft Passport or Liberty Alliance).

This pattern is being applying, with minor adjustments, in some of our current projects, namely the PUC (<http://berlin.inesc-id.pt/projects/puc/>), Rent@School ([www.rentaschool.org](http://www.rentaschool.org)) and ArtGate (<http://berlin.inesc-id.pt/projects/artgate/>) projects.

The proposed pattern is mainly based from our experience in R&D projects as well as from other proposals such as CWM (OMG, 1999) and OSF (OMG, 2001).

# PERSON Pattern

---

## Context

Every organization and its related information systems have to deal with information about people. Organizations maintain people information due to different reasons, such as: to pay salaries to their employees; to better understand their customers and improve sales; or just to satisfy their shareholders' interests. However, a person can easily perform different roles regarding an organization. (For example, Mary Brown can be simultaneously a customer and an employee regarding the ABC company, and a shareholder of XYZ company. Or in a university organization, John Black can be simultaneously a student, a teacher and a student's parent.)

Furthermore, people keep a set of common information (such as name, gender or date of birth) as well as a set of contacts and addresses varying from home and business addresses to emails, web addresses and phone numbers.

There are numerous applications where management of people, their roles and their skills are critical, varying from human resources systems to Web portals, from project management to e-learning systems.

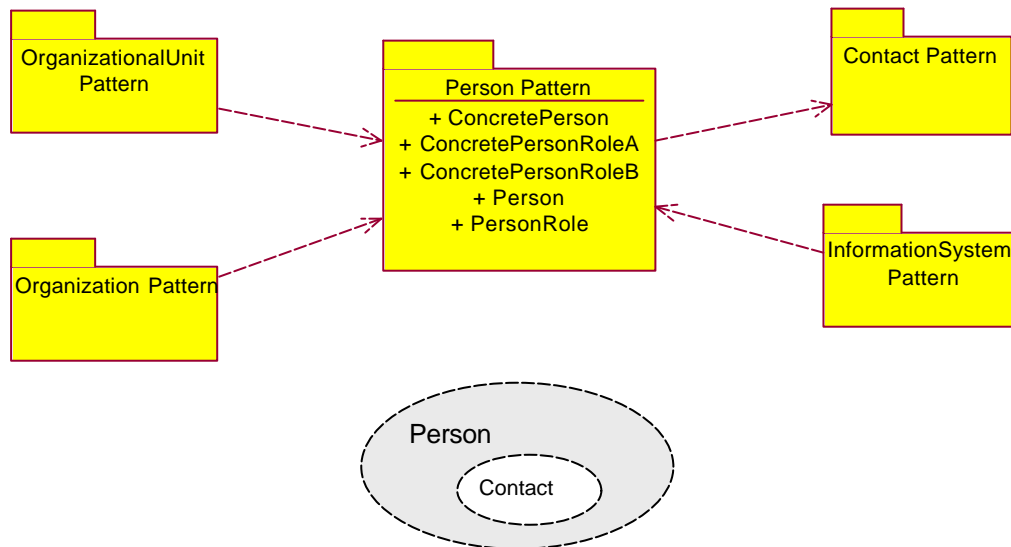


Figure 9: The PERSON pattern and the other patterns in the language.

**Context/pattern relationships:** Figure 9 shows the relationship of the PERSON with the other patterns in the language. It is evident that the CONTACT helps to complete the PERSON pattern. Use PERSON pattern when you want to model, for instance, an OrganizationalUnit or an Organization.

## Problem

How do you capture and represent person information in an organizational context, knowing that people *can perform different roles regarding one or more organizations or even regarding the society?* How can you *manage flexibly and elegantly the disparate number of possible roles* that a person can perform, with the involved specific information and behaviours?

## Forces

A person has a set of common information, such as name, gender, date of birth, nationality or national identification. She can also have a disparate number of contacts.

A person can perform different roles depending the situation (i.e., according the *space* constructor) or even along its history (i.e., according the *time* constructor). Every person role has a distinct set of information and a specific behaviour. For instance, currently I can be a “professor” and a “scientific coordinator at the department” when I am in the University, a “researcher” and a “group leader” when at the INESC research lab, or even a “bank-customer” when at my local bank...

We don’t want to have neither an explosive number of classes regarding the support of all possible combinations of person roles, nor a unique and big class collecting all the attributes (and behaviours) associated to all the roles.

## Solution

**Model a Person as an entity with a varying number of PersonRoles. Implement this by making the PersonRoles decorators for Person.**

A flexible solution to this problem is shown in Figure 10. There are two main aspects to be considered. First, because a person can perform diverse roles along the space and the time (e.g., can perform in a given period of time and depending on the situation, roles like customer, shareholder or collaborator), which can be attached or detached dynamically, we adapt the design pattern DECORATOR in order to support this unpredictability. [The DECORATOR pattern (Gamma et al, 1994) allows attaching dynamically additional responsibilities or roles to an object.]

Second, because a person can have a variety of contacts we detach them from its nuclear definition. That is the reason the `Person` class derives from the `Entity` class. Hence, we allow the dynamic association of contacts according the `CONTACT` pattern (see `CONTACT` pattern presented previously).

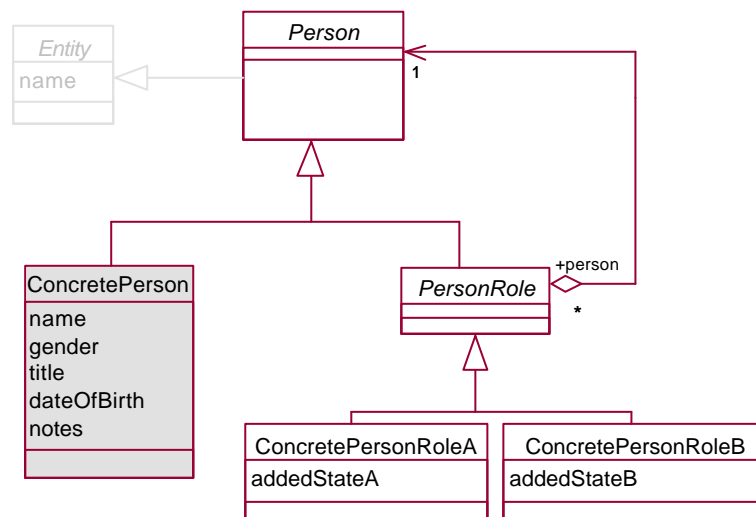


Figure 10: PERSON pattern, the structural definition.

Figure 11 shows the generic collaboration corresponding to the PERSON pattern. Figure 12 depicts two PERSON pattern applications: (1) the left side, regarding the business application domain; and (2) the right side, regarding a classic academic application domain.

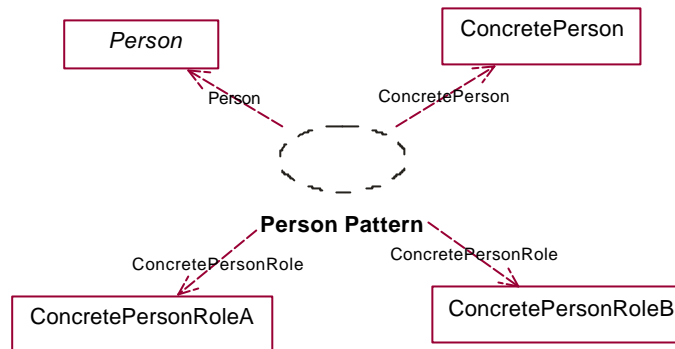


Figure 11: PERSON pattern, the generic collaboration.

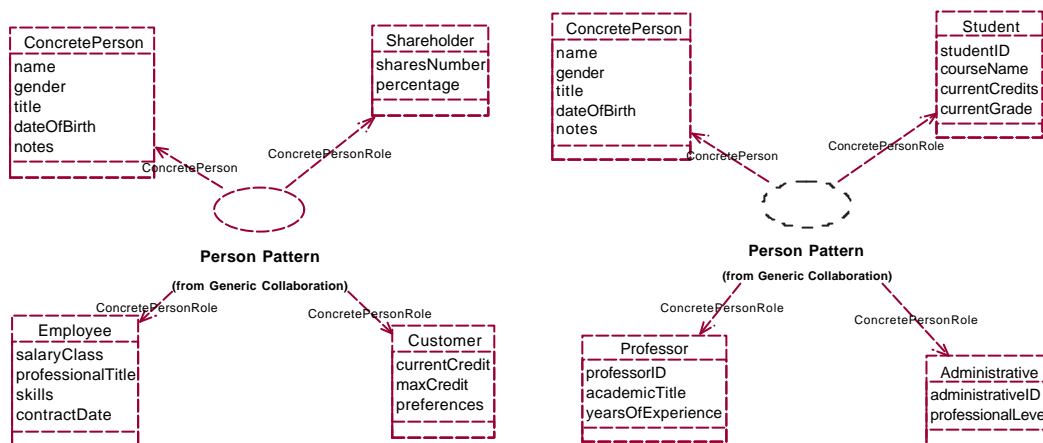


Figure 12: PERSON pattern applied to different application domains.

The next box shows the PERSON pattern's relational schemas specified in the compact format.

### Data Model for the Person Pattern

```
// Entity and Contacts
// see Contact pattern's data model
//
// Person Core
Person(PersonID, EntityFK, Name, Title, Birthday, Gender, NationalId, Notes, ...)
// Roles
PersonRoleA(PersonRoleA, PersonFK, addedStateA, ...)
PersonRoleB(PersonRoleB, PersonFK, addedStateB, ...)
e.g., applied in the business application domain:
Customer(CustomerID, PersonFK, CurrentCredit, LimitCredit, Preferences, ...)
Collaborator(CollaboratorID, PersonFK, SalaryClass, ProfessionalTitle, ...)
Shareholder(ShareholderID, PersonFK, SharesNr, Percentage, ...)
...
```

### Related Work

The PERSON pattern is inspired mainly on the DECORATOR pattern (Gamma et. al., 1994) and has a tight dependency with the CONTACT pattern. Also the ROLE OBJECT pattern (Baumer et. al., 2000) could be considered because it has the same structure of the DECORATOR pattern, but allows introducing new operations dynamically. However, because the PERSON pattern is essentially structural (i.e., it does not focus on operations) the DECORATOR pattern seems to be a more simple and natural adoption. The PERSON is used by the ORGANIZATIONALUNIT and

ORGANIZATION patterns (because organizational units and organizations have relationships with people through their different business roles) as well as the INFORMATION SYSTEM pattern (because information systems are used by people/users).

The domain model of OSF (OMG, 2001) also expresses the Person concept and its respective roles.

As it was referred for the CONTACT pattern, the PERSON pattern can be used in a extensive set of information systems such as personal information manager systems (e.g., Microsoft Outlook, Elefante or Chandler), human resource and payroll systems, customer relationship managers or application service provider (ASP) based systems. From our own experience, the majority of information systems have to deal with person and person roles information. For instance, this pattern is being applying, with minor adjustments, in some of our current projects, namely the PUC (<http://berlin.inesc-id.pt/projects/puc/>), Rent@School ([www.rentaschool.org](http://www.rentaschool.org)) and ArtGate (<http://berlin.inesc-id.pt/projects/artgate/>) projects.

## ORGANIZATIONAL UNIT Pattern

---

### Context

Organizations are internally structured according a varying number of schemas. These structures are based on organizational-units, which are called differently depending the situations, namely: departments, sections, services, project groups... Moreover, these organizational-units can establish different kind of relationships between themselves, as well as can be arranged hierarchically in order to better reflect responsibility, power or management relationships (For example, the “administrative department” of a common organization can be arranged around two or three specific sub-organizational units, called “sections”).

Organizational-units offer a suitable abstraction to deal with people, skills, roles and business processes. The traditional representation of organizations is based mainly on the identification of organizational-units as well as on their respective relationships and, sometimes, on the involved leaders or managers.

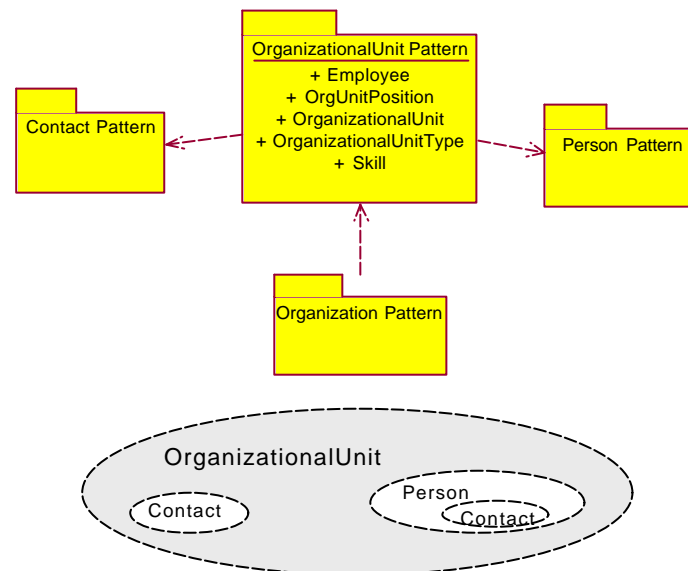


Figure 13: The ORGANIZATIONALUNIT pattern and the other patterns in the language.

**Context/pattern relationships:** Figure 13 shows the relationship of the ORGANIZATIONALUNIT with the other patterns in the language. It is evident that the PERSON and CONTACT helps to complete the ORGANIZATIONALUNIT pattern. On the other hand, the ORGANIZATIONALUNIT pattern can be used in the context of Organizations.

### Problem

How do you *capture and represent internal structures in an organizational context*, knowing that different organizations can have different internal arrangements? How do you *capture hierarchies and or embedded arrangements among organizational-units*? How do you specify the *human resources required and or used currently by some organizational-unit, as well as the involved skills required*? How do you *capture the people that work and or manage a specific organizational-unit*?

## Forces

An organizational-unit has a set of common information, such as name, type, budget, and credit limit. Usually, it has also a leader (which can be called differently, such as president, director or manager), and one or more contacts.

An organization can be structured around multiple organizational-units; some of them can be embedded inside others.

People work for organizations only if they are affected to some organizational-unit. Of course, people can be affected simultaneously to multiple organizational-units. However, it can be important to keep the reasons that justify the recruitment of a given person, or the original set of required skills for a given position.

## Solution

**Represent the internal structures of organizations with OrganizationalUnits. OrganizationalUnit offers specific positions, which can be filled by Employees, if they have the convenient Skills. Additionally, OrganizationalUnits can have different arrangements between themselves.**

Figure 14 shows the solution for the referred problems, which are relevant two main concepts. On one hand, the concept of *organizational-unit* with properties such as name, contacts, manager and a set of required and or occupied positions. The “subOrgUnits” reflexive association allows the definition of organizational-unit hierarchies.

On the other hand, the concept of *employee*, which is a generic role that every person working for any organization should have. Both organizational-units’ positions and employees are described by their respective skills. Hence, this solution allows capturing the people competence model and consequently can provide some support for the matching between the positions and the employment offered.

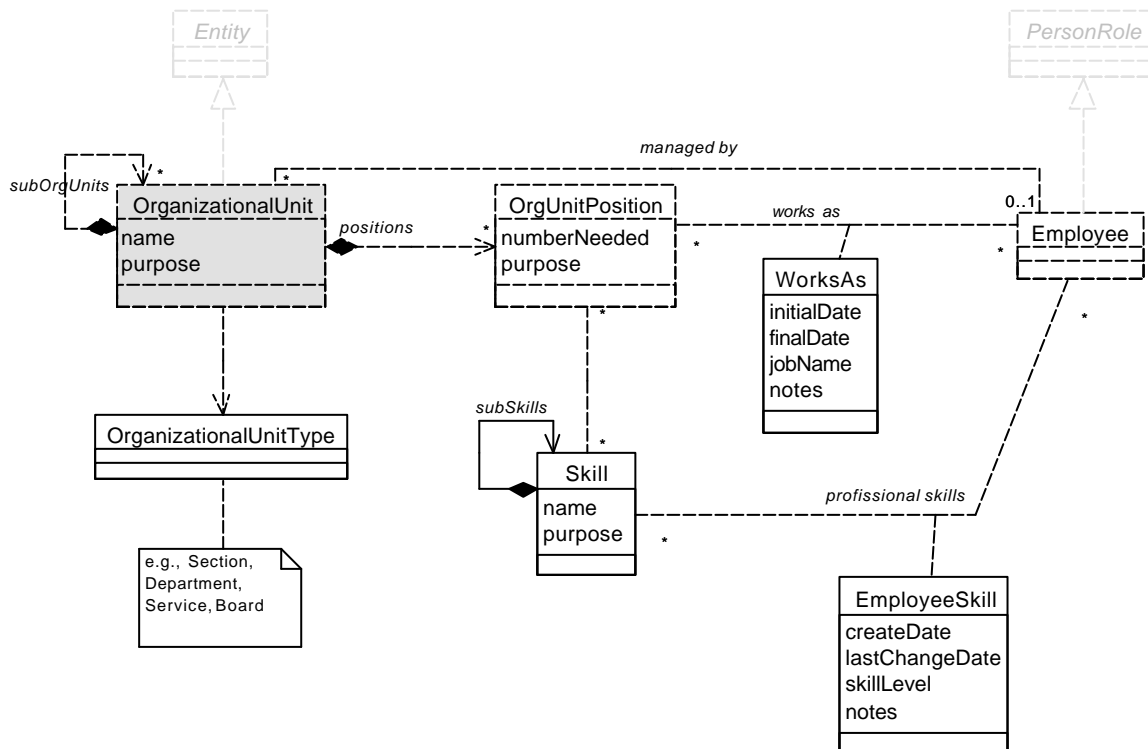


Figure 14: ORGANIZATIONALUNIT pattern, the structural definition.

Figure 15 shows the generic collaboration corresponding to the ORGANIZATIONALUNIT pattern.

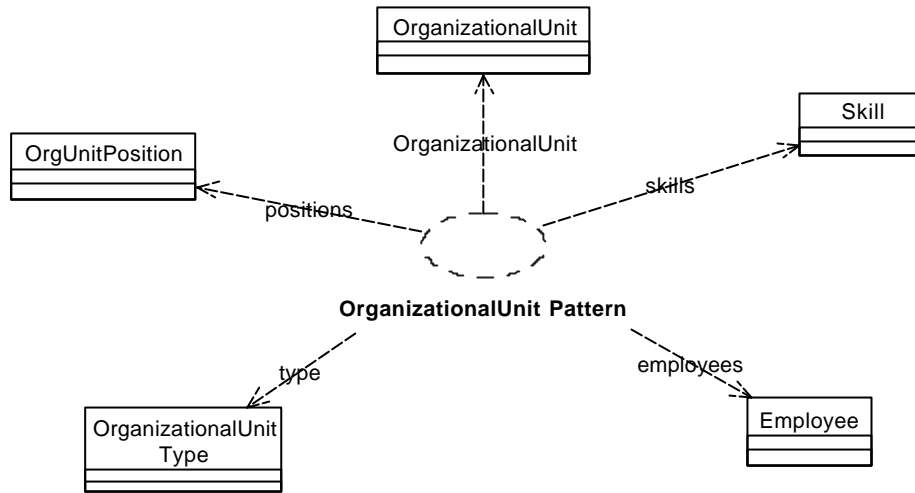


Figure 15: ORGANIZATIONALUNIT pattern, the generic collaboration.

The next box shows the ORGANIZATIONALUNIT pattern’s relational schemas specified according the compact format.

**Data Model for the ORGANIZATIONALUNIT Pattern**

```
// Entity and Contacts
// see Contact pattern’s data model
//
// Person and Person Roles
// see Person pattern’s data model
Person(PersonID, EntityFK, Name, Title, Birthday, Gender, NationalId, Notes, ...)
Employee(EmployeeID, PersonFK, SalaryClass, ProfessionalTitle, ...)
...
// OrganizationalUnit
OrganizationalUnit(OrganizationalUnitID, Name, HierarchyLevel, OrganizationalUnitTypeFK,
  OrgUnitFatherFK, ManagerFK, ManagerRoleName, Purpose)
OrganizationalUnitType(OrganizationalUnitTypeID, Name)
// Positions
OrgUnitPosition(OrgUnitPositionID, OrgUnitFK, NumberNeeded, Purpose)
Skill(SkillID, Name, Purpose)
OrgUnitPositionSkill(OrgUnitPositionSkill, OrgUnitPositionFK, SkillFK)
EmployeeSkill(EmployeeSkill, EmployeeFK, SkillFK, createDate, lastChangeDate, skillLevel, ...)
WorkAs(WorkAsID, EmployeeFK, OrgUnitPositionFK, InitialDate, FinalDate, JobName, ...)
```

**Related Work**

The ORGANIZATIONALUNIT pattern is tightly coupled with the other patterns presented in this paper. In particular, this pattern can be used to describe the internal structures of organizations.

The Organizational Structure Facility (OMG, 2001) specifies a data model and a set of CORBA interfaces related to organizations modeling. However, in spite of its robustness and flexibility, OSF tends to be hard to understand and to apply in practice.

The OIM's Business Engineering Model (MDC, 2000) specifies a business metamodel, which also introduces some main concepts that we adopted. This metamodel describes the actors and resources of a business and their relationships. It captures who should perform what activity, the necessary skills, the reporting structure, and the responsibility structure. But also, BEM is focused on the internal view of organizations, and considers only two types of resources (i.e., BusinessUnit and OrganizationalRole).

The ORGANIZATIONALUNIT pattern can be used in several systems such as those referred for the PERSON pattern, namely for human resource and payroll systems.

# ORGANIZATION Pattern

## Context

Organizations represent the central concept of the organizational engineering domain. Nevertheless and despite organizations' information systems have to manage a high volume of information about other related organizations, that is not well known represented, designed and or implemented in concrete information systems. Traditionally, organizations are understood and represented following their internal view, through well-known diagrams that just show their organizational structures and the corresponding relationships. However, these representations are not enough because there are other organizational perspectives that deserve to be designed and analysed, such as the vision of the business roles, the people involved on (e.g., employees and shareholders), or the external vision of the organization.

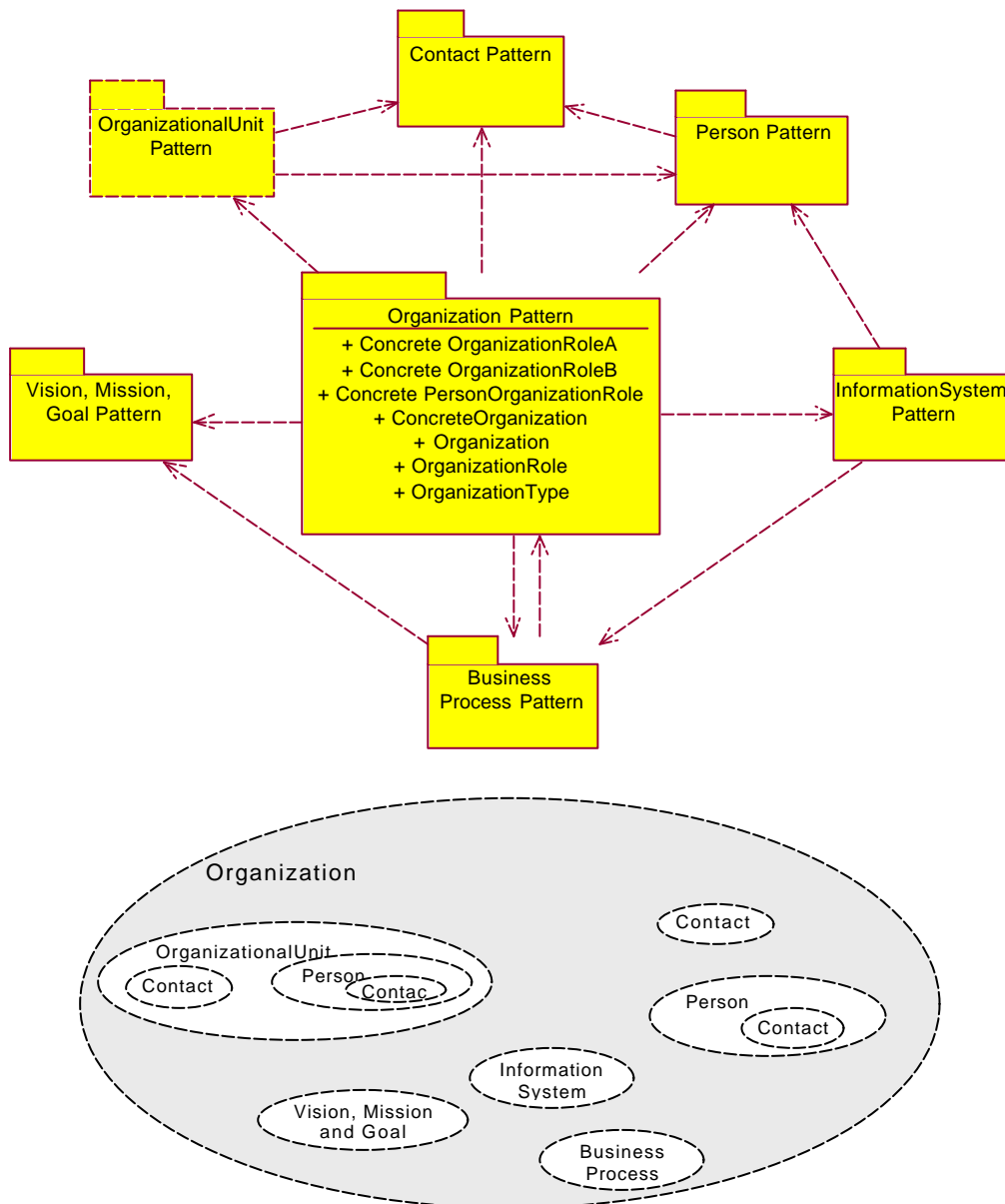


Figure 16: The ORGANIZATION pattern and the other patterns in the language.

Organizations have different information systems with distinct data models that try to represent and maintain information concerning organizational and human entities. However, following the organizational engineering, it is important to have a common, application-independent and consolidated data model. Of course, that common data model should capture in a flexible and elegant way organizational and human entities.

This kind of information is used in many situations and supported by distinct types of information systems, such as ERP and CRM systems; strategic management or business reengineering support systems; or even simulators and games for management or economy domains.

**Context/pattern relationships:** Figure 16 shows the relationships of the ORGANIZATION with the other patterns in the language. It is evident that this pattern performs a central role in the pattern language, with many dependencies with the other patterns. Particularly, PERSON, CONTACT and ORGANIZATIONALUNIT help to complete the ORGANIZATION pattern.

## Problem

How do you *identify and capture the relationships between organizations and people* as well as *between organizations performing different roles* (e.g., customer, supplier or partner)? How do you *capture the internal structure of an organization*, with all its complex organizational structures, internal business roles, needed skills, and people involved on? How do you *capture the external view of an organization* compound by a multitude of relationships with other organizations and or people performing different business roles?

## Forces

Organizations have some common information, such as name, nationality, fiscal identification, and contacts. They have also internal organizational structures based on which they support their own business processes and activities.

Organizations have to deal with resources in order to carry out its business. Resources can be arranged in structures and have relationships with each other. However, there are a wide variety of resources, such as tangibles (e.g., people, buildings, machines and equipment, cars, materials, money) and intangibles (e.g., information, knowledge, research and innovation politics, know-how).

Organizations establish several relationships with people and also with other organizations.

Organizations can perform different roles, namely they can act as public administrations, partners, subsidiaries, suppliers or customers.

## Solution

**Model an Organization as an entity with a varying number of OrganizationRoles. Implement this by making the OrganizationRoles decorators for Organization. An Organization can establish different kind of relationships with other Organizations as well as with Persons. Additionally, an Organization has a Vision (and consequently Missions and Goals), is structured internally around a set of OrganizationalUnits, and supports multiple BusinessProcesses.**

The generic solution to the referred problems is depicted in Figures 17 and 18, and it is based in the unambiguous identification of two key concepts: persons and organizations, as well as their respective business roles. *Persons* are entities with individual and multifaceted interests, who are designed according the PERSON pattern, presented previously. On the other hand, *organizations* are viewed as socio-economical entities that can establish different kind of relationships with people and other organizations. Because an organization can perform different roles in the same or different situations (e.g., can perform simultaneously roles like

customer, supplier, subsidiary and business partner) we adapt the DECORATOR design pattern in order to support this unpredictability. The DECORATOR pattern (Gamma et. al., 1994) allows attaching dynamically additional responsibilities or roles to an object and hence, it is very adequate to support this situation.

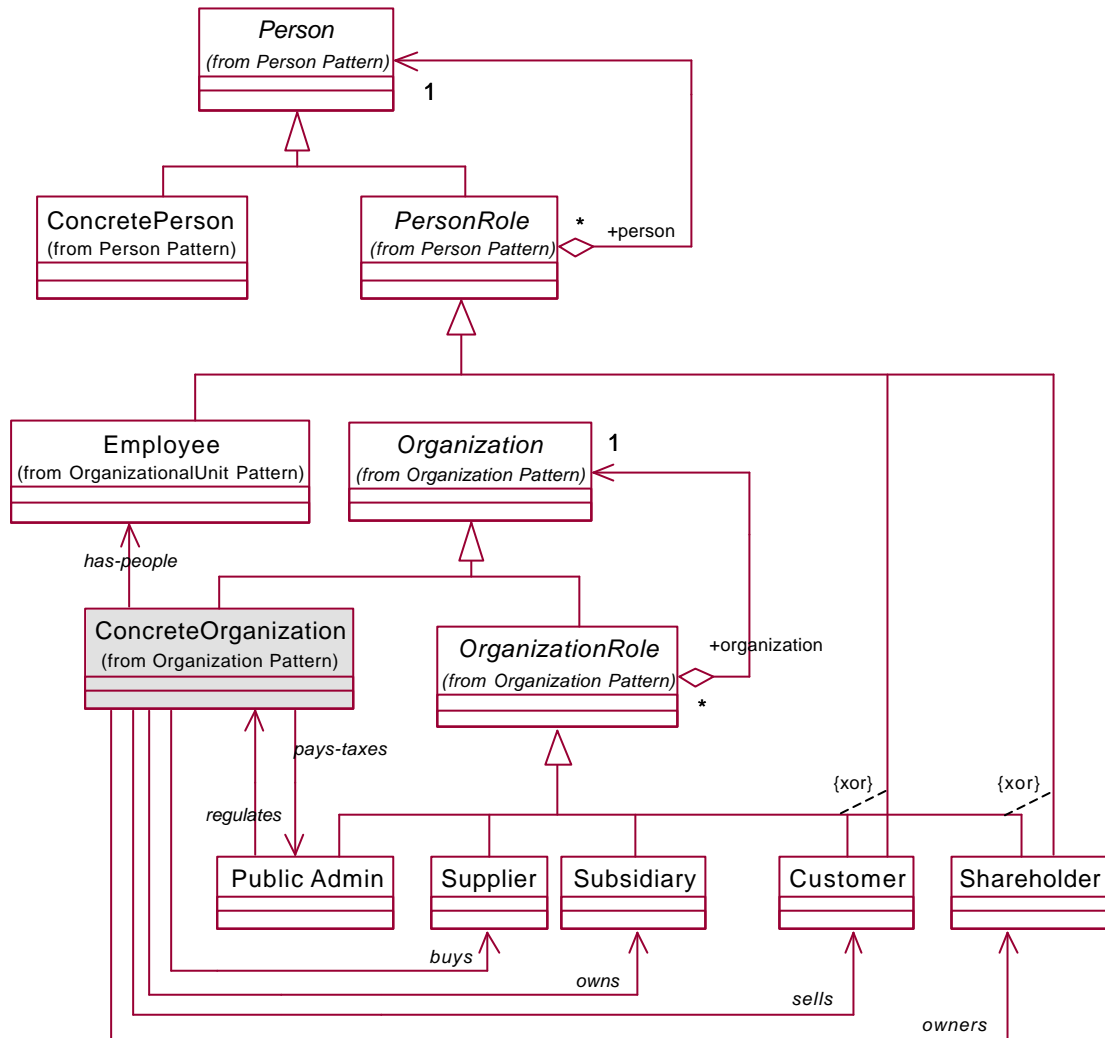


Figure 17: The ORGANIZATION pattern applied in a business context.

People can be identified as different roles through the Person-PersonRole inheritance, such as suggested in Figure 17: Employee (e.g., worker, collaborator, consultant, project manager), Shareholder (e.g., enterprise owner) or Customer (e.g., anonymous or well-known customer). The adoption of the DECORATOR pattern means that a person can be simultaneously an employee, a shareholder and or a customer.

An organization can perform disparate roles (defined according the Organization-OrganizationRole inheritance), such as customer, supplier, shareholder, subsidiary and or public administration. Generically, an organization establishes relationships with the Public Administration (e.g., to pay taxes or to satisfy legal/social obligations), buy and sell products/services respectively to suppliers and customers, and can work with other organizations in partnerships agreements. Finally, organizations have to motivate and manage their employees in order to satisfy the goals and requirements of their respective shareholders.

Figure 17 depicts two {xor} constraints that deserve a special note. Those constraints are needed to specify that some roles (e.g., Customer or Shareholder) can be performed either by an organization or by a person, but not both simultaneously. For example, the {xor} constraint between the inheritance relationships PersonRole-Customer and OrganizationRole-Customer means that either an organization or a person can be a customer in a given context. By exclusion, we assume that only organizations can perform roles like public administration, supplier, subsidiary or partner.

As discussed previously for persons, organizations should also have different kind of contacts in order to support conveniently their businesses. This information is not represented in Figure 18 for the sake of simplicity. However, this is evident in the CONTACT pattern through the Entity-Organization inheritance (see Figure 6).

Organizations can be classified according a specific type (i.e., the OrganizationType class), such as a company, university, private or public institute.

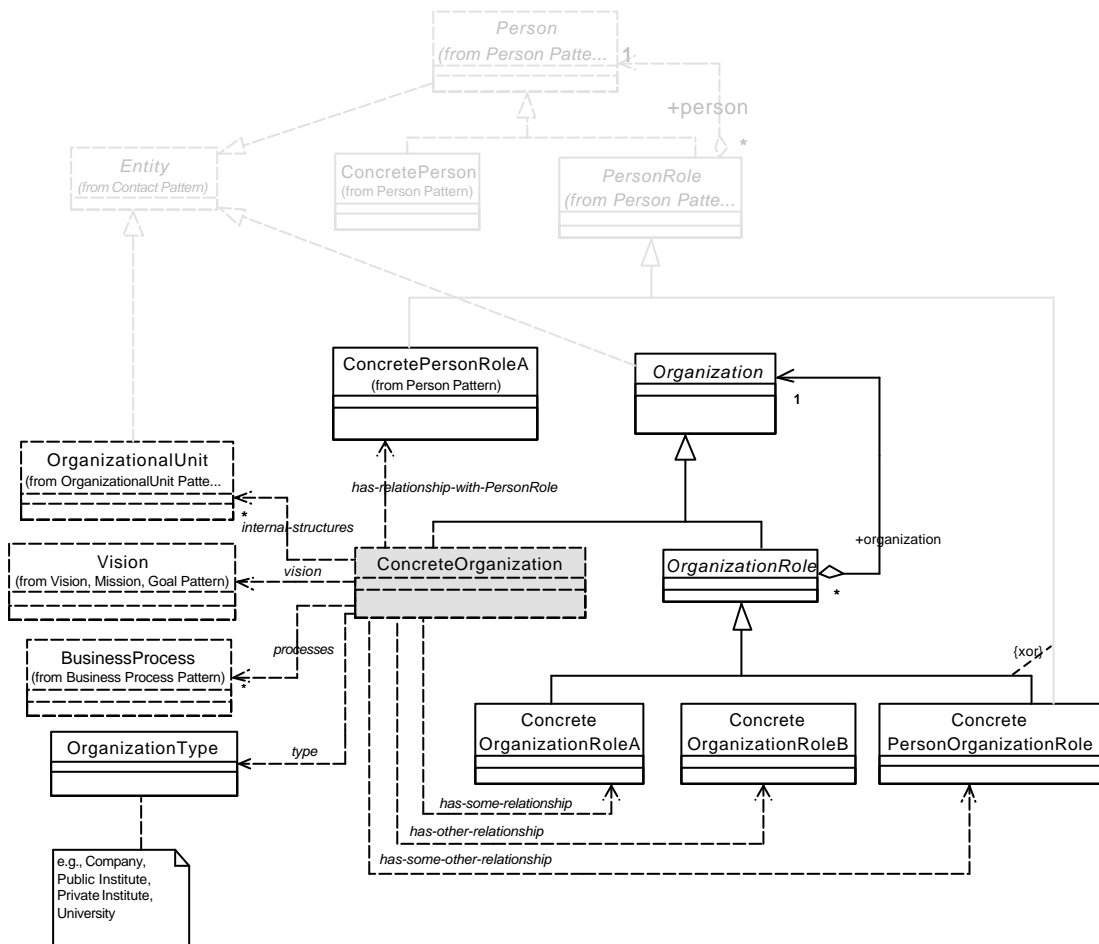


Figure 18: ORGANIZATION pattern, the structural definition.

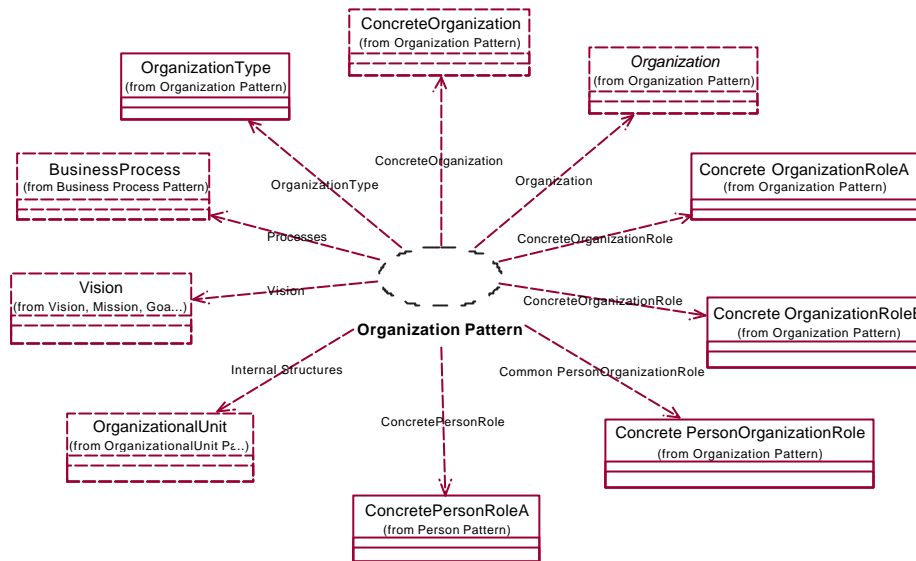


Figure 19: ORGANIZATION pattern, the generic collaboration.

Additionally, organizations are better represented through two complementary perspectives: the internal view and the external view.

### External View

According to the *external view*, there should be captured all the relationships between the source organization and all target organizations and people involved on. This information can be obtained from the relationships between the `ConcreteOrganization` and all the person and organization roles (i.e., `ConcreteOrganizationRole` and `ConcretePersonOrganizationRole`). Of course, some of these relationships, in real situations, would involve more information and consequently relationships with other entities. A simple but suggestive application of the Organization pattern is depicted in Figure 17.

### Internal View

On the other hand, according to the *internal view*, there should be captured the organizations' internal units (e.g., sections, departments and services), employees and their respective relationships (see ORGANIZATIONALUNIT pattern). It can also be captured the organizations' vision, mission and goal statements (see the VISION, MISSION AND GOAL pattern) as well as their internal and external business processes and involved activities (see the BUSINESS PROCESS pattern).

The next box shows the ORGANIZATION pattern's relational schemas specified according the compact format.

### Data Model for the ORGANIZATION Pattern

```
//
// Entities and Contacts
// see CONTACT pattern's data model
// ...
// People
// see PERSON pattern's data model
// ...
// Organizations
Organization (OrganizationID, EntityFK, Name, Acronym, FiscalNumber, Notes,
             OrganizationTypeFK, ...)
```

```

OrganizationType(OrganizationTypeID, Name)
// Organizations Internal View
// see ORGANIZATIONALUNIT pattern's data model
// ...
// Organizations External View
OrganizationRoleA(OrganizationRoleA, OrganizationFK, addedStateA, ...)
OrganizationRoleB(OrganizationRoleB, OrganizationFK, addedStateB, ...)
PersonOrganizationRoleB(PersonOrganizationRoleC, [PersonFK | OrganizationFK], addedStateC, ...)
e.g., applied in the business application domain:
Customer(CustomerID, [PersonFK | OrganizationFK], InitialDate, FinalDate, CurrentCredit,
    LimitCredit, Preferences, ...)
Shareholder(ShareholderID, [PersonFK | OrganizationFK], SharesNr, Percentage, ...)
Partner(PartnerID, OrganizationFK, InitialDate, FinalDate, ...)
Subsidiary(SubsidiaryID, OrganizationFK, InitialDate, FinalDate, PercentageOwner, ...)
Supplier(SupplierID, OrganizationFK, InitialDate, FinalDate, Notes, ...)
PublicAdmin(PublicAdminID, OrganizationFK, Notes, ...)
...

```

## Related Work

The ORGANIZATION pattern is tightly coupled with the patterns presented previously, namely it is completed by the CONTACT, PERSON and ORGANIZATIONALUNIT patterns. Like the PERSON, it is based on the DECORATOR pattern due to its flexibility and elegance regarding the management of organization roles. Also the ROLE OBJECT pattern (Baumer et. al., 2000) could be considered because it has the same structure of the DECORATOR pattern, but it allows introducing new operations dynamically. However, because the ORGANIZATION pattern is essentially structural (i.e., it does not focus on operations) the DECORATOR pattern seems to be a more simple and appropriate adoption.

The Organizational Structure Facility (OMG, 2001) specifies a data model and a set of CORBA interfaces related to organizations modeling. However, in spite of its robustness and flexibility, OSF tends to be hard to understand and to apply in practice. On the other hand, OSF is primarily focused on the organization's internal views and, as a consequence, it does not propose mechanisms to model and to manage explicit relationships between organizations and their roles, such as it is proposed in this pattern.

The OIM's Business Engineering Model (MDC, 2000) specifies a business metamodel, which also introduces some main concepts that we analyzed. This metamodel describes the actors and resources of a business and their relationships. It captures who should perform what activity, the necessary skills, the reporting structure, and the responsibility structure. But also, BEM is focused on the internal view of organizations, and considers only two types of resources (i.e., BusinessUnit and OrganizationalRole).

The use of the ORGANIZATION pattern is twofold. First, from its internal perspective, it can be used in several systems such as those referred for the PERSON pattern, namely human resource and payroll systems, or application service provider (ASP) based systems. Second, from the its external perspective, this pattern can be used by used in management simulators and games, virtual societies, and the majority of e-business based systems such as e-marketplaces, customer relationship and supply-chain management systems.

## 4 Conclusions

As state initially in this paper, patterns are about *proven* solutions, not new or unique ones. Patterns represent years of application development, observation and experience. In spite of the long experience designing and developing software systems (with the consequent wide range of well-known software patterns (Rising, 1999)) there are not so much experience in the organizational engineering domain, at least described according the “pattern approach”.

In this paper we argue that an organizational pattern language is need for a better design, understanding and (re)engineering of organizations as well as their associated information systems. In particular, we analysed and discussed in detail the CONTACT, PERSON, ORGANIZATIONALUNIT and ORGANIZATION patterns. Other patterns, for instance VISION, MISSION AND GOAL, BUSINESS PROCESS, and INFORMATION SYSTEM patterns, were drafted and would be discussed and proposed in future work.

The most important contribute of this paper was the proposal and discussion of organizational patterns in a novel way, bringing the pragmatic and effective approach of software patterns (Vlissides, 1996-2003) to a more recent and yet conceptual area of organizational engineering (Scheer, 1999; Bider & Khomyakov, 1998; Malone, 1999; Eriksson & Penker, 2000; Adams et al., 2001; MIT, 2000). Our main goal is to start a reflection, with consequent activities, concerning the issue of modelling organizations through generic and high-level concepts and structures.

Furthermore, the proposed patterns are important by themselves, they can be analysed and applied together or individually into a large range of information systems and situations as was referred along the paper.

Finally, it is important to state that the patterns proposed are just an open and incomplete version of the intended “*organizational pattern language*”. Consequently, for us it is clear that more work and research should be done in the future. Among others, we identify two issues that deserve to be handled. First, this pattern language should be applied at different levels into different case studies and real scenarios in order to obtain positive feedback as well as to improve itself. Second, this pattern language can eventually becomes the basis for an organizational metamodel (i.e., an UML profile) with main concepts (i.e., stereotypes or metatypes) such as Organization, Person, OrganizationRole, PersonRole, Goal, Business Process, and Information System. Of course, with that metamodel, it would be possible to associate easy-to-read icons to those stereotypes, and consequently to improve the expressiveness of the produced models as well as the communication among the involved stakeholders.

## Acknowledgments

I'd like to thank my EuroPloP'2003 shepherd, Steve Berczuk, for his pertinent suggestions and guidance, which help to improve the readiness and deepness of this paper.

I also would like to thank my colleague José Tribolet, for the discussions around the “organizational engineering” subject, from which I recognized the relevance of defining a minimum set of organizational constructors.

## References

- Adams, J., Koushik, S., Vasudeva, G., Galambos, G. (2001). *Patterns for e-business: A Strategy for Reuse*. IBM Press.
- Alexander, C., Ishikawa, S., Silverstein, M., et al. (1977). *A Pattern Language*. Oxford University Press.

- Berczuk, S, Coplien, J, Devos, M., Harrison, N.(--). *Common Pattern Language of Organizational Patterns*. To be published in a forthcoming Prentice - Hall book. See <http://www.easycomp.org/cgi-bin/OrgPatterns?OrgPatterns> and <http://easycomp.info.uni-karlsruhe.de/~jcoplien/HarrisonCoplien.pdf>
- Bider, I. and Khomyakov, M.(1998). One OO Approach to Business Modeling, *Proceedings of OOPSLA '1998*. ACM Press.
- Baumer, D., Riehle, D., Siberski, W., Wulf, M. (2000). Role Object. In *Pattern Languages of Program Design 4*. Addison Wesley.
- Booch, G, Rumbaugh, J, Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley.
- Eriksson, H-E. and Penker, M. (2000). *Business Modeling with UML – business Patterns at Work* John Wiley and Sons, Inc.
- Fowler, M. (1996). *Analysis Patterns: Reusable Object Models* (Object-Oriented Software Engineering Series), Addison-Wesley.
- Fowler, M. (2003). Martin Fowler: Articles. <http://www.martinfowler.com/articles.html#ap>
- Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley.
- Hillside Group. (n.d). Patterns Home Page – Patterns Conferences. URL: <http://www.hillside.net/conferencesnavigation.htm>
- Malone, T. et al. (1999). Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. *Management Science*. 45(3) pp 425-443. March 1999.
- Meta Data Coalition (MDC). (2000). *Open Information Model*. <http://www.mdcinfo.com/OIM/index.html>
- MIT (2000). *MIT eBusiness Process Handbook*. MIT Sloan School and Phios Corporation.
- OMG. (1999). *Common Warehouse Metamodel Specification*. URL: <http://www.omg.org/technology/cwm/>
- OMG. (2001). *Organizational Structure Facility Specification*. URL: <http://www.omg.org/docs/dtc/01-09-04.pdf>
- Penker, M., Eriksson, H. (2000). *Business Modeling With UML: Business Patterns at Work* John Wiley & Sons
- Ramakrishnan, R., Gehrke, J.(2002). *Database Management Systems*, Third Edition. McGraw-Hill.
- Rising, L. (1999). *The Pattern Almanac* Addison Wesley.
- Scheer, A (1999). *ARIS – Business Process Modeling*, Second Edition.Springer-Verlag.
- Sharp, A., McDermott, P. (2001). *Workflow Modeling: Tools for Process Improvement and Application Development*. Artech House.
- Silverston, L. (2001). *The Data Model Resource Book, Vol. 1 and 2: A Library of Universal Data Models for All Enterprises*. John Wiley.
- Wurman, R. (1997). *Information Architects*. Palace Press International.
- Vlissides, J. (Series Ed.). (1996-2003). *The Software Patterns Series*. Addison Wesley. URL: <http://www.awl.com/cseng/swpatterns>.
- Zachman, J. (1987). “A Framework for Information Systems Architecture”. *IBM Systems Journal*, vol. 26(3).