A Method to Implement Information Systems by Reusable Components

VINCENZA CARCHIOLO, MICHELE MALGERI AND GIUSEPPE MANGIONI
Dipartimento di Ingegneria Informatica e delle Telecomunicazioni
Facolta' di Ingegneria - Universita' di Catania
viale Andrea Doria, 6 - I95125 Catania
ITALY
{

Abstract: – Development of information systems for the management of complex organisational structures is a well known problem, it is interesting in particular in presence of many figures with different roles that interact with the system.
Traditionally the development of an information system includes a phase of analysis of the requirements, a phase of analysis of the existing system (independently from the fact that it is computerised) up to the phase of implementation, by generally using database engine and tools for the generation/management of the human-machine interface.
This paper describes the approach used by the authors to implement the information system for the management of the engineering faculty and how it can be extended to a whole class of systems. The informaton system is modeled in terms of resources that can be demanded and be assigned to persons or groups, on the basis of the roles that they hold inside the system. Particularly, we have faced the phase of analysis and logicalization just one time for the ”generic system” developing also an engine for its management.

Key-Words: – Information Systems, Software Engineering, Web-based Software Engineering, Reusability, Internet based Data Bases, Software Design and Development

1 Introduction

The development of Information Systems for the management of complex organizations, both in terms of data to manage and in terms of involved people, is a well known matter [1]. The traditional approach to the development of information systems usually comprises a phase of analysis of the existing system, a phase of definition of new system requirements, and at last phase of implementation of the physical system using database engines and ad-hoc interfaces [2]. In this paper the authors present a design methodology that has been suggested by their experience in the development of the information system of engineering faculty. Such design methodology introduces an additional abstraction level in order of simplify the implementation stage. The first attempt to analize the information system of enginnering faculty was made in a traditional way, finding several entities and relations, that carried out a rather complex schema. This fact pointed out the needs to a more general description in the attempt to design a reusable system; we tried to find such entities able to model the system at more abstract level. This operation allowed us to observe that such abstract entities not only exist, but they allow to model an entire class of systems. The information system is modeled in terms of resources that can be demanded or be assigned to persons or groups, based on the roles that they hold inside the system. Particularly, we have faced the phase of analysis and logicalization just one time for the ”generic system” developing also an engine for its management. In this way, every time that the goal is to develop an information system belonging to the class of those based on resources, the designer has to create just an istance of the generic system, populating it with the data characterizing system being developed and managing it using the engine.

The novelty of the approach lies on the capability to introduce a further level of abstraction that permits us to analyse the system in terms of resources without boring with details of their management. Of course, to deploy the system is needed to implement the human interface which also has in charge to carry out the functions typical of system being developed. Then, the work we carried out was:

1. analysis of the generic system and deploying of the system into a relational database (i.e. creation of the schema and definition of the *tables*);

2. development of a set of functions working on resources and actors, i.e. a *RMF (Resources Management Framework) - engine* that hide some implementation details to designer;

Having completed above described work, to develop a new system belonging to the same family, it is sufficient:

1. locate all the resources and actors and characterise them;

2. list all the operations that can be performed by each actor to each resource;

3. instantiate the system (practically populate the database);

4. develop the interface using the resource-engine already implemented.

Section 2 discusses our framework (RMF). Section 3 presents the main functions provided by the engine and the general architectures of the information system. Section 4 shows an application of our framework to a real problem.

2 A system design based on RMF

This section highlight the history of the design of the information systems of the engineering faculty pointing out the reasons that drove us in developing the proposed strategy. At very beginning we approached the design of the faculty information systems adopting the traditional strategy based on entity-relationship and dataflow analysis. We find several entities, as, for instance, student, teacher, classroom, administrative staff, department, building, all needed to cover some requirement of current system. It is intuitive the meaning of each of the previous entities and why they was selected. However, the analysis pointed out that each entity can be described thank to a couple of general entities allowing us to drastically reduce the complexity of the system being developed but also covering an entire class of systems. The entities candidate for the purpose highlighted above are: resource and actor. Actor is used to model entity able to perform actions. Resource deals with the entity that can be used by someone (the actor) to do something. Choosing the resource and actors are the main aim of our analysis. Of course , the higher abstraction level of the entities chosen has to cover with the problem to describes the involved element by the same set of attributes without loosing of generality. This fact, bring us to the development of an information system with deals just with resource and actors and, therefore, is always usable when the real system can be described in those term. Moreover, we have developed a resource engine that aims to hide the details of the resource/actor implementation.

2.1 Resources-Oriented View

Definition of generic resource-oriented system was the main concern of the first step of analysis of the system. Thus we were able to model the system using just two entities and one relation between them (see Fig.1). Moreover we took into account not only material resources but also *someone (or something) that can be used by someone to reach his (its) goal.* This reasoning allows us to delay the characterisation of the system being developed up to the end, performing it just populating the database.

In the above paragraph we discussed the properties of the resource, now we have to introduce the second key entity: the actor. It represents the generic entity that use some resources, as, for instance, a student (actor) that uses a computer (resource) for an exercise. The meaning of the relation between actor and resource represents what is the operation that actor make on the resource. The action is modelled as a generic one in order to represents the whole set of possible actions inside a system belonging to this class; also the action will

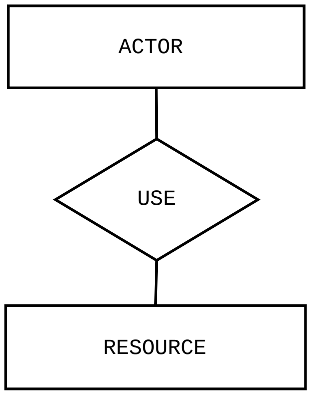

Figure 1: The generic Resource Oriented System model

be defined during the instantiation.

At this point we are able to answer to following question:
What are the common background among the entities able to model a resource-oriented information system ?

2.2 Detailed Design

Actual resource-oriented system often exhibits a hierarchical behaviour, mainly when the system is much complex. We aim to introduce this aspect also in the generic system we are developing without reducing the generality of the approach. Practically, we aim at describing group of person (people working in the same office, or collaborating to the same job, or having the same function). Note that same actor can belong to several groups, but also groups may act as actors allowing us to collect them into other groups and so on. The model we adopted is shown in Fig.: 2 where any actor may belong to other actor or may be a set of other actor. Actual we do not introduce any new entity.

Similar consideration can be done about resources, they also exhibits a hierarchically behaviour then we model them as a hierarchy of resources representing a resource in terms of other resources. For example, the resource COMPUTER can belong to a CLASSROOM which is also a resource.

The hierarchical description permit us to simplify the design because we do not introduce any new entities but permit to extend the complexity of the system to be developed. Therefore the resulting system is able to model either simple system or very complex one.

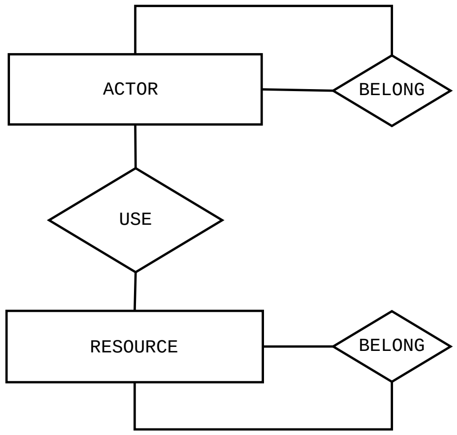

Figure 2: A more refined Resource Oriented System model

At this point we are able to answer to following question:

1. there exists a hierarchical dependence among the actors (an actor can be contained inside another one)?

2. there exists a hierarchical dependence among the resources (a resource can be contained inside another one)?

3. is it possible to easily represent these hierarchies?

2.3 Adding privileges

The model described in the previous sections is good enough to representing all requirement of the systems that we are considering, including all the relations between the actors (or the groups of actors) and the resources (or categories of resources). What was still not modelled are the constraints on the actions that actors can execute on the resources. What we want to introduce is a method that allows to model, in general way, the permissions associated to each group (or each actor). It must permit to express the permissions only during the instantiations phase and, above all, it must allow a dynamics management permitting to change, also in run-time, privileges associated to the actors and/or resources. The adopted solution is an *Access the Control List*-like approach, which permits to create a grid of authorisations in a simple and detailed way. Therefore, one new entity is necessary to describes *privileges* and relations between actors and privileges and between actions and privileges.

The engine has been developed leaving actions undefined; these actions will be added during the instantiation phase depending on the information systems we want to implement. This task is accomplished by interface that supply the functions implementing the events related to a given action execution.

Fig. 3 shows the entity-relations description of the whole system, that is the system that keeps in consideration all the requirements.

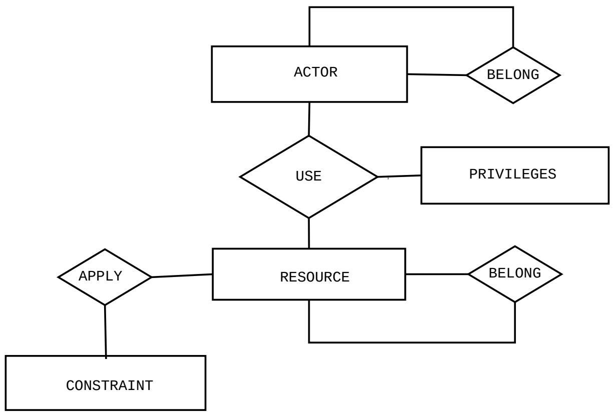

Figure 3: Entity-relations description of the whole system

The introduction of these entity and relations allows us to answer the following questions:

1. What actions are allowed for each resource?
2. Which people (or groups of people) are qualified for executing operations on resources?

3 A web based engine implementing RMF

The schema presented in the above sections permits us to model the system being implemented, but to make actual the abstraction we also implemented an engine which permit to hide the operations on resources. Presence of schema and engine allows to design an information system resource-oriented without boring with details of tables, basic operations and all management action that will be delegated to RFM-engine.

The main functions provided by the engine can be grouped based on the entities that manipulate. For example, there are functions for the insertion, elimination and selection of the resources, categories, groups, actors and so on. In Table 1 are summarised the main categories of functions provided by the engine.

Insertion, elimination and selection of categories
Insertion and elimination of categories-resources links
Insertion, elimination and selection of resources
Resources booking management
Resources hierarchy management
Insertion, elimination and selection of groups
Insertion, elimination and management of actors
Operations definition and management
Groups privileges allocation
Actors privileges allocation
Management of operations-resources-actors links

Table 1: Engine functions

Obviously, the implementation of the engine does not place constraints neither on the choice of the programming language, neither on the platform and/or the architecture. However, we have decided to develop the RMF engine using the PHP (PHP: Hypertext Preprocessor), and then integrate it inside the web-based client-server architecture.

Fig. 4 shows the architecture of the proposed information system. It looks like the WWW architecture which is the winner in the client-server arena[3]. The figure shows the three components of the architecture: server-side, Internet e client-side. First is implemented by an application which is able to manage several *pages* according to the rules of http protocol. Third is also an application running on the client machine which provide a architecture independent graphical interface: the *browsers*. In the last decade the capability to interfacing the http server to database in order to manage data in the server raised and several environment and languages has been developed. All ingredients to cook an information system which is architectural independent, quite general, distributed are present. The advantage we have are:

- user interface easy to do;
- standard user interface;
- cross-platform system;
- high degree of flexibility;
- high level of modularity;
- independence from commercial software;
- no installation is required on client side

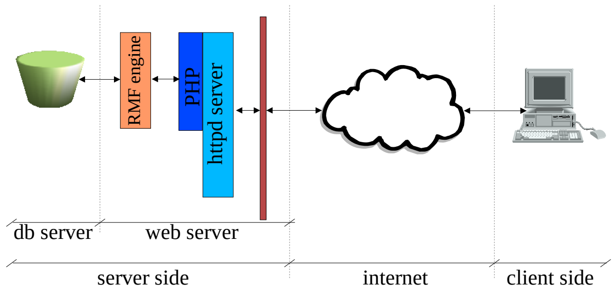

Figure 4: *RMF architectures*

As previously said, the function needed to interface with general system was implemented in PHP, which is a scripting language server-side, cross-platform and HTML-embedded. Thus, it is possible to interface with engine via web-server, the engine is independent by the platform (both hardware and software), and, at last all the engine resides into the server.

4 A complex application of RMF

The goal of this section is to show the use of engine to manage a real problem. In particular we present the management of a part of the computer centre of the faculty. Following the methodology shown in precedence, the first step to implement the information system is the analysis of the system in order to identify the resources and the actors. The computer centre is composed by three rooms of computers, a classroom, a print room, an office as well as two rooms for the servers. There are two types of rooms (named A and B) for the computers. These two rooms do not have internet access. The C room is a room of Linux machine. Some of the identified actors are: sysManager (computer centre manager), teachers, students,... Besides some of the identified resources are: roomA, roomB, roomC, PC1, PC2,... Fig. 5 highlighted a part of the tree that expresses the hierarchy of the resources.

The second step of the methodology consists in the identify the operations executed by each actors on the resources, the constraints on the operations and the requested privileges. In particular we have identified that:

- A resource can not be booked if its father is already booked. For example, a computer in the room A can not be booked if the room is already booked for the same period.

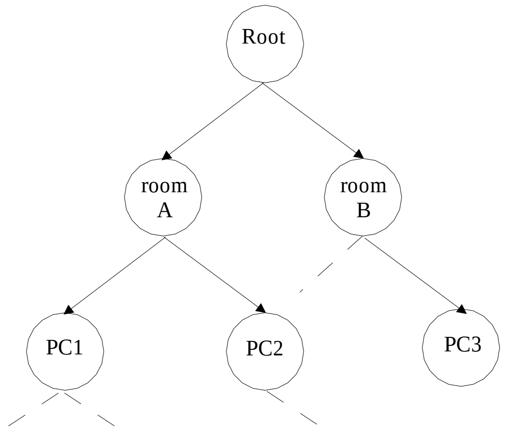

Figure 5: Hierarchy of the resources

- Students can only reserve a computer per day.
- Students can reserve a computer for the day of the booking demand and for the next day.
- Students can reserve a computer as far as the period of booking has not started for more than 10 minutes.

To manage the booking of PC, we first need a new operation. We decided to create one specific operation called *bookingpc* that will be dedicated, in the interface, to the booking of the pc. It will be associated to all pcs of the computer centre that can be reserved by the students. So, we can assert very precisely who will be able to make this new operation. For example, we can assign this operation to a set of user by the privilege *world*. This privilege is the basic privilege for all the people of the faculty. If we associate this privilege to the operation *bookingpc*, all the people of the faculty will be able to reserve a pc. At the end of the second step, all the operations and privileges of the information system of the computer centre are defined. In Table 2 are shown some of the privileges and the relative operations.

Privileges	Operations
world	bookingPC
didactic	bookRoom, bookMaterial
soft	install, deinstall, update
manager	activate/de-activate

Table 2: Privileges and operations

At last in the Table 3 is shown the correspondence between actors and privileges.

Actor	Privileges
sysManager	manager, didactic, soft, world
teachers1	didactic, world
...	...
student1	world
...	...

Table 3: Actors and privileges

The last step of the methodology was the interface development. Functions that control the interface with the actors (students, teachers, sysManager,...) use the functions provided by the engine to control and manipulate the database, populated with the information about the resources, the actors, the privileges and the operations belonging to the information system of the computer centre.

5 Conclusion

The paper discussed the development of the information system of engineering faculty. The proposed system is analized focusing on the concept of resource, therefore the system is modeled only in term of resources and actors. In summary, our first goal, development of a definite system was switched into a more general one: find a general approach able to model a whole class of system implementing, also, an engine. The purpose of the engine is to provide a set of function supporting common requirement of the resource-oriented system.

The peculiarity of the system lies into the capability to cover several system reducing the effort needed to perform the deploying of the system saving the time to develop the kernel of the analysys that was done once.

A complete example of a subsystem of the engineering information system is discussed in the last sections aiming to highlight the milestone of the proposed approach.

References

[1] A. Silberschatz et Al. - Strategic directions in database systems - breaking out of the box. ACM Computing Survey, Vol. 28, No. 4, Dec. 1996, pp. 764-778

[2] C.J.Date An Introduction to Database Systems. 6th ed. Addison-Wesley, Reading, Massachussetts, 1995

[3] D.E. Comer Internetworking with TCP/IP, Volume I: Principles, Protocols and Architecture. 3rd Ed. Prentice Hall, Englewood Cliffs, New Jersey, 1995