Designing Distributed Databases from an Organisational Perspective

António Rito Silva, Helena Galhardas,
Paulo Sousa, Jorge Silva
and Pedro Sousa
IST/INESC, R. Alves Redol n°9, 1000 Lisboa, PORTUGAL
Tel: +351-1-3100287 – Fax: +351-1-525843
email: Rito.Silva@inesc.pt, Helena.Galhardas@inesc.pt

This paper will be presented at the European Conference on Information Systems (ECIS '96), Lisboa, Portugal, July 2-4, 1996.

Abstract

In this paper we present a case study of the database system design in the context of an organisation with special emphasis on the replication concern. This case study allows us to identify abstractions and concepts of organisational modelling which are reusable in similar cases. We used the TASKON/OOram method and tool to support the case study development.

Keywords: Methodologies and modelling for CoopIS, Systems Integration and Interoperability.

1 Introduction

Perceiving organisations in terms of their information systems is a common attitude. The information systems structure, often called schema, gives a partial description of the organisation, which integrated with other such descriptions, is held to be the organisation model. This bottom-up approach for perceiving organisations has been proven to be unsuitable to meet the new challenges faced by organisations: improvement and change in order to constantly adapt to new environment requirements. The bottom-up approach does not offer a global view of organisations where changes can easily be applied. Thus, a top-down view of an organisation is required. In this view, information systems are the result and not the cause of the organisation model.

The organisation model, defined independently of the information systems, permits a global description of the organisation, where management requests apply using the correct level of abstraction. Moreover, the organisation model is the reference when integrating other information systems in the organisation.

The design of information systems in general and database systems in particular can take advantage of an organisational perspective. Once the global organisation model has been defined, new database schemas should be designed and integrated as parts of the global one. Furthermore, the organisation model permits the description of interoperability requirements among databases since inter-database semantics is maintained.

Organisations replicate information for two main reasons: circumstance and design. Replication by circumstance happens, quite often, when the organisation grows and changes in an uncontrolled way. It is locally decided when and how to replicate, based on subjective factors, e.g. communication between departments. The amount of replicated information brings the possibility of inconsistency. Nevertheless, replication is necessary for the normal behavior of an organisation. Replication by design corresponds to a conscious attitude where replication is driven by both computational and organisational factors. Traditionally, computational factors for replication are identified as availability and reliability: availability permits access to replicated information without remote accesses; reliability hides failures of remote databases from the local computation. Organisational factors for replication can be driven by geographic distribution, strategic decisions, security concerns, process tracing, and so on.

Inside organisations, sometimes it is permissible that replicated information is not consistent. Inconsistency can be allowed since most information is not critical and it is clear from an organisational perspective where to find up to date information and, if it is the case, when to reestablish consistency. We call replication models to the different perspectives an organisation has about its own replicated information. The replication models should be described within the organisation model. This way, consistency of replication policies can be inferred from the organisational model achieving an interoperable system.

Information system design from an organisational perspective should be integrated in a development process which uses a set of concepts, a notation and a set of stages.

Concepts, notations and stages are based in the approach described in [Jacobson 95]. We use the concept of use case, and ideal and real object model to describe the organisation. We enrich these concepts with special views, relevant for perceiving replication, and special kinds of object, relevant for representing replicas and functional departments.

We apply the proposed development process to a case study: the process of purchasing equipment at INESC. The case study is used to exemplify the reverse engineering stage, where replication by circumstance is identified, and the forward engineering process, where replication by design is defined during the construction of the real object model. Finally, the real object model is used as part of the distributed database development requirements.

In order to support the case study development process we used the method and tool described in [Reenskaug 96]. The OOram method is suitable for the proposed task as it supports the concepts and stages of our development process. The OOram method is built upon a software engineering concept which is described using four perspectives. In the first one (organisational perspective), several roles/actors are described as well as their tasks and responsibilities. The second one determines the work-process stages (user need analysis, requirements specification, design and implementation). In the third one, a set of tools (role modelling, business modelling, etc) supports these four stages. Using these tools to support work-process stages results in an information model (fourth perspective). Some possible directions in the use of a framework for replication, as a reusable component of the implementation model similar to the Model-View-Controller example in [TASKON 94], are pointed out.

The paper is structured as follows. The concepts we use for modelling organisations are described in section 2. In section 3 a model for an organisation use case is presented. The use of the OOram tool to support the integrated process of organisational modelling and information systems design is presented in section 4. Conclusions are presented in section 5.

2 Context

This work integrates two different areas: organisational engineering and distributed databases.

2.1 Object-Oriented Organisational Engineering

Organisational engineering, as described in [Jacobson 95], encapsulates two concepts: organisation reengineering and organisation improvement. With organisation reengineering, organisational processes are questioned and, perhaps changed. New functionalities may be added to the initial organisation structure. After reengineering its processes, the result should be maintained and optimised, taking into account costs and quality factors, in other words, organisation improvement.

In order to support the reengineering and improvement tasks, detailed and systematic information about the organisation should exist. Models are able to describe information in such a unified way, so the company can easily be understood by all levels of the hierarchy. In particular, an organisation model describes the company environment (every entity with whom the company interacts to accomplish its functions) and the interfaces between the company and that environment. This model shows what the company does, how it does and when it does.

Object oriented models are also revealed to be suitable for describing organisations as they can model the company in a way that is easily understood. Furthermore, objects in object oriented organisation models can be mapped directly into information system objects. Thus, the language spoken between organisation, system and software developers is the same, which certainly reduces the risk of errors.

During object oriented organisation modelling, a reengineering process encloses four steps that are called reverse organisation engineering, envisioning, forward organisation engineering and installing the new organisation.

During the reverse engineering step, a model of the actual organisation is built. This allows us to analyse it and find the organisation processes that need to be reformulated. After that, an envisioning task should be performed. Its result is a specification of the new organisation targets, based on the company organisation strategy and user requirements. The objective of the forward engineering step is to produce a view of the new company, to build an information system to support it and to test it on a small scale. Finally, the new processes are installed in a pilot project.

The organisational models must include two consistent views of understanding the company: the outside and the inside views. The outside perspective describes the company and its surrounding environment using use cases. The organisational procedures are explained >From the point of view of the services delivered to clients. The inside view describes the organisation from the point of view of the functional and hierarchical structure, the tasks performed and the different types of resources used. The notation employed here is object oriented.

2.2 Distributed Databases

The interoperability of distributed databases is a challenging demand of organisational information systems infrastructure. The interoperability preserves local autonomy of databases and allows the existence of data replication [Litwin 90].

This paper focuses on the issue of data replication in the context of an organisation and its information systems. The replication of data is desirable but should be controlled according to some consistency criteria.

In [Silva 95] we proposed an approach to the specification and design of distributed applications using three levels of abstraction: model, policy and mechanism. Models describe the expectation of users about the application, or system behavior. Policies define algorithms which support application models. Mechanisms define functionality which is used by policies to implement their algorithms.

Some work has been done in the multidatabase area in what concerns the need of maintaining consistency of related data. In particular, [Rusinkiewicz 91] proposes the specification of constraints among multiple databases defined in terms of time, data state, and operation occurrence. The specification is done using interdatabase dependency rules, which do not compromise any algorithms. These kind of rules are suitable for the specification of models for the organisation replicated data.

Several policies have been designed to support the needs of data consistency. For instance, primary copy [Budhiraja 93, Ladin 90] is a well known algorithm, which supports a model where copies state never differ.

3 Modelling the Organisation

We use our own organisation, INESC, as a study case to verify the advantages of a development process which integrates organisational modelling with the specification and design of the information system replication.

We chose the process of purchasing and registering goods. People can request the purchase of goods, as computers or desks and chairs. Before office supplies are ordered, the correspondents requests must be approved. Approval depends on the type of the item and its necessity and also on the available budget. Once approved the item is ordered from an external entity, the supplier. When the item is delivered at INESC it has to be registered. If they have been moved, the registered goods are checked and their location is updated, annually.

3.1 Reverse Engineering

During reverse engineering, the organisation must be modelled to be understood. The first model to be constructed is the use case model which describes the system from an external point of view.

The use case model uses three types of concepts: organisation system, actor and use case. An organisation system symbolizes an organisation and defines the boundary between what is organisation and what is environment. An actor is a role that someone or something from the environment performs when interacting with the organisation. A use case consists of a sequence of actions the organisation must execute to satisfy the customers' requests. It must describe exactly what the organisation does.

We identify several use cases in this process: Purchasing of Goods, Purchasing of National Goods, Purchasing of Foreign Goods, Lending Goods, Goods Entrance, INESC Goods Entrance, Non-INESC Goods Entrance, Repaired Goods, Goods Identification and Goods Identification Checking as is shown in the figure below.

A use case (figure 1) is represented by an ellipse. An actor is illustrated by a stick figure. Cooperation between actors and the use case is shown by an arrow. The "extends" association between use cases is represented by a dashed arrow. It means that, for example, Purchasing of Goods description can, in the case of buying a national product, conform to the description of the Purchasing of National Goods use case.

In this paper we detail the Purchasing of Goods use case since it involves several departments from INESC and their information systems.

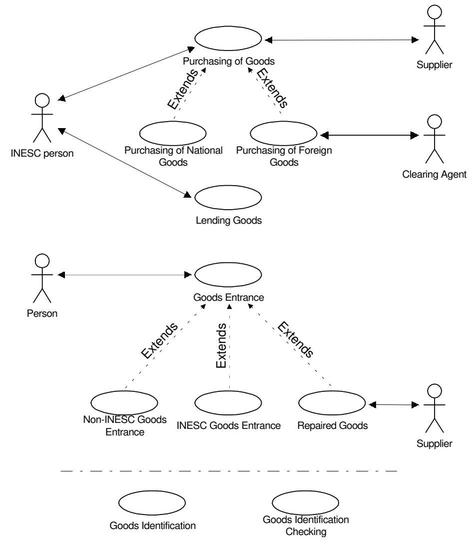

Figure 1: Use Case Model.

A person that works at INESC regularly buys office supplies. To request its purchase, he/she has to fill in a form (Model 26) and deliver it to his/her department manager. This form will travel over several INESC departments that will authorise, control, technically evaluate it and finally order the goods. After being delivered to the person that ordered it, the goods are paid for.

The internal description of the use cases is done in the object model.

The object model shows how the system is internally structured in order to supply a set of services to customers. It employs three types of concepts which are: entity objects, control objects and interface objects. The entity objects represent the products and things handled in the organisation. Control and interface objects represent tasks in the system. Interface objects represent a set of operations executed by a resource in the organisation. These objects communicate with the organisation environment. Control objects also represent a set of operations but are executed without direct contact with the exterior.

We propose new views and concepts to be used in the object model. These views are relevant for the understanding of the organisation and their information systems.

The first view describes the tasks associated with the use case. It can be easily obtained from the use case description and corresponds to the first attempt to identify objects.

The actors involved (figure 2) are INESC person and supplier of goods. They interact with the use case through the interface objects "Form model 26 filling in" and "Purchase processing", respectively. After filling in form model 26, a sequence of control objects that represent several tasks to be executed takes place. The purchase order must be authorised by the department manager. Next, the form is given to the secretary who must deliver it to be processed and archive a copy of it. Financial evaluation follows and then, technical approval is asked for and is either given or not. Afterwards, direction authorisation is asked for and is either given or not. Then the money amount that corresponds to the purchase is marked as reserved. After this process is ended, form model 26 is duplicated and the copies are archived by the logistic department. Then, the form is numerated sequentially. Depending on whether the item is national or foreign, the correspondent use cases take place. In both cases the purchase is processed and the supplier is contacted.

The second view describes resources and data involved for each task. This view permits the identification of informational objects and departments. Data which is accessed by the task, is represented by entity objects. Departments, which are responsible to fulfill the task, are represented by the special kind of object, that we introduced, called the resource object.

Resource objects permit the identification of INESC functional departments. For instance, the resource secretary belongs to an administrative department.

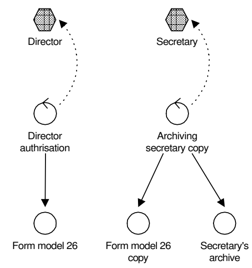

Figure 3: Resources and Data View.

On the left side (figure 3) the resource involved is the director and the task performed is the authorisation of the purchase. It is given on the form model 26 which is the entity object. On the right side, the resource is the secretary. The activity is the process of archiving the form copy. The entities handled are a form copy and the secretary's archive where it is to be saved.

The third view describes all points where information is replicated. This view is relevant to understand the information consistency inside the organisation. We introduced a new kind of relationship between entity objects called the replication relationship, and a notation to represent consistency rules between replicas. The latter is based on [Rusinkiewicz 91].

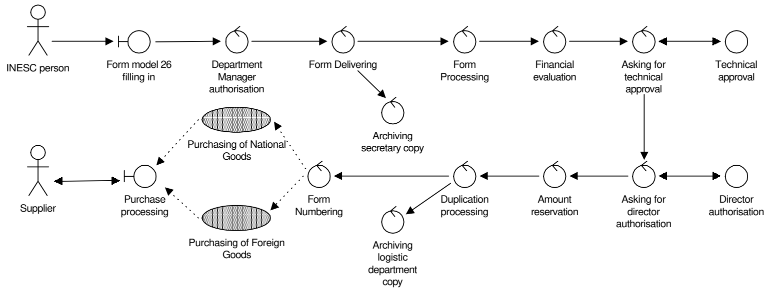

Figure 2: Tasks View.

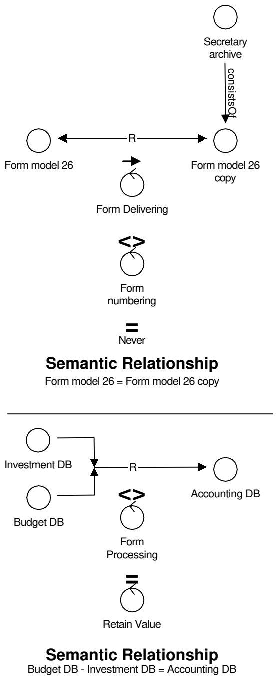

Figure 4: Replication View.

Figure 4 shows two excerpts of the information model that illustrates two data replication model cases. Each replication model includes the replicas identification and the consistency rule for the replicated data. Each consistency rule is composed of a semantic relationship that describes the initial relationship between the values of the replicas; a “->” rule that indicates when the replicated value is created; a “<>” rule that tells when replicas become inconsistent; and an “=” rule that establishes when the replicas become consistent again.

At the top of this figure is specified when the piece of paper that constitutes form model 26 is duplicated. Namely, it is duplicated by the form delivering control object. A line with an "R" links form model 26 original to its copy. The creation of the copy is represented by the “->” above the duplication processing control object, which must point to the copy entity (creation rule). The model 26 copy is saved inside the secretary’s archive.

The “=” rule means: "when will form model 26 original and its copy have equal values". This will never happen, in fact, because the secretary will save model 26 copy at the beginning of the whole process of handling the purchase request. This copy will never be updated with the authorisations that will be stamped on the form afterwards.

At the bottom of this figure, we have another example of replication. In this case, we have three databases (computational archives) that exist already in the system. The budget database saves the financial annual predictions for all departments, the investment database saves the outgoings for each department and the accounting database saves the organisation’s general accounting. Besides other details, the accounting database gathers information that comes from the other two databases.

Data saved in the investment database is modified whenever there is the intention of ordering a purchase. Budget database is accessed in order to check if there is enough money for the purchase. The accounting database is updated after the form model 26 is authorised by the director.

Following the figure, investment database values are modified when the form processing control object is instantiated (defined as the "<>" rule). Meanwhile, accounting database values are not updated. They will become updated when the amount reservation task is executed. When there is consistency, the three databases are semantically related: the accounting database saves the actual values, the budget database saves the predicted budgets at the beginning of each year and the investment database saves all the expenses. These last two work as "sub-parts" of the first one.

Finally, the fourth view shows the decomposition of entity objects in terms of their components. Furthermore, it associates the information systems with the departments which are responsible for them. It is useful if we want to understand the existing information systems.

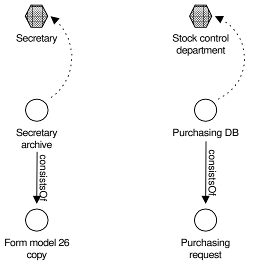

Figure 5: Information Structure View.

On the left hand side (figure 5) we can see that the secretary archive is composed of form model 26 copies and is handled by the secretary resource. The "consists of" association between the secretary archive and form model 26 copy entity object represents the secretary archive composition. On the right hand side, the composition of the purchasing database is explained. It is accessed by the stock control department and saves several purchasing requests.

3.2 Forward Engineering

The goal of our study case is to adapt and optimise the existing organisation. This is achieved by redefining the information system and improving some use cases. This redefinition and improvement are based on the targets established during envisioning.

In this section, we present the improvement of the previous use case, Purchasing of Goods, and the redefinition of some of the information systems it handles.

The forward engineering step defines two different object models: the ideal and the real models. The ideal model explains the organisation without taking into account how it can be implemented. For instance, it does not consider the fact that the company might already have an organisation running or that it might be geographically distributed. The real model considers all these factors.

3.2.1 Ideal Model

The ideal model is described using the task view. It considers only the tasks that add value to the final result and it includes the intrinsic functions only.

Each INESC person (figure 6) wants to purchase some goods. He/she should fill in a form model 26. This action of filling in is represented by an interface

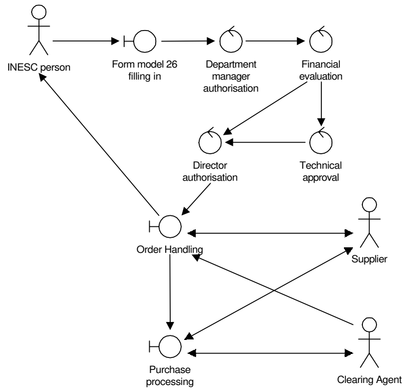

Figure 6: Ideal Model.

object. Then, the department manager authorisation control object is instantiated. During this task, the manager must read the form and authorise its contents. Then, a financial opinion must be given. This is achieved by the financial evaluation control object. During this task, the budget is checked to find if there are available funds. If the item is a computer, a technical opinion must be given. This is executed by the technical evaluation control object.

For all kinds of goods, the director must read the form and approve the purchase. This is represented by the director authorisation control object. Afterwards, goods must be ordered. This is done during order handling which is an interface object as it communicates with the actor supplier. Besides, it also makes the interface with the INESC worker, who made the request initially, to deliver the products.

In the end, a purchase processing interface object is instantiated. It illustrates the payment that must be given to the supplier and, makes the interface with the clearing agent when goods are received from an outside country.

3.2.2 Real Model

The real model is driven by the integration of previous systems and the existence of a previous organisation.

The first step consists of the definition of the information system infrastructure and the replication model.

The information systems needed for the real system are driven by two main reasons. Firstly, each key department has its own database that gathers the important data for that department's function. Secondly, some existing software systems are kept. This happens to the accounting database that maintains the whole organisation accounting information that should be integrated in the new system.

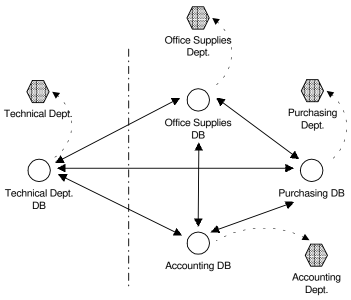

Figure 7: Architecture.

Thus, the databases that support the real system are (figure 7): the office supplies database, the purchasing database, the accounting database and the technical department database. Each of them is accessed by a different department in the organisation.

In particular, each INESC technical department has a database that saves the form model 26 contents corresponding to all the purchase requests issued by that department. The accounting database is a legacy database as we have already referred.

The flow of information between databases and consequent replication allows that each department leader is aware of all the details relating to the process of handling the purchase; the office supplies department is aware of the important accounting dates (amount reservation dates, supplier and clearing agent payment dates) and purchase details (order stuff, date of item reception, receipt date); and the purchasing department is able to access important accounting dates. Hence all the functional departments keep track of the state of the process of handling the purchase request, each with a differing amount depending on its role in the organisation.

There is a strong need to replicate data between these databases. This replicated data consists mainly on the important accounting dates, authorisations given and purchase details. In figure 8, we give two examples of consistency rules for the databases aforementioned. These rules are described using the notation introduced in section 3.1. The first example of consistency rules relates to the replicated information between each technical department database and the office supplies database; and the second one relates the replicated information between the accounting database and the purchasing database. The "=" rule for both cases should be triggered by the underlying computational system.

In the first case, both databases save information about the form model 26 contents and should also save information about the state of the process of handling the purchase request (namely the authorisations given, the important accounting dates and the purchase details). If the correspondent values saved in both databases are consistent, then the state of the process known is the same (semantic relationship). The "->" rule indicates when the form model 26 is replicated in the office supplies database. The "<>" rule indicates the tasks that, when executed, add new data to the office supplies database copy of the form (tasks where authorisations are given). The office supplies department warns each department about these modifications only when it collects all authorisations ("=" rule) - then the consistency is re-established. But each technical department manager may want to know how far the process has progressed. So, after two days without any news, the technical department may issue a request for information to the office supplies department. Then, this one should provide information about the actual state of the process ("=" rule).

In the second case, the idea is similar. Data that is replicated is, once again, about form model 26 contents plus information about the state of the process of handling the purchase request. The "=" rule tells that the accounting department should warn the purchasing department about modifications everyday.

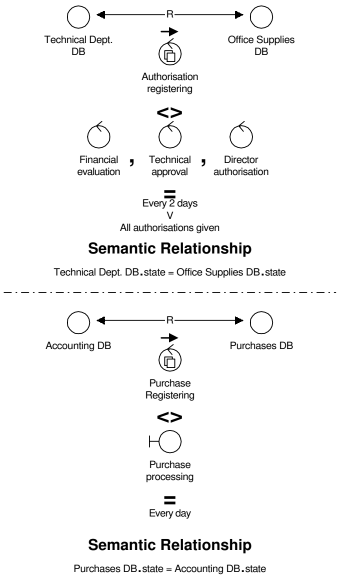

Figure 8: Replication Model.

Next step is the redefinition of the object model tasks taking into account the department and information system infrastructure.

Figure 9 is intended to indicate the functional departments found as essential to the real model. In addition, the various control objects responsible for the task forwarding and replication of data in the real system are identified. A new notation to represent the last ones is introduced.

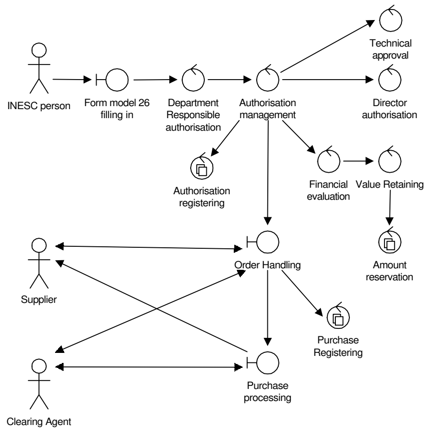

Figure 9: Task View of Real Model.

The most relevant differences between this task view of real model and the task view of ideal model (figure 6) are the introduction of a composed control object (authorisation management) and of a duplication control object.

The authorisation management task is executed by a department that coordinates the activities of the technical, senior board and accounting departments, in what refers the authorisations given. The authorisation management task is considered to be finished only after all the three authorisations are completed.

The duplication control objects are instantiated whenever the form model 26 is duplicated into the office supplies, accounting and purchasing databases. The symbol used to represent this object means, by itself, the replication of form model 26. The text that comes below these objects indicates the additional information that is saved in each of the databases.

4 Using the OOram Method

The TASKON/OOram method is used to support the organisation business modelling of our study case: INESC's purchasing of goods. Due to space limitations, we have chosen to show only the forward engineering process using OOram, as it is the most interesting in the context of distribution issues.

The OOram system development process involves four models which are built in sequence: system user model, system requirements model, system design model and system implementation model. During the system user model, the exact needs of people who are going to use the organisational system as well as customer needs are determined. The system requirements model objective is to describe the system as it is seen from its environment. It must use the results from the previous stage and refine them into a specification that emphasizes the interface for the users and the information modelling. The system design model describes the system as it is seen by its developers adding some technical details to the system requirements. In this phase, special focus is given to the reuse of designed modules. System implementation converts all these specifications into a real system.

The OOram tool supports these four models through a set of views. The ones we use are: collaboration, process, scenario and semantic views. In the collaboration view, different roles and their corresponding responsibilities are defined. The roles interfaces, and messages changed between roles, are also described. The process view shows the flow of data changed between roles and the consequent actions taken by them, in a scale of time. The scenario view pictures the sequence of messages changed, events triggered and conditions imposed, in a scale of time also. The semantic view captures the relationships between roles that model entities.

We use all these concepts to support and represent the object oriented organisation model for our study case that is explained in section 3.

4.1 System User Model

A mapping between the system user model and the ideal model (sectiown 3.2.1) was done. In both, the goal is to identify and model the main organisational processes that must be running to satisfy customer desires, regardless of the computational support and organisation structure.

The entities presented in the ideal model were the control and interface objects, which means that entity objects were not included. This agrees completely with the system user model where information entities are not modelled.

During the system user modelling the collaboration and scenario views were produced.

4.2 System Requirements Model

The requirements modelling includes the task-tool, the information and the non-functional requirements specification. These kind of necessities match the real model descriptions, where a possible previous organisational structure's running is taken into account. From the system user model results and from the envisioning step, we can conclude there are activities that need to have computational support and that are not implied by the process functions - these are mainly architectural requirements. These tasks are due to the existence of distinct departments that are geographically separated and that need to archive data and have different visions of the same item of information.

According to figure 7 (section 3.2.2), the four databases identified correspond to four database tools (accounting, purchasing, office supplies and technical department database tools). Each of them interacts with a functional department as was said before. This creates a collaboration view where departments and tools are modelled as roles.

Each of these tools handles its own data. For instance, the purchase database saves the contents of form model 26 and a set of data that mirrors the state of the process of handling the purchase request as was said in section 3.2.2. For every tool, an information (semantic) view that models data handled by the tool is done. It would be interesting to be able to represent completely the consistency rules using the OOram available views, but this has not yet been achieved.

As we have already demonstrated, there is a need of data replication between the four databases cited. So, the service that we are most interested in is the one that is connected to that need - the replication service. This replication service must be provided by the four databases and is differently instantiated for each of them.

In figure 10, the purchasing database tool and its replication service are shown through the collaboration view as well as the information model for the replication service. This informational view models the data saved by the purchasing database that needs to be replicated.

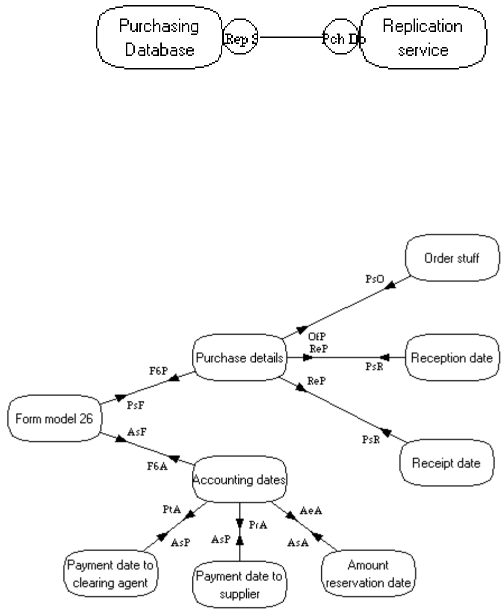

Figure 10: Purchasing Database Tool.

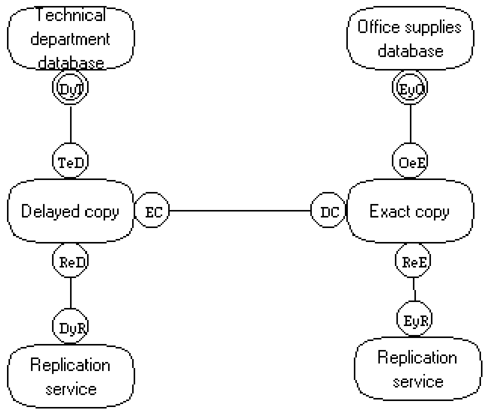

Figure 11: Consistency Rule Implementation.

4.3 System Design Model

The key issue in system design is reusing. The service that is being focused is the replication one. The system design phase main aim is to include patterns of replication to describe the tool design. In particular, the relationship between the technical department and the office supplies databases is developed in figure 11. The correspondent consistency rule as was described in figure 8 is implemented here. An exact copy-delayed copy pattern is introduced to explain the embedded replication policy.

The office supplies database maintains information about the state of the process of handling the purchase request. The technical department database also maintains this information, but the updates are not received at the same time. In first place, the office supplies database updates its copy and afterwards, this update is transferred to the technical department one. This propagation takes place whenever the office supplies department collects all authorisations (a message is sent from this department to each individual department through the DC port) or whenever each technical department asks for actualised information (a message is sent in the inverse direction if two days pass without any updating message, from the EC port). This last message is sent if each technical department wants to know which authorisations have already been given. The replication services supplied by each of these tools are now associated to the copies. The messages changed between the exact and delayed copies correspond to the accomplishment of each of the replication services, respectively.

5 Conclusions

This paper presents a study case of an organisation modelling activity which is used as the base of the organisation distributed database specification and design. This approach suggests that the global perspective employed by the organisation model allows a smoother integration of the interoperable information systems. The issue of data replication was particularly emphasized.

We proposed some new concepts and views to be included in the organisation models. Some of these new concepts and views were driven by the need of separating the models, policies and mechanisms of replication.

We adopted the OOram method to support the description of the organisation models and of the information system databases. OOram seems to be suitable for the predicted work, but it was not possible to describe the replication models, since we have used a declarative notation.

Lastly, we believe that the results already achieved should be tested in more study cases in order to be checked and improved.

The design and implementation of patterns of replication appeared a promising area for future work.

References

[Budhiraja 93] N. Budhiraja, K. Marzullo, F. Schneider, and S. Toueg. The primary-backup approach. In S.J. Mullender, editor, *Distributed Systems, 2nd Edition*, ACM-Press, chapter 8. Addison-Wesley, 1993.

[Jacobson 95] Ivar Jacobson, Maria Ericsson, and Agneta Jacobson. *The Object Advantage Business Process Reengineering with Object Technology*. Addison-Wesley, 1995.

[Ladin 90] R. Ladin, B. Liskov, and L. Shrira. Lazy replication: Exploiting the semantics of distributed services. In *Proceedings of the Ninth Annual ACM Symposium of Principles of Distributed Computing*, pages 43–57, Quebec City – Canada, August 1990. ACM.

[Litwin 90] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[Reenskaug 96] Trygve Reenskaug, Per Wold, and Odd Lehne. *Working With Objects: The OOram Software Engineering Method*. Manning Publications Co., 1996.

[Rusinkiewicz 91] Marek Rusinkiewicz, Amit Sheth, and George Karabatis. Specifying Interdatabase Dependencies in a Multidatabase Environment. *Computer*, 24(12):46–53, December 1991.

[Silva 95] António Rito Silva, Pedro Sousa, and José Alves Marques. Development of Distributed Applications with Separation of Concerns. In *Proceedings of the 1995 Asia-Pacific Software Engineering Conference APSEC'95*, pages 168–177, Brisbane, Australia, December 1995. IEEE Computer Society Press.

[TASKON 94] TASKON. OORAM/WE methodguide draft version. Technical report, TASKON Work Environments, 1994.