Collaboration Techniques to Design a Database

Sergio Gálvez, Antonio Guevara, José Luís Caro, Iván Gómez, Andrés Aguayo

Sergio Gálvez
ETSI Informática, Universidad de Málaga
Málaga 29071, Spain
galvez@lcc.uma.es

Antonio Guevara, José Luís Caro, Iván Gómez, Andrés Aguayo
EU Turismo, Universidad de Málaga
Málaga 29071, Spain
{guevara, jlcaro, ivan, aguayo}@uma.es

Abstract. In this paper, we describe methods for the analysis and design of Cooperative Object-oriented Information Systems. These methods involve users in the development of general Information Systems much more than traditional ones like Metrica or UML. We introduce a set of cooperative tools, based on a powerful and user friendly graphic interface that facilitates user participation in new stages of system construction and in re-engineering processes. We focus on the database definition tool: CDB (Cooperative Data Base), and show the need for cooperation among users and the specific collaboration characteristics of this tool.

1 Introduction

The SICUMA[1] (Cooperative Information Systems of the University of Málaga) research group is currently developing a project whose objective is to change the approach to the development of Information Systems (IS) for tourist sector companies and organizations. Our fundamental objective is to apply CSCW (Computer Supported Cooperative Work) methods to the development of IS to increase the quality and reliability of business information systems. In order to accomplish this, the aim is to fully engage the users themselves in the design and construction of the information system they will later work with, in order to diminish long-term costs and increase the competitiveness of the companies in the sector. The present work is geared in this direction.

Section 2 presents the basic characteristics of the Cooperative Methodology and Tools needed to implement the foregoing in practice, create higher quality software, and be more user-friendly for the final user. Section 3 focuses on the most important of these tools: the CDB (Cooperative Database), whose function is to provide the necessary mechanisms for users to cooperate with each other regarding the decisions

[1] SICUMA Research Group is financed by the Junta de Andalucía with Registration number TIC-160

about what data they want to have stored in the IS. Simultaneously, an operational database is automatically created, which can then be fine-tuned by IT engineers, should this be required. Section 4 is focused on the collaborative issues included into CDB and how these have influence over the quality of generated prototype. Finally, Section 5 deals with the conclusions and the relevance of our cooperative methodology for the development of software applications.

2 Cooperative Methodology. Cooperative and Object-Oriented CASE Tools

In order to produce applications with a minimum degree of quality, the IS department of any company (SICUMA group focuses on Tourism companies) has to apply a Software Engineering Methodology. Our work considers the evolution from Structured Methodology to Cooperative Methodology ([13], [14] and [22]), emphasizing at each point how data can be created in a cooperative way.

Our methodology seeks the participation of all these members, so they have to agree about the best way to split the IS into several subsystems that can be handled more easily. The interrelationships existing between such subsystems requires that the development methods are applied cooperatively. The cooperative tools [15] that will be described further on, will be used in each subsystem and then the outcome will be integrated to set up the global system.

Equally, we propose a radical change in the analysis and design of the system: up to now IS departments have been in charge of carrying out the design, whereas in our proposal, the users do this in a cooperative way. This is achieved by implementing cooperative and object-oriented tools which assist users in analyzing and designing their system while maintaining control over the entire process. Such tools should have powerful but friendly user interfaces capable of handling the transfer of information between the different subsystems of the final IS.

Finally, constructing and testing the system will be done simultaneously with the cooperative creation of a set of instructions that will allow the final user the optimum use of the system. This means that code will be generated, databases implemented, and integration tests carried out. Therefore, this stage has to be done co-operatively by all the agents involved in the analysis process, and will be assisted to a great extent by automated computer processes. This is clearly illustrated in figure 1, where we may see the sequence of stages of the methodology.

The SICUMA group is developing a coordinating and cooperative module [14] based on this methodology. The key element in this module is the Cooperative Data Dictionary [15] which allows us to carry out the entire process in a cooperative way, and handle all the information about the objects described in the system. The cooperative functions carried out by this module are:

- Giving advice on how to personalize the user interface on the basis of the operations carried out by other users.
- Cooperative consolidation of the data introduced by users.
- Automatic consistency control during all the design processes.

- Integration of the different sub-IS designed by users.
- Generation of a relational model based on the user's data needs.
- Coordination between data models and process models.
- Code generation from the merging and description of primitive processes.

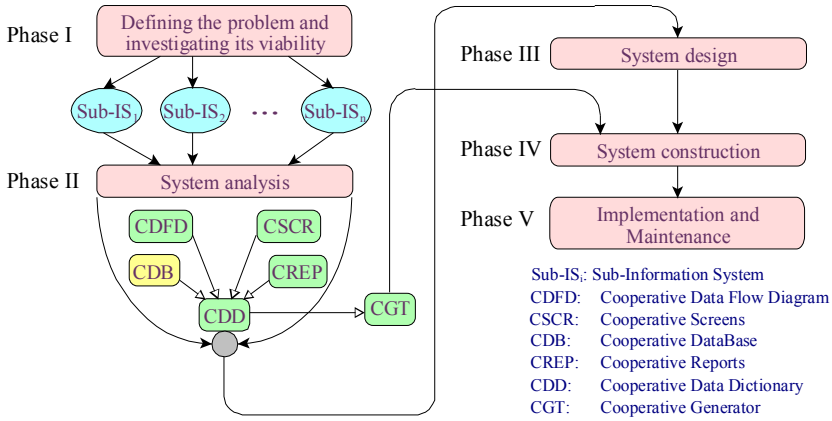


Fig. 1. Cooperative tools in the cooperative life cycle

In order to manage all these tasks, the coordination module is made up of a set of strongly interconnected tools, all of which make use of the Cooperative Data Dictionary. These tools are:

- CDFD: a cooperative data flow diagram tool.
- CDB: a tool for cooperatively designing the database.
- CSCR: a tool for designing the visual appearance of the application.
- CREP: a tool for generating reports.
- CDD: the cooperative data dictionary.
- CGT; a code generator.

The purpose of our work is to develop the set of tools that make up the cooperative, coordinating, object-oriented module. We also focus part of our attention on the workflow methods needed to facilitate the distribution of tasks in the cooperative design of the IS ([3] and [4]). At present, we have built the prototypes of CDFD and CDB which allow the user to specify elements and later classify and group them by areas of interest. In the following section we describe the basic characteristics of the CDB.

3 Cooperative Tool for Database Design

One of the main elements of any Information System (IS) is the database. The role of the database is not only to store information regarding the different elements of the

system (clients, items consumed, expenses, etc.), but to establish relationships between them which will enable obtaining useful reports [16]: statistics on the months with more clients, departments with a deficit, control of resource availability, etc.

Therefore, among the tools mentioned in the previous section, the CDB -- a cooperative tool for the definition of the database -- has the greatest significance.

3.1 CDB (Cooperative Database)

The CDB follows the Cooperative Methodology philosophy, as it promotes the shared participation of users who, after all, are the ones who know best what data they need and how it should be related. Once achieved, the aim is to automatically create a useful, high quality database. The CDB is based on three main ideas [11]:

- It should be easy to use for designing the necessary data elements.
- Elements used by multiple users have to be designed in cooperation.
- The engineering aspects of the database are transparent to the user.
- Storing of visual information.

3.2 SMF (Semantic Model of Forms)

The SMF [10] is a semantic model whose basic component is the form, or more particularly, the form structure, which we call design. It enables the modeling of entities, their attributes/fields, and relationships. SMF enables us to achieve two basic objectives:

a) Obtaining data requirements. To do this, we use the CDB tool (Cooperative Data Base), which is basically a user-interface that allows the creation of designs (i.e., form structures). The task of storing designs and verifying their consistency is carried out by a Form Management System (FMS).
b) It works as a prototype. The user should be able to work with specific electronic forms, i.e., they must be able to create, delete, and relate them to each other. In this way, a feedback process takes place that allows the fine-tuning of designs already built based on actual usage. In order to support this aspect, the power of CDB and FMS tools has to be enhanced.

The merging of the SMF model, the CDB interface, and the FMS data management system gives rise to a fully user-oriented cooperative architecture [11].

3.3 The need for cooperation

The CDB tool simulates the production of forms on the computer, which will be filled in later and stored in a register, taking advantage of computing power at each stage to avoid insertion of redundant data, and speed up data access. The users interact with the CDB and design forms electronically, in a way that is not so very different to

what they do when they have to design them to order them from the printers. The definition of a form by a given user generates part of the database in an automatic and transparent way, so that when another user fills in a form with specific data (in the system validation stage), such data will be stored in the database created. With this method, we can make it easy for users to describe the information they need for their work without having to know anything about databases.

However, the same design may have as many views as users do actions over it, and only the Data Processing Department (DPD) can analyze the design as the fusion of all its views (master form, also named family or entity). Figure 2 shows an instance of this global view by using the diagrams proposed in [11].

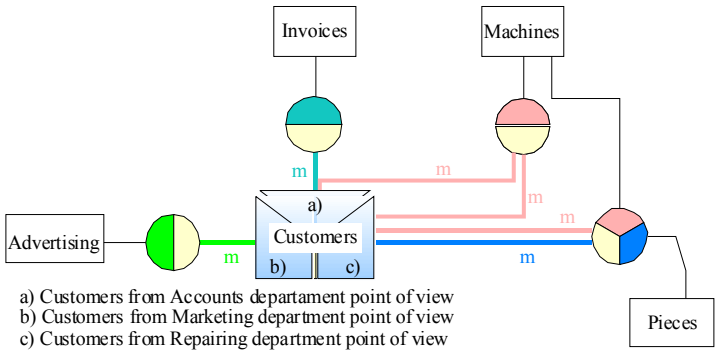


Fig. 2. A global view of the entity Customers composed by three sub-views

Thus, the need for cooperation appears, which is based on the following points:

- Use of a single database common to all designs.
- Consistency of information shared by several users.
- Punctual accesses to forms defined by others users to share data
- The CDB has to maintain this data consistency when changes are made in data shared by several forms.
- Maintenance of independent views for each user.

In this working method, the analysis and design stages are closely related, because users can use the forms they have created to introduce data. This means that the actual structure of the form is a preliminary design or prototype which can be used to validate the system before proceeding to its final construction.

3.4 Basic Operations for Form Design Building

The user can carry out four basic operations to build a form design:

a) Creating a new design. A design is associated with a regular entity, which we call family. Several users can have different views of the same family. Each view is a design.

b) Inserting fields in a design. Each field has an associated name and domain. Available domains must be meaningful for users: letter sequences, numbers, dates, and not much more.
c) Transferring fields from one design to another. This is not a simple operation of copy & paste, but sharing data at a logical level. This enables the creation of relationships between entities (designs) in a transparent way.
d) Creating homogeneous data lists, called repetition groups. In this way, one-to-many relationships between entities -- even anonymous entities -- can be established.

All these operations involve a inter-user collaboration, either explicitly or implicitly. A collaboration is implicitly started when CDB recognizes, through conflict detection algorithms, that a user task (usually, design or field creation) may have some relation with another users tasks in the same or another sub-I.S. Explicit collaborations allow users to obtain information about other users' designs, only for reading purposes, and to include pieces of such designs into the own designs. Also, a user may do a copy of a somebody's else design and include it into the own system's view, in order to do some modifications over it, leaving the original unaltered. Using color codes simplifies the perception of the shared data blocks.

CDB can be understood as a visual interface. Other types of interfaces are currently being taken into consideration by our research team [1].

CDB is a multidocument interface that holds information about each active design, along with data regarding the selected component, and a structural view of the current design by means of a field tree. Users can insert several kinds of atomic components in a document: checkboxes, text fields, lists, and choices, thus producing the H block in the level in which the components are inserted. For each of these components, users can choose between introducing text, numbers, dates or Yes/No words. They can also introduce repetition groups (R blocks) and insert into them any components, which permits the building of designs with every possible tree structure. Each repetition group may have a name and a relationship assigned to it, so it can be used as a design on its own. Subdesigns are introduced by drag'n'drop, and a color-based technique is used to easily identify the source of each subdesign. In this prototype, colors play an important role in helping users to distinguish between the different blocks that make up the design. CDB supports any sentence from data structuring/manipulation languages, although modifications are sent to FMS one by one to make FMS implementation easier. Any conflict detected by FMS during the execution of a requested sentence is communicated to CDB via an exception, so every design involved is restored to its previous state.

Once all the information system designs are built, users may switch to working in form mode. When this switch takes effect, FMS generates all the relational tables that will store the form information inserted by users. This is a reversible switch, but involves the consequent disappearance of such tables and their contents. In any case, this may be the most rewarding phase from the users' point of view, because they can see how their effort is translated into something real and capable of giving solutions to their data needs. The mechanism that allows the use of the same relationship in several designs supplies the model with cooperative capacity, so data inserted into a form by one user is automatically reproduced in any other design that uses the same

relationship. In addition, the concept of subdesign makes the insertion of data much easier: a simple double-click over a subform will show a window containing all the forms of the source design. The user selects the desired one, and the requested data is automatically transferred to the subform.

Figure 3 shows a form when fulfilled. The design it belongs to includes data blocks coming from other designs (note the use of different colors in different sections of the design). Tools and Projects. Even it is allowed to make an autoreference: an employee may have inside data about another Employee: his/her manager in a Task.

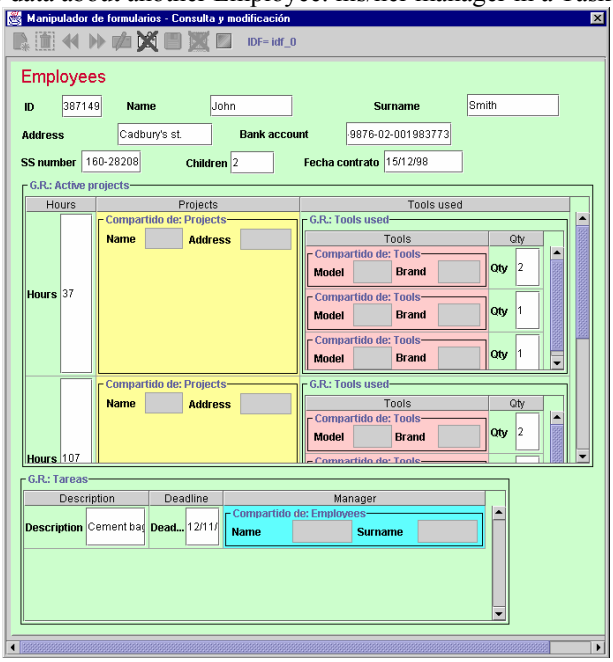


Fig. 3. An employee's form created and fulfilled with CDB. Each colour identifies a different subdesign created by a drag'n'drop action

4 Collaborative Characteristics of CBD

In order to test the model viability, we carried out a study with 50 students from the Tourism School at University of Malaga. Half of the participants were trained in E-R models as outlined in Métrica v.3. The other half were trained in SMF (CDB). First of all, both groups were asked to solve problems ranging from basic- to medium complexity using only a ball-pen and paper. The basic exercises were successfully solved by 60% of the E-R group and by 76% of the SMF group, while the more complex ones were solved by 12% and 36% of the E-R and the SMF groups, respectively. When users were supplied with computers, the E-R group worked with Access (i.e., an E-R model that only allows one-to-one and one-to-many relationships). Both groups were provided with a new set of problems of similar

complexity. In this new situation, the basic exercises were solved by 68% and 92% of the respective groups, whereas the more complex ones were solved by 24% and 56% of the groups respectively. The statistic have been obtained by using a wide set of quality evaluation criteria similar to one used on several subjects in the Tourism studies. Concerning SMF, the main complaint from users was that its interface was rather slow. Concerning CDB, the main problem encountered was the faults in conflict detections (we will see this later). In spite of this drawback, the use of the SMF prototype led to a large improvement compared to E-R models. Therefore, even greater advantages are to be expected once the efficiency of the prototype improves.

Once we have exposed the framework and the results of the usage of the architecture SMF, we will proceed to remark the main characteristics of CDB concerning to communication, collaboration and coordination among users in order to design a database.

Table 1. Results from the model E-R vs SMF evaluation

	On paper		**With a computer**	
	E-R	**SMF**	**E-R**	**SMF**
Basic	60%	76%	68%	92%
Complex	12%	36%	24%	56%

4.1 Adaptive Interface

Usually each group of users need to manage a different set of tools from the whole set of the interface. For example, when solving complex problems, one of the most executed actions against the interface is to create data lists. When solving another kind of problems, it is very useful to create multimedia items (mainly photos in our case of study). Even more, the framework is different according to the stage the user is working in: creating designs or testing the automatically generated prototype. If there have been created many multimedia components in the design stage, when the user is going to test the prototype the interface has to make easy to insert these kind of data type.

As is proposed in [18] an appropriate system of tailorability may be essential in order to achieve a successful groupware system like CDB. In our work, tailorability has been introduced in two ways:

- Direct: a user may change the location of the buttons in the tool bar, as well as include new buttons to carry out -a priori- seldom operations.
- As advices: each action a user accomplish on the interface has a path, where the shortest path is for actions started by typing keys, and the largest path is for actions contained in menu items with depth 2 with access to a dialog (e.g., to insert a bitmap as the background of a design). By using the length of the paths, CDB controls the frequency of each action and give advice to the user about to insert new buttons into the tool bar in order to minimize the inverted effort.

If the user accept the advice given by CDB to insert a new button, the interface do not insert it instantaneously but do acquire the mouse and keyboard control and, with the Robot class of Java, carries out exactly the manual operations needed to add the button. In such way the user is trained and stimulated.

4.2 Dimensions of the users' tasks

We do not focus uniquely in the user to build accurately a set of designs but these designs to be correctly integrated in the cooperative information system also. This is because the inter communication mechanisms must be managed very carefully (Klein et al deals in [19] with the design of complex systems). In spite of design operations usually are asynchronous (due to that several users work on different designs simultaneously), it is obvious that they must to contact among them in several ways:

- Synchronous: with instant messaging tools.
- Asynchronous: with e-mails.

On the other hand, the communication may be started by the user or by CDB. CDB start a communication with another user when it detect a conflict between two designs, and it recommends the actions to be taken.

In addition, there is a interesting collaboration regarding to the use of the designs as if they were applications prototypes. When a user insert a form (a form is a design fulfilled with data), s/he is enhancing the view that other users have on related data.

Finally, the test carried out with students of Tourism had a very high component of urgency, due to the limited time to model the problems. This had an influence upon the concentration of the participants in the developed tasks, as is exposed in [20].

4.3 Conflict detection

The designs created by different users are interrelated by two main ways:

a) Sharing data blocks: a user take a piece of another design and incorporates as is to his own (this action generates a one to many or many to many relationship between the participant designs).
b) Creating different designs that belongs to the same family or entity. For example, each user may has a different view of the design of Employees (though FMS maintains an unified view, CDB controls the different views; e.g. figure 3 shows a typical view of an Employee form).

There are many authors who provide different categorizations of conflicts in a wide sense. For further information, [9] is a good reference. From the point of view of a System construction, the main error source is derived from the heterogeneity of the data schemes managed by each group of involved users. This heterogeneity appears

when when there area incompatibilities between the attributes that describe a same entity in differents contexts.

As mentioned in [6] two main kinds of incompatibility may be distinguished: structural and semantic.

4.3.1 Structural incompatibility

A structural incompatibility appears when the same attribute is defined in different schemes but in different ways. The sources of this kind of incompatibility are:

- Type conflict: this conflict appears when the same concept is defined with different type constructors in different schemes. For example, a concept is represented as an entity in a scheme, but as an attribute in other scheme. Or a single attribute in a scheme corresponds with a set of attributes in other scheme. This problem is very frequent in Spain where a person has two surnames: a scheme may merge the two surnames into one attribute but other scheme may define two attributes: first (father) surname and second (mother) surname.
- Format conflict: several schemes may use the same entity with different input/output formats: the first day of a week may be Sunday (in USA) or Monday (in Spain).
- Unit conflict: several schemes may use the same entity with different units of measure: millimeters or inches.
- Granularity conflict: data regarding measures may differ in granularity; e.g. month sales or week sales.

4.3.2 Semantic incompatibility

Semantic incompatibility is usually due to attributes defined in a similar way in different schemes, but with actual data being different in each of them.

In any cooperative process in which there is involved many people in design tasks, may appear the problem of Unique Name Violation (UNV) due to a lack in a standard terminology to denote the shared concepts. Each participant is involved in building up a piece of the system, by using subjective identifiers to name entities and concepts. This subjectivity affects our work because the designs created by each user have to be integrated into a whole and there is not an infallible communication channel to resolve this kind of conflicts. This problem get worse due to the asynchronous nature of the design process. In this sense, when several users work on the same design, probably in different locations and time slices, it is frequent they to employ different vocabularies, so two main kinds of inconsistencies may appear:

- Homonyms: this appears when the same identifier is used into independent schemes to mean different entities. For example, the identifier **State** may represent the state of birth of an employee in a scheme, but other one may may use the **State** to identify the situation of an employee: full or partial time.
- Synonyms: this appears when different identifiers are used in different schemes to mean the same entity. For example, **DateIn** and **DIn** may be used in two different schemes to mean the date on check-in of a hotel guest.

In order to solve the UNV problem, several works has been analyzed, and most of them conclude that there is no solution if analysts and users do not focus in depth on the problem. Perhaps, the most suitable work we have deal with is [1], where natural language is used to define a set of functions to characterize the properties of each entity; also, the work exposed in [21] has an interesting approach: Palapoli & al. use graph algorithms to group the concepts that characterize a same entity.

In particular, CDB has a conflict management system that detects when a user starts to build a component very similar to other already built by another user. When this is detected, CDB pops up a warning message and shows the document that includes the similar component, giving an advice to the user: it is strongly recommended to start a communication with the owner of the displayed document. The tools and algorithms to detect conflict are a priority for us in order to ensure the success of CDB [12].

4.4 Coordination made by the DPD

Our architecture deals with three important problems. In one hand, participation of unskilled users into analysis and design techniques to build an Information System. On the other hand, the construction of user interfaces to make easy to develop and test prototypes. And finally, the collaboration among users in order to the development to be global, complete and consistent.

Leaving this kind of system to work in autonomous way is not realistic, if we hope a high quality degree results (at least at first stages of building and using designs). Thereby it is essential that a Data Processing Department supervises what the users do, trains them adequately, gives advices in critical design stages, and even refines the users' designs when needed (with express approval of the designs' owners).

In the usage tests of CDB, the members of SICUMA acted as DPD, and this participation gave us a feedback to fine-tune the interface and the methods to trigger automatically the intelligent participation of CDB in the communicative process (like the usage of Robot class previously mentioned, and fine-tuning basic conflict detection by name matching).

4.5 Sharing components

We have stated in section 4.3 that users may share pieces of their designs in three different ways:

a) An user may insert into an owned design any piece of another design, yet of hisown or not. This insertion is carried out by a drag'n'drop gesture, but it is not a copy&paste action, because CDB and FMS maintain a link to the original component. Hence FMS knows that there is a relationship between both designs (one to one, one to many, or many to many). And even more, FMS knows which concrete components are shared. This allows us to estimate how a modification in the original design may affect the designs that refer to it. To try such modification gives a report of all the users that share components with the original design, and is given an advice to start a communication session (perhaps asynchronous).

To make easy sharing pieces of designs, a user may incorporate any other user's design into his/her view for read only.

b) Many users may build different designs that refer to the same entity (named family in SMF). By means of conflicts detection, CDB can find out that two users are modeling the same entity from different points of view (figure 3). When this happens, any drag'n'drop operation between such designs must be interpreted as a copy&paste action, because a relationship between an entity and itself has no sense (nevertheless CDB allows reflexive relationships when finds out duplicated components after a drag'n'drop action). Hence, the effort performed by a user in a design is reused by others.

c) Users may test the usefulness degree of their designs by using them to insert actual data (see section 4.2). Thereby, the relationships created in the design stage are used in the usage stage. For example, if an Invoice contains a piece of the Customer's design, when an actual invoice (named form) would be fulfilled, a double clic on the shared customer's components (emphasized with a different background color) will pop up a dialog with all the actual customers already inserted; once the user had selected the correct one, its data will be automatically inserted into the invoice form.

   In addition, users cooperate indirectly when they fulfill a form. The same relationship may be seen in different ways in different designs. E.g., the relationship between a customer and the resources of an hotel (sauna, tennis playfield, etc.) may be used in several designs, as Customer's one and Resources reservation's one. If an employee of Reservation adds the customer X as user of resource Y at hh:mm, this information will appear automatically into the from of the customer X that Reception holds in order to register it when cheking-out X (this reservation may disappear if X cancels the reservation). Information about the time of reservation may be not included into the Reservation's form of the customer. The system is integrated.

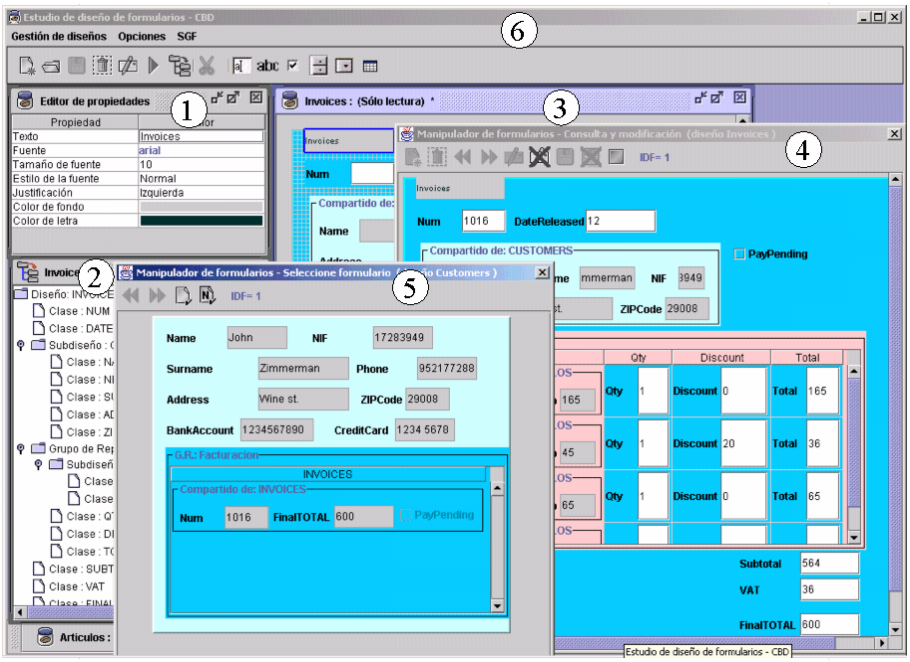
Figure 4 shows most of the main components of CDB. The development/execution environment is similar to other commercial tools, such as the Form Builder of Oracle. Design and execution are included in the same environment, so it is easy to fine-tune the designs once they have been used.

4.6 Training on CDB

The main reason, CDB has been put in action with students of the last courses of Tourism is that they have a profile very similar to the users this tool is oriented to: they have a basic knowledge about computers, they knows well their work, but they are unskilled users of analysis and design technical tools.

Thereby, training users about how to use the tools is very important in order to achieve a high degree of success, mainly in the first steps of usage [17]. Training has been divided into three stages:

- Initial training. In a training session, the users were introduced into the main characteristics of the tool: first the standalone working methods, and last the collaborative methods.



1.- Component's properties manager: color, type, relationships selector, etc.
2.- View of the fields tree os selected design
3.- Current design. In this window is made any design modifications.
4.- Current form. Window associated to a design, that allows to insert, modify and delete forms.
5.- Window to select forms. In this is selected a source form to fulfill a subform.
6.- Development/execution multidocument environment.

Fig. 4. A view of CDB

- Support during the usage stage: the members of SICUMA acted as DPD, aiding users with problems derived from tool usage exclusively: we were "ignorants" of the problem universe to model.
- Automatic support by CDB: tool usage (by using the mentioned Robot class), conflict detection (there is manual ways to watch the designs built by other users), and advices to start a communication process (a user can start a communication with other user or with any member of a group when he/she wants).

4.7 Technology used

Technological issues implemented into CDB have influenced notably in the obtained results with actual users ([8] is an interesting study about how different technology issues affects groupware systems when designing a wide range of products, yet physical or services). In fact, the main problem encountered by users when using CDB has been the slow response in several actions.

This is due to CDB has been developed in Java with a RDBMS based on Oracle 8i, what requires many more resources than supported by the computers used. In addition, the prototype of CDB is version 1.0 and efficiency was not the main goal. Although CDB is a complete functional prototype, we minimized the impact of complexity when developed it, so several details of CDB implementation may overload FMS and the event queue, making inefficient the response time and leaking productivity.

In addition, with the FMS prototype currently developed, the usage stage of forms requires an automatic phase to generate a layer of relational structures to support actual data. This intermediate phase impacts negatively in management comfort.

5 Conclusions and further work

As we have seen, the SICUMA group has developed a prototype of CDD, a groupware tool for many users to collaborate in order to define their data needs by means of electronic forms. The SMF architecture provides the formal layer.

The tests of CDB with users with knowledge about touristic companies management and unskilled in computer science, have proved that SMF improves appreciably the requirements analysis phase of an Information System. This improvement is wider when CDB comes into scene, in contrast with other tools with different approaches, like the widespreadly used E-R model.

When using CDB, the most complex problems need a DPD to support users to resolve them, in order to design the most complex structures of the SMF.

At present, we are working in a PSEE framework (Process-Centered Software Engineering Environments) [7] based on a workflow management system that is actually a meta-CASE tool to make a high level integration of the results obtained with the tools proposed in the cooperative methodology of section 2.

In this sense, a user will manage CDB conducted by the workflow management system in a cooperative framework with an underlaid map of workflow processes. This map models the cooperative methodology, focused in building up the system data model as of captured requirements by means of CDB's designs [5].

References

1 Bhargava, H.K., "Using Quiddities for detecting semantic conflicts in information systems", Journal of Organizational Computing, 5(4), 379-400 (1995)

2] Carrillo, A., A.Guevara, S.Galvez, J.L. Caro, Interacción Guiada por Objetivos, en: Proc. of INTERACCION 2002, Leganés-Spain 68-75 (2002)

3 Caro, J.L., A. Guevara, A. Aguayo, L. González, "Workflow Technlogy: An Application for Tourism Management", Proc. of ENTER'97, Vienna-Austria 307-317 (1997)

4 Caro, J.L., A. Guevara, A. Aguayo, S., Galvez, "Tecnología workflow aplicada a los sistemas informáticos de gestión hotelera" Proc of Turitec'99 Málaga-Spain 145-158, (1999)

5 Caro, J.L., A.Guevara, "Workflow: a Solution for the Cooperative Development of an Information System", Business Process Management, 9(2), 208-220 (2003)

6 Chatterjee, A., A. Segev, "Data manipulation in heterogeneous databases", SIGMOD Record, 20(4), 64-68 (December-1991)
7 Derniame, J.C., B.A. Kaba, and D. Wastell (Eds.): "Software Process", LNCS Springer-Verlag Berlin-Heidelberg 117-164 (1999)
8 Divitini, M., B.A. Farshchian, T. Tuikka, "Internet-based Groupware for User Participation in Product Development", ACM SIGCHI Bulletin, 32(1), 31-35 (2000).
9 Easterbrook, S., "CSCW:Cooperation or Conflict?", Springer-Verlag. ISBN 3-540-19755-9 (1993)
10 Gálvez, S., A. Guevara, A. Aguayo, J.L. Caro, "Forms Management System", Proc. of ADBIS'99, Maribor-Slovenia, 171-177 (September-1999)
11 Gálvez, S. "Participación del usuario en el diseño cooperativo de bases de datos. Metodología y herramientas", Ph.D. dissertation, Dept. of Lang. y Computing Sciences, Univ. Of Málaga, Spain (2000)
12 Gomez, I., A.Guevara, A.Aguayo, J.Falgueras, S.Galvez, "Objects relations on cooperative environments: dealing the vocabulary problem", Proc. of ISD'98, 305-316 (1998).
13 Guevara, A., "Planning Methodology of Information Systems Based on Bottom-up and Topdown Strategies Under Cooperative Design", 12th World Computer Congress IFIP-92, Madrid-Spain (1992)
14 Guevara, A., "Planning Methodology of Information Systems under Cooperative Design", Computer Science 2, Ed. Ricardo Baeza Yates, Plenum Press, New York (1993)
15 Guevara, A., A. Aguayo, I. Gomez, S. Galvez, J. Falgueras, "Intelligent and Cooperative CASE Tool Kit for Development of Information Systems", Proc. ISD'94, Bled-Slovenia, 609-703 (1994)
16 Hull, R., R. King, "Semantic Database Modeling: Survey, Applications, and Research Issues", ACM Computing Surveys, 19(3) 201-260 (1987).
17 Huysman, M., C. Steinfield, Chyng-Yang J., K. David, M. Huis in 't Veld, J. Poot, I. Mulder, " Virtual Temas and the Appropiation of Communication Technology: Exploring the Concept of Media Stickiness", CSCW: The journal of Collaborative Computing, 12(4), 411-436 (2003).
18 Kahler, H., A.I. Mørch, O. stiemerling & V. Wulf, "Introduction to special issue on tailorability", CSCW: The journal of Collaborative Computing, 9(1), 1-4 (2000).
19 Klein, M., H. Sayama, P. Faratin, Y. Bar-Yam, "What Complex Systems Research Can Teach Us About Collaborative Design", Proc. CSCW in Design, 15-19, Canada (July, 2001)
20 Mulder, I., Y. van Houten & H. Ter Hofte, "Dimensions of comprehensive CSCW settings", Telematica Instituut (1999). https://doc.telin.nl/dscgi/ds.py/Get/File-398/GigaCSCW110.doc
21 Palapoli, L., D. Saccá, G. Terracina, D. Ursino, "A unified graph-based framework for deriving nominal interscheme properties, type conflicts and object cluster similarities", Proc. of CoopIŚ99, IEEE Computer Society Edinburgyh-United Kingdom, 34-45 (1999)
22 Triguero, F., A. Guevara, A. Aguayo, J. Falgueras "User Participation in Information Systems Development. Techniques and Tools", ACM SIGOIS Bulletin, 16(1), 68-78 (August, 1995)