Objects Dynamic Behaviour for Graphical Interfaces in VIM = An Application Case Study =

Francisco Jose Negreiros Gomes
Arlindo Gomes Alvarenga
Luciano Lessa Lorenzoni
Universidade Federal do Espírito Santo
Vitoria
BRAZIL

Helder Jorge Pinheiro-Pita
Luis M. Camarinha-Matos

Universidade Nova de Lisboa
Quinta da Torre-2825 Monte Caparica
PORTUGAL

Abstract: An approach to model the dynamic behavior of objects is discussed in the context of graphical interfaces. The problem of visual interactive modeling for the optimized leather cutting task is considered as the application domain. The UNL's concepts of dynamic objects, roles, views and interlocutors are evaluated as supporting mechanisms for the UFES requirements in terms of visual interactive modeling.

1. Introduction

In leather cutting, we are dealing with an open class problem, i.e. a situation where

- The boundaries of the problem are not clearly stated;
- There are no strict rules to characterize the boundaries;
- There are more objectives than one which can be conflicting.

In an open problem-solving, one often applies non-visual modelling techniques which thrust upon the modeller the necessity for specifying a closed problem which approximates, as closely as possible, the open problem. Solving a closed problem (well defined boundaries and objectives) is usually trivial; but such techniques do not help in the process of defining the closed problem from an open one. This can be thought of as a representation gap. An attempt at narrowing this 'gap' is to use a framework for Decision Support System (DSS) design, which suggests that a DSS should have three subsystems: an user Interface System (IS), a Knowledge System (KS) and a Problem Processing System (PPS). This framework is represented in Figure 1.

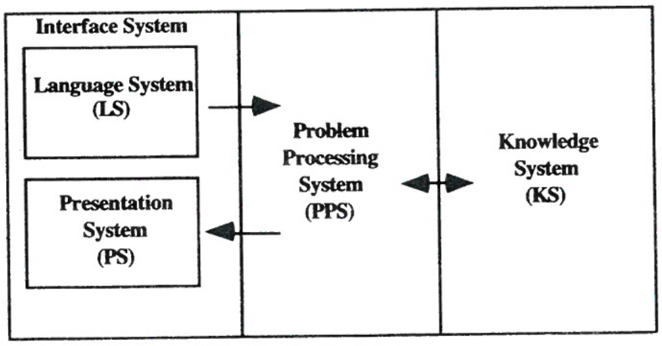

Figure 1. DSS Framework

The KS contains the knowledge about the problem domain, i.e. data models and algorithms, rule sets, forms, templates, etc. The PPS is a processor capable of accepting problems stated in the Interface System and of manipulating knowledge in the KS to generate appropriate responses to the decision -making process. The LS includes all the linguistic facilities made available to a decision-making process by a DSS. The responses from the PPS to the user are the means thereby the DSS communicates with the user and are referred to as the Presentation System (PS). Like the LS, the PS is a system of representation. Together, the LS and PS characterize the user interface. If a DSS is to be used by many different users for different tasks, its interface must be easily adaptable. In this way, we use the UNL's concept, Pinheiro-Pita/Camarinha-Matos [1], of object's dynamic behavior applied to the design of adaptive interfaces.
In this paper we deal with the design of a DSS for the leather cutting problem focussing on the Interface System in which the elements are represented by dynamic objects.

The second section makes a characterization of the leather cutting objects to be visualised, of the desired user interaction model and of the classes of users. Third section introduces the dynamic behavior model concerning the basic concepts (objects, roles, views, interlocutors) and implementation methods. Applicability is discussed in the fourth section.

2. Interface Requirements for VIM

In this section the requirements of the graphical system interface which supports the visual interactive modelling (VIM) environment are described.

2.1 Characterization of the objects to be visualised

The use of an approach based on visual interactive modelling (VIM) for the development of Decision Support Systems, gives the user the

opportunity of interacting with the system during problem formulation and modeling phases so as to establish a critical dialog on the generated solutions. In that sense, it is quite natural that the system interface is required to provide facilities for identifying the dynamic features of the various entities setting up the solution space of the problem.
Next, we describe the classes of objects to be identified and visualised in such environment, in the context of the leather cutting problem. The classes are:

- Leather Piece : Based on an appropriate input device (camera, scanner, etc.), that captures the geometry of the piece, we define the 'leather piece class' by an abstract representation of a two-dimensional bounded region.

Associated with the 'leather piece class' we define the following derived classes:

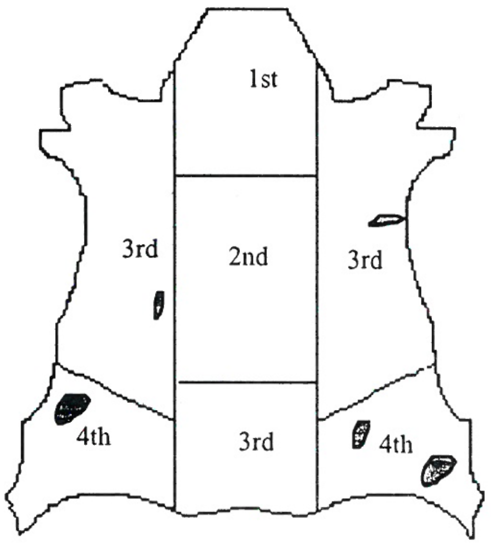

Figure 2. Graded and Defective Leather Piece

- Graded Leather Piece : From a technical point of view, it is well-known that in a piece of leather there exists a classification (1st, 2nd, 3rd and 4th) in sub-regions based on the quality of the material.

The derived class, 'Graded Leather Piece' should therefore contain additional attributes such as texture, thickness and boundary type. A partition operator may be defined on a leather piece object in order to generate, based on a quality criterion, new objects of this class.

- Defective Leather Piece : A sub-region of the leather piece that encloses a defect such as a stain, hole, scratch, etc.

A Defect operator may be defined mapping the leather piece class into objects of this derived class.

Figure 2 illustrates those derived classes.

- Mould : An abstract two-dimensional region with additional attributes that identifies the pieces that will be produced by the layout, the quality of the material required for their manufacture, the directional properties as well as other details.

On analysing the classes 'leather piece' and 'mould', in the sense of establishing a dialogue between the objects of those classes, the following operators are defined:

- Overlap : Its purpose is to detect an overlap between the moulds over the layout generated by the system on a leather piece;
- Direction : To inform a 'mould' object on the feasible set of directions for realizing the layout of the leather piece;
- Identification : Based on the technical specification, there will be learned whether a mould object may be manufactured from a given 'graded leather piece' or not;

After executing the cutting process, the production line needs establish the scheduling of operations to be carried out on the final product. In this sense, the representation and manipulation of the mould objects that belong to a given final product are better defined using an abstract graph object. In this graph, nodes represent the mould objects, while arcs establish the relationship between moulds and common boundaries.
Job scheduling, using this representation, is implicitly due to the order of the nodes in the graph.

2.2 Desired Interaction Model

The solution space must be represented by pictures. This provides for an open-problem methodology as long as the pictures can easily be re-designed. It would be wise to provide a 'core' of visual interactive facilities, which can be used to build any visual interactive model. Examples of such facilities include windows, icons, menus, tables, variables, etc. Extensions of these facilities can be envisaged for this particular application area (leather cutting).

The user should deal with a modeling system which combines WIMP-style (Windows, Icon, Mouse, Pull-down menus) workstation capabilities with the extension of conventional optimization algorithms.

These combinations allow the end user to explore a model from several perspectives within an animated display.

On an initial implementation, the set of objectives will include:

1- accessibility by end users, without technical intermediaries;

2- Flexible user interaction, allowing for incremental development, exploration and editing of a cutting plan;

3- Fluent and strongly graphical presentation of the results on user selectable subset in the model.

The Interface

The system combines the functionality of spreadsheets and optimization algorithms with a graphical presentation of the results. The interaction method is through a WIMPB (windows, icon, mouse, pull-down menus and browser) interface. Control over the evolution of an interactive section can be almost entirely mouse-driven, and even the entry/amendment of numerical values can be largely achieved by using the mouse in conjunction with an analog display of variable value.

End-users could operate with a number of defined models to be used with different data at different times. After loading a model, the user can edit it in a number of ways. For example:

a) by clicking the mouse on an action and selecting the target figure (piece) and putting it over the sheet (leather) to perform the desired action.
b) use a non-visual modelling language to define variables, constraints and parameters.
c) filling-up dialog boxes which are linked to defined models.

To create a new model, a definition facility translates the user input (in the form of a collection of visual language and algebraic expressions) into an internal representation of the system. Also the user is provided with the ability to incorporate heuristic rules appropriate for model-based reasoning procedures.

The desired model interface should have four regions:

1 - Graphic Objects Menu: this is composed of visual objects (Icons) which the user can select in order to execute a desired action.
2 - Working Area: user area reserved to the problem selection, interaction and solution presentation.
3 - Actions Menu: it displays the provided built-in operations (Methods) that can be applied to a previously selected object.
4 - Status Line: this area is reserved to display warning/error messages, the current process status, environment state and so on.

2.3 Classes of Users

The leather cutting system has four classes of users:

- Design expert: this kind of a user is responsible, for instance, for the shoe design model definition. He prepares the

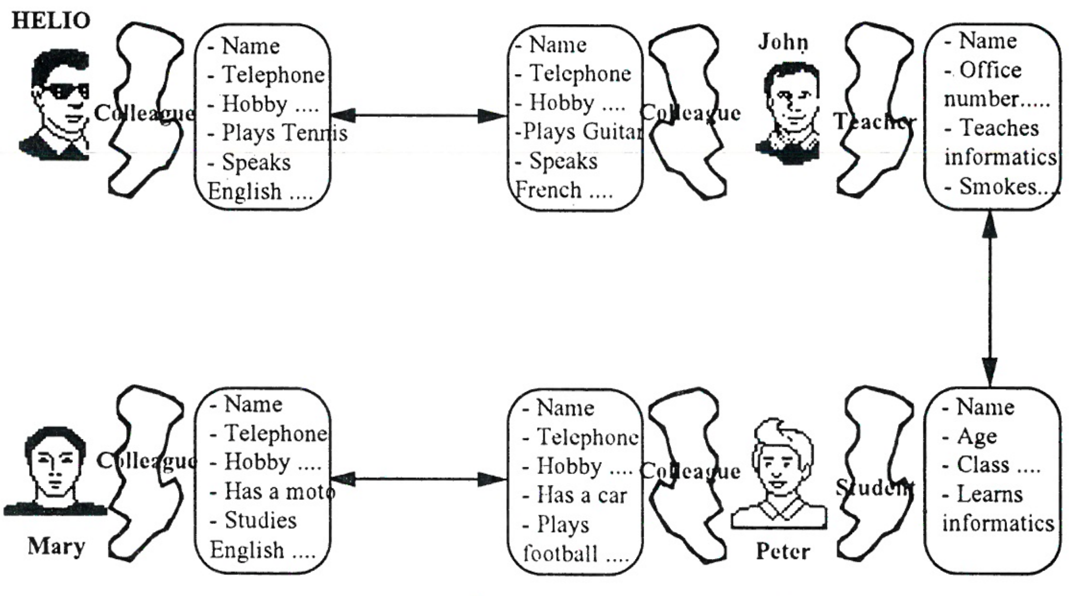

Figure 3- Dialogue Among People and Played Roles

pieces to be cut, the piece region possible allocations, precedence constraints, tolerance parameters for the piece/plate placement.

- Planner: this is the end-user of the system. He is responsible for obtaining an 'optimal' cutting plan to the specific model. Leather piece image acquisition and cutting plan generation.
- Administrator: this user performs such system maintenance action as: user definition, models catalog, knowledge base updates, product order.
- Client: this user can see a model solution. He gets the appropriate sequence of cutting steps within an order and its visualisation.

3. Dynamic Behaviour Model

Let us now introduce the UNL's approach to object's dynamic model, making an analogy with the world of people. We can admit that every person in a society is playing roles. Persons are teachers, students, chiefs, workers, parents, sons, and so on. On the other side, when a dialogue sets in between two persons, each one playing a specific role, the "access" one gets to the information of his interlocutor (including the actions he can perform) is a function of his status against the role assumed.For instance assuming the role of a teacher, John can obviously evaluate Peter, who assumes the role of student, but it is not so obvious that he can evaluate Helio assuming the role of his colleague. A simplistic view of the society, will let us conclude that when a person communicates assuming a role, the information as well as the functionalities made available, are a function of the status of its interlocutor. We call this sub-set of information and functionalities a **view** of the role. Figure 3 shows an example of people's communication.

Dynamic Behavior is, therefore, the facility that a person has in order to make different sub-sets of data/functions available as a function of the status of its interlocutor.

Taking these ideas into account, let us go to the objects' world. Under the Object Oriented Methodology, an object has a behavior defined by a set of methods that composes its interface. Sometimes, when an application is developed using that methodology, the objects are, at the same time, instances of multiple concepts. This is particularly important when we are using a frame oriented approach, where multiple relationships

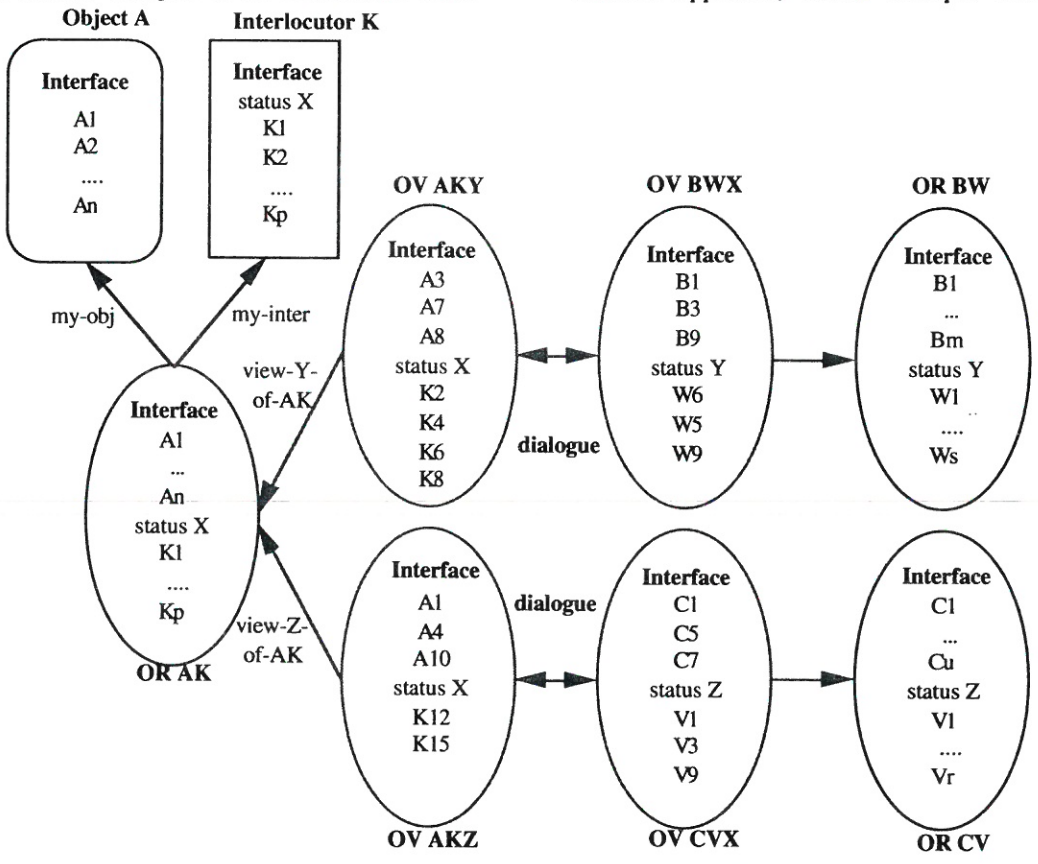

Figure 4 Dialogue Structure of Interactive Objects

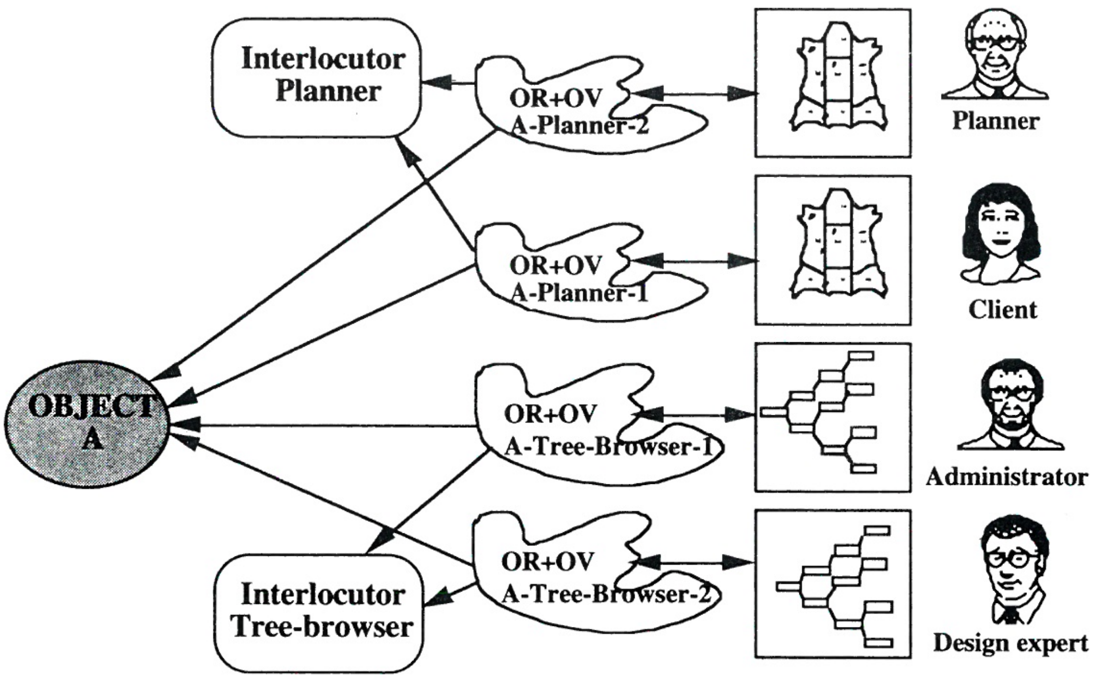

Figure 5 Typical Interface of a Complex System

among objects with different rules of inheritance can be defined. Using this mechanism, the programmer defines a structural behavior of an object. Making an analogy with the concepts introduced for the people's world, we can imagine a system where the objects also play roles . In addition, we can also imagine that during the execution of an application only a sub-set of the inherited data/methods of an object is relevant in that context, taking into account some status variables, while in other context a different subset will be relevant.

An algorithm that supplies a basis to support objects with dynamic behavior was developed by UNL's team. It is introduced in the next paragraph. Let us suppose that in an application we have defined a set of objects which can dialogue with other objects - **Interactive Objects** - and a set of auxiliary objects that model the roles that an Interactive Object can play - **Interlocutors**. When someone wants to dialogue with an interactive object assuming some role, a new object is created as an instance of both this object and of the interlocutor that models the desired role. We call this new entity **Object-Role** (OR). However, only a subset of the inherited attributes will be available for dialogue as a function of the status of the object that has started the communication. Therefore, a view of that new entity will be created. We call that view - **Object-View** (OV). Figure 4 shows an example of this dialogue structure. The OR AK is an instance of the object A and of the interlocutor K, while OR's BW and CV are, respectively, instances of objects B and C and of the interlocutors W and V. As the status of OR BW is Y and the status of the OR CV is Z, two object views of AK were created, respectively, AKY and AKZ.

Let us look in more detail on how the OV's AKY and BWX were created:

1- The OR BW sends a message to the Interactive Object A saying that it wants to dialogue with A playing the role K and also informing A about its status - *(do :interlocutor K :my-status -is Y)* ;

2- The object A sends a message to interlocutor K asking it to create an OR for the status Y- *(start-OR :status-is Y)* ;

3 - The Interlocutor K starts a set of actions:

3.1 - if an OR AK does not exist, it is created;

3.2 - if an OV AKY does not exist, the relation View-Y-of-AK is created; a rule of inheritance from OR AK

specifies only the attributes/methods that are available for status Y;
3.3 - if the OV AKY does not exist it is created;
3.4 - The interlocutor K returns to object A the name of the created view and its status.
4- The object A sends a message to OR BW containing the name of the created OV and its status - *(Done :view-name OV-AKY :my-status X)*;
5 - If an appropriate view does not exist then steps 3.1 to 3.3 will be performed at OR BW;

An important application of this algorithm is in the management of flexible user interfaces. Here the problem can be simplified since the dialogue always starts from the human expert to the system, i.e. only the user starts the dialogue. Therefore, the interactive objects are internal entities that can dialogue with an expert via different kinds of windows (roles). Another important aspect, met as a consequence of that simplification, is that it only makes sense to talk about user status. In this system there are four kinds of users. Figure 5 shows that idea.

- Design Expert : view including the shape, dimensions, type of raw material, etc.
- Planner : view including associated polygon from the leather piece image acquisition, allocation constraints and related cutting tools, demand, etc.
- Client : view including the cutting order of the piece, assembly constraints, etc.

4. Discussion on Applicability

In Section 2 we presented the interface requirements for a system that uses the visual interactive model environment designed to solve the leather cutting problem in the shoe industry. In this section we discuss an abstract representation of the aforementioned system.

In Figure 6 we describe the distinct processes involved in this abstract representation.

- User Interface Module: Provides the user with facilities like windows, icons, menus, tables, variables, etc., forming a dynamic visual interactive modeling environment for problem formulation and resolution phases;

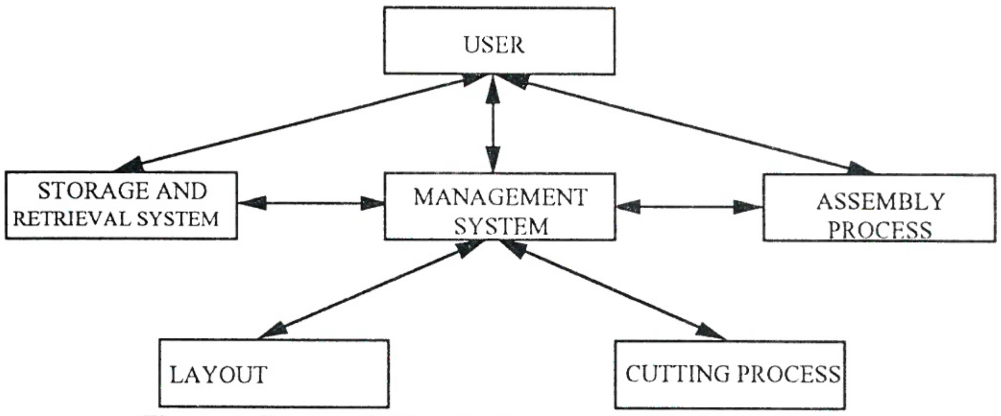

Figure 6 - Leather Cutting Environment for the Shoe Industry

Now, considering the characteristics of the leather cutting problem, we must emphasize that the use of the object's dynamic behavior model appears to be a promising approach to such a problem. In this way it is advisable to relate different views that we can have on the problem, given different processes involved, such as layout, cutting, assembly, etc.

In order to illustrate this usefulness let us consider the following example. Suppose that, in a shoe industry, the design department proposes a new shoe model. The dynamic object "leather piece" has several views depending on how the user interacts with it: for instance:

- Management System: Coordinates the entire system establishing a link between several processes;
- Storage and Retrieval System: Manages the automated storage and retrieval of the raw material and of the manufactured pieces. As to retrieval, not only the material flow and control are concerned, but also the geometry acquisition of the leather pieces, necessary for the layout problem.\
- Layout Process: Using a production system for problem representation as a state graph, this process generates a layout of moulds on a given leather piece, based on a nesting algorithm for irregular shapes.
- Cutting Process: Manages the interface

between the overall system, in particular with the layout process, and the numerical command engine responsible for cutting the pieces generated in the layout.

Assembly Process: Based on the user requirements and on the cutting plan generated by the system, such operations as colouring, details, sewing, etc., to be realized during the final product assembly are scheduled.

The system presented here requires a strong interaction with the user. For this purpose a flexible interaction model as described in chapter 3, seems promising and is being evaluated.

5. Conclusions

In this study we have described the interface requirements when applying the UNL's concepts to dynamic objects for the construction of Decision Support System -DSS, based on Visual Interactive Modeling (VIM). The leather cutting problem in the shoe industry was considered as an application area and an abstract representation of the system was also analysed.

Considering that the proposed system should be used by different people with different tasks and views and the universe of problems in CIM area that could be handled in a similar way (layout, sequencing/ scheduling, grouping and so on), we argue that the use of VIM together with dynamic object's behavior concepts is a good approach to dealing with such flexible problems.

Acknowledgments

This work is funded in part by the European Communities (CIMIS.net project) and JNICT (Sarpic and CIM-CASE projects).

REFERENCES

1. 1.PINHEIRO-PITA,H.J .& CAMARINHA-MATOS, L.M. - **Comportamento de Objectos Activos na Interface Grafica do sistema CIM-CASE**. 4s Jornadas de PPPAC - Lisboa 17-19 de Maio 1993
2. ALVARENGA,A.G., NEGREIROS,F.J.& DAZA, V.P . **The Irregular Cutting Stock Problem and Its Application to the Shoe Industry** - Working Paper - Department of Informatics - UFES - 1993.
3. BELL,P.C. , **Visual Interactive Modelling : the Past, the Present and the Prospects** - European Journal of Operations Research, Vol. 54, 1991, pp. 274-286 .
4. NEGREIROS,F.J. **SAD's Inteligentes para Planejamento em Manufatura: uma abordagem via Modelagem Visual Interactiva** - Robótica e Automação, nº10 Novembro 1992, pp. 91-93.