ASA, An Interactive Assistant to Architectural Designers

BENEDETTO COLAJANNI[1]
ANDREA FORNARELLI[2]
ALBERTO GIRETTI[3]
BERARDO NATICCHIA[4]
GIUSEPPE PELLITTERI[5]

ABSTRACT

In the management of information for the design case reasoning seems the best fit for simulating the real designer's behaviour. In order to construct a plausible interactive assistant to architectural designers three main problems are to be solved: the way of encoding and indexing technical knowledge in order to easily recover the best starting case; the way of giving semantics to sketches; the way of coming to terms with inconsistencies generated during the process. An interactive architectural assistant is proposed based on case reasoning, managing architectural information encoded in a memory of real instances of the architectural type of reference and technical information encoded according the SfB system. Its architecture is made of two main parts. The first includes case memory, case recovery engine, abstraction engine and the design board. It manages the general and specific case knowledge in its abstract and semantics given form. The second includes the tools to generate single objects composing the architectural organism both directly and in parametric form, constraint management and consistency checking. The representation of the state of the object is twofold: as a drawing in the drafting board, as a symbolic representation in the design board in which all the attributes of the object are recorded together with their relationships. The graphics of the assistant is implemented in AutoCAD environment while the alphanumeric knowledge is implemented in Kappa. The general architecture of ASA and the single modules are described, followed by a simulation of a session of work.

Key Words

case reasoning; architectural assistant; knowledge engineering

1 *Dipartimento di Progetto e Costruzione Edilizia, University of Polermo.*
2 *Consiglio Nazionale delle Ricerche, Instituto di Edilizia, University of Ancona.*
3 *Instituto di Informatica, Faculty of Engineering, University of Ancona.*
4 *Instituto di Exilizia, Faculty of Engineering, University of Ancona.*
5 *Dipartimento di Progetto e Costruzione Edilizia, University of Palermo.*

INTRODUCTION

The aim of this paper is to describe an interactive design tool, an Architectural Symbolic Assistant (ASA) able to help a designer from the conception of a functional and only qualitatively dimensioned schema, expressed in a wire-frame sketch, to the realisation of executive drawings. The chosen approach is case reasoning. Since the pursued goal is having a tool giving its assistance both in the design phase during which the functional and organisation choices are made and in the phase of technological choices, two kinds of data bases are to be created. The first is the case memory of items of the architectural type the designer is concerned with. The second is the data base of the technical elements from which the ones most apt to realise the architect's aim are chosen. The first has a range of variability and a virtuality of growth qualitatively different from the second. To be really useful it has to be rich in cases covering a wide range of design situations. In the first implementation of ASA we thought it convenient to limit the knowledge only to one architectural type: row houses.

The goal of entrusting some part of the design process to an automatic tool has always been a very hard one. Much theoretical work has been done on the subject, both on the attempt to define the very nature of the process and on the implementation of tools performing some particular design tasks. In the run the focus has shifted from problem solving techniques to case based reasoning.

Case based reasoning is a good way to try to simulate human behaviour in the field of design. Designers always attempt to recall from their memory (their personal knowledge base) when they are faced to a specific design problem in a specific context situation. The main difference between human knowledge base (the memory) and any kind of artificial one lies in the diversity of constitution. The first is, by its nature heterogeneous or, perhaps better, ill-structured. The various objects that form it have different degrees of definition. Degrees cannot be ordered in levels, because these form a continuum. Further, they are intrinsically heterogeneous, as the various parts of a single object are generally remembered with different degrees of exactness. Another characteristic, in a sense a consequence of this heterogeneity, is a certain degree of randomness of the access to memory; this allows in a given situation the reuse of something (parts or entire structures, as well as only some particular effect, or the image) from an object which no rational criterion can be applied. Hence the difficulty, if not the impossibility of building up a case memory and algorithms of retrieval simulating human memory in the field of architecture.

On the contrary, a case memory, if we want the retrieval to be efficient and rapid, needs to be highly structured. Structuring means stiffening the description of the object, taking decisions on the relative importance of the

various characteristics of it. The main difficulty remains the way of the access to knowledge base via indices which depends on the description of the objects that constitute the knowledge base. However, the description depends on what the designer is allowed to do with the retrieved case. This use can be simply looking at it as a source of ideas for further manual designing or can be considering it as a starting solution on which to work with change operations till the solution is reached. Hence the representation of the case is to be as articulate as possible in order to be able to operate on single elements of the item. As design decisions are taken at different levels of abstraction, the case memory must contain different levels of representation. As far as the row houses are concerned three levels are provided: an accessibility graph among rooms, a wire-frame representation, a fully drawn representation in plan and in section. The specification of how each level is implemented is given in the description of Case Memory. Keys of access to the data base can be any subset of the characteristics constituting the description of the case.

The characteristics by which a case is described are aimed at two purposes: the retrieval of the case (or cases) best approaching the performance the designer wants to obtain; the possibility of modifying the description of the retrieved case in order to reach the goal. The first purpose implies a previous organisation of the domain of the cases in order to give it a structure that minimises the research. For every architectural type the most abstract description is the accessibility graph. At the second level of abstraction in the field chosen for developing ASA, the row houses, an empirical research has shown that the elements of the plan which has the most influence on the final disposition are: number of floors; type of staircase (one flight, two flights, winding); position and orientation of the staircase; position of kitchen and bath: shape of the living room. Such information is then supplied for every case. For the second purpose, the widest possibility of modifying the case retrieved, the entire scheme is to be represented through the composition of single rooms. These, in turn, will be represented in their case base in a parametric way with dimensional constraints. A set of constraints manages the consistency of the entire case both in its initial shape and in the course of the modifying process. Less problems are met in the technological data base. A good organisation of it can be achieved using the SfB classification. Less difficulties are also met in the second phase of the process: the passage from the wire-frame representation to the execution drawings as this phase substantially attributes a technical specification to the abstract elements already defined. The technical case base contains also the knowledge able to solve the connections among elements in the various predictable situations.

THE ARCHITECTURE OF THE ASA SYSTEM

The system accomplishes the functionalities of an Assistant to the architectonic design through the interaction between various modules, each able to fulfil specific functions (see Figure 1). The modules can be grouped according to four macro functionalities:

- Interface.
- Parametric CAD: Drafting Board, Parametric Geometry System, Parametric Architectural Objects Engine.
- Symbolic Reasoning: Case Engine, Abstraction Engine.
- Quantity Reasoning: Quantity Reasoning System.

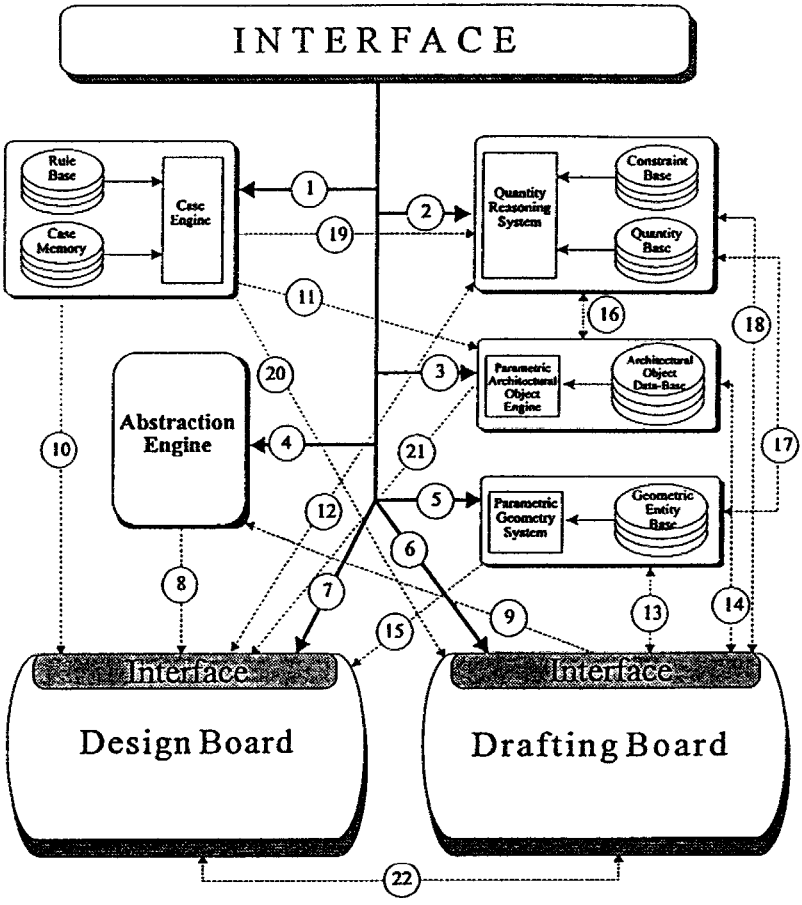

Figure 1. The Architecture of the System ASA

Drafting Board

The Drafting Board module is a data base containing the instances of the geometrical entities and their groupings generated during the design process. It facilitates the visualisation of the represented design objects. The module supports the activities and the interactive functions typical of a two-dimensional CAD, as well as the logical connections with the other representations of the same design object in other modules, for instance, the Design Board.

Design Board

The Design Board is a data base containing the symbolic representation of the actual state of design process. The data base is conceptually divided into parts containing instances of: architectonic objects, geometrical shapes, scenes, relationships between objects, parameters and constraints. Each object in the Design Board is described by a set of numerical and alphanumerical attributes and a set of methods that capture its functionality. Objects are always instances of classes pre-defined in other data bases of ASA. An object in the Design Board that can be represented graphically is logically connected with the corresponding object in the drafting Board. The set of objects and values of the attributes describes the state of the design process. The Design Board manages the temporal sequence of the design phases (considered as a sequence of states) according to the following scheme:

- a new state is created every time a new design action is made; the state is put in front of a stack of states;
- if the user requires it, the effects of the last action are undone and the last state is restored.

Case Memory

This memory is a data base with a structured organisation of stereotypical solutions of design situations. The basic element of the Case Memory is the scene. A scene represents a design event through the representation of its effects. Its nature is episodic. If it is to be possible to modify a scene in order to adapt it to a new situation it is necessary to abstract some general outlines from the event, in order to be able to manipulate and to define the non essential lines of the scene. For some design themes it seems possible to define a language, based on objects and primitive relationships, with which to describe the scene, allowing learning processes. In a scene the following items are described:

- the applicability of the scene;
- the abstract description of the represented context;
- a real instance of the represented context.

The implementation of a data base for the design of row houses is

described. This conceptual structure may be formalised in terms of frame (in Kappa language by Intellicorp) as in the scheme referring to the class "ROW_HOUSE" in Figure 2. The slots content is specified here after:

- the slot "NAME" contains the mark of the scene;
- the slot "PRECONDITIONS" contains a list specifying the limitations for the applicability of the scene. The arguments of this list refer to instances of the classes "POSITION_MATRIX" and "TYPE_OF STAIRCASE";
- the slot "SPECIFIC" describes the characteristics distinguishing the scene from other ones equally apt to be applied to the design problem. Preconditions distinguish "design variants" while the specifics characterise the possible variations of a variant. The arguments of the list contained in the slot are in the classes "ACCESSIBILITY_GRAPH" and "POLYGONS";
- the slot "PLOT" has the structure of a list containing the instances constituting the abstract description to which the scene is applied. The monitor "IF NEEDED" contains the procedural structure in terms of "CALL" to the procedure of instantiation of the design solutions characterising the scene;
- the method "EVALUATE" contains the call to the procedure evaluating the preconditions. Its list of parameters accepts only instances of the objects in the classes pertaining to preconditions. These instances are abstracted interactively from the design scheme. The method contains only the "CALL" to the procedure of evaluating the consistency between the given list of objects and the list in the slot "PRECONDITIONS". The result is of Boolean type (True, False);
- the method "CHOICE" contains the "Call" to the procedure of evaluation of the consistency of the specifics with the context to which the scene is applied. It accepts as arguments objects of the same classes as in the slot "SPECIFIC". The procedure shall evaluate, in a not binary way, an index of applicability of the scene to the design scheme.

The class of objects "POSITION_MATRIX" contains in the slot "ELEMENTS" a list of instances of the class "FUNCTIONAL_SPACES". The objects in the slot "POSITION" are instances of the class "PLAN_POSITIONS". In the Case Memory of the Row Houses the Position Matrix contains only the positions of kitchen (K), bath (B) and staircase (S) if it exists.

The class "ACCESSIBILITY_GRAPH" contains in the slot "ELEMENTS" instances of the class "FUNCTIONAL_SPACES". The objects contained in the slot "ACCESSIBILITY" are instances of the class "ACCESSIBILITY_RELATIONSHIP". The class "FUNCTIONAL_SPACES" constitutes a hierarchy of scenes.

The slot "FUNCTION" of the class "FUNCTIONAL_SPACES" contains one of the symbols marking the functions typical of the spaces of the

dwelling (see Figure 2). The slot "SHAPE" contains an instance of the repertory of polygonal shapes. The method "FURNISH" contains the "CALL" to the procedure that instantiates the furniture of the dwelling.

The two slots of the class "ACCESSIBILITY_RELATIONSHIP" contain instances of the class "FUNCTIONAL_SPACES". The method "SOLVE_ACCESSIBILITY" contains the procedure of instantiation of the access. The method is based on the evaluation of the amplitude of the adjacency and on the values of the slot "SPACES". The slot "POSITION" in the class "PLAN_POSITIONS" contains a value from 1 to 9 whose reference is bound to the scheme of Figure 2. The matrix schematises the positions on the plan of the dwelling. Positions (1,4,7) are situated on the front, positions (2,5,8) in the middle, positions (3,6,9) in the rear of the dwelling. Positions (1,2,3) and (7,8,9) are situated on the blind sides of the dwelling.

The slot "TYPE" of the class "TYPE_OF_STAIRCASE" contains one of symbols of Figure 2 representing an instance of the repertory of types of staircases shown in the figure.

A first Case Memory of row houses has been implemented through a collection of cards in the class "ROW_HOUSE". A sample card, the SP25 of the collection is analysed in order to explain the process of implementation of the repertory. Each card is marked with a name written in bold letters in the square in the top left corner. Near the square there is some information referring to the position in the plan of kitchen, bath and staircase. Of this last also the type is shown. In the card are reported: an accessibility graph of the scheme, a wire-frame plan and a real instance of the represented dwelling. The implementation of a subsequent section of the Case Memory, provided in the next release of the system shall allow the recursive application of scenes to the wire-frame scheme, thus obtaining a progressive process of refinement of the description leading to a representation level comparable with the drawings of the instances.

Case Engine

The functions of the case engine, aimed both at assimilating new situations and of adapting known situations to new ones, are the following:

1) Indexing: some relevant characteristics of a situation in input are used to build a set of indexes and store the situation in the case memory after a consistency check.

2) Retrieving situations similar to the desired one through the use of indexes. In a design session the case retrieved becomes the initial solution.

3) Modification: the initial solution is adapted to the new context, generating the new solution.

4) Test: the proposed solution undergoes a consistency test.

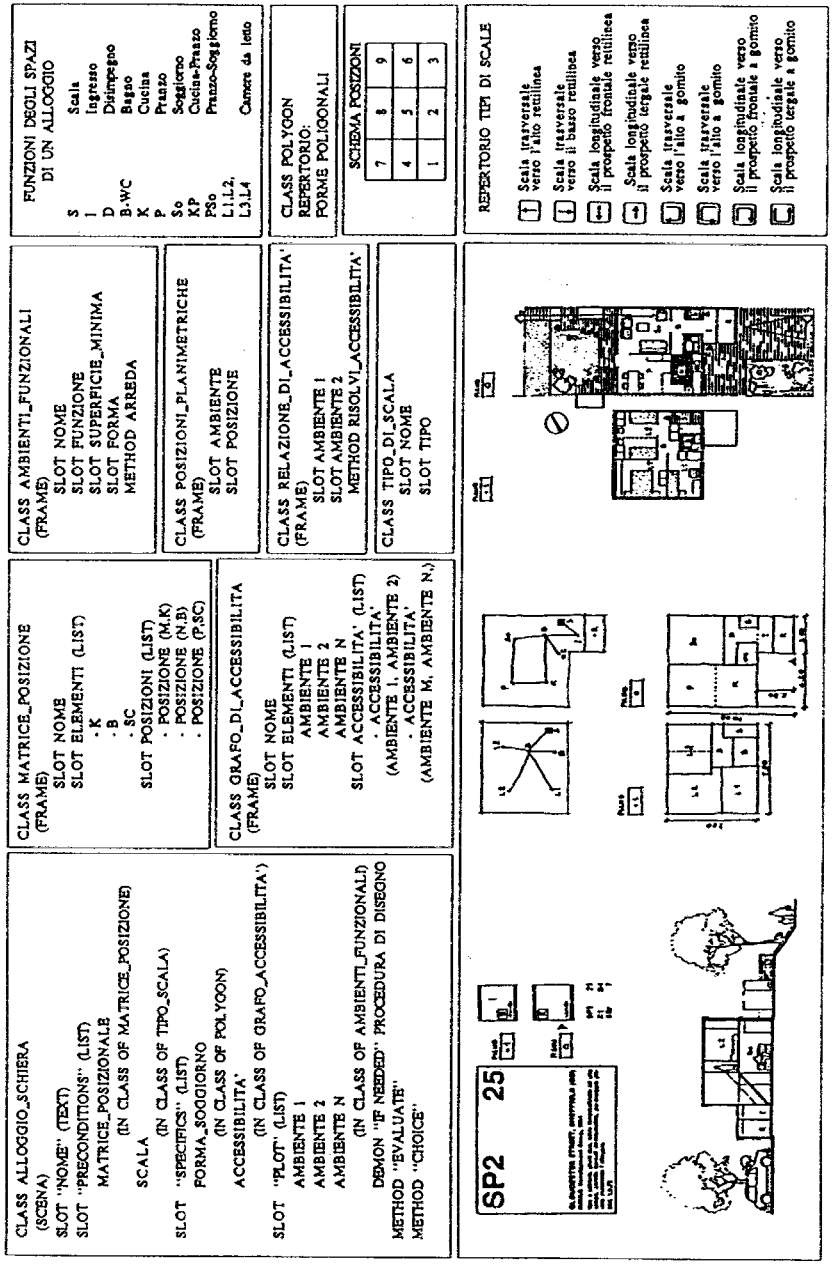

Figure 2. The Card of a Row House in the Case Memory: An Example

5) Explication, Correction and Test: if the solution is inconsistent an explication of the failure is sought, identifying the cause of the inconsistency, the solution is adjusted and undergoes a new consistency test.

Those functions are supported by the following data structures:

- Indexing rules: they identify the characteristics that can be used in a predictive way in order to realise the fit indexes in the case memory.
- Case memory: it represents the episodic memory, constituting the data base of the experience.
- Similarity metrics: it is the base for a selection function when the case memory proposes several prior solutions.
- Modification rules: they adapt the prior solution when, as it is the typical case, the prior solution only approximates the actual problem solution.
- Repair rules: they encode the rules that allow modifications when a solution is recognised as inconsistent.

Quantity Reasoning System

In ASA the objects composing a scene can be defined also in a parametric way: a typical interval of some attributes is given, while the exact value is specified in the instantiation. The module QRS supports the management of the parametric attributes and their relationships. Two types of parameters are possible: numerical and symbolical. The numerical parameters can be considered as variables assuming values from a domain which, in turn, may consist of one or more intervals. Thus an algebra of intervals is also necessary. A relationship among n numerical parameters consists of n functions defining each parameter in function of the others.

The domain of a symbolic parameter is constituted by a set of symbols. A relationship among parameters is defined extentionally through a set of n-tuples representing all the combinations satisfying it, or intentionally through a code segment which verifies if a certain n-tuple satisfies it. For both types the management of the relationship settles the value of each parameter in function of the present values of the others.

Parametric Geometry System

The Parametric Geometry System manages parametric geometry in which the shapes are defined only implicitly trough a set of points and a set of constraints on their positions.

Parametric Representation of Shapes

The geometry of an object is defined through a set of n characteristic points (both on the border and internal to the body) and a set of equations among them. The border of the shape is defined through the m generative equations between the characteristic points.

The PGS only translates the geometric relationships into a system of inequations, taking into consideration positional constraints, whose management is left to the QRS.

Parametric Architectural Object Engine

The Parametric Architectural Object Engine (PAOE) implements the functions of drawing and manipulating the building technical elements that can be classified according PC¦SfB system.

Technical elements are described through four types of information:

- classing: necessary for a complete and unique classing of the object as an instance of a class of technical elements: class of membership, number of the instance;
- graphical: position in the space of general reference and the primitive graphical entities necessary to the representation of the element;
- numerical: the instances of the parameters that allow the representations;
- alphanumeric: referring the object to particular classes or subclasses as "working", "materials" and so on.

The technical elements can be represented in three ways:

- executive: the complete drawing of the element. It is possible only if all the information necessary to a full instantiation has been given;
- symbolic: the system represents it either if requested or if it has not been given all the necessary information. This representation shall be employed when one has the intention of putting a technical element in a particular position without fully defining it. In this case the system will use the default values of not instatiated parameters;
- 3D: if requested; again default values shall be used in lack of the necessary information.

Every technical element is defined through a set of geometrical-dimensional parameters, alphanumeric attributes and a set of functions referring to the operations possible on it. The parameters are of two types: external, that relate the elements to the entire drawing (typically the co-ordinates of the insertion point or some dimensional parameters depending on the class of the element) and internal. The instantiation of an element implies the instantiation of all the constrains connected with it.

Abstraction Engine

The module of the Abstraction Engine operates a geometrical interpretation of not structured graphical data in input. From a set of lines on the Drafting Board not connected with any object on the design board, the interpretation process tries to define possible organisations of spaces implied by such lines, the possible shapes of those spaces according to a data base of pre-defined shapes and the relationships between the defined shapes (see

Figure 3). The process then creates on the Design Board the instances of the convenient classes "Geometrical Shapes" and connects them logically to the lines in the Drafting Board. This first interpretation can be followed, through the access to the Case Base, by a further interpretation that will produce, in the actual design context, hypotheses of signification of the datum, not structured in input.

The Data Bases

Figure 1 shows all the data bases of the system. They are of two types: permanent data bases and working data bases (Boards).

Permanent data bases are not modified during the work sessions except in particular phases such as learning. They represent the knowledge of the system in the specific application domain. Permanent data bases are: Architectural Object, Geometric Entity, Constraint, Quantity, Rule Bases and Case Memory.

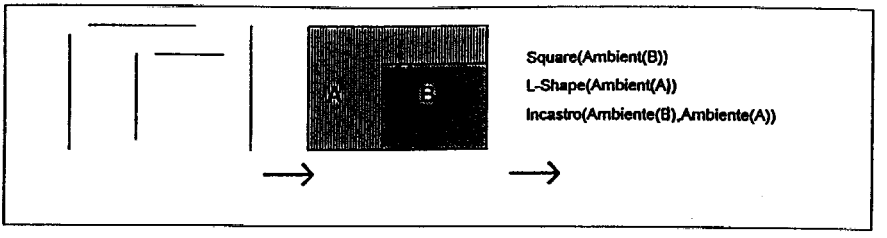

Figure 3. The Interpretation Process

Working data bases contain the description of the state of the design process: they are empty at the beginning of the work session and are continuously updated. Of great importance is the function of "undoing", the capability of bringing back the system to the state preceding the last operation.

Flows of Data and Control

In this section we shall take into consideration the global behaviour of the system, examining the flows of data between the modules and the control of the system. ASA gives the user the possibility of direct access both to the functionalities of a single module and to an organised sequence of them. The possible basic functions are:

- Sketching: the user can trace sketches directly on the Drafting Board without any correspondence with the entities and shapes contained in the data base of ASA. This function implies all the possibilities supplied by the 2-D vectorial drawing packages: drawing of the main geometrical entities, editing

and so on.

- Parametric Drafting: this functionality is generated by the juxtaposition of the modules QRS, PAOE, PGS and Drafting Board and consists in the possibility of executing parametric drawings according to the last tendencies of the variational CAD.
- Case Generation and Case Reasoning: the activity of Case Reasoning involves a great part of the system. The Case Engine manages the historical memory and presents, according with the input keys, the possible design solutions (typically scenes). In order to fit the proposed scenes to the dimensions of the design problem they have to be represented in parametric form. This role is fulfilled by the Parametric Drafting. The possibility of fitting the scene to the design context is entrusted to the union of the constraints implied in the scene with those already existing in the design context. The triggering of routines searching consistent solutions produces the necessary fitting of the scene to the context. The start proposal can be interactively modified by the user or employed as a further key to a subsequent process of search to a higher detailing level. For instance, if the first solution was proposed at wire frame level, the refinement process produces a solution with details at a geometrical level.
- Constraint Posting and Consistency Checking: these functionalities are related directly to the QRS module and allow the user to pose new constraints to the design parameters, in addition to the ones posed each time by the system.
- Abstraction: the functionality of abstraction, when it is activated, proposes to the user possible interpretations of the sketches present on the Drafting Board. The abstraction process is well defined as it provides only one flow of information from the Drafting Board to the Design Board and a local control limited to the interior of the Abstract Engine.

In the following the main informative flows present in the system is described. By informative flow we refer to the interaction between two modules that can be generated both by the transit of information and by the access to specific functionalities (see Figure 1).

Twenty-two main flows can be identified:

- Flows (1-2-3-4-5-6-7): they represent the interaction of the user with all the modules of the system. The temporal sequence of the interactions accomplishes the global control of the elaboration. Through these channels the user can reach the interface functions of every module allowing him to have access to the data and/or to activate the functionalities.
- Flows (8-9): they represent the communications between Drafting Board, Abstraction Engine and Design Board. Flow (8) implies the abstraction procedure of lines sketched in the Drafting Board while flow (9) represents the instantiation of the interpreted datum on the Design Board, which in turn

is connected dynamically to the corresponding lines on the Drafting Board.

- Flows (10-11-19-20): they pertain to the functionalities of generation and definition of a case. After the activation of a case search through a key, the Case Engine generates on the Design Board all the instances of the object present in the particular case retrieved and on the Drafting Board their geometrical representation linking the two. At the same time the QRS generates all the parameters and their reciprocal constraints.
- Flows (12-18): flow 12 represents the bi-directional interaction between QRS and Design Board. The QRS finds in the Design Board the instances of parameters and constraints related to a certain design moment. Flow 19 allows the updating of the parameters of the design unit when the set of constraints is made consistent (ie when the procedure of the consistency checking is triggered or a possible instance of the design geometry is searched). Flows are bi-directional as the user can vary directly both parameters and constraints.
- Flows (13-14-15-16-17-18-21): this considerable set of flows is generated by the typical activities of the functionalities of parametric drafting present in ASA: Flows (13-14-15-21) are generated by the activity of instantiation of forms and parametric objects present in the Drafting and in Design Board. The parametric nature of the objects requires the contemporary generation of the parameters and the relative constraints nets.
- Flow (22): represents the dynamic link between the entities present in the Drafting Board and the related class instances in the Design Board. The dynamic link consists of the possibility of an asynchronous updating of the attributes in the Design Board depending on variations in the corresponding entities in the Drafting Board and vice-versa.

AN EXAMPLE

An example of a work session exploiting ASA's capabilities as an architectural assistant follows. The exposed sequence is only one of the possible way of exploiting ASA's capabilities.

- The designer sketches a scheme of both floors of a dwelling in the Drafting Board, using the interface and the drawing tools of the Board.
- The Abstraction Engine interprets the sketch translating it to a configuration of polygons.
- Conceptual images of the polygons are created in the Design Board. A wire-frame plan of the floors is returned on the Drafting Board.
- The designer assigns a value in the class "FUNCTIONAL_SPACES" to the polygons of the scheme on the Drafting Board together with the accessibility requirements.
- The Abstraction Engine derives from the wire frame sketch, as already interpreted by the Design Board and from its semantics, preconditions and specifics corresponding to the case on the Boards; the QRS constructs the

constraint set that will be handled by the CMS.

- The Case Engine searches in the Case Memory the most approaching cases, according to the similarity metrics, the requirements of the case; the designer chooses the one on which he will work.
- Changes in the retrieved case are operated in the Drafting Board through the drawing tools allowed by the interface; QRS and PGS control and manage the consistency of the proposed scheme.
- The further specification of the scheme from the wire frame state to the executive level is carried on with the help of the PAOE.

Now we will describe a practical example of a session. On the Drafting Board a sketch is drawn. The Abstraction Engine gives back a plan consisting of rectangles and the adjacency graph (right of Figure 4). A semantic is given to the polygons of the wire-frame whose representation is both in the Drafting and in the Design Board. The accessibility conditions are input either directly on the adjacency graph (bold lines in Figure 4 right) or on the wire-frame plan (Figure 4 left).

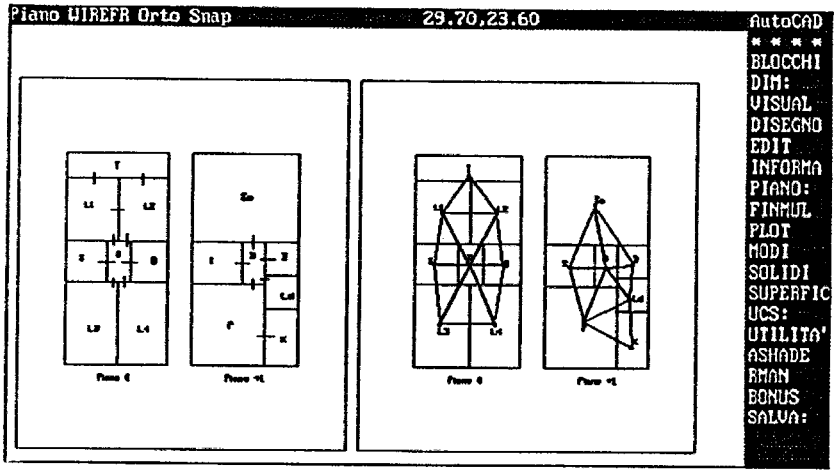

Figure 4. To the plans already encoded as set of polygons and given a semantics as functional spaces the accesibility conditions are added; the accessibility graph is generated as a subgraph of the adiacency graph

The Case Engine searches the cases satisfying the preconditions and the constraints that the abstraction Engine has in the meantime derived. The selected cases in the Case Memory of row houses already implemented are 18.

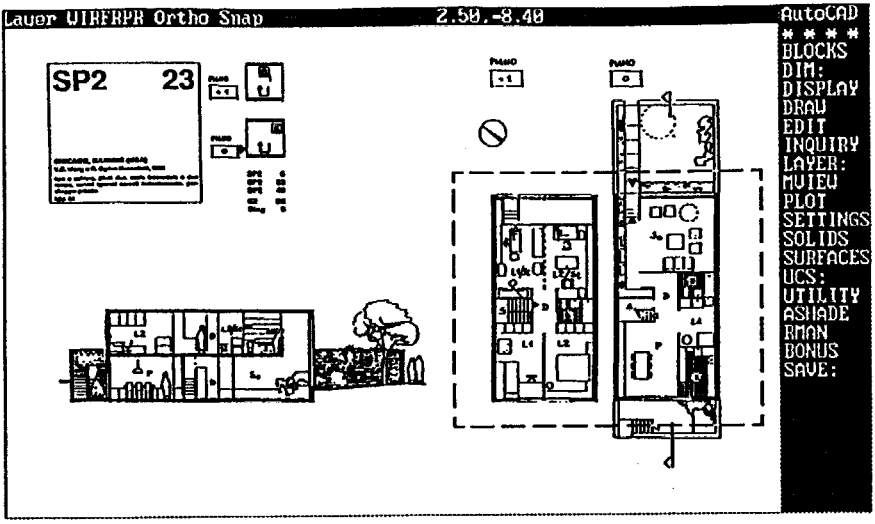

Figure 5. The Case Retrieved by the Case Engine

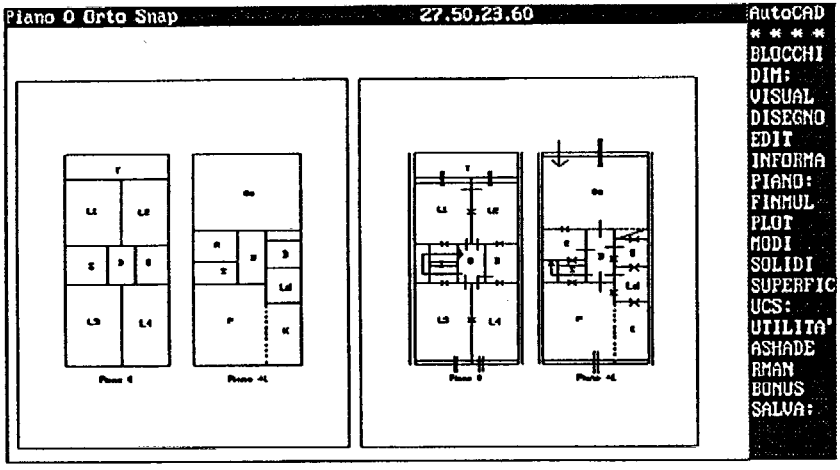

Figure 6. The Wire-frame Representation of the Case Returned as Modified by the Designer

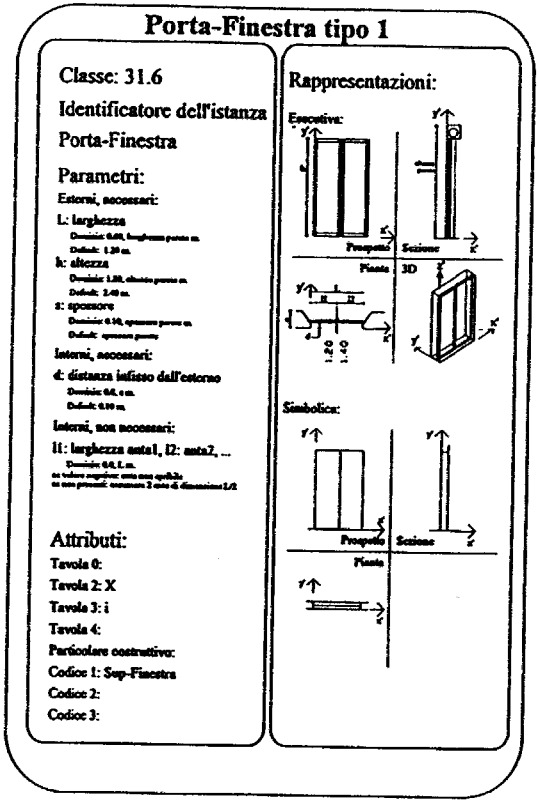

Figure 7. The Card of a Technical Element

Imposing the specifics leads to a further selection of six cases which, when presented to the Drafting Board lead the designer to choose the case n. 23 (see Figure 5) for further elaboration. Interactively from the Drafting Board the scheme is modified as in Figure 8 (left), always under the control of QRS. Eventually on the Drafting Board a technological semantic is given to the lines (Figures 6 right and 7) in order to allow the PAOE, with the help of PGS, to instantiate the architectural objects necessary to give the designed plan an executive representation (Figure 8).

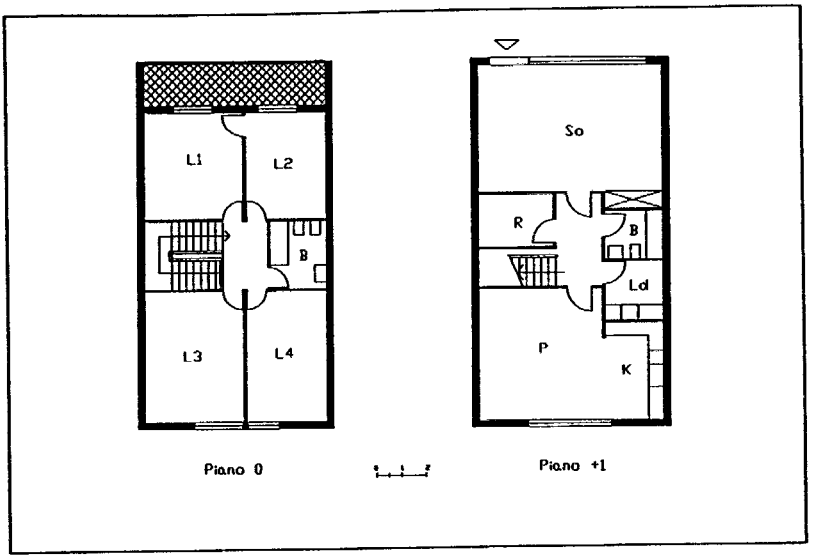

Figure 8. The Final Output: The Designed Plans are Fully Instantiated

CONCLUSIONS

ASA is a tool without pretensions of creativity but tries, through the connection of existing functionalities to extract from the existing functionalities to partially automate the work of a conscientious assistant. Its main feature is the continuous interaction between Drafting and Design Board, that is the immediate conceptualisation of every object drawn on the Drafting Board, drawing on the functionalities of the various modules. Starting from its actual capabilities, much work can be done in order to exalt some virtual possibilities, for instance, to directly entrust it with some classes of updating operations via a fit extension of the similarity metrics, and an enhancement of the capabilities and the collaboration of QRS, PAOE and PGS.

References

Colajanni, B, M De Grassi, M Di Manzo and B Naticchia (1991), *Can Planning Be a Research Paradigm in Architectural Design*, Proceedings of First International Conference on Artificial Intelligence in Design, Edinburgh.

Davis, E (1987), *Constraint Propagation with Interval Labels*, Artificial Intelligence 32, pp. 281-331.

Giretti, A, and B Naticchia (1992), *ASPIDE: A Constraint Oriented Approach to Geometrical Modelling in Architectural Design*, C.I.B. World Congress, Montreal.

Gossard, D C, R P Zuffante and H Sakurai (1988), *Representation*

Dimensions, Tolerances and Features in MACE System, IEEE Computer Graphics & Applications, March, pp. 51-59.

Hayes, P J (1979), The Logic of Frames, in *Frame Competitions and Text Understanding* (D. Metzing ed.), Walter de Gruyter and Co., Berlin, pp. 46-61.

Kolonder, J L, and C K Riesbeck (1986), *Experience, Memory and Reasoning*, Lawrence Eribaum Asoociates, Hillsdale, New Jersey.

Kolonder, J L (1992), *An Introduction to Case Based Reasoning*, Artificial Intelligence Review n.6, pp. 3-34.

Kurnar, V (1992), *Algorithms for Constraint Satisfaction Problems: A Survey*, AI Magazine, Spring, pp. 32-44.

Roller, D (1991), *An Approach to Computer Aided Parametric Design*, Computer Aided Design, vol.23, n.5, June.

Slade, S (1991), *Case-Based reasoning: A Research Paradigm*, AI Magazine, Spring, pp. 42-55.

c. 1993, Management of Information Technology for Construction, K. Mathur et al (Eds), World Scientific Publishing Co., Singapore.