Design and Implementation of a Natural Logic System in Conjunction with a Relational Object Oriented Database as Applied to the Architectural Engineer

Marty Diamond, and Arthur L. E. Schumacher

Phoenix Advanced Software Systems
201 Park Place, Suite 105
Altamonte Springs, FL 32701, USA

KEYWORDS

Natural Logic, Relational Object Oriented Database, Computer Aided Design and Drafting (CADD).

ABSTRACT

Two critical areas needing major improvements remain in present CADD applications. The solutions to these problems will make CADD an extension of the human creative process instead of a restriction.

The first area of concern is the user interface. New systems will adapt to the user's procedures and work flow. Based on the natural logic concept, these systems will require a trivial length of time to learn and will be able to adapt to the reality of an ever changing work flow.

The second area of concern is database structure. Present applications use an object representation for graphic manipulation to the detriment of associated informational data. The new generation of systems will be based on a relational object oriented data structure. This structure allows for optimal performance whether the data is used for graphical representation or associated data processing information.

The architectural engineer looks to CADD to enhance his/her abilities and productivity. The future of the CADD industry is dependent on the adoption of these radical new principles to meet these expectations.

DESSIN ET MISE EN OEUVRE D'UN SYSTÈME DE LOGIQUE ORIENTÉ CONJOINTEMENTE AVEC LE FONDEMENT DE DONNÉS ORIENTE AUX OBJETS RAPPORTÉS A L'INGÉNIEUR ARCHITECTONIQUE

Dr. Martin Diamond and Arthur L. E. Schumacher

Phoenix Advanced Software Systems
201 Park Place, Suite 105
Altamonte Springs, FL 32701
U.S.A.

MOTS-CLES

Logique au Hassard, Fondement de Donnés orienté aux Objets/Rapportés aidé par la machine a Calculer pour se dessin (CADD)

SOMMAIRE

Il n'y reste que deux sujets délicats dans les applications du CADD en besoin de majeurs ameliorations. Les solutions pour ces problèmes occassioneront une extension du procés createur humain au lieu de restriction.

Le premier sujet à regarder est l'interfase d'usaguer. Les nouveaux systèmes s'adapteront aux procés d'usageur et au flux de travail. Basé sur le concept logique au hassard, les systèmes requeriront une durée de temps pour l'apprentissage et adaptation la realité d'un flux de travail qui change constantament.

Le seconde sujet à amelliorer est la structure du fondement de Donnes. Les applications actuelles utilissent les representations des objets pour manipulations graphiques pour le detriment des Données d'information associé. La neuve generation de systèmes sera basee sur une structure de Données rapportées aux objets rapportes. Cette structure permet que les Donnés soit utilisé pour la representation de les Données associé pour la meilleur accomplissement.

L'ingènieur Architectonique regarde CADD pour relever ses habilités et productivité. Le future de l'industrie de CADD depends sure les adoptions de ceux principes radicaux à la recherche de ceux expectations ci.

Introduction

This paper describes some advances in computer applications that will have a significant impact on Computer Aided Design (CAD) systems. They will have a particular impact on architectural design and construction packages. The advances originated with users and designers of previous CAD systems and were prompted by a desire to make CAD a tool that matches the needs and procedures of users. The resulting new generation of CAD systems is intended to be Computer Aided Design as opposed to Computer Aided Drawing. The new developments are the following:

1. To make systems object or concept oriented rather than line oriented.

2. To allow users to move freely through various functions in a sequence that is determined by the user according to his or her design procedure rather than by a system imposed structure.

3. To store data in a concept-oriented hierarchy that matches the designer's conceptualization, thus allowing objects to be referenced in an order that is consistent with the design.

4. To provide report definition procedures that allow users to specify reports in such a way that they visually represent the report logic. The specifications not only provide tools that make good logic structure, but actually enforce good logic structure and readability.

Concept Oriented CAD

Historically, CAD was a tool to replace the draftboard. Its functions were intended to duplicate those of a draftsperson rather than those of a designer. The basic drawing functions will, of course, have to be part of any usable CAD system, but new functions are being added to satisfy the needs of designers. For instance, it is now possible to draw a simple figure and identify the lines as being part of a particular construct or of a particular generic type. Then, at any later time, the generic type can be changed to a specific type and the CAD system will automatically replace the lines with the representation required by their specified types. Thus, for example, lines could be drawn as a generic wall and later changed to a specific wall type. The single lines would then be automatically replaced by the appropriate multiple-line representation of the specified type. The set of multiple lines would still be known to the system as a wall and could again be changed to yet another wall type with yet another graphic representation.

The fact that lines can be entered as part of larger constructs allows an advanced CAD system to identify items of a given type within a given area. For instance, it is possible to identify all doors in a room, a wing, a floor, or a building. This identification could be for highlighting on the screen or for use in a report. Being able to identify all items in a room for a report allows analysis of such things as consistent fire ratings in an area.

The most significant point here is that the items maintain a conceptual relationship determined by the designer. The designer is thus able to work with walls and rooms instead of lines.

Natural Logic

The next advance presented in this paper is in system logic flow. Whereas previous systems have imposed a rigid structure on the user, advanced systems allow the user to move through the functions as desired. An example will illustrate the differences. Suppose you wanted to put a circular island in the middle of a kitchen. You would have to specify a center and a radius. In traditional CAD systems, you would have to:

1. Determine where the center is to be and what the radius should be.

2. Start circle construction and specify this center and radius. (It would, of course, be possible to specify an arbitrary circle and then edit it into place.) In an advanced system, you could decide that you want a circle and enter the circle construction module. You would then have to specify a center and radius just as before. But now you could start some entirely new function to determine the center without having to exit circle construction. When the center is determined, you would exit the secondary function and return to circle construction. You might, then, start another secondary function to determine the radius. The result is the same, but the procedure is determined by the designer according to his or her thoughts instead of by some predetermined order imposed by the CAD system. That is, the logic followed is natural logic rather than computer-imposed logic.

The program structure that is employed to do this is a well-known nesting procedure in which pointers are stored to keep track of the return points, the items already defined and those yet to be defined. Indeed, some computer games where the players work their way through a cavern picking up, putting down and consuming objects encountered do this. Syntax parsers also do it to some degree. But what is revolutionary is the use of these procedures in real problem interactive design systems. Thus, although there is no algorithmic breakthrough, there is really a user interface breakthrough.

Data Base Structure

We now consider some advances in CAD data base structure. A combination of relational data base functions and internal network structure can provide a lot of power in a CAD attribute data base. The structure is created as the designer inputs data and specifies the relationships between different items in a drawing or between an item and some properties. The data base is thus structured in a near optimal way since relations that will be required by reports or interactive requests will already be recorded in individual records. The relations can be made in any order and between any items with the only restriction being that they make sense to the designer.

Relations input by the user forms links in a network structure. There can be many links to any item and many links from any item. Furthermore, there can be link paths of different lengths between two particular items. There could, for instance, be a direct link between two items and also a link from the first item through some intermediate items and finally to the second item.

Report Writer Facilities

The data base system report writer could automatically structure reports using the network links to determine a hierarchy. Thus, breakdowns for bills of materials according to any specified selection would be automatically produced. The exact contents of the printed output would be specified in a report definition supplied by the user or as a standard feature of the CAD system. The searching, sorting, selecting, and structuring would, however, be an automatic function that required no programming by the user. Alternately, a where-used report could be generated automatically with the exact output format specified by a report definition.

Report Definitions

All reports are produced by collecting information, sorting it, structuring it in some way, doing calculations according to some criteria, and, finally, outputting results. As mentioned in the previous section, advanced CAD systems can automatically structure reports. It is, of course, possible

to specify some other non-standard structuring in a report definition.

The procedures for collecting data and later doing calculations according to some criteria requires first a search of the data and then a selection procedure. The report writer can automatically search a portion of the network if a starting point is specified and the type of search or report structure is chosen from a list of options.

Although specifying the selection procedures is a little more complex than simply choosing something from a list, it is done in a way that is a significant advance in the computer user interface. Historically, selection specification has been done through a series of computer program IF tests. Even sophisticated query languages and report writers have used these tests. The original IF developed into IF THEN or IF THEN ELSE statements with indentation to improve readability. But the basic programming language syntax remained. Furthermore, the individual conditions were made of Boolean combinations specified by the operators AND, OR, NOT, and possibly XOR. Unfortunately, a condition containing only a few AND's and OR's in a single IF can be confounding even to experienced programmers, causing them to read such statements two or three times to be sure they have comprehended the condition. The situation is aggravated if the condition specified is not correct or contains NOT's. Those who are not programmers have an even more difficult time specifying, reading, or debugging such statements.

The following procedure for specifying conditions has been used very successfully in a special purpose report writer that has not been marketed and is, thus, virtually unknown. Conditions are specified in a tabular form that graphically displays the tests and makes them easy to specify, read, and debug. There are three principal features. The first is that conditions are prefixed by dots that determine the AND OR structure. The second is that conditions are defined and named in one step and grouped together in an implied IF, THEN, ELSE-IF, ELSE structure in another step. The third is that both the condition specification and the grouping together are done on form-fill-in displays that limit the number of things that can be specified at one time. Since it is possible to include conditions within the definition of other conditions, the limit does not restrict the conditions that can be specified. It merely enforces a structure on the specification. This is, ultimately, a benefit in that it makes the specifications more understandable and imposes a top down structure on the user. This is something like a computer program architecture's "less is more."

The fact that the system imposes a structure is also a benefit. There have been many higher-level languages that have provided tools for developing top down programs. But, in reality, it has always remained the obligation of the programmer to use the tools and create the top down structure. The languages themselves could not guarantee such structure. The procedures described above, together with other report definition forms that limit what can be specified at one time, guarantee a top down structure. Actually, they do not force the user to create definitions by using a top down procedure although it is the most natural thing to do when using the forms. But it is guaranteed that the end result will have a canonical top down structure that would be seen when anyone read the report specification.

The Boolean logic implied by prefixing dots is specified as follows:

1. A test item is specified.

2. Conditions on the item are specified in blocks.

3. A block can be broken into sub-conditions that are themselves blocks.

4. Each level of block nesting is indicated by one dot. Thus, a block that is n levels down would be prefixed by n dots.

5. All blocks at level n+1 within a block at level n are OR'ed together.

6. The OR'ed combination of blocks at level n+1 is AND'ed with the containing block at level n.

7. The following three examples that specify a letter is a consonant and not a vowel illustrate the syntax of the conditions:

Example 1

```
(`B'-`H'), (`J'-`N')

.NE`E'
(`P'-`T'), (`V'-`Z')
.NE`Y'
```

Example 2

```
(`B'-`N')
.NE`E'
..NE`I'
(`P'-`Z')
.NE`U'
..NE`Y'
```

Example 3

```
NE`E'
.(`B'-`H')
.(`J'-`T')
..NE`O'
.(`V'-`Z')
..NE`U'
..NE`Y'
```