The Design of Multimedia Languages Based on Teleaction Objects

G. Polese[1], S. K. Chang[2], and G. Tortora[1]

[1] Dipartimento di Matematica ed Informatica, Universita' di Salerno,
Via S. Allende, 84081 Baronissi (SA), Italy,
email: {giupol}@dia.unisa.it
[2] Department of Computer Science, University of Pittsburgh,
214 Mineral Industries Building University of Pittsburgh, Pittsburgh, PA 15260,
USA, email: {chang}@cs.pitt.edu

Abstract. We describe a methodology for the design of multidimensional languages to be used for multimedia applications. The methodology supports the generation of multimedia languages and applications starting from the generic description of a problem. The design framework extends traditional methodologies for visual language design and relies on Teleaction Objects (TAO) as the basic model for specifying and controlling multimedia presentations.

Keywords:.

1 Introduction

The increased employement of multimedia computing raises several problems and issues. The inherent complexity and size of many multimedia applications requires the introduction of proper software engineering techniques, languages, and tools for mastering the specification, the development process, and the dynamics characterizing their presentation. A contribution in this direction is being given from the visual language community. Visual languages have been extended in order to capture the dynamic behavior of multimedia objects [10]. The extended visual languages are called *multidimensional languages* and are still based on the concept of generalized icons, but the physical appearance of icons can be not only images, but also sound (earcon), motion (micon), text (ticon), and video (vicon). The user can access and animate multimedia information by composing multidimensional sentences, that is, by combining icons, earcons, micons, and vicons, according to some spatial and/or temporal relations. Other extensions of visual languages aimed to enable the development of *teleaction objects* (TAOs) [6, 16], multimedia objects that automatically respond to events in order to perform tasks, and are particularly suitable for developing and controlling

* Correspondence to: G. Polese, Dipartimento di Matematica e Informatica Università di Salerno, 84081 Baronissi (Sa) Italia, tel. +39–89–965391, fax +39–89–965272, email: {giupol}@dia.unisa.it

multimedia presentations. In particular, Multidimensional languages and their underlying syntactic mechanism can be used for specifying the static structure of a TAO and for controlling the multimedia presentation modeled by the TAO [2]. As the multidimensional sentence is parsed, semantic routines are invoked to play the media types as they are processed by the parser.

In this paper we face the design of the multidimensional languages for specific application domains. We have gradually extended the design framework for iconic languages described in [11] to include the design of multidimensional languages. In particular, we have first extended the framework to accomodate the design of visual languages other than iconic [18], and then we have adapted it for the design of temporal visual languages [12]. For visual languages the framework was used to design the set of visual sentences (Icon Dictionary) to be used for visually encoding the concepts of a given application domain. Some constraints could be set on the target visual language, such as the maximum number of icons, and the maximum length allowed for a visual sentence, both affecting the quality of the final visual interface and its usability.

The framework presented in this paper can be used for the design of complex multidimensional languages. As opposed to visual language design, here we need to consider more complex objects from the application domain, and the final goal might not be a precise coding of domain concepts through a multidimensional sentence, but it might require more an approximate coupling. The concepts of the application domain are first modeled using the object oriented paradigm and then clustered based upon a distance or similarity function. The icon design phase [11, 5] has been extended since we have not only to design icons, but also earcons, vicons, etc. Thus as we will see, we need to use appropriate iconic operators [10] to manipulate extended generalized icons. As for the back-end part of the design framework, starting from the multidimensional icon dictionary (MID) we design the visual grammar to implement the language parser. Here we can use positional [13] or SR grammars [19, 20, 21]. The multidimensional sentences can be parsed to generate their TAOs, addressing both the static hypergraph and the dynamic active index. The TAO can in turn be parsed by using a TAO parser [2].

We present an example on the use of the design framework to transform textual lectures into their corresponding multimedia presentations.

The papaer is structured as follows. In Section 2 we describe the TAO model and the extended generalized icons and icon operators. In Section 3 we review the concepts underlying a visual language design methodology. The extended methodology is presented in Section 4, whereas the example is described in Section 5. Finally, conclusions are given in Section 6.

2 The TAO Model for Multimedia

The specification of interactive multimedia presentations requires the use of multimedia languages, that is, languages that let users specify and merge different media types such as text, audio, image, and video. Thus, multimedia languages

have to contain mechanisms for specifying both static and dynamic aspects of multimedia presentations. These characteristics of a multimedia language are effectively supported in the teleaction objects (TAO) model [6, 16]. TAOs are multimedia objects capable of automatically reacting to events and messages. The structure of the multimedia object is represented through an hypergraph G, whereas the event driven dynamic structure is represented through a knowledge structure K called *Active Index* [9, 16].

More formally, G is a graph $G(N, L)$, where N is the set of nodes and L is the set of links. A node can represent a media type or even a TAO itself. In the first case we have a *base node*, whereas in the second case we have a *composite node*. Links represent spatial and temporal relations among graph nodes. In particular, links can be of the following types: *attachment*, *annotation*, *location*, and *synchronization*. These can be further specialized, so for example the synchronization can include the temporal relations *co-start*, *co-end*, *before*, etc [1]. An example of hypergraph structure is given in Figure 3.

The knowledge structure K of a TAO is based upon an *active index* (IX) [16], that is a set of *index cells* (IC) from an *index cell base* (ICB), each defining the reactions of the TAO to events. In particular, an index cell can be seen as a type of finite-state machine, since it accepts input messages, performs some action, and sends output messages to a group of ICs. The messages sent will depend on the state of the IC and on the input messages.

The physical appearance of a TAO is described by a *multidimensional sentence*, which is a spatial/temporal composition of generalized icons [8, 10]. The syntactic structure underlying a multidimensional sentence controls its dynamic multimedia presentation. The multidimensional sentence may be location-sensitive, time-sensitive, or content-sensitive. A multidimensional language is a set of multidimensional sentences.

2.1 Generalized icons and icon operators

In a visual language *generalized icons* are dual objects with a physical appearance represented by an icon image and a meaning representing the interpretation of the icon image. More formally, a generalized icon x is defined as $x = (x_m, x_p)$ where x_m is the meaning and x_p is the physical appearance. As said above, in visual languages the physical appearance x_p is an icon image. In multidimensional languages the concept of generalized icon has been extended to represent all the different type of media [10]. The following types of generalized icons have been defined:

- Icon : (x_m, x_i), where x_i is an image;
- Earcon : (x_m, x_e), where x_e is a sound;
- Micon : (x_m, x_s), where x_s is an image;
- Ticon : (x_m, x_t), where x_t is text (can be regarded as a subtype of icon);
- Vicon : (x_m, x_v), where x_v is a video clip (video icon);
- Micon : (x_m, x_s), where x_s is an image;

In [2] a new type of generalized icon is introduced to represent a multimedia sentence:

Multicon : (x_m, x_c), where x_c is a composite icon or multimedia sentence.

In visual languages generalized icons are connected through icon operators to form visual sentences [8]. Also icon operators are dual objects (op_m, op_i), with a physical part (op_i) and a logical part (op_m). The physical part combines the icon images, whereas the logical part combines their meanings. For visual languages we mainly use spatial operators, such as *ver* (vertical composition), *hor* (horizontal composition), *ovl* (overlay), *con* (connect), and so on. Multidimensional languages include a broader class of icon operators. Multidimensional sentences are constructed by combining generalized icons through *earcon operators* such as *fade_in* or *fade_out*, *micon operators* such as *zoom_in* or *zoom_out*, *ticon operators* such as *text_merge* or *text_collate*, and temporal operators. The latter include *co_start*, *co_end*, *overlap*, *equal*, *before*, *meet*, and *during*. For example, to play a video clip and at the same time have a background audio it can be requested that a vicon *co_start* with an earcon. The combination of generalized icons through temporal operators can yield a composite icon of a different type. For example the temporal composition of a micon with an earcon yields a vicon.

In TAOs Generalized icons are represented by nodes in the hypergraph, whereas operators are represented by links.

As an example, let us consider the multimedia presentation *Salerno Multimediale*, a CD rom on the city of Salerno [15]. The presentation begins by displaying a cover image, inviting the user to touch the screen in order to proceed. After the touch, a background sound is played and an animation starts at the same time, it is made of a background image that has a rotating label *"Salerno Multimediale"* on it. After few seconds the rotating label fades out and a falling curtain starts covering the background image. The animation ends and gives place to another background image with a menu overlayed on it. Some screendumps for this portion of the CD rom are shown in Figures 1 and 2, whereas the associated hypergraph is shown in Figure 3.

3 A Methodology for visual language design

There are several types of visual languages which makes it difficult to define a common theoretical framework to capture their characteristics. In [8] a classification of visual languages is made based upon the visual or textual nature of the language itself and of its application domain. Nevertheless, the different types of visual languages can all be seen as addressing different aspects of generalized icons [8]. Thus, a methodology for visual language design should be centered on the concept of generalized icons. At University of Pittsburgh we have developed a methodology for the design of iconic languages [11], a subclass of visual languages. We have used such methodology for developing complex iconic languages to be used by speech impaired people [3, 4]. Successively, the methodology has been extended to accomplish the design of general visual languages [18] and temporal visual languages [12].

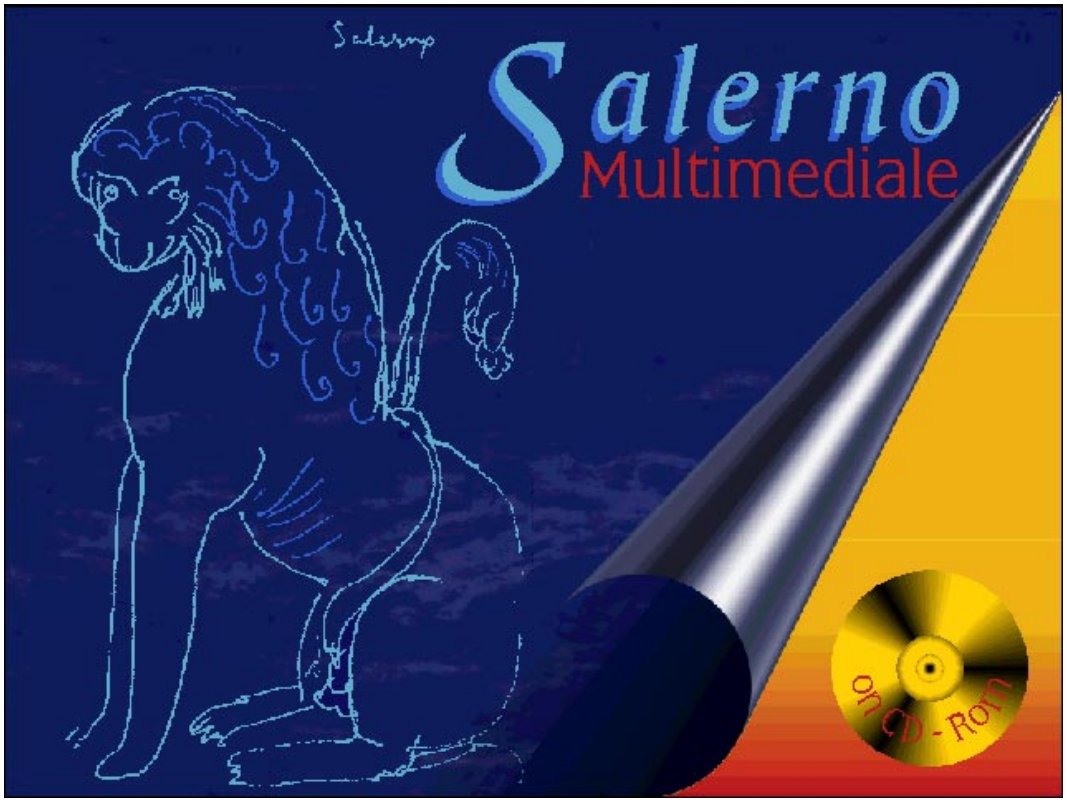

Fig. 1. The cover image of the CD rom *Salerno Multimediale*

.

The design problem for visual languages is to encode the elements of an application domain through visual sentences semantically close to them, according to a certain metaphor. Moreover, there might constraints on the characteristics of the final visual language affecting its usability, like for instance the number of generalized icons allowed for the visual language and the maximum length allowed for the visual sentences. As an example, the Minspeak^{TM} [3, 4] iconic language for the speech impaired people allowed a maximum of fifty icons and sentences with no more than three icons, because the target user could have other impairments hindering him/her in striking too many icons to form a sentence.

Let K be the set of elements from the application domain that need to be visually encoded, our design methodology accomplishes the task through the following phases:

- Domain and Knowledge Construction.
- Clustering of K.
- Icon Design.
- Coverage and encoding.

Fig. 2. Screens of the multimedia presentation *Salerno Multimediale*

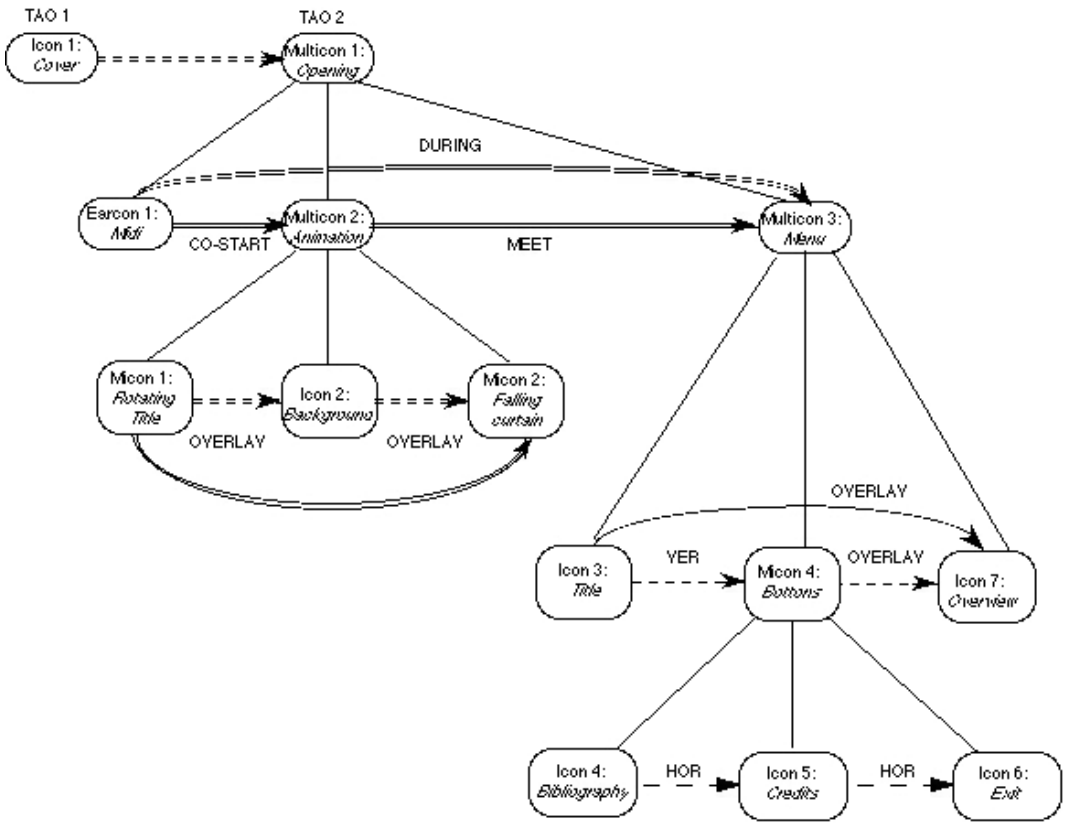

Fig. 3. An example of hypergraph structure.

- language testing.
- parser and/or syntax directed editor generation.

At the start we need to build the domain K and a reduced knowledge base to better characterize its elements. Then,we cluster K according to an adaption of the K-means algorithm to obtain a reduced set K_B of clusters representing basic concepts of the application domain. The clustering algorithm is based upon a conceptual similarity or distance function [11, 18]. Successively, the phase of icon design is performed to sketch a visual representation for the elements of K_B. Speaking in terms of generalized icons, for each word or concept $w_i \in K_B$ during the icon design phase we must sketch a visual representation for w_i to derive a new generalized icon $x = (x_m, x_i)$ such that x_m includes w_i. At the end we obtain a set I of basic generalized icons. In the next step we should combine the icons in I through the operators of icon algebra [8] to form visual sentences, for the sake of visually encoding the whole set K. Some operators are visible and some are not. Invisible operators have an empty physical part. At this point, we should encode each element w_i from the language domain through

a visual sentence made of icons and operators. Visual sentences are themselves generalized icons $S = (S_m, S_i)$. Thus, during the Coverage and encoding phase we construct a visual sentence, run inference engines to derive its meaning part S_m from the meaning parts of the icons and icon operators composing it, and then use S_i to encode the domain element w_i similar in meaning to S_m. The process is iterative and ends when all of the words from the application domain have been visually encoded. The final set of visual sentences or generalized icons form the generalized icon dictionary. Each record of this dictionary contains a domain element, the visual sentence encoding it, and a rationale explaining the association between the word and the sentence. In the language testing phase we use special tools for testing the language characteristics to verify usability issues for the intended users. Thus, we can inquire about the number of elements that have been encoded with visual sentences of a certain length and if the coding is not satisfactory we can invoke other iterations of the design phases to perform adjustments. If there are only minor problems the iteration can affect low level phases such as icon design (redesigning some icons can change their meaning and can make them associable to other elements from the language domain). Otherwise, we might have to repeat the clustering.

In order to implement the new visual language we need to construct a parser and/or a syntax directed editor. For this sake we can use several techniques. We can use parser generators [13], systems with automatic grammatical inference and syntax directed editor generators [7].

The methodology and its phases are sketched in figure 4. An high level procedural description of the methodology follows.

Procedure Design_Iconic_Language(K)
Begin
Let n be the maximum number of icons allowed for this language
Repeat
Repeat
Build_Knowledge(K);
K_B =Cluster(K, n);
1: **Repeat**
I =Design_Icons(K_B);
Encode(K_B);
Apply_Icon_Operators(I);
Cover($K - K_B$);
Let $f(k_i)$ be the set of visual sentences similar to $k_i \in K$
If the number of empty $f(k_i)$ sets is less than $Threshold_1$
Then set $Threshold_1$ to the number of empty $f(k_i)$ sets;
Until there are no empty $f(k_i)$ sets
or their number is greater than $Threshold_1$;
Until there are no empty $f(k_i)$ sets;
LID =Encode($K - K_B$);
If the number of uncoded concepts is less than $Threshold_2$
Then

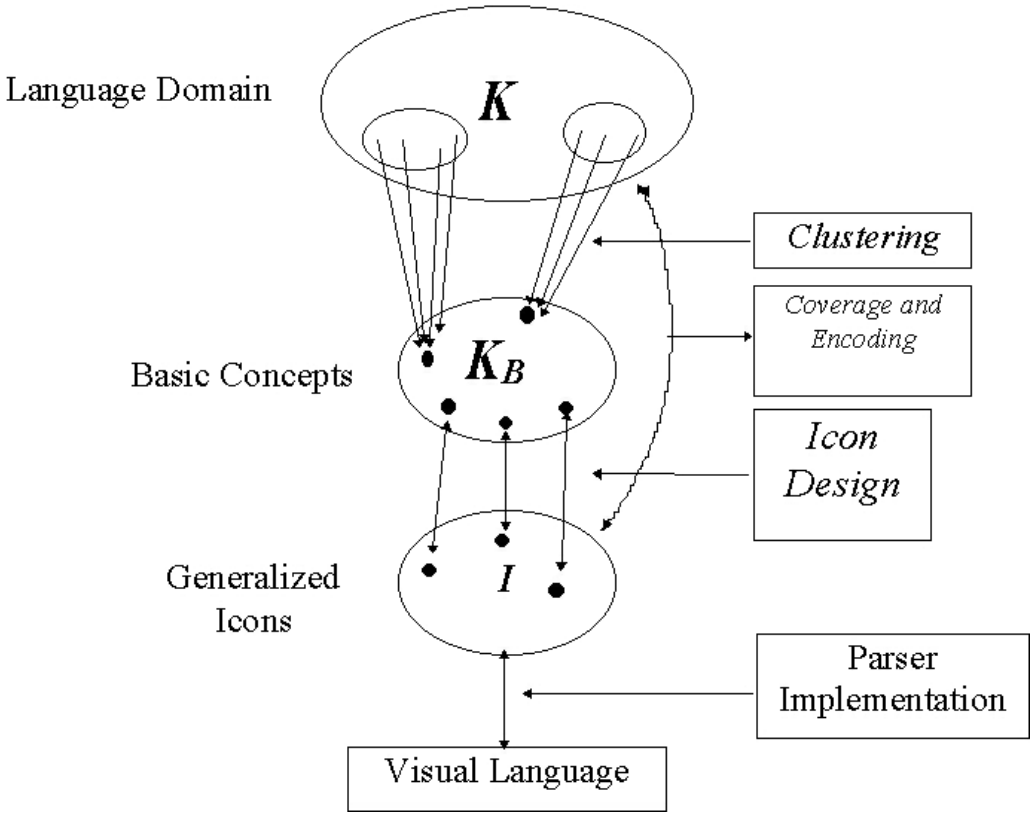

Fig. 4. The methodology for visual language design.

Begin
set $Threshold_2$ to the number of uncoded concepts;
GOTO 1;
End
Until the number of uncoded concepts is 0;
G =Construct_visual_grammar(LID);
Generate_Parser(G);
Return(LID, G);
End.

The procedure *Design_Iconic_Language* takes in input the set K returns the encoding function that assigns a visual sentence to each element in K. At the end of this process a *Language Icon Dictionary (LID)* and a visual grammar G are returned. LID contains the set of visual sentences, the domain elements they visually encode, and an explanation of the coding choice through the generalized icons and icon operators that have been used. The visual grammar

$G = (I, N, OP, S, P)$ is constructed from LID, where I is the set of icons, N is the initial set of nonterminal symbols, $S \in N$ is the start symbol, OP the set of operators and P is the set of production rules.

3.1 Domain and Knowledge Construction

As a first step of the design methodology we have to construct the language domain K, that is, the set of elements to be visually represented. These can be commands of an operating system, instructions of a textual query language, words of a natural language. We can also think of D as another type of visual language or an hybrid visual/textual language that needs a different visual representation. After the selection has been completed, we need to build a frame based knowledge base, that will be important to perform the clustering of D and to apply the conceptual similarity function.

3.2 Clustering of K

The goal of this phase is to partition the elements of the language domain K into clusters based upon their conceptual similarity. Moreover, in our case we also find the central element of each cluster C_i, which is the one minimizing the conceptual distance with respect to the other elements in the cluster. Clustering is progressively iterated until the number of clusters becomes close to the maximum number of generalized icons allowed for the visual language.

The clustering algorithm we use is an adaption of the well known *K-means* algorithm. Let K be the language domain and N its cardinality, the goal of clustering is to partition K into k disjoint subsets. The algorithm randomly selects k elements $z_1, .., z_k$ from K which will be the the central elements for the k newly created clusters. Then, each element $x_i \in K$ is put in the cluster C_j that minimizes the conceptual distance $d(x_i, z_j)$. It is important that the initial selection $z_1, .., z_k$ produce a uniform distribution of the domain elements among the clusters. Successively, the algorithm recomputes the new central element for each cluster, which will be the one minimizing the distances with respect to the other elements in the cluster. This operations might change the distribution of the elements among clusters, since each element might have become closer to another cluster center z_j. Thus, elements are moved across clusters and this might require new adjustments to the cluster centers. The number of clusters k depends upon the maximum number of icons max allowed for the visual language to be designed. For some languages k coincides with max and in general it is close to that value.

If **d** is the chosen conceptual distance function with values in $[0, 1]$, the final K-means algorithm for visual language construction is described in the following:

Procedure Cluster(K, k)
Begin
Select k cluster centers $z_1, ..., z_k$ from K randomly;
While there are no changes in the composition of clusters

```
        For each x_i ∈ K
           Put x_i in cluster C_j that minimizes d(x_i, z_j);
        For each cluster C_i
        Begin
           mindist = 1; // mindist is initially set to the maximum value for d
           For j 1 to |C_i|
           Begin
              d_j = Σ_{r=1}^{|C_i|} d(x_j, x_r)
              If d_j < mindist Then
              Begin
                 mindist = d_j;
                 z_i = x_j;
              End;
           End;
        End;
     Enddo;
End.
```

where d_j represents the total conceptual distance between x_j and each of the remaining elements in the cluster. The variable $mdist$ contains the minimum d_j observed at a given moment. At each step, the value of $mdist$ is updated with that of d_j if $d_j < mdist$.

3.3 Icon Design

The design of images for generalized icons is a crucial step in the process of designing a visual language. The first goal here is to draw images to visually represent the central concepts of the clusters determined in the previous phase, taking into consideration the uncertainty that these may not have been selected properly. The remaining elements of the clusters will be visually encoded starting from the image associated to the cluster and combining it with other images through icon operators.

When designing an icon, the following guidelines should be taken into consideration [11]:

- The icon should clearly describe the basic concept.
- The icon should be conceptually rich. It should be created in such a way that when joined to other icons together they cover a substantially broader set of concepts.
- The icon should be related to the way of using it.

In this phase we can use the operators of Icon algebra [8] to explore the meanings that can be covered by an image, in order to encode a broader set of elements from the domain K. For example, the designer can consider the following issues:

- Can this icon be spatially combined with another icon to cover a new concept?
- Does this image contain a detail that can be emphasized to cover a new concept?
- Does the inversion of this image derive a new concept?

by using the spatial operators, the marking operator, and the inversion operator, respectively. Some operators have been embedded in the inference engine of the system DEVIL (Design Environment for Visual Languages) [18]. We first construct the meaning parts of the generalized icons and then apply logical operators to verify the coverage of the elements in K. In other words, if a visual sentence is obtained through the application of a physical operator OP_i to the generalized icons of the sentence, the logical operator OP_m can produce inferences to derive the meaning of the whole visual sentence, that can be used to find the elements of K that are conceptually close to it by using a similarity function.

3.4 Coverage and Encoding

Once a basic set I of generalized icons has been designed, with both physical and logical parts, we need to combine them through icon operators, in order to ensure the covering and encoding of the whole set K. During the covering process, for each element $k_i \in K$ we compute a set $f(k_i)$ of visual sentences conceptually similar to it, i.e. those visual sentences α such that $d(k_i, \alpha)$ is less than a predefined threshold.

A failure in the covering process occurs when either at least one set $f(k_i)$ is empty or it is not empty but all of its visual sentences have been used to visually encode other elements. When a failure occurs, we first try to review the *icon design* phase without changing the clustering, peraphs the images have not been designed properly. However, if the number of uncovered elements is greater than a predefined threshold, we might have to modify the clustering process on K.

3.5 Language Testing

During this phase we query the language icon dictionary to verify the characteristics of the visual language. Queries include the length of visual sentences, the use of spatial and temporal relations, etc. For instance, we can ask for the number of visual sentences containing more than x icons, or the sentences with more than y icons arranged using the vertical combination, etc. If adjustments are required we need to go back to a previous phase and review some of the steps.

3.6 Parser Generation

The language icon dictionary provides the visual sentences (i.e generalized icons) of the new language VL and their meanings. In this phase we provide either a

parser or a syntax directed editor to recognize the sentences in the dictionary. The logical part of the visual sentences correspond to elements of the language domain K, and the correspondence is enforced by the semantic routines associated to the productions of the visual grammar for VL.

3.7 The System DEVIL

The system DEVIL (Design Environment for Visual Languages) [18] is an interactive design environment with a set of tools that provide automated support for the visual language design methodology just described. It is divided in a front-end module and a back-end module. The former provides support for constructing or importing the language domain K, the knowledge base, the distance function and for clustering the language domain. The back-end module provides support for icon design, visual coding, language testing, and has a prototyping environment for the application of icon operators and inference routines to derive the logical part of visual sentences. It can be connected to several tools for the generation of the parser or syntax directed editor [7, 13]. We have used the methodology and the system DEVIL to design visual languages for handicapped people [3, 4, 11], visual languages for specifying security policies [12], visual languages for querying databases [18], and so on. As expected, the front-end part of the methodology was necessary only for complex visual languages. When there are few elements to be visually encoded it is sufficient to apply the methodology starting from the icon design phase.

4 The Extended Methodology for Multidimensional Languages

Multidimensional languages have been used for specifying the syntactic structure, knowledge structure, and dynamic structure of complex multimedia objects such as TAOs [10]. They are particularly useful for specifying the external appearance of the TAOs, that would be otherwise a difficult task, due to their complex nature. Multidimensional sentences can be parsed in order to control the associated multimedia presentation [2]. Generalized icons and icon operators are central concepts for multidimensional languages, they have been defined by extending similar concepts of visual languages. We extended our design methodology for visual languages in order to design multidimensional languages. We will often refer to the phases of the visual language design methodology to explain how they have been extended for multidimensional languages.

The design problem for multidimensional languages is more complex than it is for visual languages. In particular, here the association between the elements of an application domain and multidimensional sentences is more approximate. In fact, since we aim to provide a TAO representation for the elements of an application domain, the association between these elements and the TAO is also dynamically ruled by the knowledge structure associated to the TAO. We need a methodology to derive a multimedia representation for the domain elements. In

general, this process includes *content selection*, *media allocation*, and *media realization* [19]. We see this process as the derivation of a certain number of TAOs representing the elements of the application domain in multimedia presentations. Thus, we have defined a design process to derive the multidimensional language for expressing the TAOs for a given domain.

As an example, we have used the methodology for transforming a textual lecture into a multimedia presentation.

In subsections that follow we describe the phases of the new methodology. The process can be customized for the different applications. In particular, as for visual languages also for multidimensional languages high level design phases can be discarded for simple applications.

4.1 Domain and Knowledge Construction

In this phase we have to build the multidimensional language domain K. As opposed to visual language design here the language domain includes more type of elements, such as images, sounds, etc. Also the frame structure for the knowledge base is more complex. Some of the attributes used for the design of visual languages apply to multidimensional languages. These are the *Sound*, *Time*, *Location*, *Color*, and *Shape* attributes. Some other attributes depend upon the application domain and are used to express content. Moreover, we need an additional attribute *image*. Their attribute values include not only text but also images and sound. Complex attributes such as Sound and Image can also have a special index [14] to be used for similarity matching. In fact, the similarity function to be used for multidimensional languages needs to find similarity in sound or image, for which it can use well known approximate matching techniques developed for multimedia databases.

4.2 Clustering

In this phase is no more sufficient to merely cluster the elements of the application domain through a conceptual similarity function. We first structure the domain elements by using object oriented modeling techniques. It is important in this phase to capture the different types of relationships, expecially Part-Of and ISA relationships, because they will be very useful when designing generalized icons. Successively, we perform Clustering by using the class diagram and a special distance function. The latter still compares attributes to determine the similarity, but it will be using more sophisticated and approximate matching techniques because of the presence of complex types of data. For example, if a text mentions the painting of Leonardo *Monnalisa* and another contains a figure with the image of MonnaLisa, then these two elements should be considered close and then clustered together. At the same time, in the some cluster should also fall other images with some visual characteristics similar to the painting of Monnalisa, such as colors, shapes, etc, for which the function used well known techniques for image retrieval [14]. As a result, new relations can be added to the class diagram, some of which can be translated in the knowledge structure

of the TAO. In fact, similar images or text related to images may be part of a *prefetch_related_info* message for the active index [9].

4.3 Generalized Icon Design

The input to this phase is the class diagram, the object diagrams and the clusters determined in the previous phase. We need to translate their information in terms of generalized icons. Thus, we first construct the physical appearance of generalized icons and then their logical part. We select central elements of clusters and design the physical parts of image icons, earcons and ticons. For instance, we can decide to provide both a visual and a sound representation for a textual information according to a certain metaphor.

After have sketched generalized icons for elementary information we apply icon operators to compose multidimensional sentences. We exploit the relationships of the class diagram and the clusters to understand the appropriate operators to apply. For instance, if the *Time* attribute of the frame for the Transparency2 of a lecture indicates that it follows the Transparency1 and can be recalled by the Transparency10, then we build a multidimensional sentence by applying appropriate temporal operators and run inferences to derive the logical parts of the sentence. For temporal operators we have provided a visual representation using the calendar metaphor as shown in Figure 5. The generalized icons can be put on calendar cells to express their temporal ralationships. Moreover, we can zoom each icon to set its specific attributes using the lock transcription [12].

4.4 Approximate coverage

For each element of the domain we compute the set $f(k_i)$, but we do not have to perform a precise coding choice. We will only maintain a set of multidimensional sentences and a score indicating the type of similarity with a rationale associated. This is done at different levels of granularity. For example, we might consider a chapter as an element and see how it can be represented, or we can go at the level of its sections.

This information will be used for TAO construction. Some of the operators used for composing multidimensional sentences are associated to inference routines which will modify the active index.

4.5 TAO generation

Having constructed a set of multidimensional sentences covering the language domain, we can compose them and select their semantics from the associated sets $f(k_i)$. The parsing of the sentences will control the multimedia presentation.

As for the knowledge structure, visual languages have been proven to be useful for specifying it [10]. Part of the knowledge for the TAO is automatically generated by the inference routines associated to the icon operators.

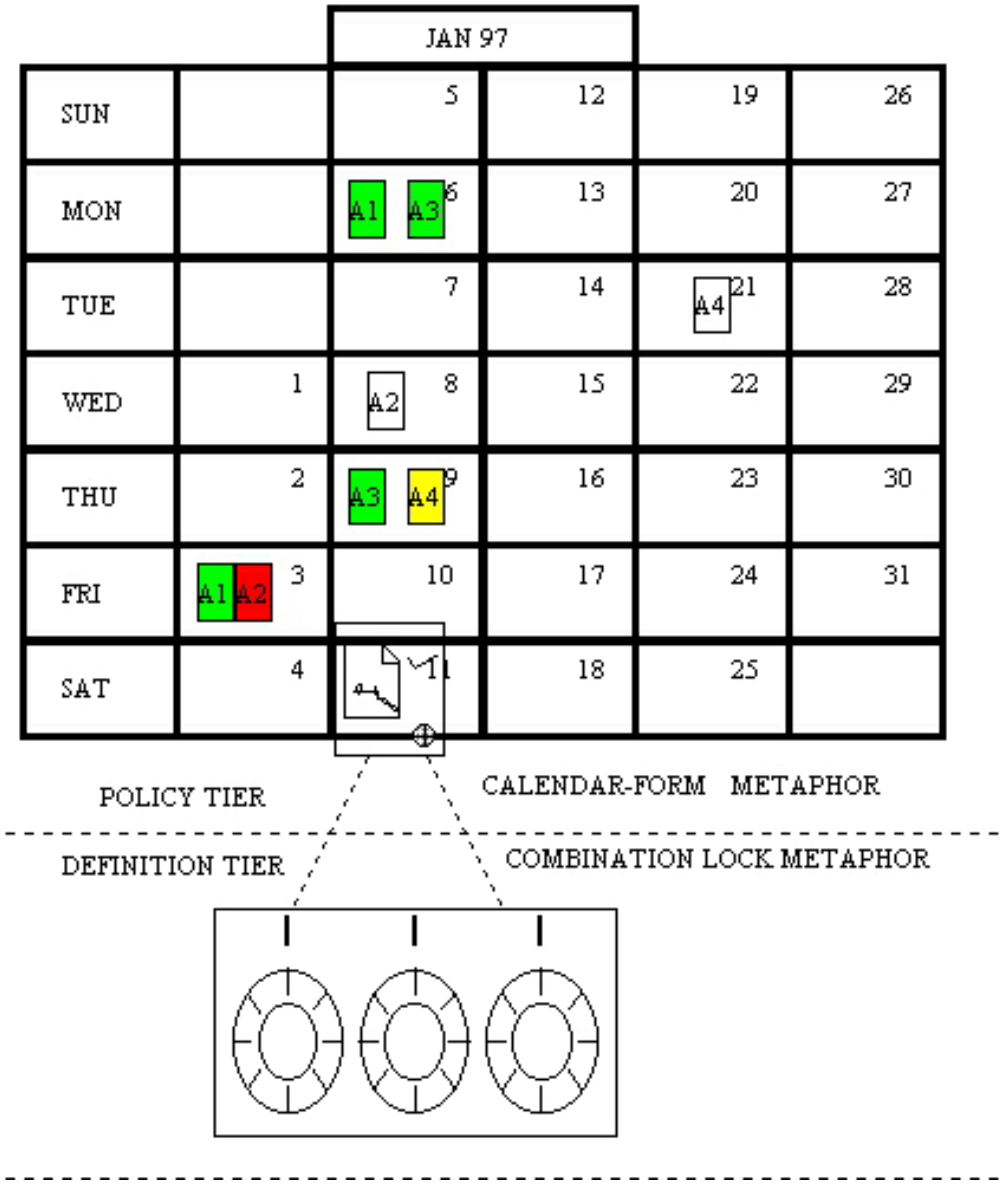

Fig. 5. Physical appearance of the temporal operators.

5 Example

The methodology presented in this paper has been used experimentally for the development of a multidimensional language to transform a lecture into a multimedia presentation format. Moreover, we have used it to design visual query languages for multimedia databases. In the first example we started from a domain language made of textual lectures with transparencies comprising text, figures, tables, plus there could be movies on specific subjects. The goal was to derive the multidimensional sentences expressing the TAOs for the multimedia presentation of the lecture. An abstract class diagram for this example is shown in Figure 6.

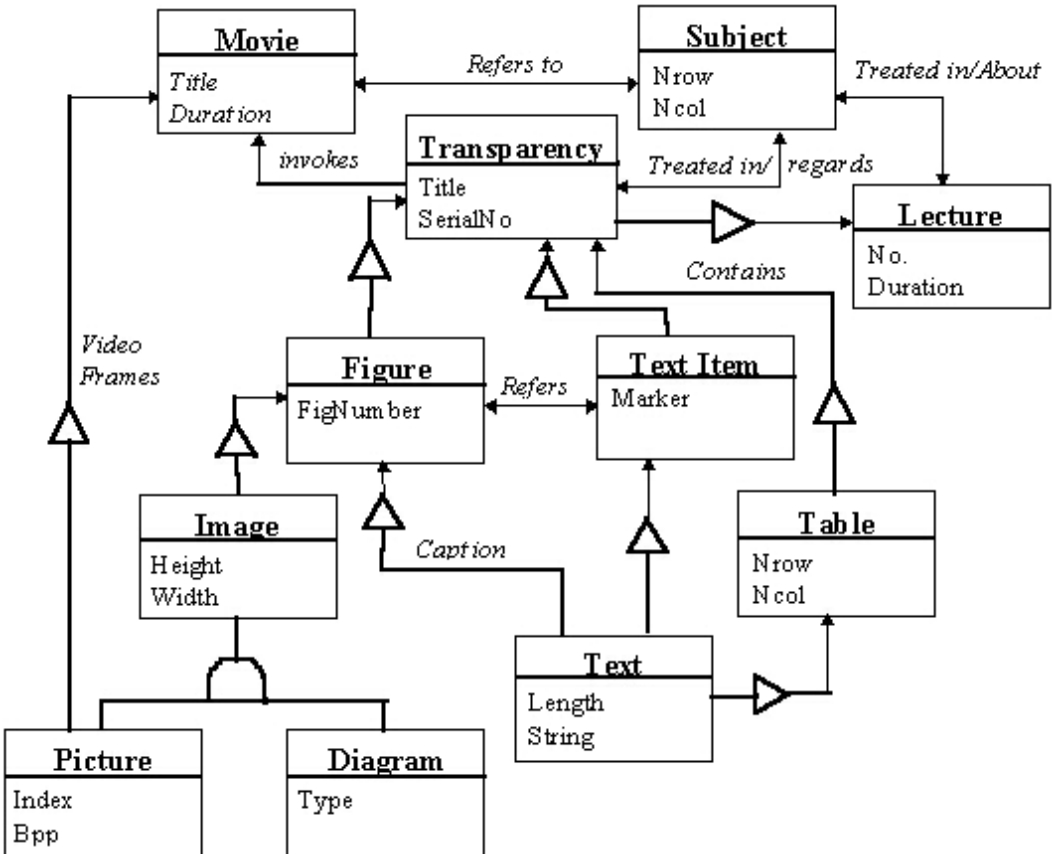

Fig. 6. The class diagram for the Lecture example.

Moreover, in the frame-based knowledge base we provided additional attributes to better characterize the domain elements, so as to be able to apply distance functions and clustering algorithms. For instance, a textual element

were further characterized in terms of its meaning, its spatial and temporal information, and the possibility to associate it to an image or a sound.

Let us consider a transparency from a medical lecture on meniscal surgeries as shown in Figure 7.

In the object diagram there will be an instance *Meniscal Cuts* of the class *Transparency*, which is connected to the instance *Meniscal Surgery* of the class *Subject* through the relationship *Regards*, to the instance *Movie1* of the class *Movie* through the relationship *Invokes*, and is composed of three instances of the class *Text Item*. Additional attribute values for each object are provided in the knowledge base. For example, the text item *Umbrella handle* has associated the following frame of attributes:

SLOT:	**VALUE**
NAME:	**Umbrella handle**
SHAPE:	Sketch(Umbrella handle)
COLOR:	Monochromatic
LOCATION:	Middle of Knee
IMAGE:	Overlay (This.Shape, Knee_CT_ Scan)
TIME:	Before(Before(Co_start(This.text, This.Image), "Circle shape"), Movie1)

Notice that the *Image* attribute is a query to an image database [14, 17]. Since the language domain included a collection of medical images we could run such queries on the system *FIRST* [17] during the multidimensional clustering. In this way we could cluster images with similar meniscal anomalies together. Moreover, clustering also detected other textual items talking about knee.

During the *Generalized Icon Design* we produced a vicon *Movie1* and a ticon for each of the three text items in the transparency. Then, the *part-of* relationship between the transparency and its three text items suggested the introduction of a multicon for the whole transparency *Meniscal Cuts* and the application of the *attach* operator for each of the three text items. We applied the spatial operator *ver* to combine the ticons. The *Invokes* relationship and the TIME attribute from the frame structure associated to the transparency suggested the application of the temporal operator *before* to combine the multicon with the vicon. Moreover, the SHAPE attribute of the the textual items "Umbrella handle" and "Circular shape" suggested the sketch of two icons each depicting one of the two shapes. The sketch queries contained in the IMAGE attribute caused the linking during the clustering phase to examples of CT scan images reporting similar knee anomalies. We could then decide to enrich the presentation by combining the two ticons with either the two icons or even with some of the CT scan images (represented as icons) resulting from the queries. A further decision was to be made on the spatial and temporal operators to use for combining ticons and icons. For example, each ticon could be combined with the associated icon by using the spatial operator *overlay* and the temporal operators *co_start, co_end.*

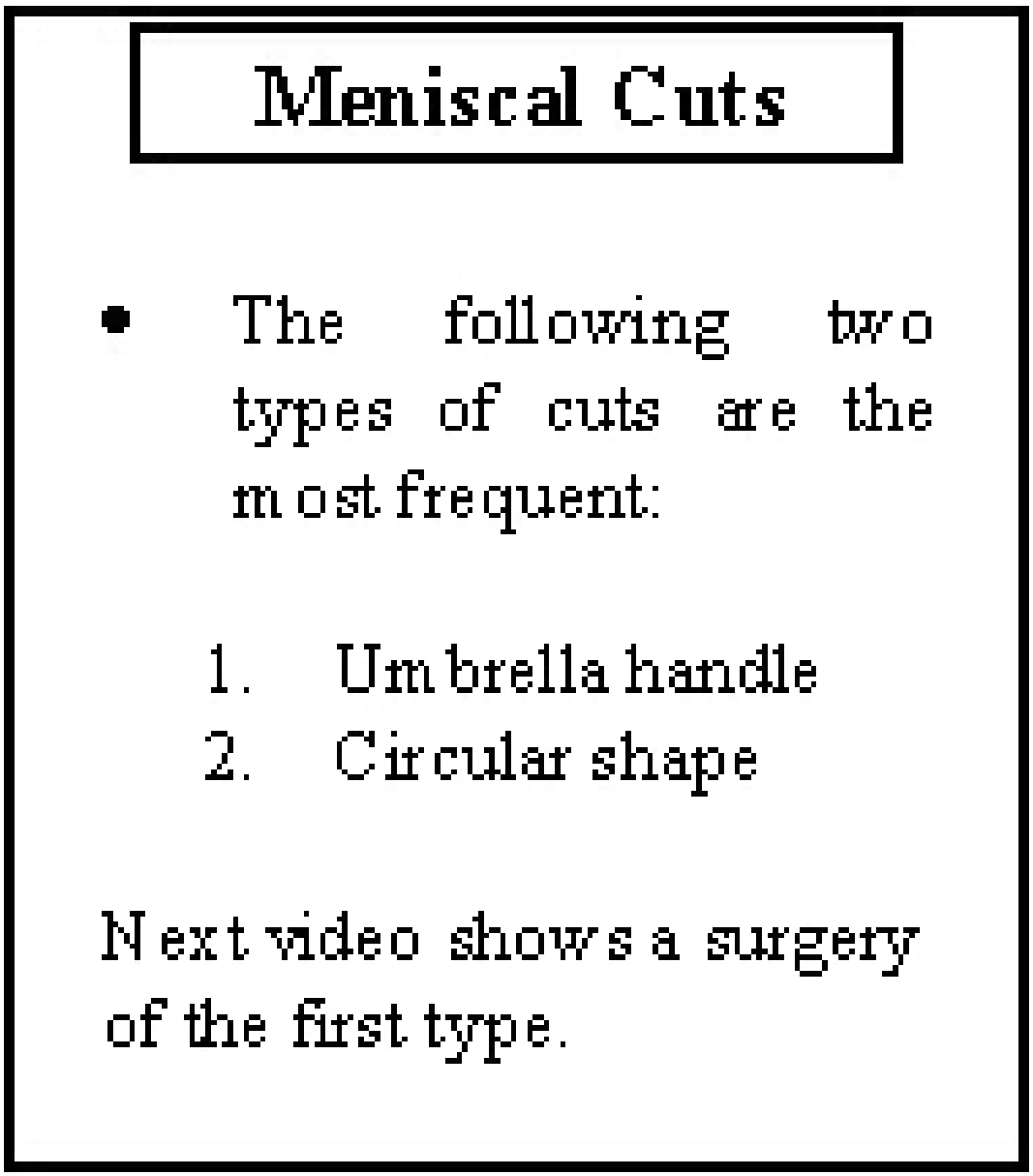

Fig. 7. A transparency on Meniscal Surgeries.

As noticed, we could produce several multidimensional sentences to represent the given transparency. The entry *Meniscal Cuts* in the multidimensional icon dictionary will have associated all of these multidimensional sentences and the rationale for each of them. An example of rationale follows: [Multicon[Meniscal Cuts] *attach* [[ticon[umbrella] *overlay* + *co_start* + *co_end* icon[umbrella_cut]] *vertical* [ticon[circular] *overlay* + *co_start* + *co_end* icon[circular_cut]]] *before* Vicon[Movie1].

From the rationales in the multidimensional icon dictionary we can easily generate the associated TAOs.

6 Conclusion

We have extended a methodology for visual language design to perform the design of multidimensional languages to be used for visually specifying the structure of Teleaction Objects. We are currently extending the tools of the system *DEVIL* to provide automated support for the phases of the new methodology. Some extensions have been made on parser generation tools. We need to further investigate similarity and clustering algorithms and use the methodology for generating a broader class of multidimensional languages. Moreover, we need to refine logical icon operators to directly generate the knowledge part of the TAOs.

References

1. Allen J.F., Time and time again: The many ways to represent time, International Journal of Intelligent Systems, vol. 9, No. 3, June 1991, pp. 413-428.
2. Arndt T., Cafiero A., Guercio A., Multimedia Languages for Teleaction Objects, Proceedings of the IEEE Symposium on Visual Languages, Sep. 1997, pp.322-331.
3. Baker B. R., Minspeak, A Semantic Compaction System that makes Self-Expression easier for communicatively Disabled Individuals, Byte, volume 7, number 9, pages 186-202, September 1982.
4. Baker B. R. , Schwartz P. J., and Conti R. V., Minspeak, Models and Semantic Relationships, Fifth Annual Minspeak Conference Proceedings, Seattle, WA, November 1990.
5. Blankenberger S., Hahn K., Effects of Icon Design on Human-Computer Interaction, International Journal of Human Computer Interaction, Vol. 35, 1991, pp. 363-377.
6. Chang H.J., Hou T.Y., Hsu A., Chang S. K., The Management and Application of Tele-Action Objects, ACM Multimedia Systems Journal, vol. 3, No. 5-6, 1995, pp. 204-216.
7. Chang S. K. , Costagliola G., Pacini G., Tucci M., Tortora G., Yu B.,Yu J. S., A Visual Language System for User Interfaces IEEE Software, vol. 12, No. 2, 1995, pp. 33-44.
8. Chang S. K., Principles of Pictorial Information Systems Design, Prentice-Hall, 1991.
9. Chang S. K., Toward a Theory of Active Index, Journal of Visual Languages and Computing, vol.3, No. 3, 1996, pp. 18-26.

10. Chang S. K., Extending Visual Languages for Multimedia, IEEE Multimedia Magazine, Fall 1996, Vol. 3, No. 3, pp. 18-26.
11. Chang S.K., Polese G., Orefice S., Tucci M., A Methodology and Interactive Environment for Iconic Language Design, International Journal of Human Computer Interaction, Vol. 41, 1994, pp. 683-716.
12. Chang S.K., Polese G., Thomas R., and Das S., A Visual Language for Authorization Modeling, Proceedings of the IEEE Symposium on Visual Languages, 1997, pp. 110-118.
13. Costagliola G., De Lucia A., Orefice S., Tortora G., A Parsing Methodology for the Implementation of Visual Systems, IEEE Transactions on Software Engineering, vol. 23, n. 28, 1997.
14. De Marsico M., Cinque L., Levialdi S., Indexing Pictorial Documents by Their Content: A Survey of Current Techniques, Image and Vision Computing, vol. 15, 1997.
15. De Roberto P., Ermellino F., Patria G., Tortora G., Salerno Multimediale, Laurea Thesis, Dipartimento di Informatica, University of Salerno.
16. Lin C.C, Xiang J.X, Chang S.K., Transformation and Exchange of Multimedia Objects in Distributed Multimedia Systems, ACM Multimedia Systems Journal, vol. 4, No. 1, 1996, pp. 2-29.
17. Nappi M., Polese G., Tortora G., Fractal Indexing and Retrieval System, Image Vision and Computing, 1996.
18. Polese G., MMDMS:A Framework for the Design of Spatial-Temporal Visual Languages, Ph.D. Thesis, University of Salerno, Italy.
19. Weitzman L., Wittenburg K, Automatic Presentation of Multimedia Documents Using Relation Grammars, Proceedings of ACM Multimedia 94, ACM Press, New York, pp. 443-451.
20. Weitzman L., Wittenburg K., Grammar-based Articulation for Multimedia Document Design, ACM Multimedia Systems Journal, Vol. 4, No. 3, 1996, pp. 99-111.
21. Weitzman L., Wittenburg K., Ralational Grammars: Theory and Practice in a Visual Languages Interface for Process Modeling, Proceedings of AVI 96 International Workshop on Theory of Visual Languages, http://www.cs.monash.edu.au/ berndm/TVL96/tvl96-home.html.

This article was processed using the LaTeX macro package with LLNCS style