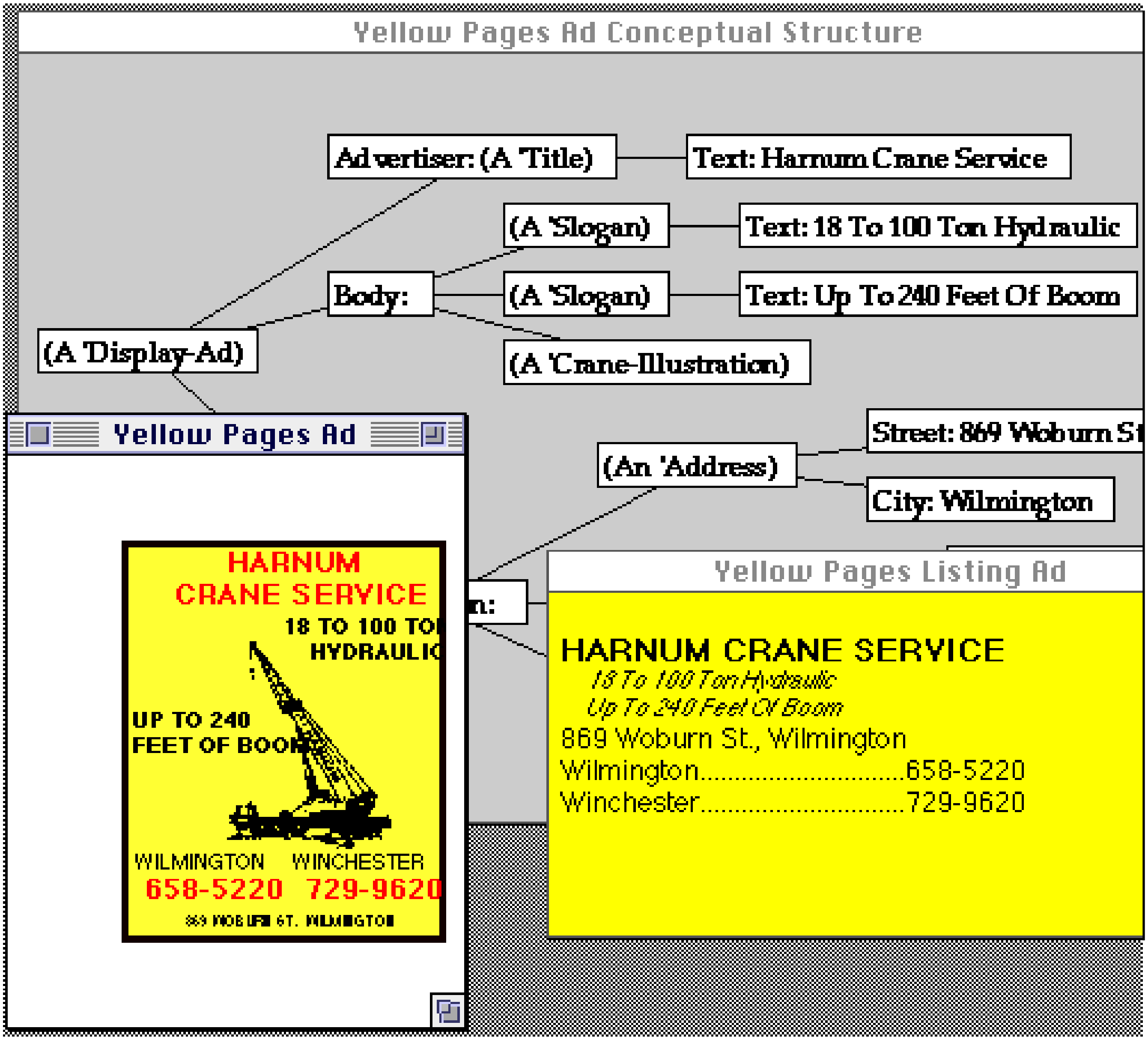

Yellow Pages Ad Conceptual Structure
Advertiser: (A 'Title)
Text: Harnum Crane Service
(A 'Slogan)
Text: 18 To 100 Ton Hydraulic
Body:
(A 'Slogan)
Text: Up To 240 Feet Of Boom
(A 'Display-Ad)
(A 'Crane-Illustration)
Yellow Pages Ad
Street: 869 Woburn St
(An 'Address)
City: Wilmington
HARNUM
CRANE SERVICE
18 TO 100 TO
HYDRAULIC
UP TO 240
FEET OF BOOM
WILMINGTON
WINCHESTER
658-5220
729-9620
Yellow Pages Listing Ad
HARNUM CRANE SERVICE
18 To 100 Ton Hydraulic
Up To 240 Feet Of Boom
869 Woburn St., Wilmington
Wilmington............................658-5220
Winchester............................729-9620

Making Programming Accessible to Visual Problem Solvers

Chapter 7

by
Henry Lieberman
Media Laboratory
Massachusetts Institute of Technology

Introduction

Graphic designers, architects, multimedia designers, animators and users of CAD/CAM systems are experts in visual problem solving. To date, computer-based graphic editing tools provide these people with the means to display and edit visual representations of their designs, but provide little explicit support for the problem-solving processes that visual designers perform. A study of how knowledge is actually communicated in visual design domains shows that visual designers rely heavily on the generation and critique of visual examples. Programming by demonstration is a technique that permits the construction of programs directly from the presentation of graphical examples, together with some explanation of them. This paper presents a scenario of how a particular demonstrational technique, graphical annotation, can be used to convey the structure of a design to permit a system to perform layout tasks automatically.

The goal of a graphic designer or multimedia designer is often to create a graphic artifact that serves to communicate some information to a viewer. The designer must encode his or her communication intentions in some structured representation of media objects that reside in a graphical editor.

The problem is that the current generation of graphic editors keeps only the *results* of design choices, that is, only the arrangement of media elements the designer has chosen. This is fragile, because if any of the assumptions responsible for the current set of graphic choices changes, the design may no longer be valid as a solution to its communication goals. If, for example, the amount of space on the screen or page changes, correct response to such a change is not simply to stretch or shrink the design, but to change decisions about typeface, aspect ratio or placement of elements. The designer must redesign the artifact "as if" the new situation had been present from the beginning, while preserving as much of the original reasoning [not necessarily the original data!] as possible. This process is tedious and error-prone, and it is here that computer assistance could be of real help to the designer.

Culturally, designers are comfortable with direct manipulation of the visual elements of a design, and not comfortable at all with textual input of abstract concepts. So a possible approach might be to *use concrete visual examples as the representation of abstract concepts in the domain of visual design*.

Representing An Example Design: A Yellow Pages Advertisement

I will use as an example design an advertisement from the Yellow Pages telephone directory. This format is known as a display ad. Subtle decisions concerning typefaces and placement are made to emphasize certain aspects of the information and maintain stylistic consistency. It is the task of communicating those decisions from the designer to the machine that poses the challenge for constructing intelligent design assistants.

An important component of the representation activity that knowledge engineers do involves establishing the structure of the concept domain. Part-whole hierarchies indicate relationships between components of the design. An inheritance hierarchy links more specific to more general concepts. These are expressed in terms of a textual object-oriented description language, of which a sample appears below.

Figure 1: Example Yellow Pages advertisement

```
(defob yellow-pages-ad)

(defob graphic-ad-component () (ad-component graphic))

(defob display-ad
  ((advertiser (a 'graphic-ad-component))
   (body (a 'graphic-ad-component))
   (information (a 'graphic-ad-component)))
  (yellow-pages-ad))

(defob listing-ad
  ((advertiser (a 'graphic-ad-component))
   (address (a 'graphic-ad-component))
   (phone-number (a 'graphic-ad-component)))
  (yellow-pages-ad))
```

Writing such descriptions is problematic for graphic designers as a means of communicating the conceptual information in a design. The syntax is difficult to remember, and the connections between the concepts and the graphic elements are not immediately apparent.

Meta-Graphic Annotations Communicate The Structure Of Designs

We propose an alternative. We would like to allow the designer to communicate taxonomic knowledge about a design by annotating concrete graphical examples with what we will call *meta-graphics*, graphic objects whose purpose is to describe other graphic objects in the editor. *Meta-graphic editors* can distinguish between object-level graphics and meta-graphics, either indicated explicitly by the user, or through a visual parser, such as found in [Lakin 87] or [Weitzman 92]. They can provide the ability to attach knowledge structures to meta-graphics that can be used in automated design systems.

Callouts are textual meta-graphic labels introduced into a picture to serve as visual pointers to the conceptual function of graphic elements in a design. Below, the user annotates the yellow pages ad by attaching graphic labels to each element, using the same operations provided by the graphical editor. Note that labels themselves can be labeled, to produce multilayered structures.

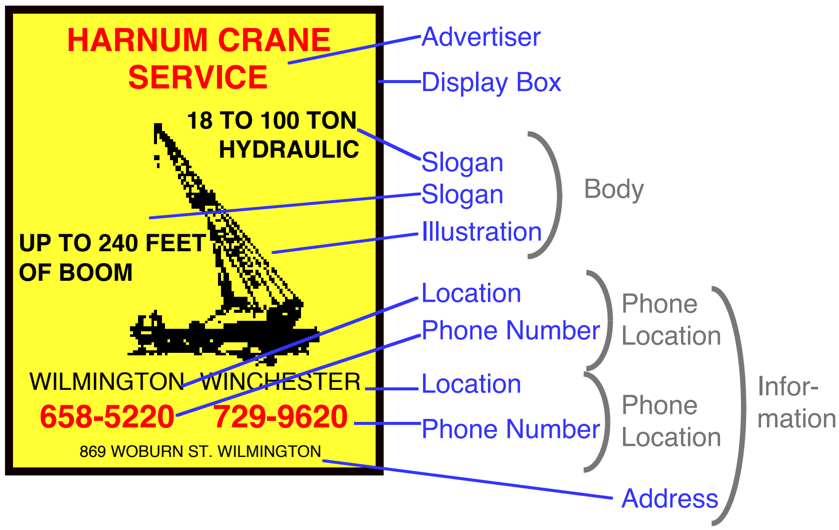


Figure 2: A meta-graphic annotation of a Yellow Pages ad

The task of the meta-graphic editor is to build associations between the conceptual structure of the design and the primitive graphic elements, such as lines, text, or bitmaps, and attributes of those objects, such as fonts and color. Below, we illustrate the graphic structure for our example ad, seen through an *inspector* [or *browser*] interface.

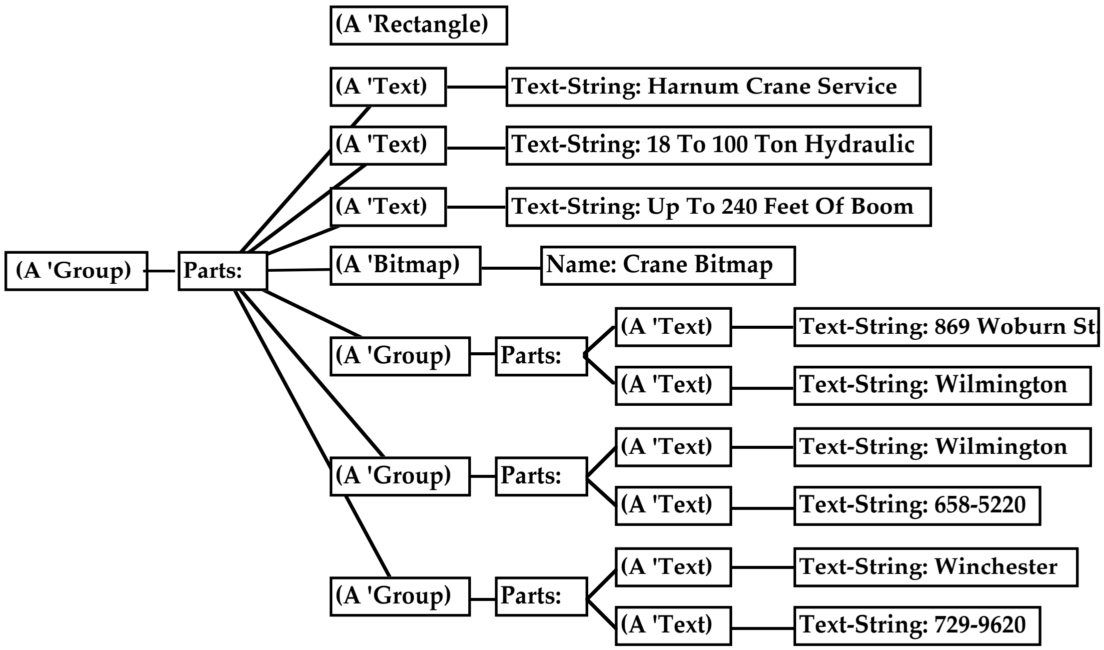


Figure 3: Purely graphic object structure for Yellow Pages ad

By simply noting the position of the labels and lines in the meta-graphic annotations, a very simple visual parser can produce a hierarchical structure that corresponds to the annotations, and build cross-links between the parts of the conceptual structure and the graphic structures that serve as examples of them. Annotations allow the designer to visually verify that the knowledge structures have a clear correspondence with the graphic structure.

Below, the conceptual structure obtained from the meta-graphic annotations.

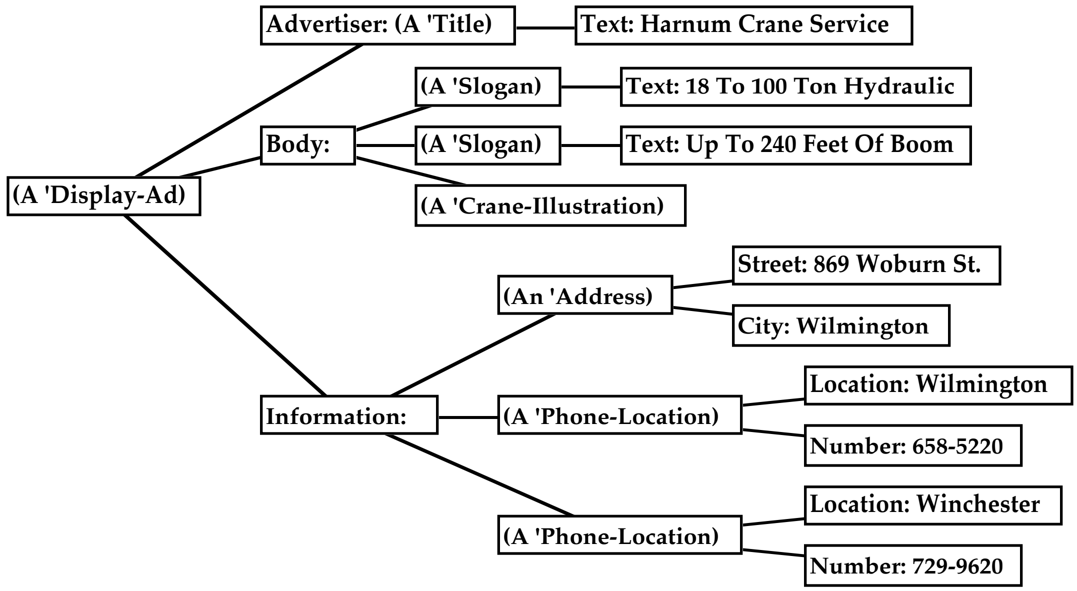


Figure 4: The conceptual structure of the Yellow Pages ad

Annotations and Programming by Demonstration

Several systems for *programming by demonstration*, such as the author's Mondrian, Maulsby's Metamouse, and Kurlander's Chimera have been designed to operate in a graphic editing domain. Such systems record user actions in a graphical editor, such as the creation of lines or text, and attribute changes such as color or alignment. Augementing these systems with annotation capability would allow them to relate conceptual structures to graphic structures. The same conceptual structure may then be used to generate a graphical structure that is visually quite different. If a graphical operation is performed on an annotated item, that can indicate to the system that an analogous operation should be performed on whatever item may receive the corresponding annotation in a future example.

In the Yellow Pages, another format, the *listing ad*, contains only text, not graphics, and displays the essential information of the name of the advertiser, address and phone numbers. Typeface and alignment relations indicate the conceptual function of each component, as did placement relations in the display ad format.

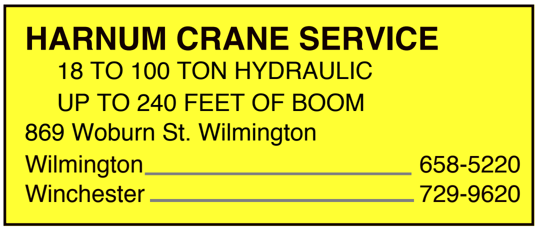


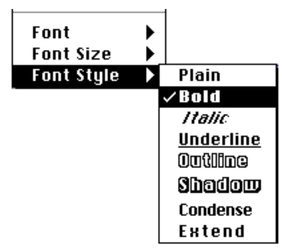
Figure 5: Yellow Pages listing ad

To teach the system how to make listing ads from display ads, the designer can transform the Harnum Crane display ad into the format of the listing ad shown above by modifying the properties of the graphic elements. The system records the modification steps, and generalizes them according to the relations specified by the annotations.

For example, an interactive type style change to the text element representing the address "869 Woburn St. Wilmington" in the Harnum Crane ad would be represented in the recorded procedure as the code,

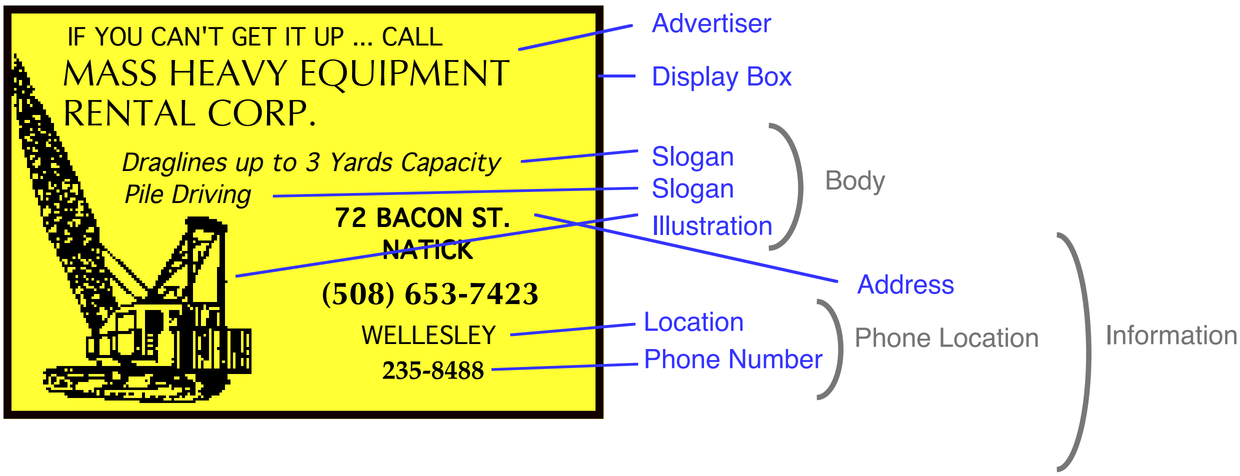
```
(setf (font-style (address display-ad)) :bold)
```

Figure 6: An interactive change to a typeface



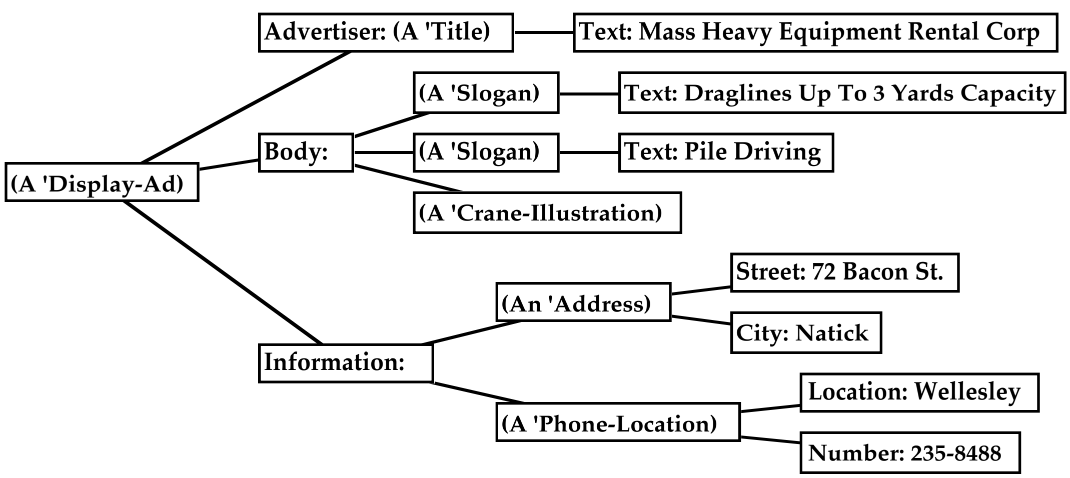
referring to that element by its annotated function in the design rather than by its graphic structure. The recorded procedure would then take a display ad as an argument, and produce an analogous listing ad. We supply a new example by annotating a new ad with a similar conceptual structure. We can take advantage of our original annotation by copying most of the annotation meta-graphics from the original example, and adjusting them as necessary to fit the new example.

Figure 7: Annotation of another display ad



The result of this is a conceptual structure for the Mass. Heavy Equipment ad analogous to that of the Harnum Crane ad.

Figure 8: Conceptual structure of second display ad



The representation of the conceptual structure of the display ad allows the system to automatically generate the corresponding listing ad.

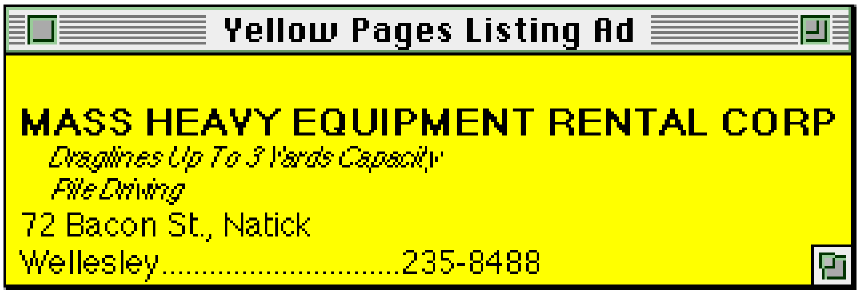


Figure 9: Automatically generated listing ad

Some care must be taken both in how the user demonstrates the procedure, and what aspects of the graphic operations the system decides to record. If the user centers a graphic element by eye rather than by calling the centering function of the graphic editor, the system will record a move action, and it might not recognize the intent to center the object. Such a recorded procedure might not have the desired centering effect in subsequent examples. Some systems such as Peridot, contain rules for comparing graphic states that would be capable of recognizing centering relations. Similarly, the system must not record the operations too literally, for example by embedding absolute cursor positions in the code.

This paper has presented a scenario of how concrete visual examples can serve as a means for communicating design knowledge in a graphical editor. Annotations serve as the graphical counterpart of variables in programming. Systems that can record user actions and generalize on graphical annotations can go beyond the current generation of graphical editors in providing support for automatically generating design alternatives based on the functionality of components of the design. Programming by demonstration systems can help visually oriented users access the procedural power of computers, which would otherwise require programming in a textual programming language.

Acknowledgments

The Visible Language Workshop at the MIT Media Laboratory is supported by research grants from Alenia, Apple, DARPA, Kansa, NYNEX, Paws, and Digital.