The Architecture of Information

Interpretation and presentation of information in dynamic environments

Louis Murray Weitzman

Bachelor of Architecture,
University of Minnesota, 1974
Master of Architecture in Advanced Studies,
Massachusetts Institute of Technology, 1978

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy at the
Massachusetts Institute of Technology

February 1995

© Massachusetts Institute of Technology, 1995
All Rights Reserved

Author: [illegible]
Program in Media Arts and Sciences
January 13, 1995

Certified b
Ronald MacNeil
Principal Research Associate, Media Laboratory
Thesis Supervisor

Accepted by [illegible]
Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Rotch

[illegible] 2[illegible] 1995

The Architecture of Information

Interpretation and presentation of information in dynamic environments

Louis Murray Weitzman

Abstract

Design of information presentation is undergoing significant changes. Documents are information interfaces that must dynamically reconfigure themselves based on their content, the medium in which they are displayed, and the intended use of the information they present. Increases in computational power and the increased bandwidth of interconnected networks provide greater access to information. These factors, combined with the realization that not all of this information can now be pre-designed, necessitate new tools and techniques to ensure the effective presentation of computer-based information.

This dissertation exploits the structure of information to support the design of dynamic documents. From this structure, visual languages are created which support the process of building an *Architecture of Information.* Relational Grammars, an extension to traditional string languages, is the formalism in which these visual languages are constructed. This formal approach affords a number of different interaction techniques, three of which are examined in this research. First, information is automatically presented from predefined languages. This dynamic layout reconfigures the same information accounting for the constraints of different delivery environments. Second, the authoring of information is supported by incremental improvements during the design process. These improvements help the user explore the design space with incremental design decisions. Third, these visual languages are constructed by demonstration. An authoring tool to modify these languages without coding is presented.

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, on January 13, 1995, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Supervisor
Ronald MacNeil
Principal Research Associate, Media Laboratory

This work was performed at the MIT Media Laboratory. Support for this work was provided, in part, by the Alenia Corporation, JNIDS, and News in the Future Consortium. The views expressed within do not necessarily reflect the views of the supporting sponsors.

Doctoral dissertation committee

Thesis Advisor: [illegible]
Ronald MacNeil
Principal Research Associate, Media Laboratory
Massachusetts Institute of Technology

Thesis Reader:
William Mitchell
Dean, School of Architecture and Planning
Massachusetts Institute of Technology

Thesis Reader:
Joe Marks
Research Scientist, Mitsubishi Electric Research Laboratories
Cambridge, Massachusetts

Acknowledgments

Dissertations are about passion. And this dissertation is no different. I found a unique environment in the Visible Language Workshop in which to explore my passion. Co-founded by Muriel Cooper and Ron MacNeil, the VLW has been a focus of design theory and applications for many years. It has been a great environment in which to investigate visual languages to support design.

I was first introduced to Muriel and Ron in '76 as a graduate student in the Architecture Machine Group. Little did I know, 15 years later I would become their first PhD student. It is a great privilege and honor to have pursued my passion in the VLW. Muriel was a strong influence on everyone she met. This research, in part, is a result of that influence. I am very fortunate to have had the opportunity to study with her and I dedicate this work to her.

I would like to thank my committee for their assistance: Ron MacNeil, who continually made me expand my vision instead of narrowing it; Bill Mitchell, who pioneered work in this area and provided keen insight along the way; and Joe Marks, who supplied important guidance and support throughout the process. I would also like to acknowledge Ken Haase, who inspired and encouraged me early on and Henry Lieberman, for his unique perspective on this work.

Kent Wittenburg played a special role in supporting this research. Acting as the unofficial member of my committee, Kent opened up a new world of looking at design as a formal language. He provided the technical support of a computational linguist, extending the parser to incorporate new requirements of the visual languages I produced. He also encouraged and focused the research into new and interesting directions. I am forever grateful.

I would especially like to acknowledge Barry Arons who provided the inspiration for me to return to school. Barry's own PhD effort was a role model that helped me throughout this process.

One of the most rewarding experiences in graduate school is the interaction with other students, and I would like to thank all of those with whom I shared this experience. In particular, Didier Bardon, Maia Engeli, Suguru Ishizaki, Ishantha Lokuge, YinYin Wong, and Xiaoyang Yang who provided useful critiques on both the content and form of the work; and Alan Turransky who provided the first tool to create a VIA grammar rule by demonstration. I would also like to thank members of the extended VLW family, Anil Chakravarthy, Janet Cahn, and Sara Elo, that helped in countless ways.

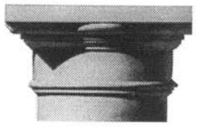
I would also like to acknowledge others who have influenced and supported me, Linda Peterson, for an excellent job, in a sometimes thankless position between student and faculty; Amy Freeman and Nancy Young, for the assistance with making it all happen on a day-to-day basis; Mark Rosenstein, for his friendship, support and comments over the years; Soizick Lesteven, who helped me survive that first semester; Terry Swack, who answered when the universe called; and of course, my parents and family, never questioning my sanity and always there when I needed them.

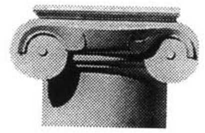
I would like to thank all of the sponsors of this work during my tenure at the VLW including: the Alenia Corporation with Alessandro Muro, the Joint National Intelligence Development Staff (JNIDS) with Captain John Hilbing and the News in the Future Consortium and all of its individual sponsors. I would also like to thank Bellcore's Computer Graphics & Interactive Media Research Group, directed formerly by Jim Hollan and currently by George Furnas for all their support. Jim played an important role in providing a research environment in which I explored early prototypes of design support systems.

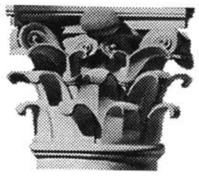
Portions of this dissertation previously appeared in IEEE Symposium on Visual Languages as [Weitzman and Wittenburg, 1993], © 1993 IEEE, and ACM Multimedia'94 as [Weitzman and Wittenburg, 1994], © 1994 ACM, and are reprinted here by permission.

Contents

ar•chi•tec•ture
(ar' ki-tek' cher)
... 4. *Design or orderly arrangement perceived by man:* the architecture of nature.
The American Heritage Dictionary







Introduction

1

The document as information interface

Knowledge is a servant of thought and thought a satellite of feeling. It is for the architect to derive from the very nature of things what a thing wants to be.
LOUIS I. KAHN,
WHAT WILL BE HAS ALWAYS BEEN,
THE WORDS OF LOUIS I. KAHN

Design is in transition. This is most evident in the domain of publishing. No longer can we view publishing as the creation of static printed documents. Instead, "*documents are interfaces, used to access and navigate through collections of information.*" [Haimes, 1994b]. Documents are interfaces that must dynamically reconfigure themselves based on their content, the medium in which they are displayed, and the intended use of the information they present. In addition, the luxury of individually crafting each document is not possible. We can, however, exploit the inherent structure of the information creating an **architecture of information**. This structure can be used to define visual languages which support both the interactive process of design, with a designer in the loop, or autonomous design, when the designer can not be present. This dissertation investigates new theories, tools and techniques to assist in the design process under these new conditions.

Today, the challenge of graphic design is becoming more critical as we move from the printed page to the design of dynamic media. Dynamic, multimedia applications are increasingly part of our daily life, caused by the push of technology's increased power and decreased costs, and the pull from the media companies who are "*selling and reselling as many bits as possible.*"[1.1] Multimedia applications and access to vast quantities of data will soon be possible through the equivalent of one's own television and cable network. The added dimension of the dynamic media introduces new variables that the designer must take into account when creating this new class of documents. The document is transformed into an interface through

1.1 Nicholas Negroponte
WIRED, November, Issue 1.5, 1993, p. 128.

which information is viewed. The distinction between static documents and computer interfaces is vanishing. Information is no longer static and documents will constantly be updated. Interfaces are becoming the primary vehicle for acquiring information. Effective design is the key to making this information accessible and understandable in the digital environment.

In order to improve the effectiveness of information presentation, its structure will be exploited. Throughout history, people have searched for ways to give structure to ideas and concepts, "*to store knowledge in graphic form, and bring order and clarity to information.*"1.2 William Addison Dwiggins coined the phrase *graphic design* in 1922 to describe the process of bringing structural order and visual form to printed communications.

Xerox was one of the first corporations to address this notion of information structure and use the "*elegant and inspirational phraseology, the architecture of information*" to define its new corporate mission. [Smith and Alexander, 1988]. From this vision, Xerox produced an unprecedented set of inventions that have fundamentally changed the way in which we view computing. As described by then Xerox's president, Peter McColough,

> The basic purpose of Xerox Corporation is to find the best means to bring greater order and discipline to information. Thus our fundamental thrust, our common denominator, has evolved toward establishing leadership in what we call the **architecture of information**. What we seek is to think of information itself as a natural and undeveloped environment which can be enclosed and made more habitable for the people who live and work within it.1.3

The power and expressiveness of graphic design is simply illustrated in an example of the packaging design for an Ivory soap bar in Figure 1.1. This illustration shows how carefully crafted design can use the form of the presentation to modify the perception of the information content. Each example evokes quite a different response from the viewer. Unfortunately, we are no longer able to provide the expertise to craft individual presentations in quite the same way. Designers can no longer be present in the production cycle of every document. This suggest tools to *meta-design* descriptions of documents. This activity of meta-design creates descriptions that can subsequently be used to automatically design information under dynamic conditions.

In addition, less qualified people are continually being required to do the design of more and more information. Powerful tools are now in

1.2 Philip Meggs, 1992. *A History of Graphic Design*, Second Edition, Van Nostrand Reinhold.

1.3 XEROX: *Searching for an Architecture of Information*, an address by C. Peter McColough, President, Xerox Corporation, before the New York Society of Security Analysts, March 3, 1970.

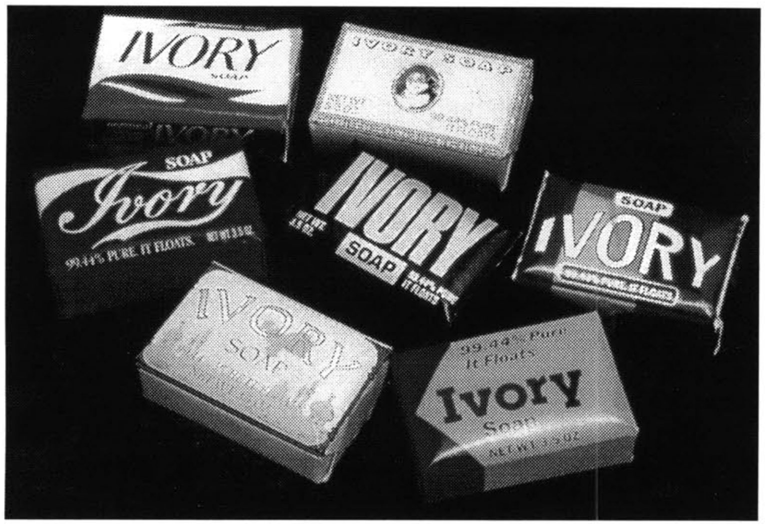


FIGURE 1.1
The power and expressiveness of graphic design to change the viewer's response given the same information content. ***Source, Inc. Chicago, Illinois.***

the hands of graphically untrained users. The expectation is that these users have the knowledge and ability to design and present information as effectively as the professionally trained. This suggests tools for interactive design to aid not only the design professional but also the graphically untrained user.

Publishing on-line

Publishers have acknowledged that they must adapt to the digital world and provide information that is more flexible. Unfortunately, current methods of on-line publishing are limited in their expressiveness, unable to maintain a publisher's identity, or too restrictive as an interface, unable to provide the controls for dynamic interaction. By separating *form* from *content* publishers are beginning to more intelligently adapt their information to new conditions.

Publishers are beginning to be more responsive to the needs of their customers accommodating and focusing on smaller interest groups. For example, R.R. Donnelley, the largest commercial printer, has developed systems to be responsive to their customers. One of their customers went from producing two full-line catalogs a year to almost 200 niche-market specialty catalogs. In the process the production cycle was also reduced from 12 weeks to 7 days. They are able to make immediate shifts of their documents in direct response to their customers market. The term *mass customization* has been applied to the process of producing information that serves the needs and preferences of particular groups or set of individuals. Siegel & Gale, the world's largest design firm, analyzes how the information relates to a company's structure. Through a process they call *document-based reengineering* they create infrastructure to support a company's own inhouse activities as well as their clients needs. [Haimes, 1994a; 1994b].

Existing tools have limited effectiveness as we move towards a digital publishing environment. Currently, there are three basic ways to access and view on-line information [Michalski, 1994]. These include page description languages (PDL), on-line services, and the use of the World Wide Web.

Page description languages, such as Adobe's Acrobat or WordPerfect's Envoy, are very expressive and can produce very high quality layouts. Their power is based on their orientation to page layout. This is also their major drawback. Page description languages do not provide a true interface and lack the flexibility for publishing new media that can be tailored to individual's preferences or needs. The only interaction available is to pan or zoom through a document.

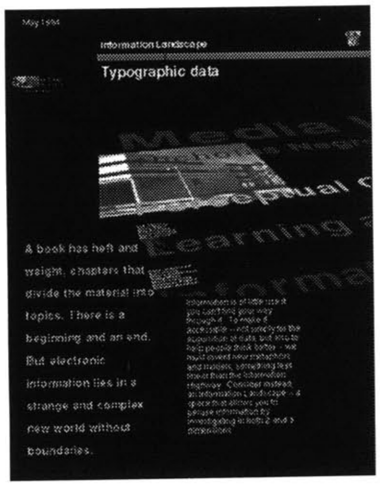
Another technique to access computer-based information is through the use of on-line services, such as *America OnLine* or *Prodigy*. Here, the drawback is that the information provider typically loses all identity during presentation. More advanced systems, such as Ziff/Davis' *Interchange,* due out in early 1995, use templates and rules to present their information with more sophistication. These applications are being extended to handle the more general case of on-line publishing, such as newspapers and magazines, but because of their initial design goals, will find it difficult to extend their capabilities to support the more general problem of expressive graphic design.

Yet another approach to publish and present information is the use of the World Wide Web and the HyperText Markup Language (HTML). Recently, this has been the focus of much attention. Companies, as well as individuals, have been using the Web to create and publish documents that others can browse. The drawback with the Web is that viewers currently do not have the expressiveness needed to control the presentation of information. Attempts are underway to correct this situation. However, these are limited to style sheets, cascading style sheets, etc. as a way to include more layout control. Unfortunately, this will be insufficient when dealing with dynamic content and display situations.

Design needs to become *ubiquitous*. By embedding knowledge of information structure and design constraints, information can maintain a *sense of itself*. This enables information presentation to be adaptive, taking into account the affordances and constraints of the delivery environment. Information sources can then present themselves appropriately under a wide variety of situations.

By exploiting the structure of the information, languages of design can be created. Similar to natural languages, visual languages are composed of primitive elements and rules for their combination. These languages can then drive different aspects of the process ranging from *meta-design*, intended to be used at a later time, to interactive design, immediately assisting user's actions.

This dissertation presents a new approach to the support of document creation where the documents are no longer static elements but dynamic publications that reflect the nature of the information they are conveying. By exploiting the structure of the information, design can be described as a construction in a visual language. The engine to process these visual languages is based on the computational-linguistic formalism of Relational Grammars. Relational Grammars provide a systematic way of describing the structure and meaning of design components. With this formalism, these issues surrounding design and the future of publishing are addressed.



A book has heft and weight, chapters that divide the material into topics. There is a beginning and an end. But electronic information lies in a strange and complex new world without boundaries.

[illegible]

FIGURE 1.2
The presentation of a multimedia presentation, including QuickTime movies, and displayed on a high resolution color display.

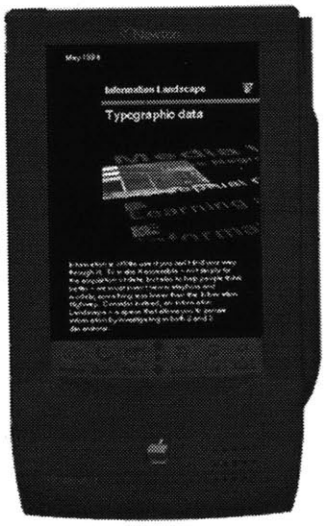


FIGURE 1.3
The same information presented on multiple pages with a personal digital assistant.

VIA: Visual Information Architecture

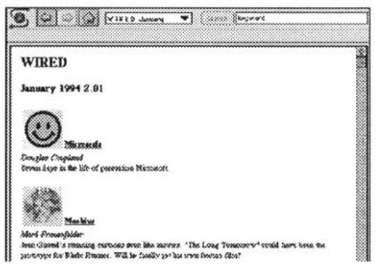
This research explores the **architecture of information** to support its design and presentation. By using the structure intrinsic to the information, visual languages can be built to support and automate the process of design. The fundamental difference in this approach compared to other systems that automate, or otherwise support design, is that it is the structure of the information and not a set of templates or rules that facilitates design. This structure is defined as a formal language which then supports various aspects of the design process. Relational Grammars is the formalism used to describe this class of visual languages and **VIA: Visual Information Architecture** is the prototype environment within which this research has been explored.

In this research, the formalism of Relational Grammars supports the design and presentation of dynamic documents. In contrast to traditional string languages, Relational Grammars handle descriptions of higher order constraints between the elements of the language. This formalism affords a number of different user interactions within the design process including:

- the automatic presentation of information
- the interactive design of information
- the creation of grammars by demonstration

First, information is automatically presented on separate delivery environments and under a variety of conditions. When designers can no longer be in the production loop of a document, they can participate in a meta-design activity. In this activity, designers create visual languages that are subsequently used to automatically present information. In this form of design, predefined grammars process data that has been classified and contains relationships between the elements of input. This process produces a complete articulation of that information given the constraints and affordances in the delivery environment. For example, Figure 1.2 illustrates a presentation on a high resolution color display that incorporates a number of different data types, including QuickTime movies. Alternatively, the same information processed by the same language is presented on a personal digital assistant (PDA), in Figure 1.3. Design tradeoffs between the display environments were made creating different articulations of the same information. On the lower resolution PDA, constraints filter out QuickTime movies and produce a multi-page presentation. This example suggests a scenario where the same information from a technical manual could be presented on a high resolution screen in an office or on a PDA out in the field while doing repairs.

Second, visual languages support the process of interactive design. By *watching* the user during the design process, the system can make suggestions and then improve an artifact under construction within the context of a given language. This language can help guide and direct a designer during the process of document creation. For example, during the authoring process, the system can clean up partial designs and produce a finished, on-line table of contents as shown in Figure 1.4. This scenario enforces a style of design within the predefined visual language. Supporting the interactive design of information includes a number of other scenarios in addition to incremental refinement including top-down expansion of design elements, continuation or completion of partial designs, verification of partial or complete designs, and zooming based on the derived structure of the information.



WIRED

January 1994 2.01

Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft

Moebius
Mark Frauenfelder
Jean Giraud's stunning cartoons look like movies. "The Long Tomorrow" could have been the prototype for Blade Runner. Will he finally get his own feature film?

FIGURE 1.4
An online table of contents authored within a style that recognizes by the constraints of the viewing environment.

Finally, tools assist designers in the creation and modification of these languages by demonstration. Designers must be supported in building these dynamic languages without having to write code. Working within the interactive scenario, Relational Grammars also assist in the creation and modification of design rules by example. These rules can then be used in the design of automatic presentations or interactively supporting other design activities. To illustrate this capability, a rule editor has been constructed to explore the creation of visual languages by demonstration. For example, the original table of contents of a popular magazine, Figure 1.5, can be redesigned by modifying the underlying grammar rule that creates individual article entries, Figure 1.6.

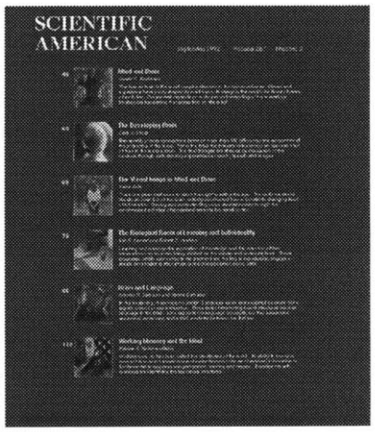


FIGURE 1.5
The table of contents of Scientific American automatically presented in a visual language that articulates its presentation style.

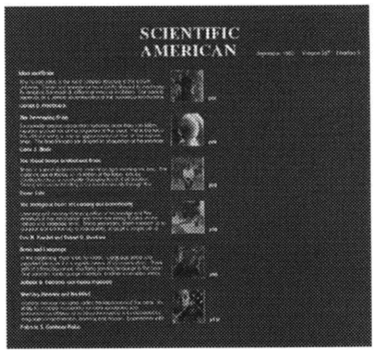


FIGURE 1.6
The same content as in Figure 1.5 after modifying the definition of the rule that forms individual article groups.

Summary

By exploiting the architecture of information, visual languages can be used to support different aspects of the design process. This dissertation describes visual languages to support the automatic presentation of information, the interactive design of information, and tools for the creation of visual languages by demonstration. This document continues with a brief description of the problems and opportunities that have motivated it. A description of the approach taken by this research with its underlying assumptions is then presented in Chapter 3. For those unfamiliar with computational-linguistics and language-based systems, this chapter provides a brief overview.

After this brief introduction, three chapters describing the different applications of visual languages are presented. These chapters can be read independently, but are meant to compliment one another. Chapter 4 describes the automatic presentation of information. This description begins with a simple static example and then presents more dynamic documents that have been automatically constructed. An appendix illustrates the complete grammar used in the main example of this chapter. The interactive design of information, in chapter 5, illustrates a number of examples of how grammars can be integrated into the toplevel design loop. Finally, chapter 6 illustrates rule acquisition by demonstration. This provides an important tool for designers enabling them to create and modify these visual languages interactively. In each chapter, partial grammars are presented with the unique requirements these separate approaches place on the role of visual parsing. In order to place this work in the proper context, related research that has influenced each approach is discussed within that chapter. The dissertation concludes with a summary including the future research suggested by this work and a brief description of the implementation.

Problems and opportunities

2

Introduction

In each period of our history, design and communication have evolved synchronously with the technology of the time. Each new medium has extended our sense of reality and each has looked to its predecessor for language and conventions, referencing and adapting its characteristics until its unique capabilities can be explored and codified.
Muriel Cooper,
Design Quarterly, 1989

This chapter highlights a number of problems and opportunities facing designers in the digital world. These problems and opportunities arise from the information-rich, networked environment in which we now create and distribute information. The presentation of this information places new constraints on design. Given the dynamic nature of the digital environment, the rate of information production is already overwhelming. Information is constantly changing and people have the ability to view information on different devices and for different uses. In addition, designers will not be able to be in the creative loop of this production cycle and untrained users are beginning to publish their own information. Users of these powerful publishing tools need support in the structuring and creation of their presentations.

However, in the digital environment there are a number of new opportunities available to us. We can begin to personalize information and customize its presentation to increase its relevance. The theories and techniques described in this dissertation suggest a new type of design. Design knowledge embedded in the information can support the effective presentation of itself in a variety of situations. Design will become ubiquitous, where information will have *a sense of itself* and be able to reconfigure its presentation in response to its environment.

Information explosion

With ever increasing frequency, we are being overwhelmed with information. Richard Saul Wurman describes the so-called information age as really "an explosion of non-information; it is an explosion of data."[2.1] More information has been produced in the last 30 years than in the previous 5,000 years and the total of all printed knowledge doubles every 8 years.[2.2] About 1000 books are published internationally every day and there are over 9600 different periodicals published in the U.S. every year.[2.3] Wurman's book describes the anxiety we all experience when we have too many facts, yet not enough information. His remedy has been to organize information to make it accessible. For example, he has published a series of travel guides organized by location so travellers are presented the information in a more relevant and comprehensible form. Albert Gore uses the term *exformation* to describe the phenomena of accumulated surpluses of data whose existence is known but is outside of our conscious awareness [Gore, 1991]. Gore states:

We have automated the process of gathering information without enhancing our ability to absorb its meaning.[2.4]

Associated with this information explosion is the growing connectivity of networked communication and information access. As newer, faster data networks are constructed, the ability to gain access to even more data increases at a greater pace. Gore supports the national data super-highways. He continues:

New technologies that enhance the ability to create and understand information have always led to dramatic changes in civilization. Now come distributed networks connecting a myriad of computers ... There is no longer any doubt that such machines will reshape human civilization. ...[2.5]

Meta-design

Publishing and the consumption of information will occur at an ever increasing pace in this new environment. The traditional approach to making information understandable is to employ a designer or graphic artist. With increased regularity, involving a designer may not be desirable or even practical. Information in this environment will no longer have the luxury of being pre-designed. Either it will be left in a raw format (as it is today), or new techniques will be employed to *meta-design* descriptions that can be used to automatically present

2.1 Richard Saul Wurman, 1990. *Information Anxiety*, p. 38.

2.2 Peter Lange, 1984. *Micro Revolution Revisited*, Rowman & Allanheld Co., New Jersey.

2.3 Wurman, 1990. *Information Anxiety*, p. 34.

2.4 Albert Gore, 1991. Infrastructure for the Global Village, *Scientific American*, September, p. 150.

2.5 ibid, pp. 150-153.

the information at a later time. Meta-design is a process in which design professionals can create flexible descriptions of designs. These descriptions will be capable of responding to the dynamics of the information, environment and intended use. They will be more flexible than a set of templates and rules and will support information presentation when information is missing, when the information is presented in different mediums, and when it is utilized for different purposes.

Information dynamics

Our computational environments are the most plastic, modifiable tools ever known. In today's dynamic computational environments, not only can the content of our information change, but so can the structure of its presentation. Computer-based documents have the ability to respond differently based on information, users, and environment. As shown in Figure 2.1, Ishizaki [Ishizaki, 1993] categorizes the dynamics of information display on three separate axes: *information characteristics*, *user interaction*, and *media presentation*. The origin represents static states of each axis while the shaded region corresponds to most dynamic state. For example, printed phone books are located near the origin, i.e., they are not interactive and the information available is static. Interactive multimedia training manuals, however, are located in the shaded region exhibiting dynamics along all dimensions. Authors must now be concerned with not only the spatial characteristics but also the temporal nature of their documents, including: animations, simulations, and the non-linear flow of interactive hypermedia links.

Information is dynamic and evolving, changing its relevance and reliability. Presentations have the potential to reflect the dynamic state of an underlying computation, simulation, or real-time event. This has become most apparent in simulation-based training systems and the scientific visualization of information. Supporting dynamic user interaction also places constraints on the way in which the information can be displayed. Information can be used by different people interested in different parts of the data or for different tasks. The same information can also be displayed on different output devices at different times. Each of these output devices might support different information display characteristics, such as size, color and dynamic (e.g., QuickTime movies) capabilities.

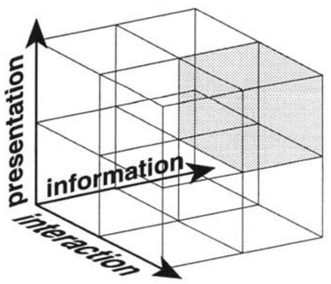


FIGURE 2.1
3D representation of dynamics in a computational environment highlighting the axes of 1) information characteristics, 2) user interaction, and 3) media presentation. The origin represents the most static position while the grayed region represents the most dynamic portion of the graph.

Personalized information

Along with this new power of information access and dynamics, we now have the capability in our computing environments to personalize information. It is very frustrating to receive a general answer to a specific question. For example, when inquiring about directions to an L.A. restaurant in a car, it may be accurate to provide directions that include information using the freeways. However, it may be inappropriate when you will be taking public transportation, a bicycle, or walking.

As mentioned previously, publishers are moving towards *mass customization* of information. By being able to respond to the needs of their customers, publishers are beginning to target smaller and smaller audiences. In the process, publishers can provide more and more relevant information. This demands a more flexible and agile publishing process in order to respond to the market forces of their readership. New techniques and tools to support this process are necessary.

Design excellence

We are beginning to harness our computational power as indicated by the success of the desktop-publishing industry. As a result, however, another problem begins to emerge. Many systems today expect, even encourage, users to make the basic graphic-design decisions of the documents they create. We have provided the tools without requiring the basic skills to control the presentation of the information. In his article on document design and desktop publishing, Jon Bentley makes fun of the misuse of powerful tools by abusing the typeface in this quotation:

"**powerful tools** can sometimes be *powerfully* ABUSED."[2.6]

Just because we have the capability doesn't mean that we should use it. Users need to understand how to apply these capabilities more effectively. This places enormous responsibility for a crucial system component of making information understandable and accessible in untrained hands. We should not expect typical users to be design experts. The results are often ineffective solutions for the presentation of information. In order to raise the quality of design, we need to begin to provide design knowledge embedded in the tools we deliver.

2.6 Jon Bentley, 1986. Document Design, Programming Pearls, Communications of the ACM, Vol. 29, No.9. **Abused** typeface adapted from original.

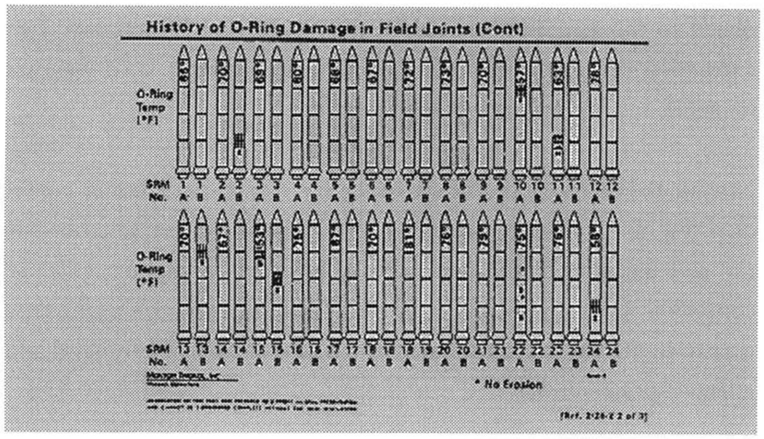


FIGURE 2.2
Chartjunk hiding the incidents of O-ring damage in a presentation of individual rocket boosters. This data was presented before the launch.[2.7]

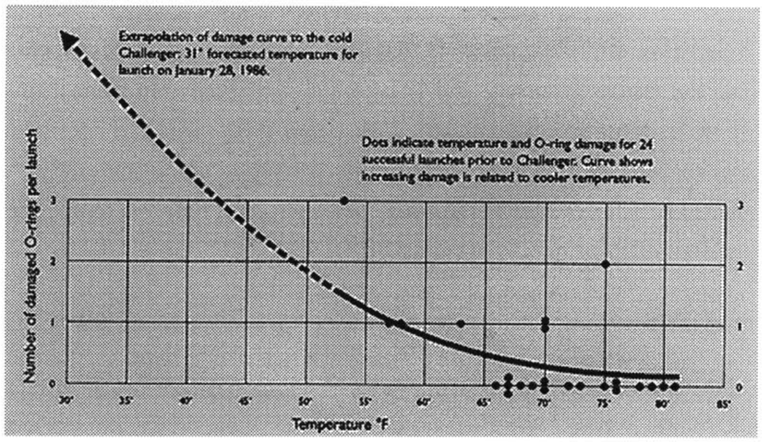


FIGURE 2.3
Post-accident analysis plots temperature vs incidents of O-ring damage extrapolating the curve to the forecasted 31 degree temperature of launch day.[2.8]

These tools can support the design and presentation of information for everyday users.

But, is good information design really necessary? Tufte suggests that it is "*important stuff*" and uses an example to illustrate the serious nature of design. Tufte [Tufte, 1992] presents an example from the disastrous January 28, 1986 cold-weather launch of the Challenger space shuttle. He suggests that if the information had been properly presented this incident could have been avoided. Figure 2.2 illustrates how the data was presented before the launch. In Figure 2.2 each of the 24 launches is presented as 2 booster rockets. The temperature at launch time and marks indicating O-ring damage are both indicated on the boosters. It is difficult to identify the relationship between temperature and O-ring damage. This presentation obscures the real meaning of the data. Tufte classifies this obscuring of information as *chartjunk*. Figure 2.3 presents the same data but instead plots the number of O-ring incidents against temperature. The curve is extrap-

2.7 Report on the Presidential Commission on the Space Shuttle Challenger Accident (Washington, D.C. 1986). Vol V, pages 895-896 ref. 2/26-2 2 of 3.

2.8 Edward Tufte, 1991. Visual Explanations, manuscript.

olated to the forecasted 31-degree temperature of launch day. This extrapolation illustrates the correlation between cold-weather launches and the history of O-ring damage.

Not all redesigns are so dramatic or have such devastating consequences, but bad design can clearly cost time and effort, and obscure the real meaning of the information. In order to support effective communication of information, a structured organization of the data is important. This structure can then be used to build meaningful and effective presentations.

Ubiquitous design

Weiser [Weiser, 1991] suggests that only when technology disappears into the fabric of society and becomes indistinguishable from it, do we see its most profound effects. *Ubiquitous computing* is the term he uses to describe this effect when applied to the computing environment. He believes that ubiquitous computing will help overcome the problem of information overload.

There is more information available at our fingertips during a walk in the woods than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human environment instead of forcing humans to enter theirs will make using a computer as refreshing as taking a walk in the woods.[2.9]

In order to make ubiquitous computing a realization, we will also need to support *ubiquitous design*, design of the presentation of information that is embodied in the tools and devices with which we interact. This will facilitate the flow of information and not just data.

Summary

Design and publishing are going through significant changes and information presentation must be flexible to accommodate these changes. As design moves from traditional media into the electronic studio, we need to provide more support for design by users of all levels of expertise in different disciplines. Applications that deal with the design process should support and liberate the user to more easily explore creative design solutions. Computational tools should support users in principled design with on-line assistance as well as automated design techniques. In addition, designers need to participate in

2.9 Mark Weiser, 1991. The Computer for the 21st Century, Scientific American, September, p. 94-104.

a meta-design process, so that they do not have to be available during the delivery of on-line documents. The tools and techniques described in this dissertation support this notion of agile publishing and dynamic presentation of information.

Approach

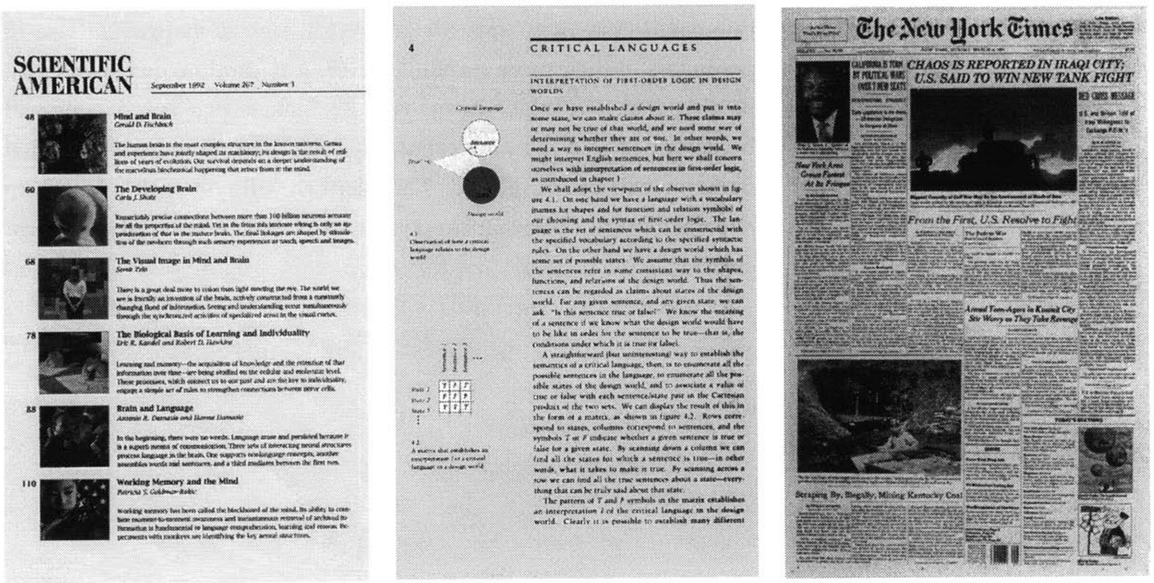
3

Introduction

The greatest crisis facing modern civilization is going to be how to transform information into structured knowledge.
CARLOS FUENTES

This dissertation applies technology that has been developed in the area of computational linguistics and rule-based systems to support the process of design. The main assumption is that graphic design, and design in general, can utilize the structure in the domain to construct a visual language to support the creation process. Figure 3.1 shows a typical page from three sources: a magazine table of contents, a book chapter page, and a newspaper front page. Each of these examples conforms to a set of rules and stylistic guidelines. By encoding layout knowledge in the form of grammar rules, the system can support different user interactions in the creation of dynamic documents. One can identify and represent the primitive elements and the rules for combination necessary to support a systematic construction

FIGURE 3.1
Formats of information presentation for different media: a magazine table of contents, a chapter page from a book, and a newspaper front page, each illustrating a specific layout and style.



SCIENTIFIC AMERICAN September 1992 Volume 267 Number 3

48 **Mind and Brain**
Gerald D. Fischbach
The human brain is the most complex structure in the known universe. Genes and experience have jointly shaped its machinery; its design is the result of millions of years of evolution. Our survival depends on a deeper understanding of the marvelous biochemical happening that arises from it: the mind.

60 **The Developing Brain**
Carla J. Shatz
Remarkably precise connections between more than 100 billion neurons account for all the properties of the mind. Yet in the fetus this intricate wiring is only an approximation of that in the mature brain. The final linkages are shaped by stimulation of the newborn through such sensory experiences as touch, speech and images.

68 **The Visual Image in Mind and Brain**
Semir Zeki
There is a great deal more to vision than light meeting the eye. The world we see is literally an invention of the brain, actively constructed from a constantly changing flood of information. Seeing and understanding occur simultaneously through the synchronized activities of specialized areas in the visual cortex.

78 **The Biological Basis of Learning and Individuality**
Eric R. Kandel and Robert D. Hawkins
Learning and memory—the acquisition of knowledge and the retention of that information over time—are being studied on the cellular and molecular level. These processes, which connect us to our past and are the key to individuality, engage a simple set of rules to strengthen connections between nerve cells.

88 **Brain and Language**
Antonio R. Damasio and Hanna Damasio
In the beginning, there were no words. Language arose and persisted because it is a superb means of communication. Three sets of interacting neural structures process language in the brain. One supports nonlanguage concepts, another assembles words and sentences, and a third mediates between the first two.

110 **Working Memory and the Mind**
Patricia S. Goldman-Rakic
Working memory has been called the blackboard of the mind. Its ability to combine moment-to-moment awareness and instantaneous retrieval of archived information is fundamental to language comprehension, learning and reason. Experiments with monkeys are identifying the key neural structures.

4 CRITICAL LANGUAGES

INTERPRETATION OF FIRST-ORDER LOGIC IN DESIGN WORLDS

Once we have established a design world and put it into some state, we can make claims about it. These claims may or may not be true of that world, and we need some way of determining whether they are or not. In other words, we need a way to interpret sentences in the design world. We might interpret English sentences, but here we shall concern ourselves with interpretation of sentences in first-order logic, as introduced in chapter 3.

We shall adopt the viewpoint of the observer shown in figure 4.1. On one hand we have a language with a vocabulary (names for shapes and for function and relation symbols) of our choosing and the syntax of first-order logic. The language is the set of sentences which can be constructed with the specified vocabulary according to the specified syntactic rules. On the other hand we have a design world which has some set of possible states. We assume that the symbols of the sentences refer in some consistent way to the shapes, functions, and relations of the design world. Thus the sentences can be regarded as claims about states of the design world. For any given sentence, and any given state, we can ask, "Is this sentence true or false?" We know the meaning of a sentence if we know what the design world would have to be like in order for the sentence to be true—that is, the conditions under which it is true (or false).

A straightforward (but uninteresting) way to establish the semantics of a critical language, then, is to enumerate all the possible sentences in the language, to enumerate all the possible states of the design world, and to associate a value of true or false with each sentence/state pair in the Cartesian product of the two sets. We can display the result of this in the form of a matrix, as shown in figure 4.2. Rows correspond to states, columns correspond to sentences, and the symbols *T* or *F* indicate whether a given sentence is true or false for a given state. By scanning down a column we can find all the states for which a sentence is true—in other words, what it takes to make it true. By scanning across a row we can find all the true sentences about a state—everything that can be truly said about that state.

The pattern of *T* and *F* symbols in the matrix establishes an interpretation *I* of the critical language in the design world. Clearly it is possible to establish many different

4.1
Observation of how a critical language relates to the design world

4.2
A matrix that establishes an interpretation *I* of a critical language in a design world

The New York Times

CALIFORNIA IS TORN BY POLITICAL WARS OVER 7 NEW SEATS

CHAOS IS REPORTED IN IRAQI CITY; U.S. SAID TO WIN NEW TANK FIGHT

RED CROSS MESSAGE

New York Area Greets Fastest At Its Fringes

From the First, U.S. Resolve to Fight

The Bedrock War

Armed Teen-Agers in Kuwait City Stir Worry as They Take Revenge

Scraping By, Illegally, Mining Kentucky Coal and presentation of these documents. For example, the grid is a standard organizing device and each layout uses a different grid for its arrangement. Some of these layout rules are generic and transcend single design layouts (e.g., equal sizing and alignment of similar elements), while others are specific to a particular design (e.g., vertical indentation for a particular layout).

Recently, work by [Mackinlay, 1986; Feiner, 1988; Marks, 1991; Kalay and Majkowski, 1987] and others have illustrated that a linguistic representation can be effectively used to support computer-based design and layout. Much of this work has been motivated by the desire for automatic layout tools for different domains. Other systems have applied a grammar-based approach to support the design process by interpreting the input of the user and suggesting improvements and alternatives [Lakin, 1986; Kochhar and Friedell, 1990].

A characterization of design systems, based on [Kochhar, et al., 1991], will be used to examine the interaction paradigms afforded by these systems. This characterization identifies different levels of user interaction, ranging from: completely manual, to constraint-based, to critic-based, to improver-based, to cooperative, and finally, to fully automatic. Relational Grammars enable interactions over many of these levels particularly, the automatic and improver-based approaches. This investigation is explored within a prototype environment, *VIA, Visual Information Architecture*, which embodies these theories and techniques.

As background to this thesis, the tradition of structuralism applied to design will be introduced. This approach has produced a number of rule-based systems to support different design domains. The use of grammars in their more traditional role of natural language will also be discussed. The requirements visual languages place on parsing will be described and the application of Relational Grammars to visual domains will be introduced. This chapter will end with a vision of document processing that supports a more agile process of document creation.

Structuralism and design

In the late 1800's, Ferdinand de Saussure proposed a new analytical view of language in terms of its structure. This branch of linguistics, *semiology*, is a general theory of signs which includes verbal as well as non-verbal systems. This structure is based on the definition of elements of the language, the relations between them, and their rules of combination. At the center of this study of language was the linguistic "*sign*." Saussure defined the linguistic sign as the combination of the representation of the token, for example the noise (i.e., the signifier) in a verbal system and the idea or object it signifies (i.e., the signified). The fundamental feature of the linguistic sign is that it is arbitrary in relation to the concept it represents. The interpretation of the sign is made only by contrast with other signs. Together they form the structural system of the language. Saussure was concerned with the difference between verbal language and the modernist's ideal of a visual language.

Search for languages of design

In contrast, modern design theory has searched for a system of signs which is universal, based on biological models of perception. The Bauhaus was a focus of design thinking in the early part of this century and was a leader in this search. The explorations into identifying a *language of vision* are part of its legacy. This language was described as being analogous to, but different and isolated from, verbal languages. Visual form was seen as "a universal and transhistorical script, speaking directly to the mechanics of the eye and brain."[3.1]

Lupton points out that Paul Klee's *Pedagogical Sketchbook* [Klee, 1953] and Wassily Kandinsky's *Point and Line to Plane* [Kandinsky, 1979] are primers for the grammar of visual languages. In Kandinsky's Point and Line to Plane, he describes a universal dictionary that would form the basis of a visual language:

The progress won through systematic work will create a dictionary which, in its further development, will lead to a 'grammar' and, finally, to a theory of composition that will pass beyond the boundaries of the individual art expressions and become applicable to 'Art' as a whole.[3.2]

Following in this tradition, Gyorgy Kepes' *Language of Vision* [Kepes, 1961] and Laszlo Moholy-Nagy's *Vision in Motion* [Moholy-Nagy, 1947] were published at the School of Design in Chicago. They use Gestalt psychology to give a scientific rationale to the *language of vision*. In addition, Karl Gerstner wrote *Designing Programmes*

3.1 Lupton, E. The ABC's of ▲■●: The Bauhaus and Design Theory, 1993, p22.

3.2 Kandinsky, W. Point and Line to Plane, 1979.

[Gerstner, 1968] which explores the structure of design as programmed systems and resultant processes rather than as a unique product. All of these explorations were ways of thinking about design in a structured way. They have laid the foundation of structured design thinking that continues today in computer-based systems. In his book, Digital Mantras, Holtzman captures the essence of structuralism.

The goal of structuralism is to formalize and make explicit these underlying rules and conventions.[3.3]

He presents a number of examples that take a structured approach to design in a wide variety of disciplines including music, art and virtual worlds.

Characterization of information

In pursuit of this structuralist approach to design support, system builders have limited the scope of the problem. A subset of the more general graphic design problem, information presentation, has been a focus for much research. By limiting the scope, a more basic understanding of the general design problem begins to emerge. In order to support the presentation of information, both automatically and interactively, researchers use a characterization of data types. This section describes the classic literature of information characterization and the mapping of this information into visual marks in a presentation. Sources for extending this characterization into dynamic data types are also discussed.

Norman's appropriateness principle emphasizes the care needed in choosing the right mapping from data to presentation technique:

The surface representation used by the artifact should allow the person to work with exactly the information acceptable to the task: neither more nor less.[3.4]

Typical characterizations of information include the differentiation between *nominal*, *ordinal*, and *quantitative* data. Nominal information describes unordered categories, such as names of elements. Ordinal information describes ordered categories such as days of the week. Ordered data is further characterized by *coordinates*, e.g., points in time or space, and *amounts*, e.g., weights or costs. Quantitative information describes numerical data and is further characterized as *continuous* and *discrete*.

A similar categorization of information is suggested by Wurman

3.3 Holtzman, S. Digital Mantras 1994, p. 50.

3.4 Norman, E. Cognitive Artifacts, 1991.

[Wurman, 1990]. He believes that there are only five strategies for organizing information which apply to almost all endeavors. These are *category*, *time*, *location*, *alphabet* or *continuum*. These principles of organization elicit new levels of understanding by taking different perspectives on the same information.

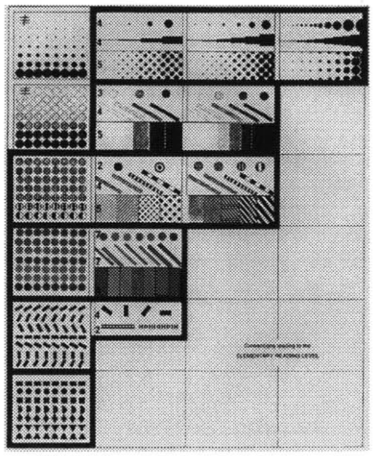


FIGURE 3.2
Bertin's categorization of data using the graphic attributes of size, value, texture, color, orientation and shape.

Jacques Bertin [Bertin, 1983] defines a methodology for the display of quantitative information based on the characteristics of the information and the ability to encode different types of data with different graphic techniques. The graphic techniques, or *variables*, he identifies include x and y dimensions, size, value, texture, color, orientation and shape. The main task of encoding data in his theory is the mapping of information type to the proper graphic variable. Certain graphic variables are better at encoding certain types of data. In fact, certain graphic variables are not appropriate at all for particular types of data as illustrated by Figure 3.2. Bertin applies the notion of *efficiency* to rate the success of a particular visualization. This is Zipf's metric of *mental cost* applied to the visual domain [Zipf, 1935].

Mackinlay [Mackinlay, 1986] was the first to use this simple characterization to map from relational data to different types of bar charts. He used the terms *expressiveness* to identify whether a particular graphic technique could express a certain type of data and *effectiveness* to describe how well that information was presented in that technique. The environments in which the information will be displayed have also been classified. [Marks, 1991; Roth and Mattis, 1990] have illustrated the effect of various characterizations of data applied to the automatic presentation of designs. Kanarick [Kanarick, 1992] extended these categories to include temporal and locative subdivisons within the ordinal (e.g., days of the week) and quantitative (e.g., hours of the day) data. Comprehensive sets of data characterization can be found in [Arens, 1993; Bertin, 1983; Norman, 1991; Roth and Mattis, 1990; Kosslyn, 1989,1994; and Senay, 1991]. This dissertation incorporates these principles implicitly. More importantly, it provides a formalism in which they can be applied to support design.

Characterization of temporal information

The extension of information categorization into temporal data is beginning to evolve. As systems concern themselves more and more with dynamic design (e.g., CD-ROMs, Interactive games, etc), this classification will become very important. Insights into the encoding of dynamic information can be found in sources such as psychology [Kosslyn, 1989; Norman, 1991], film [Bordwell and Thompson,

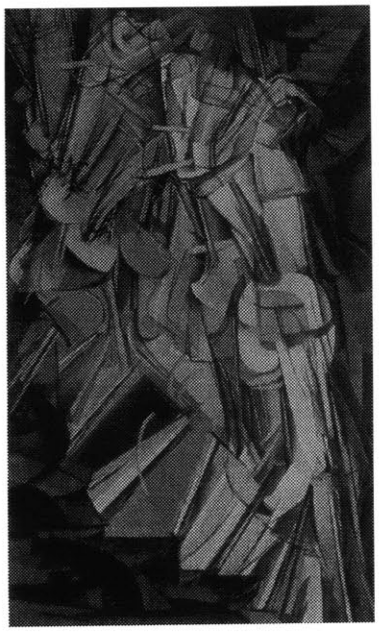


FIGURE 3.3
Nude descending a staircase by Marcel Duchamp depicting motion as repeated overlapping forms.

1990], animation [Thomas and Johnston, 1981], sequential art [Eisner, 1989; McCloud, 1993], photography [Braun, 1992], art [Gerstner, 1968; Kepes, 1961; Klee, 1953; Moholy-Nagy, 1947] and graphic design [Dondis, 1973; Bertin, 1983; Hiebert, 1992; Ishizaki, 1993; Tufte, 1983, 1990].

At the turn of the century when technology was changing the fabric of our society,

A truly modern way of depicting the new experience of space and time was called for, one that could reflect the sensation of simultaneity, speed, and dynamism that engulfed Western consciousness.[3.5]

Marey and Muybridge used chronophotography to make visible the underlying laws of nature. They thought that their images would assist artists in depicting the real world more accurately. In fact, their images became one of the key visual sources for this aesthetic. But the artists that incorporated their analytic decompositions of time, space and motion created another reality with these images. Today, we are most familiar with this body of work through the application the images by early modern artists, such as Marcel Duchamp's *Nude Descending a Staircase* in 1912 (Figure 3.3). The repeated overlapping forms remain as one of the most influential sources for representing time, speed and motion. Their work transformed the forms of chronophotography and made them into a visual language and convention to represent the dynamic sensation of time.

One technique to present temporal information is the use of *small multiples*. This storyboard technique encourages the viewer to compare information between multiple instances of the data over time. By examining traces, discrete or overlaid, the viewer can compare data to achieve a better comprehension of the information. Edward Tufte characterizes a worthwhile graphic technique as one that induces the viewer to think about the substance rather than the form, encouraging the eye to compare different pieces of data and revealing the data at several levels of detail, while avoiding distortion. [Tufte, 1983]. Today, many systems have the added capability of three-dimensional displays to dynamically simulate an event or process over time.

Film is another source for the characterization of temporal information. Time can be manipulated to evoke different responses by making the viewer actively participate in making sense of the narrative of the film. Manipulation of *order*, *duration* and *frequency* are three specific temporal factors that can be controlled in the language of

3.5 Picturing Time, The work of Etienne-Jules Marey (1830-1904), Marta Braun, University of Chicago Press, 1992. p264.

film [Bordwell and Thompson, 1990]. Temporal order includes techniques such as flashbacks. There are two types of film duration, the actual duration of events and the screen duration. Emphasis can be created by expanding screen time relative to an event that only takes a short duration. Screen duration can also be used to compress story time.

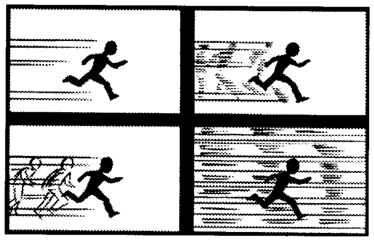


FIGURE 3.4
Techniques for the depiction of motion within a single frame of a comic. From McCloud, 1993.

Another source for the representation of time is sequential art, i.e., comics. Scott McCloud describes the interaction and depiction of time in comics. Generally, it leads to one of two motifs, sound or motion. Though comics existed for many centuries without depicting time, its incorporation into the language of comics was inevitable. Initially, this centered around the use of multiple images in sequence. More and more sophisticated mechanisms have evolved and have been explored in the language of sequential art.

In learning to read comics, we all learned to perceive time spatially, for in the world of comics, time and space are one and the same. 3.6

The depiction of sound in a static, print medium adds duration to a panel partially because of the nature of sound, but also as a result of the actions and reactions of the characters in the panel. Sound can be graphically represented by balloons or sound effects. Motion can be illustrated in the closure from panel to panel or through a number of graphic techniques within a single panel. Motion lines, multiple images, photo blurring of the subject or blurring of the background are just some of the techniques used today (Figure 3.4).

Examination and analysis of the nature of dynamics and temporal information from other domains can better support the design process. Affordances and constraints of the information and delivery environments can then be utilized to make mappings of the information into more effective presentations. These techniques are becoming more relevant to creating systems that can support multimedia authoring tools. With a better understanding of temporal data, languages of design will emerge that incorporate the expressiveness and effectiveness criteria to form the basis of a new class of multimedia design systems.

3.6 McCloud, S. 1993, p100.

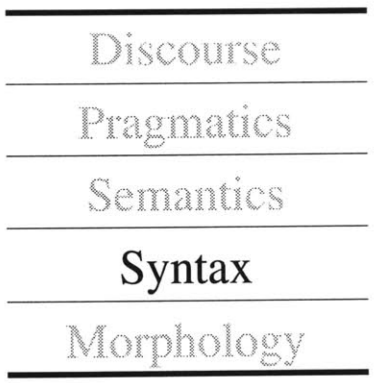


FIGURE 3.5
Layers of processing for natural language understanding and generation. Relational Grammars are concerned with the grammars and parsing algorithms at the syntax level.

Traditional natural language processing

The study of natural language has traditionally been divided into two basic categories, syntax and semantics. More specifically, however, the process of understanding and generating natural language includes a number of interconnected layers of processing. Typically these are divided into morphology, syntax, semantics, pragmatics, and discourse (Figure 3.5). Morphology, or word structure, is included in the broad definition of syntax, while pragmatics and discourse are part of the general category of semantics. Pragmatics deals with context and intentions of the speakers, while discourse is related to the use of multiple utterances and how they support the establishment of context for better understanding. This research focuses on the aspects of syntax and syntax directed parsing to support design.

Syntax

Work on syntactic models of natural language processing have drawn upon research in both computer science, which is interested in efficiency and search, and linguistic theory, which is interested in the formalisms that can be used to characterize the grammatical strings of natural language. The syntax of natural language is concerned with two interrelated concepts, *grammars* and *parsing algorithms*. A grammar is a finite specification of a possibly infinite set of languages that systematically captures the regularities of the language. A grammar can be used to generate sentences in the language. It can also be used to determine whether a given input string belongs to the language and identify the structure according to the grammar. Parsing algorithms, on the other hand, specify how to apply a grammar to a sequential input string to produce a structured representation, or parse tree. In general, a parser includes both the grammar and the parsing algorithm.

A grammar is composed of one or more rules that typically map multiple symbols to a new symbol. In natural language, for instance, a sentence (S) is formed from a noun phrase (NP) and a verb phrase (VP). This is captured in the rule:

S → NP VP

The left-hand side of the rule is the composite formed by rule application. In this case, a sentence (S) is formed. This is sometimes referred to as the *mother* or *parent* of the rule. The right-hand side of the rule is a list of the input necessary for the rule to apply. In this case, the grammar indicates a noun phrase (NP) preceding a verb phrase (VP) are necessary. These are sometimes referred to as the *daughters* or *children* of the rule.

Parsing

Parsing is the process of determining the structure of the sentence being parsed. Using a grammar that describes the structure of strings in a particular language, a parser assigns a structure to a grammatical sentence. This structure is called a *parse tree*. The parse tree corresponds to the order in which the rules within the grammar can be applied to transform a *start symbol*, or goal state, into the final structure. The start symbol of the simple language above is S, indicating that a sentence is the goal of the parsing process.

Top-down processing begins with the start symbol and applies the grammar rules forward until the symbols at the terminals of the tree correspond to the components of the sentence being parsed. This is *goal-directed* search. On the other hand, *bottom-up* parsers start with the sentence being parsed and apply the rules backward to build a parse tree whose terminals are the words of the sentence and whose top node is the start symbol of the grammar. This search through the space of alternatives is *data-driven* and builds successive layers of syntactic abstractions.

Semantics

Semantic interpretation is the process of mapping natural language utterances onto some representation of the world, or onto a model of the real or an imaginary world. Conventionally, semantics is about the truth or satisfaction conditions of a particular utterance, while pragmatics deal with the context and the intentions of the speakers. According to the principle of compositionality, put forth by the philosopher Frege, the meaning of a sentence can be expressed in terms of the meanings of its parts. The rule-to-rule hypothesis provides a framework in which syntactic and semantic rules are matched, (i.e., each syntactic rule has a semantics component). Taken together this means that semantics can be formed by taking the semantics of the rule that generated the tree and applying it (as a function) to the semantics of the constituents of the tree.

Ambiguity

Ambiguity is a major part of natural language expressions. In processing natural language one of the main goals is the reduction of this ambiguity. Ambiguity can take on three different forms. There is *lex-*

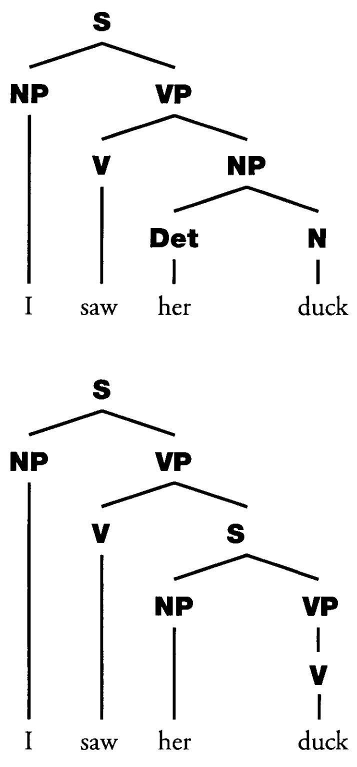


FIGURE 3.6
Lexical and structural ambiguity in natural language. From Gazdar and Mellish, 1989.

ical ambiguity, where words or phrases in one domain can express multiple senses; *structural ambiguity*, where a phrase can be analyzed such that the derivations are completely different; and *scope ambiguity*, where there may be more than one way of reading which predicates, operators and descriptions fall within the scope of negatives, conjunctions and disjunctions and intensional operators. For example, the classic sentence, "I saw her duck" in Figure 3.6 illustrates both lexical and structural ambiguity. In addition, an utterance can be underdetermined with respect to information that has not been specified explicitly. Programming languages are good examples of languages that avoid ambiguity because there is little interaction among the components of input. In natural languages, however, changing a single word can change the entire structure of the sentence.

With complex grammars and more involved input, parsers slow down significantly. Repeating work due to ambiguity is one of the most common causes of these slowdowns. To address this problem, *tabular* or *chart parsing* is used to eliminate redundant parsing of constituents in ambiguous derivations. One common parsing algorithm that forms the basis of a number of different research efforts is the Earley algorithm [Earley, 1970] that combines both top-down and bottom-up processing (sometimes called "*prediction*" and "*scanning*," respectively). It uses a chart or table to explicitly record partial parses that can be reused in constructing subsequent derivations.

Ambiguity in traditional natural language processing is problematic, producing slow and potentially inaccurate results. In design, however, the notion of ambiguity takes on another role. In the beginning stages of design, there are many options and the decision making process needs to remain flexible. This form of ambiguity can actually contribute to a better understanding of the problem and to more appropriate design solutions. The ability of parsers to maintain this ambiguity and use it to support subsequent interpretation is an big advantage to supporting the design process.

Generation

The basic task of language generation is to produce a valid string in the target language given a parse tree. Determining what words and phrases to use in describing the structured representations is one of the basic decisions. Language generation is similar to language understanding in that the initial work concentrated on conversion of isolated sentences. However, growing interest in discourse and pragmatics has led increasingly to the development of systems that deal with multi-sentence text production. In fact, the characterization of multimedia generation is similar to this expanded notion of language generation. Both of these processes include: 1) *content selection*, constructing a structure that represents the information to be communicated; 2) *media planning*, organizing the content to be communicated so the resulting discourse is appropriate to the function of the media and for the intended audience; and 3) *content realization*, how to say the sequence of "utterances" by applying lexical information and syntactic rules [Grosz, et al., 1986; Maybury, 1993].

Early work in natural language processing

An interesting comparison in natural language to design is the use of templates. Early work in natural language processing used templates to produce the appearance of natural language understanding. The classic example of the use of templates is the ELIZA program [Weizenbaum, 1966] which simulates a session with a psychologist. Because of the rigid nature of templates, the program would drop information and fail. For example, if told, "my friend's sister likes me." the program will ignore the word *friend* and produce the inappropriate response, "Tell me more about your family." The system would rate words to determine which templates to choose, and if none were applicable, elicit more information. Basically, only a superficial analysis of a sentence would be possible.

Today, many editors and graphic systems use templates to describe and control graphics. They fail in similar ways because of the rigid and superficial nature of the template representation. In order create a richer understanding, and therefore better support in the graphic domain, we need more sophisticated tools, similar to those that have evolved in natural language processing.

Another early approach to producing natural language systems was to deal with highly constrained worlds where knowledge could be represented. A milestone was Winograd's SHRDLU program in 1971. He used a restricted domain to show the possibility of natural language understanding in a restricted blocks world. This approach limits the scope of the problem to make it more tractable. In some sense, the limited scope of the domain of page layout enables the automatic and interactive design of on-line documents.

Visual language processing

Computational linguistics has developed theories and techniques which have proven useful for string languages as mentioned previously. By extending them into the visual domains, we can gain significant leverage. Using an independently motivated parser has a number of advantages. Most importantly, if we think of visual communication as a structured language, we can use the grammar and parsing technologies to extract that structure and provide a better understanding of that information. In addition, we can take advantage of all the previous work in natural language processing. As progress is made in natural language processing, those advances can be incorporated into the methods used here.

Requirements of visual languages

Both string languages and visual language systems have many similarities. However, visual languages place a number of unique requirements on conventional rule representations and parsing techniques. Within the context of this research, the following requirements have become evident.

• *The parser needs to refer to the elements of the design.*
String languages use words while visual lanaguages use objects. These elements in the interface or document form the basic lexical categories of the language.

• *Encodable rules must relate to the design space.*
Rules and constraints of the visual domain must be expressible. For instance, alignment and symmetry are useless if they cannot be expressed. Traditional one-dimensional string languages are not adequate for this task and must be extended to higher dimensions.

• *The rule language must allow arbitrary domain-specific constraints.*
The rule language must be expressive enough to allow arbitrary domain-specific constraints, not necessarily restricted to those that are just visual in nature. For example, the rule language must be able to express arbitrary relationships such as topology and specific relations that are not visual in nature (e.g., the *author-of* relation for the design of books and table of contents).

• *Arbitrary order of input of the design elements must be supported.*
Traditional string parsing assumes input is in a strict left-to-right sequence. In visual domains, however, the system cannot restrict the

user to conform to a special order of input. Design must continue in a natural way without any restrictions to the order of input.

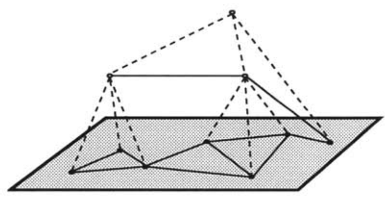


FIGURE 3.7
Primitives and composites combine in forming new structure during the parsing process. The dots on the horizontal plane indicate indicate elements in the input. Solid lines between the elements represent relations between the elements while dashed lines indicate new structure created through parsing.

• *Non-monotonic changes must be supported.*
Non-monotonic changes to the input should also be handled with minimal disruption so users can move, resize, and delete elements of their design. String-based languages are typically monotonic, because words cannot be *unsaid* or removed from the input.

Visual parsers

The basic process of the parser is to build structure, usually in the form of a tree, from the elements of input in the design. Each leaf of the tree represents lexical items in the grammar. When rules fire, composite elements are created moving up to the next branch of the tree. When the top of the tree is reached, the final composite is formed, completing the "visual sentence." This is graphically represented in Figure 3.7.

There is a wide-ranging family of higher-dimensional grammar frameworks which include array, tree, or graph grammars [Rosenfeld, 1990] and unification-based constraint grammars [Helm and Marriott, 1991]. Where string grammars generate or compose expressions consisting of one-dimensional arrays of symbols, these higher-dimensional grammars deal in structures characterized by the domain. For example, these characterizations might be based on geometric positions in two or three dimensional space, or topological connectivity, or arbitrary semantic relations holding among information objects. The motivation for using higher-dimensional grammars rather than string-based grammars is that strings alone will not be a rich enough structure to capture relationships in design domains. The information content and context serving as the input to a presentation procedure cannot be naturally or easily coerced into a string. However, the architecture used in this dissertation would still be appropriate for use with more traditional string-based frameworks such as attribute grammars [Knuth, 1968].

The Relational Grammar approach can be viewed as a graph-rewriting problem (see Figure 3.7). The relations are represented by arcs, and terminals and nonterminals of the grammar by labeled nodes. One could then define graph replacement rules that would rewrite graphs to other graphs. This approach can be seen in work on graph grammars [Ehrig, Nagl and Rozenberg, 1986]. However, as Helm and Marriott [Helm and Marriott, 1986; 1991] suggest, it is useful to provide indirection between graphical relations named by the grammar and the operations that might have to verify them as constraints relative to particular database queries on graphical objects. This indirection helps to maintain a generality to the approach by preserving an independence between the grammar and the database. However, it does complicate the determination of the computational complexity of the algorithm.

Taking Figure 3.7 as a illustration of Helm and Marriott's approach, the lines would be represented by axioms and rules of a constraint theory, while the nodes would be represented by first order logic terms with graphical attributes. The mother-daughter relations would be specified by hierarchical definite-clause-grammar-style composition rules [Pereria and Warren, 1980]. Parsing, as well as generation, could then be handled by the general resolution theorem-proving mechanism of logic programming with constraints. The goals of Relational Grammars are similar to Helm and Marriott in that both formalisms attempt to handle a very general class of graphical languages. However, Relational Grammars extend Earley-style parsing with techniques to accommodate graphic elements and sets of elements in the table lookup while Helm and Marriott use logic programming augmented with general constraint-solving techniques by extending definite-clause grammars.

Relational Grammars allow the grammar to state any number of spatial-relation constraints among the constituents of the rule body. Unlike the approach of Golin and Reiss [Golin and Reiss, 1989], Relational Grammars do not allow constraints among elements arbitrarily far in the derivation tree. Unlike the grammars of SILICON system [Crimi, et al., 1989], Relational Grammars do not confine spatial constraints to a single relation among pairs of elements that are adjacent in a rule body. As mentioned previously, Relational Grammars separate the parsing mechanism (i.e., the parse table) from any particular set of visual relations used by the grammar.

Relational Grammars are using a context free backbone as its formalism. In context free grammars, the left-side of the rule consists of only the symbol to be replaced. Nothing can influence how the replacement is done, hence the label *context-free*. But Relational Grammars use the power of attribute features and propagate these attributes from composite to composite. These are then used in rule matching. This added power is traded against the the grammar's context-freeness. In general, the Chomsky language-type hierarchy doesn't carry over to the general category of higher order languages like Relational Grammars.

Fred Lakin utilizes spatial parsing to build interpretation of user actions that enable system actions in response to the visual phrase. He states:

Spatial parsing is the process of recovering the underlying syntactic structure of a visual communication object from its spatial arrangement.[3.7]

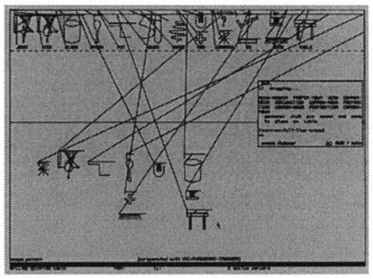


FIGURE 3.8
Fred Lakin's VIC uses pairwise comparison of the objects for parsing of phrase structure.

He uses spatial parsing for five separate applications: a visual programming language, visual communication system for aphasics (VIC), finite-state-automaton diagrams, graphic devices for organizing textual sentence fragments (SIBTRANS), and support for understanding informal conversational graphics of blackboard use.

One of the main contributions of this dissertation has been to apply the independently justified grammar formalism to the interactive and automatic support of graphic design, and to be explicit as to the role it plays in supporting the process of design. There are other systems that claim to be using grammars but don't. Much of the previous research that uses grammars for design does not articulate how grammar rules are used in their algorithms.

VIA: Visual Information Architecture

This thesis investigates the use of a computational linguistic-based parser and grammar formalism to support the design process. This independently motivated parsing algorithm forms the basis of an experimental environment for the creation and presentation of multimedia documents. This environment *VIA, Visual Information Architecture* supports the investigation into the different scenarios. Kochhar, Marks, and Friedell [Kochhar, et al., 1991] characterize the articulation of a designed artifact along the axis of automaticity, from completely manual to completely automatic. VIA provides support along this axis for various scenarios. VIA has been used to support interactive, improver-based design [Weitzman and Wittenburg, 1993], as well as the automatic presentation of dynamic multimedia documents [Weitzman and Wittenburg, 1994]. Figure 3.9 illustrates the automatic and interactive paradigms with Relational Grammars as the key processing component.

3.7 Lakin, 1986 Spatial parsing for visual languages, p.36.

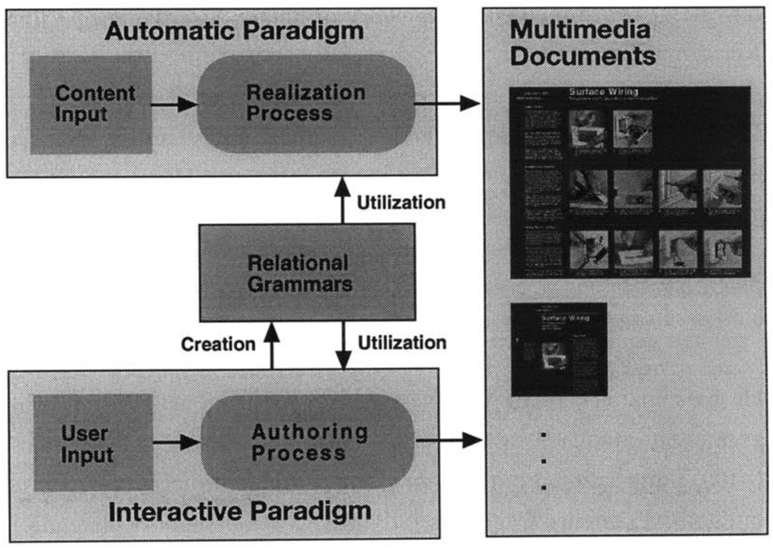


FIGURE 3.9
Vision of an integrated environment for interactive authoring and automatic presentation of multimedia documents. Relational grammars are the central computational formalism of the system.

A categorization of different interactive paradigms include:

- Incremental improvement
- Graphic design completion
- Design verification and error checking
- Syntax-directed editing
- Structural zooming, and
- Automatic presentation.

VIA uses Relational Grammars in an incremental, bottom-up parsing algorithm of [Wittenburg, et al., 1991] with an additional constraint solving module [Maloney, 1991; Freeman-Benson et al., 1990]. The parser is able to handle different interaction paradigms by modifying the way in which input is processed by the parser. These variations can be characterized into a number of distinct types, *top-down* vs *bottom-up* parsing, *complete* (cover all input) vs *incremental* parsing (cover partial input), and *depth-first* vs *breadth-first* control of search.

In addition to top-down vs bottom-up processing mentioned previously, parsing can search the space of derivations *breadth-first* or *depth-first*. A depth-first approach extends a hypothesis as far as it can before trying other options. This method is concerned with finding the first solution in a somewhat sequential manner. Alternatively, a breadth-first search through the search space maintains multiple solutions and advancing them all in turn. These alternatives have implications when applied to interaction paradigms. For instance, during automatic design, a depth-first search is used to find the first derivation covering all input. However, in an interactive mode, the parser should find any improvement, so a breadth-first search is used that doesn't need to cover all input. In this way any improvement in the design will be triggered.

The bottom-up approach is well suited to an exploratory design process. In general, a bottom-up parser is used in VIA. The grammars can also be used in a top-down algorithm enabling the semantic expansion of design elements. An interactive design system that uses Relational Grammars with a top-down algorithm has been developed by [Collier and Karlin, 1993]. The system supports the creation of structured diagrams of phone equipment. Non-terminal elements in the design provide decision points in the development of the diagram. The process is complete when all elements in the design have been replaced by terminal elements in the grammar. The top-down approach is well suited to this structured domain, enforcing conformance to known design configurations.

As can be seen from Figure 3.9, Relational Grammars play a central role in this vision. In the automatic paradigm, the realization grammar maps from sets of content objects to multimedia documents; in the interactive paradigm, the improver-based grammar watches the user's authoring actions and suggests improvements, creates composite objects, and so on. The larger vision also includes an authoring component which produces a grammar, rather than a finished document, that can be utilized in the other two paradigms.

Input to VIA

Information to be incorporated as input to VIA will have been previously filtered and classified. Other work at the Media Lab, including user profiling [NIF, 1994] and semantic-based linguistic retrieval [Chakravarthy, 1994], is concerned with these issues.

The automatic design scenarios assume the input has previously been classified by type with domain relations between the elements. The interactive scenario provides classification through menu selection while spatial relations are inferred by relative positions of the input elements. The research here is not concerned with the retrieval and filtering of data but on the structure and presentation of the information once it has been selected.

Automatic presentation

In the automatic presentation of information, the system parses the semantic relationships of the elements and creates the syntactic structure and constraints for layout. These constraints are utilized in realizing the document in its final form. Use of automatic layout to present information is enhanced by the ability to switch grammars. Separate grammars can embody individualized display styles, preferences, and standards of presentation. The ease of switching grammars dynamically supports the change of focus in the user, task, or environment, and a completely new presentation results for each new grammar. Alternatively, individual grammars can generate multiple *interpretations* where each interpretation produces a specialized presentation. Predicates within the grammar rules are used to guide the derivation of these interpretations. Predicates such as *Large-Color-Display-P* or *PDA-P*, will direct parsing to generate presentations for large, color displays or small, personal digital assistants, respectively.

In the domain of training manuals, grammars can also be created to support the various tasks that are associated with different users of the manual. In addition, grammars can support document standards for the production of variations of the manual. Training manuals, stored in databases, can dynamically display the current and most relevant information. This task-based emphasis helps the user, whether novice or expert, by illustrating information that is most relevant to their specific need. Answers to questions can now be personalized, making the use of the training manual more effective. In this domain, there are a number of typical tasks that can be identified. These include: *Overview*, providing the general description of equipment and procedures; *Usage*, describing how a piece of equipment will be used; *Installation*, describing the placement and connections of the various pieces of equipment; *Setup*, describing the calibration of the equipment once it is in place; *Maintenance*, describing the procedures for standard upkeep of the equipment; *Troubleshooting*, describing how to locate the source of a particular problem; and *Repairing*, describing the procedures on how to fix broken equipment. By utilizing different grammars for these different tasks, the system can present the information based on the situation, the level of expertise of the user, and in conformance to the standards of the intended audience. The user does not need to know that a new grammar is being used, only that the task context has changed. Remember that the emphasis here is not the selection and filtering of the information, but rather, on how to automatically present the information once it has been chosen.

Interactive design

Interactive use of Relational Grammars is symmetrical to the automatic presentation paradigm. While automatic presentation within VIA parses semantic relationships and produces the syntax of layout, interactive design parses the syntax relationships of layout and produces semantic elements of the domain. For example, in the interactive process of authoring an article for a table of contents, the relationships of how lexical elements are positioned and sized produces the semantic domain object *Article*. Many of the affordances of the Relational Grammar approach are best illustrated through an interactive design scenario. Interactive support includes simple cleanup and improvement of users' actions, top-down refinement of design solutions, semantic zooming based on composite structures, graphic completion and verification of partially complete design decisions. One of the main contributions of this work is in the innovative techniques to support interactive design.

Rule acquisition by demonstration

An important aspect of any design environment is the ability to override and change the working assumptions as the design progresses. This research also investigates the use of an editor to dynamically modify and create rules to be incorporated in the grammar as design progresses. The declarative formalism of the rule definitions facilitates the acquisition of new rules. The process of acquisition uses a generalized grammar in the interactive scenario. This grammar encapsulates rules based on effective design in dynamic environments. The grammar alone can not easily isolate the proper inferences when building new rules. In order to disambiguate these graphic inferences, menus and interactive agendas engage the designer to direct the rule creation process. Similarly, menus can support the capture of domain relationships between the elements in the preconditions of rule bodies. The output of this process produces the declarative code that defines the rule. Rule management also includes the adding/deleting, enabling/disabling, and grouping/ungrouping of rules and rule sets.

Dynamic environments

One of the most exciting capabilities is the dynamic characteristic of the environment. The techniques in this research are integrated into a unified working prototype. The user can easily switch between the different scenarios and modes, and freely incorporate any changes made in the rule definitions. Thus, this integration allows the user to easily switch between the authoring and viewing of information.

In order to exploit these characteristics of the environment and produce dynamic documents, several techniques have been incorporated into the system. First, each element has the potential to become dynamic itself. Elements dynamically reflect changes in state by modifying visual attributes like *visibility*, *position*, *color*, etc. Second, constraints within the system support the standard temporal relationships defined in [Allen, 1983]. In this way, elements can constrain their start times, end times and durations using other elements in the document. Third, hyperlinks between different elements in the document are produced during the authoring process. This permits the user to quickly jump to other information within the document. All these techniques begin to define an environment in which users can author and view dynamic documents.

Vision of document processing

A new vision of how documents will be processed in the future is beginning to emerge. This process is no longer the production of a single document, but rather the creation of an information source that can be utilized to produce multiple instances of the information appropriate for the intended viewer, delivery environment, and ultimate use.

Agile publishing

Publishing should be tightly integrated into the workflow of business processes. Rob Haimes describes a vision of content-driven document creation that separates the content from the form, resulting in a more *agile* process [Haimes, 1994a; 1994b]. Haimes describes agile publishing as:

Structuring content for repurposing and using more sophisticated templates (or rule-based design models) ultimately leads to an increasingly dynamic and automated publishing process — one that can become central to supporting a company's business objectives. 3.8

This vision enables data to be created and maintained at a higher level of quality, documents to be formatted using predefined specifications and delivery to multiple environments with quicker response to customer feedback. Michael Spindler, Apple's president and CEO, views publishing as a collaborative effort, working with quality controlled content sources, and building a technology framework to support collaboration within re-engineering organizations. Spindler calls this new publishing environment the "*information factory*," where infor-

3.8 Haimes, R. Managing workflow and content for agile publishing, *Color Publishing*, 1994, p. 29.

mation can be customized for specific audiences and uses. This vision is shared by VIA, and Relational Grammars play a key role in this new view of information publishing.

Markup languages

In an effort to support the transfer of electronic documents from one user to another and from one platform to another, the use of markup languages have evolved over the years. These languages allow the structural information of a document to be added by embedding special sequences of text characters within the stream of document text. There are currently two basic types of markup languages in use, *specific* and *generalized.*

Specific markup languages use marks that have immediate effect describing the appearance of the characters (i.e., italic or bold) or their position (i.e., centered, indented, line spacing). This is also called *procedural* markup because it describes a particular procedure for formatting a portion of the document. Generalized markup languages, however, identify the purpose of the text (i.e., heading or paragraph) rather than its physical appearance. This is also called *descriptive* markup which assigns a format independent description or label to a portion of the document. In this way, generalized markup languages can easily pass the structure of the document from machine to machine.

It is then a relatively simple task to take this structure and change the way in which the elements of the document are presented. The final appearance of the document is resolved in a system-dependent process referred to as formatting the document. The current generalized markup standard being promoted is the Standard Generalized Markup Language (SGML) [Bryan, 1988; Herwijnen, 1994]. Advantages of generalized structured documents are:

- *Support for document interchange*
  It is difficult to move documents across platforms with specific markups or proprietary systems.
- *Utility of structured document description*
  Information is hidden and lost if you concentrate on document's visual properties and not on the documents structure.
- *Flexibility of the generalized markup*
  Multiple platforms and users of the same information can more easily be supported. Uniform styles can be better controlled.
- *Unambiguous format for database storage and retrieval*

SGML and VIA have a similar vision with regard to document production. In this vision, document creation and production is more agile and not restricted to one final document. SGML uses ASCII markups to describe the nature of the content. VIA, on the other hand, uses the relations of the domain to create the distinctions in the data. These, in fact, may be exactly the same as the markups in SGML or may use other *semantic* relations in the domain of the data. For instance, here the relations might suggest the ordering of table of contents entries by the type of article.

A *document type definition*, DTD, is the method for describing document structure and element/attribute descriptions in SGML. This is similar to the grammar specification within VIA. Both define the documents in terms of a tree structure. New structures can be defined in SGML as well as in VIA for new domains and new document types.

The WYSIWYG (What you see is what you get) editor equivalent of generalized markup is the stylesheet. A stylesheet groups formatting instructions together in a instruction, like a macro. These can then be applied to portions of the document producing the desired results. This can potentially separate the structure of the document from its appearance. But if the styles are not adhered to by the author, these advantages are lost. In addition, stylesheets do not guarantee the portability of documents across platforms because the source codes depend on the particular formatter. SGML pushes off the style decisions to the end application while in VIA, the *style* is embedded in the body of the grammar rules. A more flexible use of templates may use external processes to determine the appropriate template for a given context. VIA, on the other hand, incorporates this decision process within the body of the grammar rules.

In SGML, a sequential ASCII stream of characters is parsed in order to ensure conformance to the SGML standards. This is described as a *validating SGML parser*. Only SGML errors in the document should be reported. If the parser produces any other output, this output is not covered by the standard. On the other hand,VIA parses elements within a database using relationships between these elements. The output of this process creates the composites in the language. In the SGML document, this structure has been identified by the author either by hand or through the use of a specialized editor. Conversely, VIA can infer the correct markup structure through the design relationships and user's interactions, thus avoiding a potentially laborious process. For example, consider the problem of specifying the structure of a complex mathematical expression! VIA can infer the proper structure from the primitve elements in the design. In this use of Relational Grammars, VIA can support the creation of the DTD, similar to VIA's rule editor. This is not to say that Relational Grammars can't parse an SGML document. Standard generalized markup languages are a subset of the languages that Relational Grammars can handle.

The most important distinction between SGML and VIA is that VIA is concerned about the general problem of supporting the design process with a language-based approach. The automatic paradigm within VIA, most similar to an SGML document production, is just one of a number of techniques available to the designer if the input information is based on a visual langauge. VIA is interested in the more general problem of design support and augmenting user interaction to improve the design process and the resulting artifact.

Summary

There is an emerging vision of document creation. No longer are we creating isolated documents. Instead, document production is the authoring of information customized for different audiences and media presentations. This dissertation introduces new techniques based on computational linguistics to support this vision of agile publishing. By exploiting the structured nature of information, visual languages can be defined and used within the design process.

The following three chapters investigate the use of this class of higher dimensional languages to support design. In Chapter 4, the automatic presentation of information is detailed. Various examples of dynamic documents are automatically created. In Chapter 5, the interactive design of dynamic documents is described. In this chapter, visual languages are used to support the interactive authoring of information. In Chapter 6, the creation of these languages by demonstration is explored. By providing high level tools to designers, rules can be modified without coding.

Automatic presentation of information

4

Introduction

The more we can automate the document creation process, the more we'll be talking about the document as the user interface into the information.
BRIAN DEGEN
DIRECTOR OF PUBLISHING SERVICES AND TECHNOLOGY, FIDELITY

Sometimes the best interaction is none at all. Using automatic layout, VIA presents information using predefined grammars without any user intervention. One advantage of this paradigm is that the interaction with the user is extremely simple. The user merely initiates the request for information. In addition, different contexts can be supported by the grammars encoding the style requirements and guidelines for different classes of users, tasks and delivery environments. This methodology is applicable to many domains. For example, when newspapers are delivered over the network instead of the traditional paper medium, individualized grammars can personalize the information. In addition, these techniques may be used to support document translation, parsing one document in a markup language or page description system, and translating it into another format.

A fully functioning multimedia system requires a wide range of stages to achieve effective automatic presentations. These include the processes of content selection, which identifies what to say; media allocation, which identifies what media to say it in; media realization, which identifies how to say it in these media; and media coordination, tying the different media together over time [Maybury, 1993]. However, in order to communicate effectively, adaptive multimedia systems must not merely present information, but must present information that has been specifically designed for a given context and task. The dynamics of information in the future will require a more careful crafting of the documents we author. Information will constantly be changing, users will have different requirements, and display devices on which they view the information will require vastly

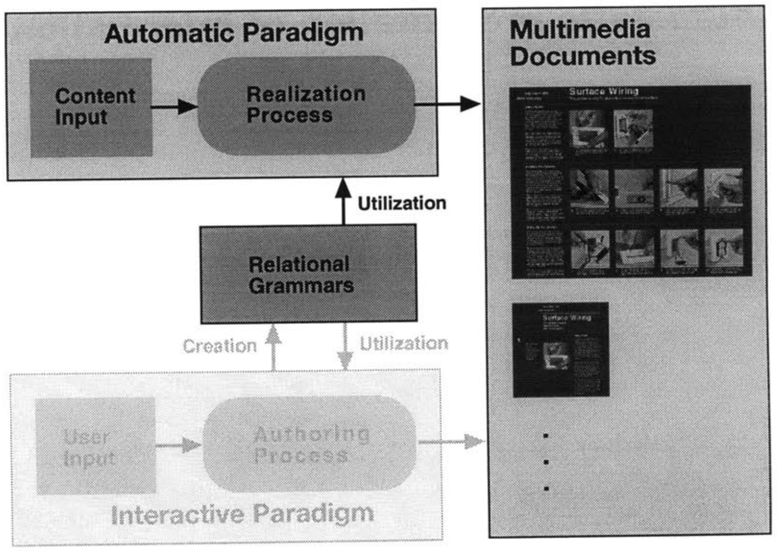


FIGURE 4.1
Overview of VIA's articulation process for the automatic presentation of multimedia documents.

different design solutions. At the same time, documents will include more structured knowledge of their content. In order to support the dynamics of this information-rich environment and exploit the nature of these structured documents, we will need new techniques and paradigms for the automatic design and presentation of this information.

This chapter focuses on the media realization phase of automatic presentation and describes the use of the Relational Grammar formalism for encoding design knowledge and a methodology for document realization. As can be seen in Figure 4.1, Relational Grammars play a central role in this vision. Information, typically stored in a database, is classified by type and includes relations within the information. Using this knowledge, the system automatically constructs the structure of the document to be presented. The semantics of this structure includes the detailed graphical constraints in order to present each element. By using grammars that are attuned to individual users, the layout is sensitive to the user's specific interests and preferences. The most current and relevant information is automatically presented without the intervention of design professionals.

This methodology is based on parsing and syntax-directed translation. Translation is followed by a constraint solving mechanism to create the final layout. Grammatical rules provide the mechanism for mapping from a representation of the content of a presentation to forms that specify the media objects to be realized. These realization forms include these constraints: graphic (e.g., font specification), spatial (e.g., relative positioning), and temporal (e.g., sequence of presentation) between elements of the presentation. Individual grammars encapsulate the "look and feel" of a presentation and can be used as generators of that style. In this manner, dynamic presentations can be supported. By making the grammars sensitive to the requirements of the output medium, parsing can introduce flexibility into the information realization process. The realization procedure may deliver different documents under differing circumstances given the same input.

After an overview of the architecture of VIA's automatic presentation system, a simple example of a multimedia on-line document is described. It takes its look and feel from the table of contents of a popular magazine. This first example, which will describe the parameters of spatial and temporal layout, focuses on the architecture of VIA's realization system. The form of the input and the output of the realization process is then described along with examples of rules that articulate these particular design styles. The realization of dynamic presentations in which the grammar constrains the elements of the presentation both spatially and temporally is then described. These examples also show how syntax-directed translation can achieve differing results depending on the hardware and software characteristics of the delivery environment.

Architectural overview

Relational Grammars with semantic attributes provide a mechanism for the articulation phase of the larger multimedia presentation problem. An overview of the automatic presentation architecture within VIA is presented in Figure 4.2 and is characterized as follows. Given a representation of the content to be communicated by some design, create one or more instances of a fully articulated design. Here, the concern is not with the important problem of accessing and filtering information. The assumption is that the information to be presented has already been chosen and relationships between the elements are known. Another process, or the user, first selects the information to be presented. The system then parses the content elements and relations building a derivation tree. The chapter's first example, presented here, corresponds to the hierarchical composition of the set of articles and headers to be included in a table of contents page.

Then, a translation phase begins. Following in the tradition of syntax-directed translation [Aho, 1986], each grammar rule has an associated set of attributes which are used to compute the output forms from a syntactic derivation tree. Here, the output determines a set of media objects to be created and a set of spatial and temporal con-

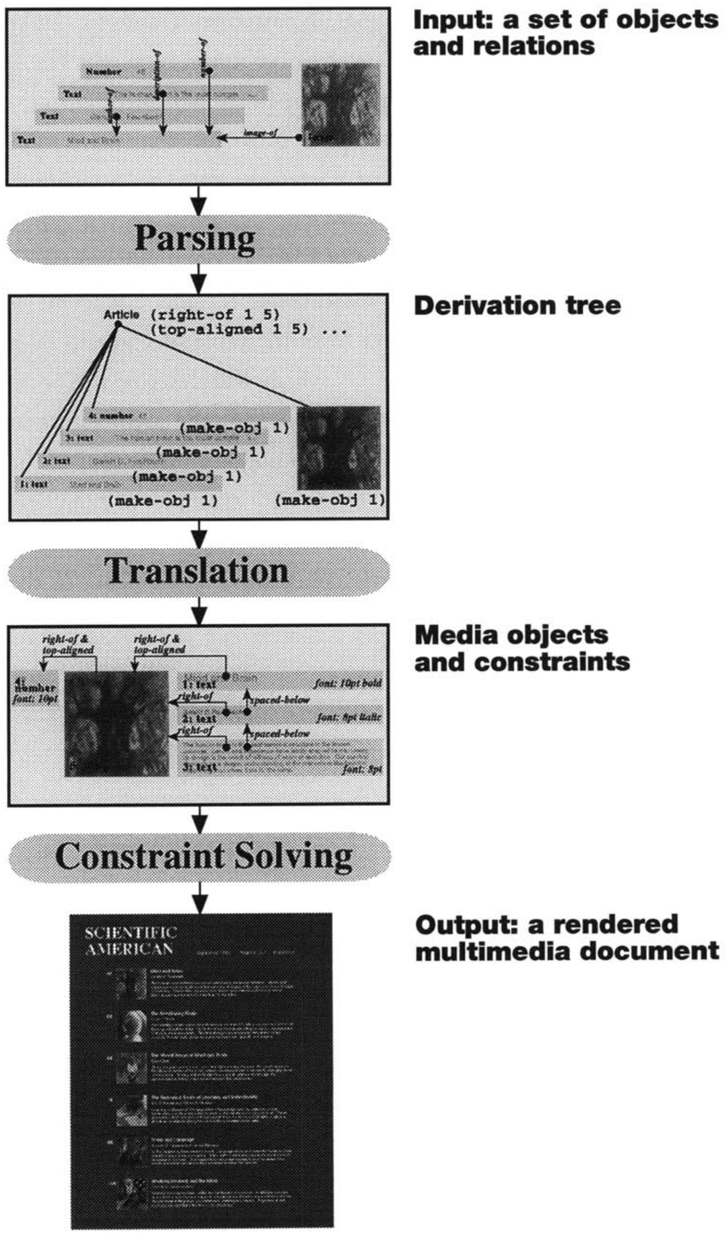
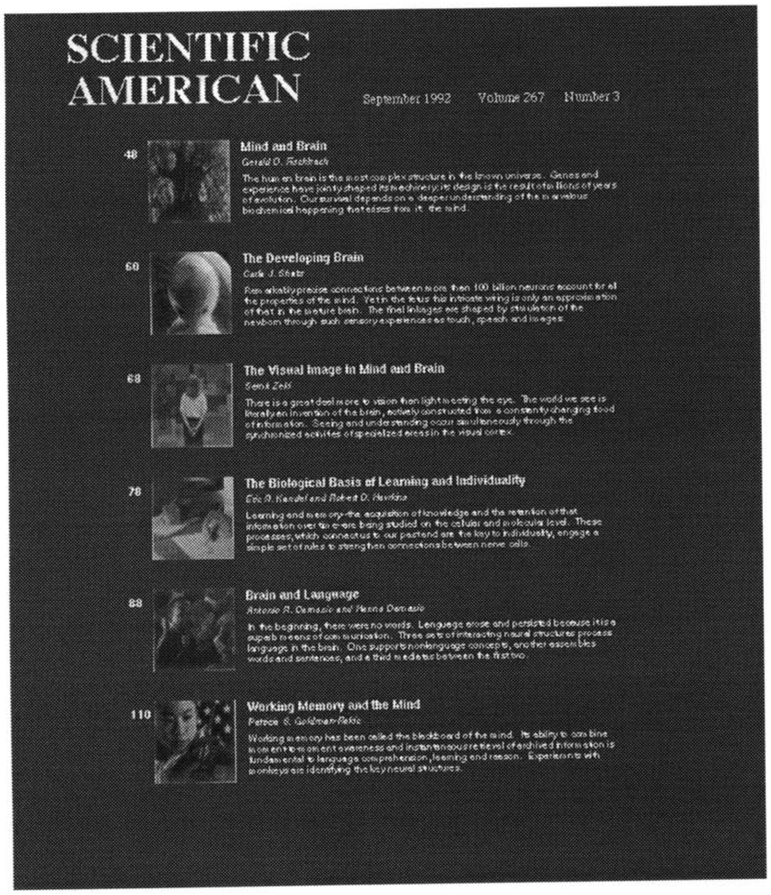


FIGURE 4.2
Overview of the process of articulating the presentation of a multimedia document in VIA.

straints to be installed. Through familiar methods of computing inherited and synthesized attributes, the semantic output of the parse tree is produced. A constraint resolution procedure is then invoked to solve the constraints among media objects that determine the actual numerical values for spatial and temporal positioning. Finally, the media objects are rendered on the display.



SCIENTIFIC AMERICAN September 1992 Volume 267 Number 3

48 **Mind and Brain**
Gerald D. Fischbach
The human brain is the most complex structure in the known universe. Genes and experience have jointly shaped its machinery; its design is the result of millions of years of evolution. Our survival depends on a deeper understanding of the marvelous biochemical happening that arises from it: the mind.

60 **The Developing Brain**
Carla J. Shatz
Remarkably precise connections between more than 100 billion neurons account for all the properties of the mind. Yet in the fetus this intricate wiring is only an approximation of that in the mature brain. The final linkages are shaped by stimulation of the newborn through such sensory experiences as touch, speech and images.

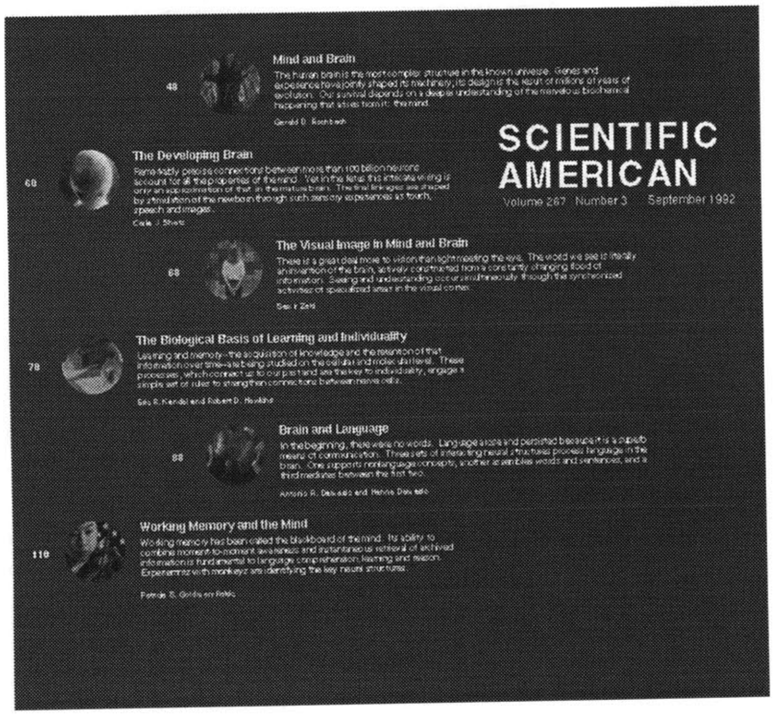
68 **The Visual Image in Mind and Brain**
Semir Zeki
There is a great deal more to vision than light meeting the eye. The world we see is literally an invention of the brain, actively constructed from a constantly changing flood of information. Seeing and understanding occur simultaneously through the synchronized activities of specialized areas in the visual cortex.

78 **The Biological Basis of Learning and Individuality**
Eric R. Kandel and Robert D. Hawkins
Learning and memory—the acquisition of knowledge and the retention of that information over time—are being studied on the cellular and molecular level. These processes, which connect us to our past and are the key to individuality, engage a simple set of rules to strengthen connections between nerve cells.

88 **Brain and Language**
Antonio R. Damasio and Hanna Damasio
In the beginning, there were no words. Language arose and persisted because it is a superb means of communication. Three sets of interacting neural structures process language in the brain. One supports nonlanguage concepts, another assembles words and sentences, and a third mediates between the first two.

110 **Working Memory and the Mind**
Patricia S. Goldman-Rakic
Working memory has been called the blackboard of the mind. Its ability to combine moment-to-moment awareness and instantaneous retrieval of archived information is fundamental to language comprehension, learning and reason. Experiments with monkeys are identifying the key neural structures.

FIGURE 4.3
Automatic layout of Scientific American table of contents in the traditional style.



48 **Mind and Brain**
The human brain is the most complex structure in the known universe. Genes and experience have jointly shaped its machinery; its design is the result of millions of years of evolution. Our survival depends on a deeper understanding of the marvelous biochemical happening that arises from it: the mind.
Gerald D. Fischbach

SCIENTIFIC AMERICAN
Volume 267 Number 3 September 1992

60 **The Developing Brain**
Remarkably precise connections between more than 100 billion neurons account for all the properties of the mind. Yet in the fetus this intricate wiring is only an approximation of that in the mature brain. The final linkages are shaped by stimulation of the newborn through such sensory experiences as touch, speech and images.
Carla J. Shatz

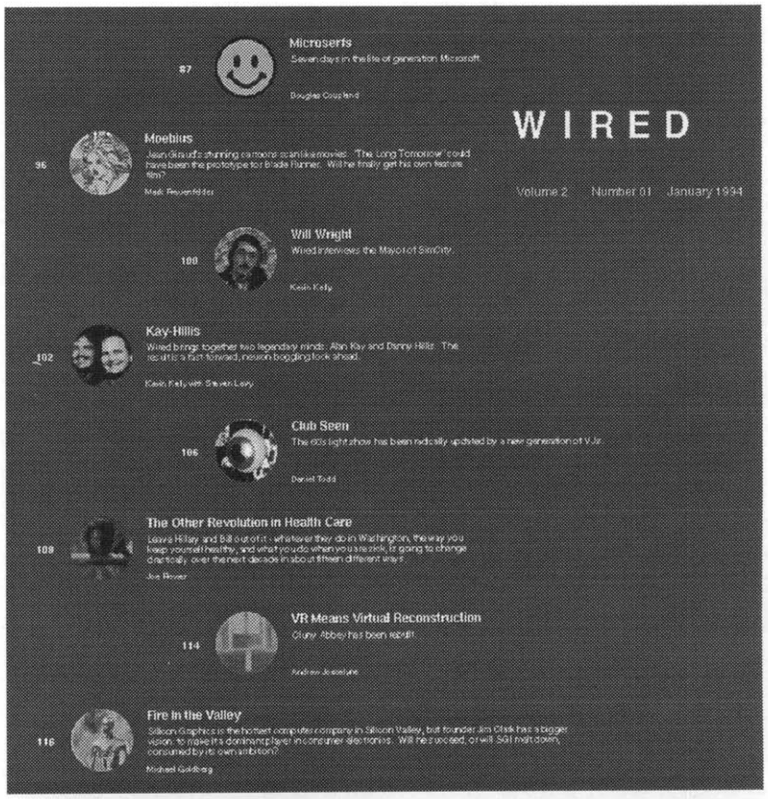
68 **The Visual Image in Mind and Brain**
There is a great deal more to vision than light meeting the eye. The world we see is literally an invention of the brain, actively constructed from a constantly changing flood of information. Seeing and understanding occur simultaneously through the synchronized activities of specialized areas in the visual cortex.
Semir Zeki

78 **The Biological Basis of Learning and Individuality**
Learning and memory—the acquisition of knowledge and the retention of that information over time—are being studied on the cellular and molecular level. These processes, which connect us to our past and are the key to individuality, engage a simple set of rules to strengthen connections between nerve cells.
Eric R. Kandel and Robert D. Hawkins

88 **Brain and Language**
In the beginning, there were no words. Language arose and persisted because it is a superb means of communication. Three sets of interacting neural structures process language in the brain. One supports nonlanguage concepts, another assembles words and sentences, and a third mediates between the first two.
Antonio R. Damasio and Hanna Damasio

110 **Working Memory and the Mind**
Working memory has been called the blackboard of the mind. Its ability to combine moment-to-moment awareness and instantaneous retrieval of archived information is fundamental to language comprehension, learning and reason. Experiments with monkeys are identifying the key neural structures.
Patricia S. Goldman-Rakic

FIGURE 4.4
Automatic layout of Scientific American table of contents (i.e., the same input as in Figure 4.3) using the style from WIRED magazine.



WIRED

Volume 2 Number 01 January 1994

87 Microserfs
Seven days in the life of generation Microsoft.
Douglas Coupland

96 Moebius
Jean Giraud's stunning cartoons scan like movies. "The Long Tomorrow" could have been the prototype for Blade Runner. Will he finally get his own feature film?
Mark Frauenfelder

100 Will Wright
Wired interviews the Mayor of SimCity.
Kevin Kelly

102 Kay-Hillis
Wired brings together two legendary minds: Alan Kay and Danny Hillis. The result is a fast forward, neuron boggling look ahead.
Kevin Kelly with Steven Levy

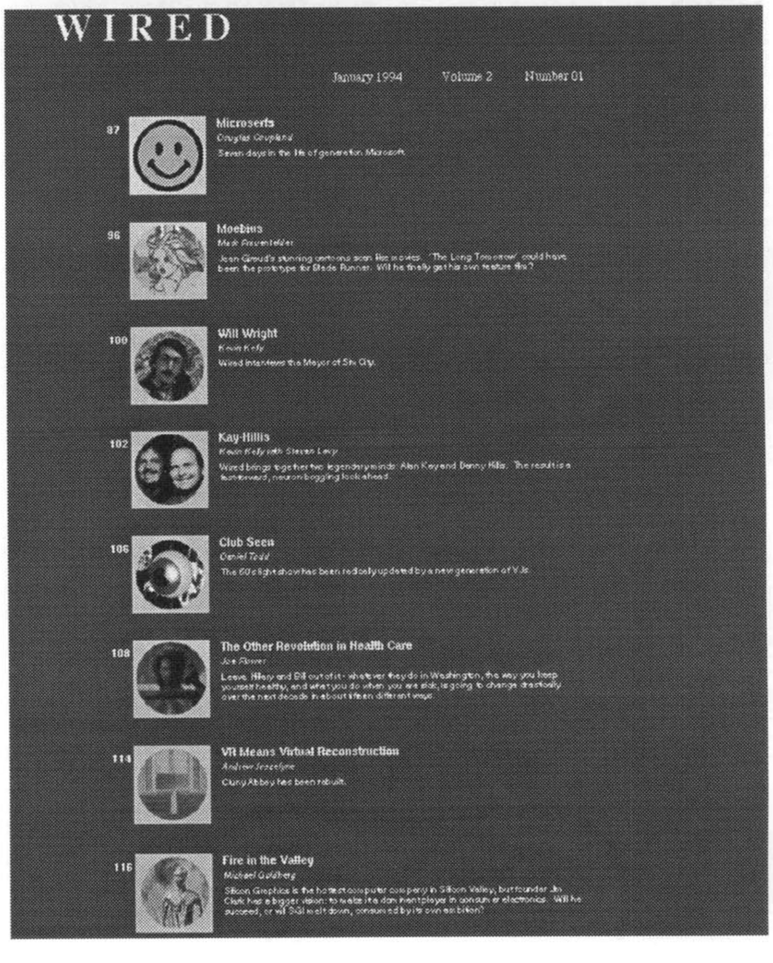
106 Club Seen
The 60s light show has been radically updated by a new generation of VJs.
Daniel Todd

108 The Other Revolution in Health Care
Leave Hillary and Bill out of it - whatever they do in Washington, the way you keep yourself healthy, and what you do when you are sick, is going to change drastically over the next decade in about fifteen different ways.
Joe Flower

114 VR Means Virtual Reconstruction
Cluny Abbey has been rebuilt.
Andrew Jossolyne

116 Fire in the Valley
Silicon Graphics is the hottest computer company in Silicon Valley, but founder Jim Clark has a bigger vision: to make it a dominant player in consumer electronics. Will he succeed, or will SGI melt down, consumed by its own ambition?
Michael Goldberg

FIGURE 4.5
Automatic layout of WIRED magazine table of contents in its ***normal*** *style.*



WIRED

January 1994 Volume 2 Number 01

87 Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft.

96 Moebius
Mark Frauenfelder
Jean Giraud's stunning cartoons scan like movies. "The Long Tomorrow" could have been the prototype for Blade Runner. Will he finally get his own feature film?

100 Will Wright
Kevin Kelly
Wired interviews the Mayor of SimCity.

102 Kay-Hillis
Kevin Kelly with Steven Levy
Wired brings together two legendary minds: Alan Kay and Danny Hillis. The result is a fast forward, neuron boggling look ahead.

106 Club Seen
Daniel Todd
The 60s light show has been radically updated by a new generation of VJs.

108 The Other Revolution in Health Care
Joe Flower
Leave Hillary and Bill out of it - whatever they do in Washington, the way you keep yourself healthy, and what you do when you are sick, is going to change drastically over the next decade in about fifteen different ways.

114 VR Means Virtual Reconstruction
Andrew Jossolyne
Cluny Abbey has been rebuilt.

116 Fire in the Valley
Michael Goldberg
Silicon Graphics is the hottest computer company in Silicon Valley, but founder Jim Clark has a bigger vision: to make it a dominant player in consumer electronics. Will he succeed, or will SGI melt down, consumed by its own ambition?

FIGURE 4.6
Automatic layout of WIRED magazine's table of contents (i.e., the same input as in Figure 4.5) using the style from Scientific American magazine.

Examples of spatial layout

Figures 4.3-4.7 illustrate the output of the VIA system in the realm of spatial layout and graphical style. Figures 4.3 and 4.4 use the same content but different grammars to achieve unique styles of presentation. Figure 4.3 is an automatically generated on-line version of the table of contents modeled directly from an issue of Scientific American.[4.1] Figure 4.4 uses the grammar based on WIRED magazine's style to present the same information.[4.2] Likewise, Figure 4.5 and 4.6 illustrate the automatic layout of the table of contents from WIRED magazine. Figure 4.5 uses a standard WIRED grammar while Figure 4.6 uses the grammar of Scientific American. The example in Figure 4.7 illustrates different content that utilizes the same grammar as in Figure 4.3 and 4.6. However, note that in Figure 4.7 less information is presented (i.e., no authors or descriptions appear in the input). The grammar in question contains rule variants that permit successful parses even though such content differences exist.

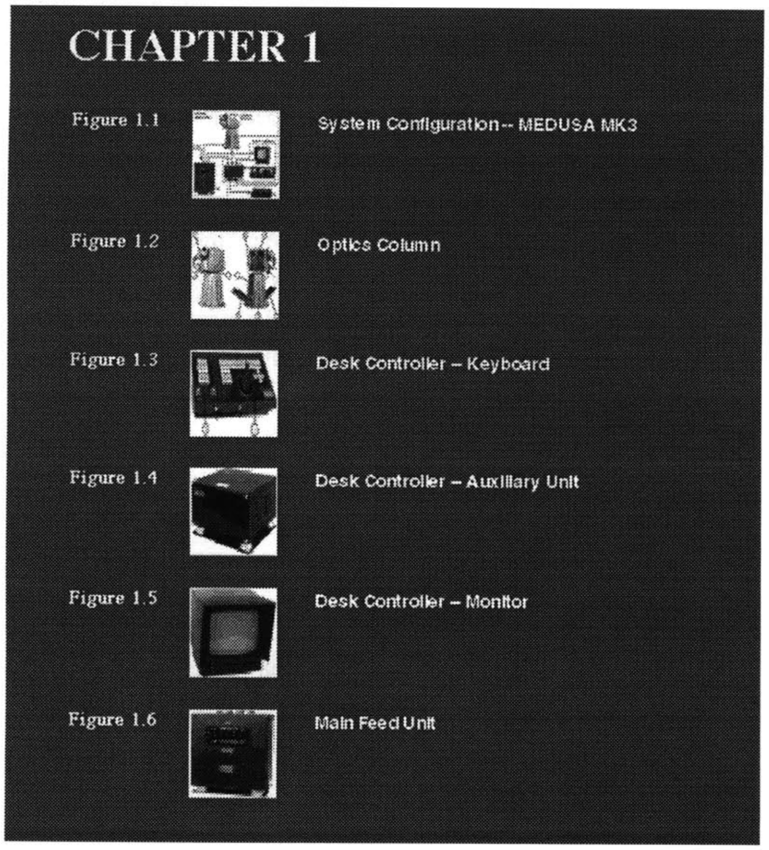

CHAPTER 1

Figure 1.1 System Configuration -- MEDUSA MK3

Figure 1.2 Optics Column

Figure 1.3 Desk Controller – Keyboard

Figure 1.4 Desk Controller – Auxiliary Unit

Figure 1.5 Desk Controller – Monitor

Figure 1.6 Main Feed Unit

FIGURE 4.7
Automatic layout of an online training manual in the traditional style of Scientific American. Variations of the basic rules parse the articles even though information is missing (i.e., no authors or descriptions available).

4.1 Scientific American, September 1992, Vol. 267, No. 3, special issue on Mind and Brain.

4.2 This layout is based on a grammar derived from WIRED, January 1994, Vol. 2.01.

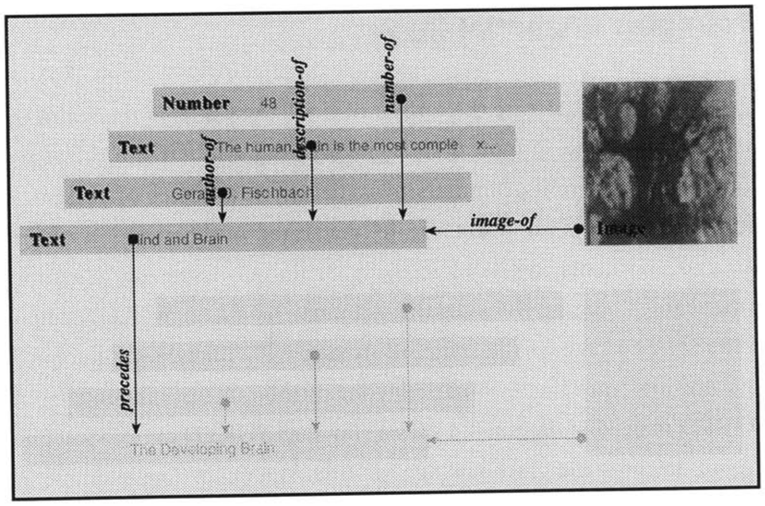


FIGURE 4.8
An example of partial input to the parser for the online magazine example. Input is a set of primitive objects (e.g., text, numbers, and images) and relations between the objects (e.g., author-of, description-of, page-of, image-of, and precedes).

Articulation components

Figure 4.3 will be used as the running example in the following discussion of the sequence of processing steps in VIA's articulation process.

Input to the parser

Input to the parser is a set of content objects as well as domain-dependent relations which hold between them. Conceptually, the input is a database, which can be thought of abstractly as a graph with primitive objects as nodes and relations as (hyper)arcs. In the on-line magazine example, VIA preprocesses a file describing the content information and constructs its own internal database. Figure 4.8 illustrates the form of the input data to the parser. For example, the figure indicates that the text, "Gerald D. Fischbach," is in the *Author-Of* relation to the text, "Mind and Brain." The basic types of objects that comprise the input to the parsing process include *Text*, *Numbers*, and *Images*.[4.3] In this example we order the articles as they are in the original publication (i.e., the *precedes* relation), but could have used other relations to determine the presentation sequence, such as importance, type of article (e.g., lead with a general science article), highest priority based on user profile, etc.

4.3 Depending on the needs and purpose of the application, the input could also contain higher structured objects such as a prestructured article composite. In this case, the system would not have to create an article composite object through the parsing process. The advantage of using more primitive elements as input is that the parser can do a certain amount of selection in assembling pre-existing pieces, an advantage not really exploited in the examples discussed here.

Grammar

Relational Grammars depend on generalized relations between the right-hand-sides of rules to constrain the rule applications and direct the parsing. In our running example, content relations such as *Author-Of*, *Description-Of*, *Page-Of*, and *Image-Of* are utilized. Figure 4.9 shows a rule which is utilized repeatedly in the derivation behind Figure 4.3. The context-free backbone of this rule corresponds to the rewrite rule:

```
Article → Text Text Text Number Image
```

Thus, 0 indicates the left-hand-side rule element and 1...n represent the right-hand-side rule elements. For details concerning the grammar formalism see [Wittenburg, 1992] and [Wittenburg, 1993].

```
(Defrule (Make-Article The-Grammar)
 (0 Article)
 (1 Text)
 (2 Text   (Author-Of      2 1))
 (3 Text   (Description-Of 3 1))
 (4 Number (Page-Of        4 1))
 (5 Image  (Image-Of       5 1))
 :OUT
 ((right-of      1 5)
  (right-of      2 5)
  (right-of      3 5)
  (right-of      5 4)
  (top-aligned   1 5)
  (top-aligned   5 4)
  (spaced-below  2 1)
  (spaced-below  3 2)
  (set-font 1 10pt :bold)
  (set-font 2 8pt  :italic)
  (set-font 3 8pt  :plain)
  (set-font 4 10pt :plain)))
```

FIGURE 4.9
The definition of the Make-Article *rule. The conditions for rule matching include relations between the elements (e.g., **Author-of**). **Article** is the resulting composite category that is created when the five basic categories (numbered 1 through 5) are matched and the indicated relations satisfied. The set of output constraints for this rule are listed after the* :OUT *symbol.*

The forms following :OUT in the rule definition represent an extension of Relational Grammars to include "semantic" attributes. Consistent with standard practice in compiler design, where attributes are used to generate compiler code, here attributes are used to generate code for creating media objects. In the current implementation, only synthesized attributes are used, i.e., the output attribute of each node of the derivation tree depends only on the values of attributes of nodes below it [Knuth, 1968], but the framework will be extended to incorporate inherited attributes as well.

Parsing

For automatic presentation, the parser's goal is to build a derivation tree that covers all of the input. In the current implementation, VIA uses a bottom-up, nondeterministic algorithm presented in [Wittenburg, 1992] with an additional control feature that allows a depth-first search, i.e., the parser returns as soon as a new derivation is found. Subsequently, parses may be sought until the search space is exhausted. The output of parsing is then one or more derivation trees, each of which yields an independent presentation.

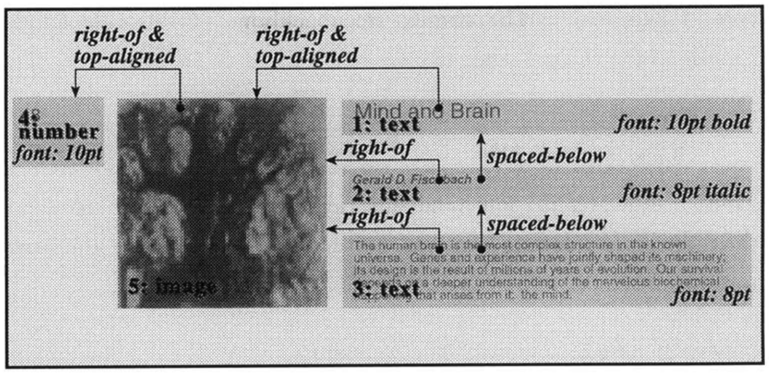


FIGURE 4.10
A part of the translation for the presentation in Figure 4.3 generated with the rule in Figure 4.9. The translation has created media objects and installed spatial constraints which must subsequently be solved.

Translation

When a derivation is found that covers all of the input, the set of :OUT forms is collected through a depth-first, left-to-right walk through the derivation tree. The *Make-Article* rule, in Figure 4.9, includes a number of forms constraining spatial (*right-of*, *top-aligned*, and *spaced-below*) as well as graphic (*set-font*) attributes. Figure 4.10 illustrates these output constraints graphically. In this example, basic lexical items include an output form which creates the realized element in the presentation.

Constraint solving

The constraint solving algorithm is a natural match with the grammar rule formalism. The grammar finds structure within the input, and then installs local constraints between the elements of the rule body. Rules for creating composite structures then create the constraints that link these smaller constraint networks together. The output forms of the final rule in the derivation then seeds this network with actual values (e.g., x and y positions) that are propagated during the constraint satisfaction phase of realization. In the final presentation, the user can interact with the elements. In the on-line table of contents examples, the user can move and resize individual elements and the constraint system interactively maintains the proper relationships installed by the grammar.

The constraint propagation system being used is DeltaBlue developed at the University of Washington [Freeman-Benson, et al., 1990; Maloney, 1991]. DeltaBlue is designed for non-cyclic constraint networks to be used in interactive applications with up to ~20,000 constraints. In the grammars built for automatic presentation within VIA, the constraints can easily be crafted to avoid cycles in the final presentation.

Examples of dynamic presentations

As mentioned earlier, the requirements for information presentation are becoming more and more dynamic. Not only does the information itself change, but so will the environment in which it will be viewed, its intended use, as well as the intended viewer of the information. In the previous examples, all output was constrained geometrically. The following examples illustrate different ways in which grammars can produce dynamic presentations that go beyond the traditional static presentation of information. Dynamics are incorporated into the output presentations by the grammars in the following ways:

- *Dynamic data types* (e.g., QuickTime™ movies, audio icons) are included in the lexicon of the presentations.

- *Temporal constraints* are used between output elements to create an automatic sequence of pages or actions presented to the viewer.

- *Hyperlinks* are installed by the grammar to produce an interactive sequence of pages through which the viewer can navigate. One way to create hyperlinked documents is through the use of HTML and presented in a viewer such as NCSA Mosaic [NCSA, 1993].

- *Presentation elements are tapped* into underlying simulations and reflect the state of that simulation by modifying graphic attributes such as visibility, color, size, and position.

Currently, *temporal constraints* in VIA are represented by intervals and their relationships to each other. These are based on the classic 13 relationships proposed by Allen [Allen, 1983]. This interval-based temporal logic includes the six relations of *before*, *meets*, *overlaps*, *during*, *starts*, and *finishes* and their inverses and the relationship of *equality*. In the more general representation of time, constraints need to include additional concepts such as sequencing, choice, iteration, and recursion.

VIA constructs a constraint network based on the relative time of an element's presentation indicated by the grammar. Then, using the same constraint satisfaction approach for spatial layout, we identify the relative time slots for realization. In VIA, every presentation ele-

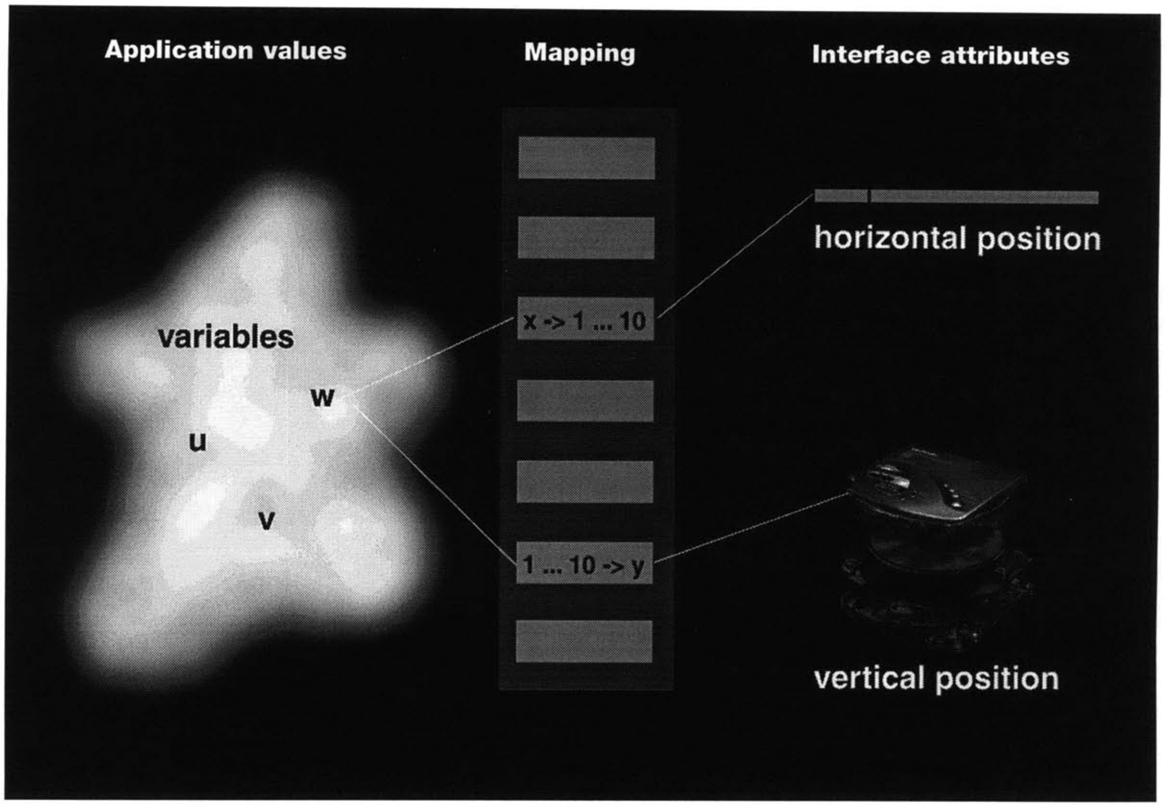


FIGURE 4.11
The general mechanism of ***tapping*** *any graphic attribute of a presentation element to an underlying simulation variable or process. Here, the horizontal position of the slider is being mapped to the vertical position of an element in an exploding diagram*

ment has a default duration time. The grammar typically "aligns" the elements to one another based on their start or end times. Then, one presentation element is seeded with a time value and the constraint system solves for the actual times to use in the presentation. This seeding typically occurs in the rule that builds the final derivation.

The ability to dynamically modify the graphic attributes of presentation elements is based on the tapping mechanism developed in the Steamer project [Hollan, et al., 1984] and generalized in the Icon Editor [Rosenstein and Weitzman, 1990]. In order to visualize complex processes, Steamer presented a graphical interface to an underlying mathematical simulation of steam propulsion. Elements in the interface were *tapped* into (i.e., connected to) the simulation maintaining the visual representation of the relevant part of the mathematical model. This connection was implemented with standardized maps that would translate interface attributes into application values and back again. This tapping process is diagrammed in Figure 4.11 illustrating a horizontal position of a slider being mapped into the application variable *w*. This variable is then mapped back to the interface controlling the vertical position of a presentation element within a dynamic exploding diagram. In VIA, the simulation is the presentation i.e., there is no mathematical simulation, only constraints modifying the behavior of the individual presentation elements. With the additional data types, constraints, links and taps, the grammar can specify dynamic relationships between the elements of an automatic or user controlled presentation. Below are a few examples of these presentations.

Home repair procedure

The first example of a dynamic presentation is a repair procedure taken from a Popular Mechanics article [Henkenius, 1993]. This repair procedure is composed of three major steps, each containing a number of minor steps. Figure 4.12 shows the layout of the complete procedure on a large, high-resolution display. However, if we consider the characteristics of the output medium as part of the input to the parser, we can make the presentation sensitive to these output requirements.[4.4]

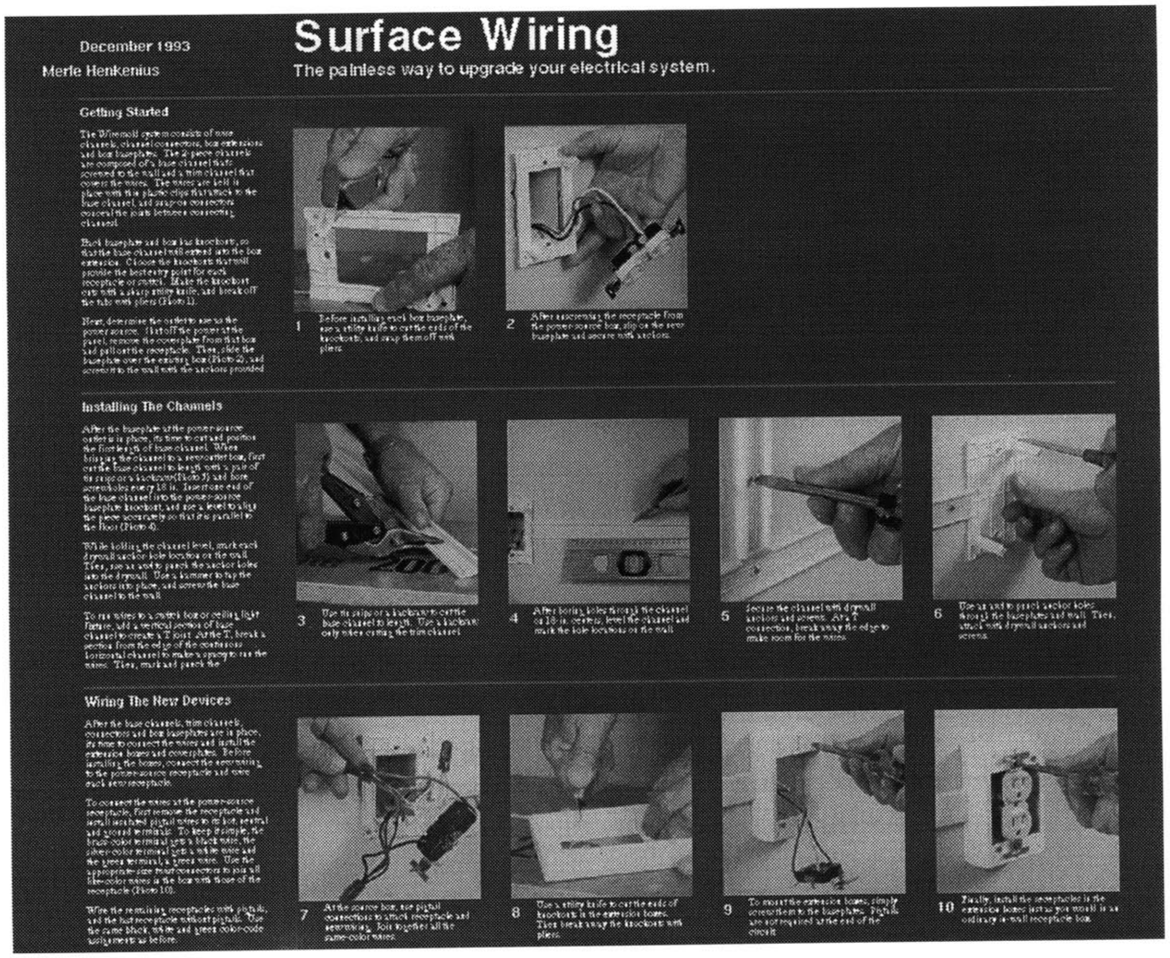
For instance, if this is a repair procedure being carried out by a person in the field with a hand-held digital assistant, the grammar can generate quite a different presentation. Figure 4.13 shows this second interpretation displaying the first step of the complete procedure. As part of the presentation, the horizontal bar at the top of the page becomes an active object which controls the presentation of elements in the repair procedure. As the user interacts with the bar, elements appear and disappear in the proper sequence. The display area of the device used to articulate the second presentation is much smaller than the first presentation. The grammar trades off the spatial resolution of the high-quality display with temporal resolution on the much smaller screen.

Note that traditional graphic design elements no longer need be static and in fact may play multiple roles in this dynamic medium. The graphic rule, or divider, is one that typically sets off areas of a layout from one another. In the above example, however, it also is used as a dynamic slider that can modify underlying simulation values. This, in turn, affects the way elements are visualized in the presentation. In design quite often elements play multiple roles expressing multiple meanings within a given design. The slider is just one of many techniques to control the dynamics of a presentation and is a common device used in many of the following examples.

On-line documentation

The next example in Figures 4.14-4.15 is similar to the first example trading off spatial vs. temporal resolution of the output environment. The small presentation automatically *plays* itself to the user. Instead

4.4 The mechanism for making a property of the output display part of the parse input is through the predicate mechanism of Relational Grammars. That is, certain rules will be fired only if a global predicate such as *large-display-p* is true or not.



December 1993

Merle Henkenius

Surface Wiring

The painless way to upgrade your electrical system.

Getting Started

The Wiremold system consists of wire channels, channel connectors, box extensions and box baseplates. The 2-piece channels are composed of a base channel that's screwed to the wall and a trim channel that covers the wires. The wires are held in place with thin plastic clips that attach to the base channel, and snap-on connectors conceal the joints between connecting channesl.

Each baseplate and box has knockouts, so that the base channel will extend into the box extension. Choose the knockouts that will provide the best entry point for each receptacle or switch. Make the knockout cuts with a sharp utility knife, and break off the tabs with pliers (Photo 1).

Next, determine the outlet to use as the power source. Shut off the power at the panel, remove the coverplate from that box and pull out the receptacle. Then, slide the baseplate over the existing box (Photo 2), and screw it to the wall with the anchors provided

1 Before installing each box baseplate, use a utility knife to cut the ends of the knockouts, and snap them off with pliers.

2 After disconnecting the receptacle from the power-source box, slip on the new baseplate and secure with anchors.

Installing The Channels

After the baseplate at the power-source outlet is in place, its time to cut and position the first length of base channel. When bringing the channel to a new outlet box, first cut the base channel to length with a pair of tin snips or a hacksaw (Photo 3) and bore screwholes every 18 in. Insert one end of the base channel into the power-source baseplate knockout, and use a level to align the piece accurately so that it is parallel to the floor (Photo 4).

While holding the channel level, mark each drywall anchor hole location on the wall. Then, use an awl to punch the anchor holes into the drywall. Use a hammer to tap the anchors into place, and screw the base channel to the wall.

To run wires to a switch box or ceiling light fixture, add a vertical section of base channel to create a T joint. At the T, break a section from the edge of the continuous horizontal channel to make an opening to run the wires. Then, mark and punch the

3 Use tin snips or a hacksaw to cut the base channel to length. Use a hacksaw only when cutting the trim channel.

4 After boring holes through the channel on 18-in. centers, level the channel and mark the hole locations on the wall.

5 Secure the channel with drywall anchors and screws. At a T connection, break away the edge to make room for the wires.

6 Use an awl to punch anchor holes through the baseplates and wall. Then, attach with drywall anchors and screws.

Wiring The New Devices

After the base channels, trim channels, connectors and box baseplates are in place, its time to connect the wires and install the extension boxes and coverplates. Before installing the boxes, connect the new wiring to the power-source receptacle and wire each new receptacle.

To connect the wires at the power-source receptacle, first remove the receptacle and install insulated pigtail wires to its hot, neutral and ground terminals. To keep it simple, the brass-color terminal gets a black wire, the silver-color terminal gets a white wire and the green terminal, a green wire. Use the appropriate-size twist connectors to join all like-color wires in the box with those of the receptacle (Photo 10).

Wire the remaining receptacles with pigtails, and the last receptacle without pigtails. Use the same black, white and green color-code assignments as before.

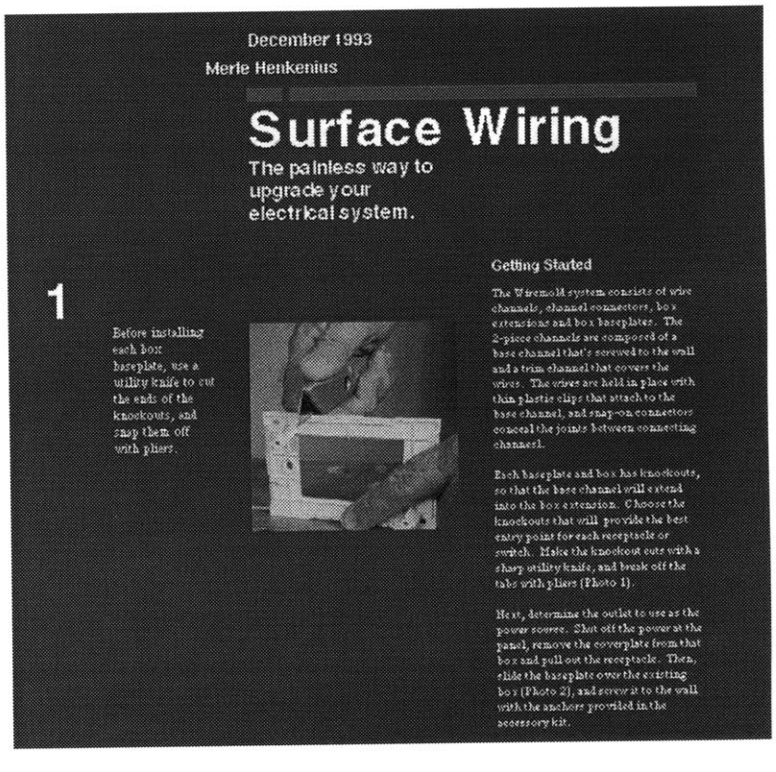
7 At the source box, use pigtail connections to attach receptacle and new wiring. Join together all the same-color wires.

8 Use a utility knife to cut the ends of knockouts in the extension boxes. Then break away the knockouts with pliers.

9 To mount the extension boxes, simply screw them to the baseplates. Pigtails are not required at the end of the circuit.

10 Finally, install the receptacles in the extension boxes just as you would in an ordinary in-wall receptacle box.

FIGURE 4.12
Presentation of a home repair procedure from Popular Mechanics magazine on a high-resolution display.



December 1993

Merle Henkenius

Surface Wiring

The painless way to upgrade your electrical system.

1

Before installing each box baseplate, use a utility knife to cut the ends of the knockouts, and snap them off with pliers.

Getting Started

The Wiremold system consists of wire channels, channel connectors, box extensions and box baseplates. The 2-piece channels are composed of a base channel that's screwed to the wall and a trim channel that covers the wires. The wires are held in place with thin plastic clips that attach to the base channel, and snap-on connectors conceal the joints between connecting channesl.

Each baseplate and box has knockouts, so that the base channel will extend into the box extension. Choose the knockouts that will provide the best entry point for each receptacle or switch. Make the knockout cuts with a sharp utility knife, and break off the tabs with pliers (Photo 1).

Next, determine the outlet to use as the power source. Shut off the power at the panel, remove the coverplate from that box and pull out the receptacle. Then, slide the baseplate over the existing box (Photo 2), and screw it to the wall with the anchors provided in the accessory kit.

FIGURE 4.13
Presentation of the same home repair procedure as in Figure 4.12 but constrained temporally as well as spatially. The user can manually step through the procedure by interacting with the horizontal bar at the top of the page.

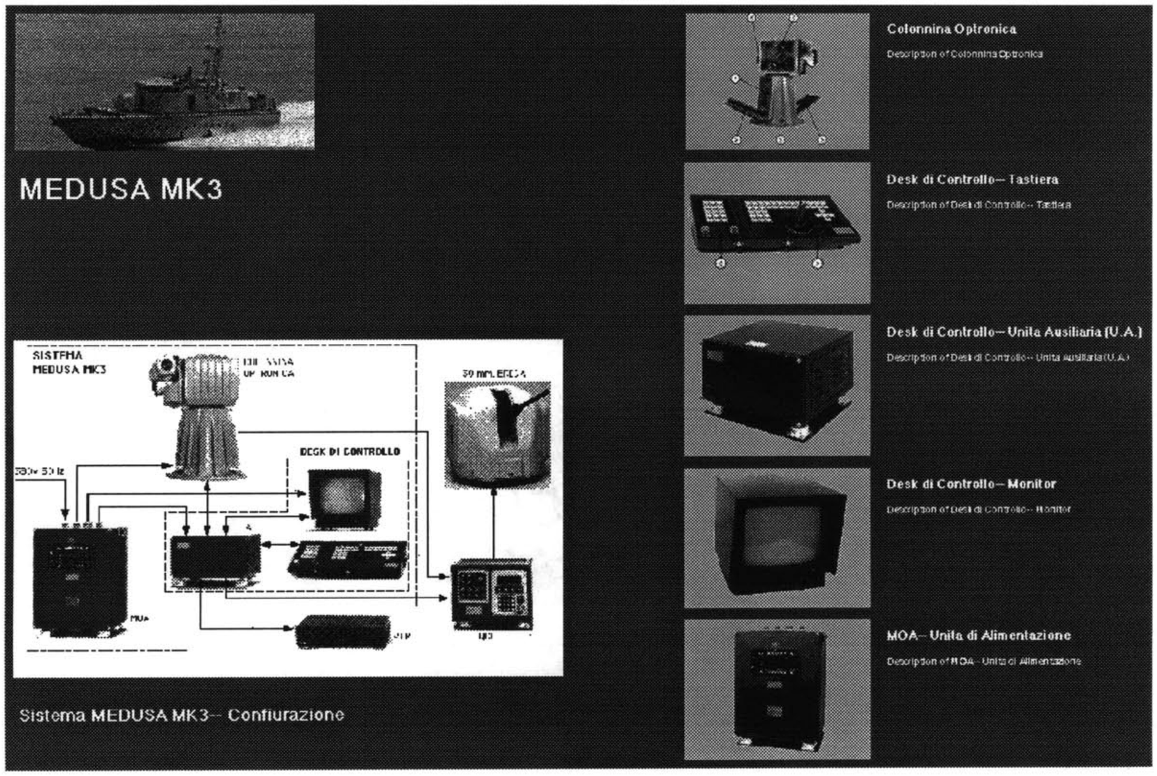

MEDUSA MK3
Sistema MEDUSA MK3-- Configurazione
Colonnina Optronica
Description of Colonnina Optronica
Desk di Controllo-- Tastiera
Description of Desk di Controllo-- Tastiera
Desk di Controllo-- Unita Ausiliaria (U.A.)
Description of Desk di Controllo-- Unita Ausiliaria (U.A.)
Desk di Controllo-- Monitor
Description of Desk di Controllo-- Monitor
MOA-- Unita di Alimentazione
Description of MOA-- Unita di Alimentazione

FIGURE 4.14
Automatic presentation of information from an online training manual. Here, the large display characteristics were taken into account displaying the complete contents.

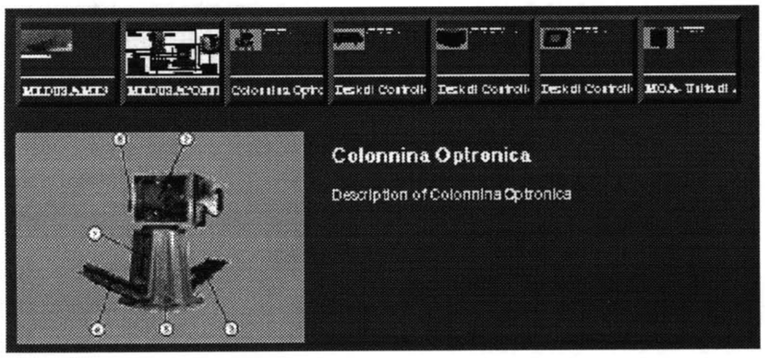


FIGURE 4.15
The same information as in Figure 4.14 is presented but on a much smaller screen. Buttons are utilized to link to other pages of the presentation.

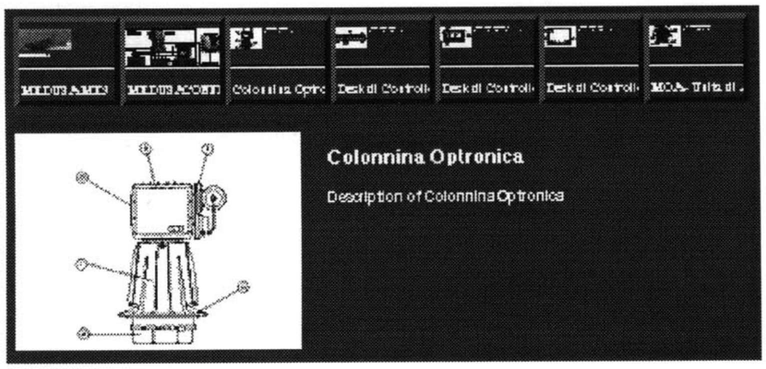


FIGURE 4.16
The grammar produces this variation in response to the user's level of expertise. Given the same input and small screen as in Figure 4.15, some filtering has occurred to layout more relevant diagrams.

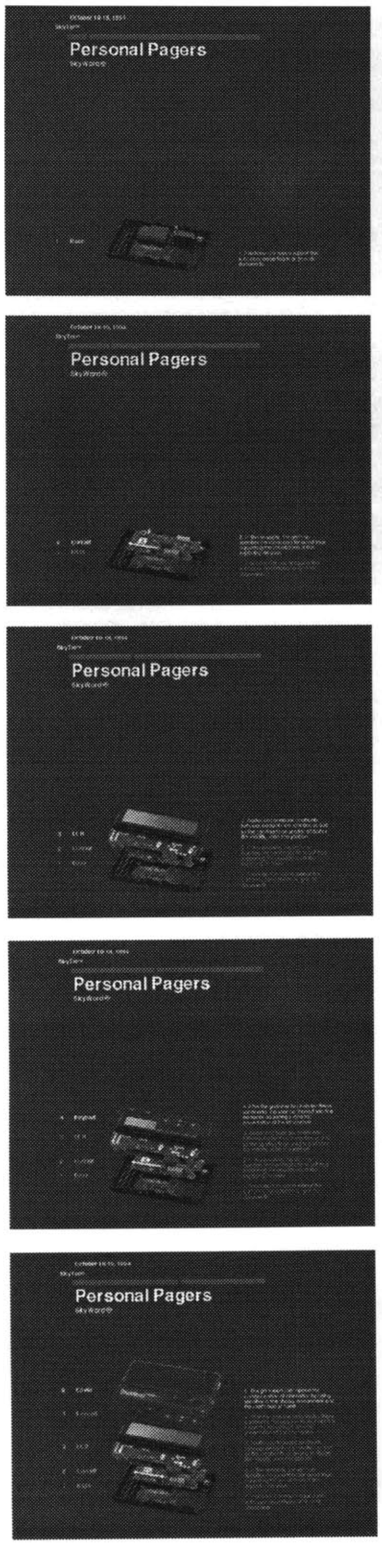


FIGURE 4.17
Exploding diagram of a personal pager.

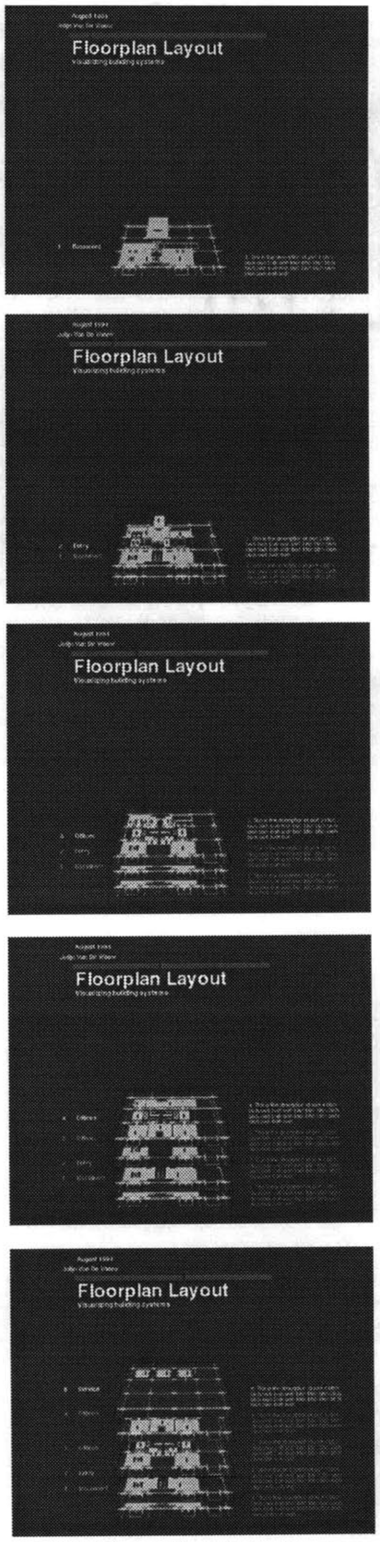


FIGURE 4.18
Exploding diagram of architectural floor plans.

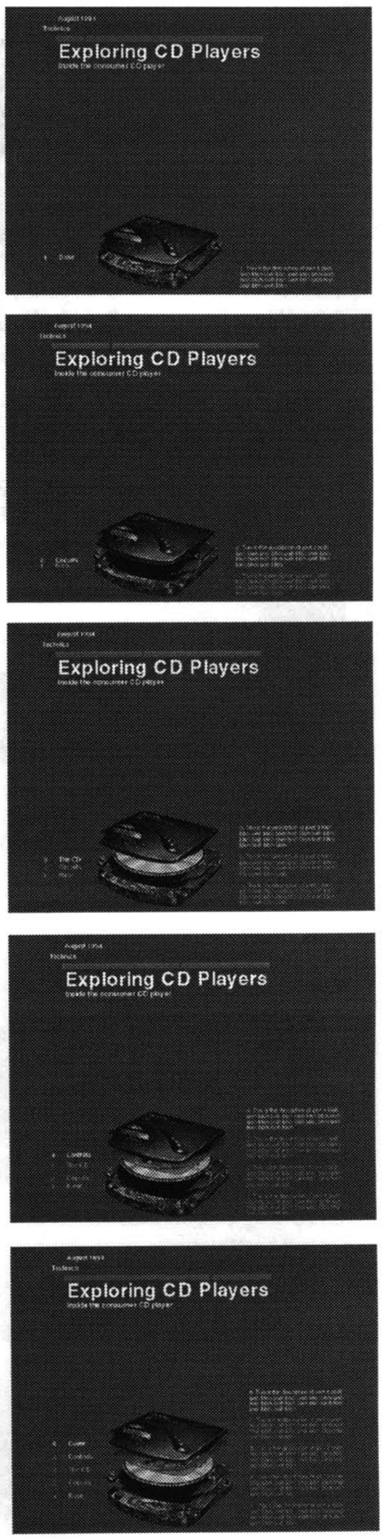
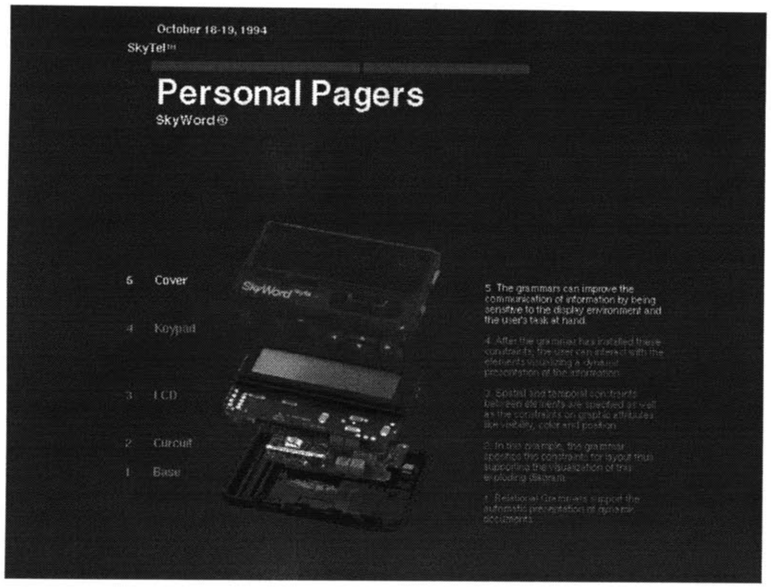


FIGURE 4.19
Exploding diagram of a portable home CD player.


October 18-19, 1994

SkyTel™

Personal Pagers

SkyWord®

6 Cover

4 Keypad

3 LCD

2 Circuit

1 Base

5. The grammars can improve the communication of information by being sensitive to the display environment and the user's task at hand.

4. After the grammar has installed these constraints, the user can interact with the elements visualizing a dynamic presentation of the information.

3. Spatial and temporal constraints between elements are specified as well as the constraints on graphic attributes like visibility, color and position.

2. In this example, the grammar specifies the constraints for layout thus supporting the visualization of this exploding diagram.

1. Relational Grammars support the automatic presentation of dynamic documents.

FIGURE 4.20
Illustration of an exploding diagram of a personal pager. The presentation changes elements visibility, color and position to create this dynamic layout (same as last image in Figure 4.17).

of using a slider, buttons are created for the various pages presented and form a context of the complete presentation. These are sequentially positioned on the screen and can be used to jump back through the document interactively.

In addition, this grammar does some filtering of data in order to customize the presentation to the expertise level of the user. The demonstration allows the user to select *Novice* (Figure 4.15) or *Expert* (Figure 4.16) to suggest grammar rules sensitive to a user's profile. Reparsing will then present different data from the same set of input. In this way, the grammars can help create presentations that are not only sensitive to output display characteristics, but also sensitive to whom the information will be presented to how the information will be utilized.

Exploding diagrams

In the last two examples, only the visibility of elements is modified to produce the presentations. Other graphic attributes of elements can also be dynamically modified in a presentation. A grammar that encodes the knowledge of exploding diagrams was created and then used on multiple datasets. These datasets can be seen in Figures 4.17-4.19. Once the grammar has been defined, it is relatively simple to create new presentations of objects with the same layered internal structure. These examples illustrate the dynamic modification of visibility, position, and color creating a simulation in order to better understand complicated information. As the user interacts with the horizontal slider at the top of the displays, the presentation moves

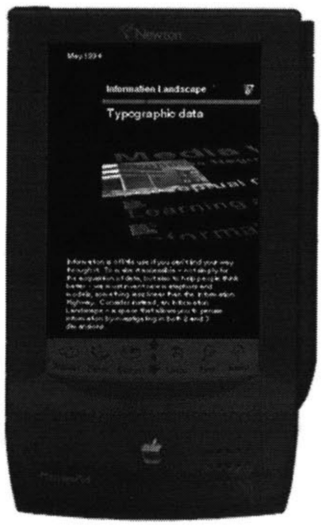


FIGURE 4.21
The presentation of the Information Landscape on a personal digital assistant. Affordances and constraints of this device have been taken into account in the design of this information. (This composite image simulates the presentation on a PDA).

pieces farther apart, new elements are revealed, and the appropriate text is highlighted with the different layers.

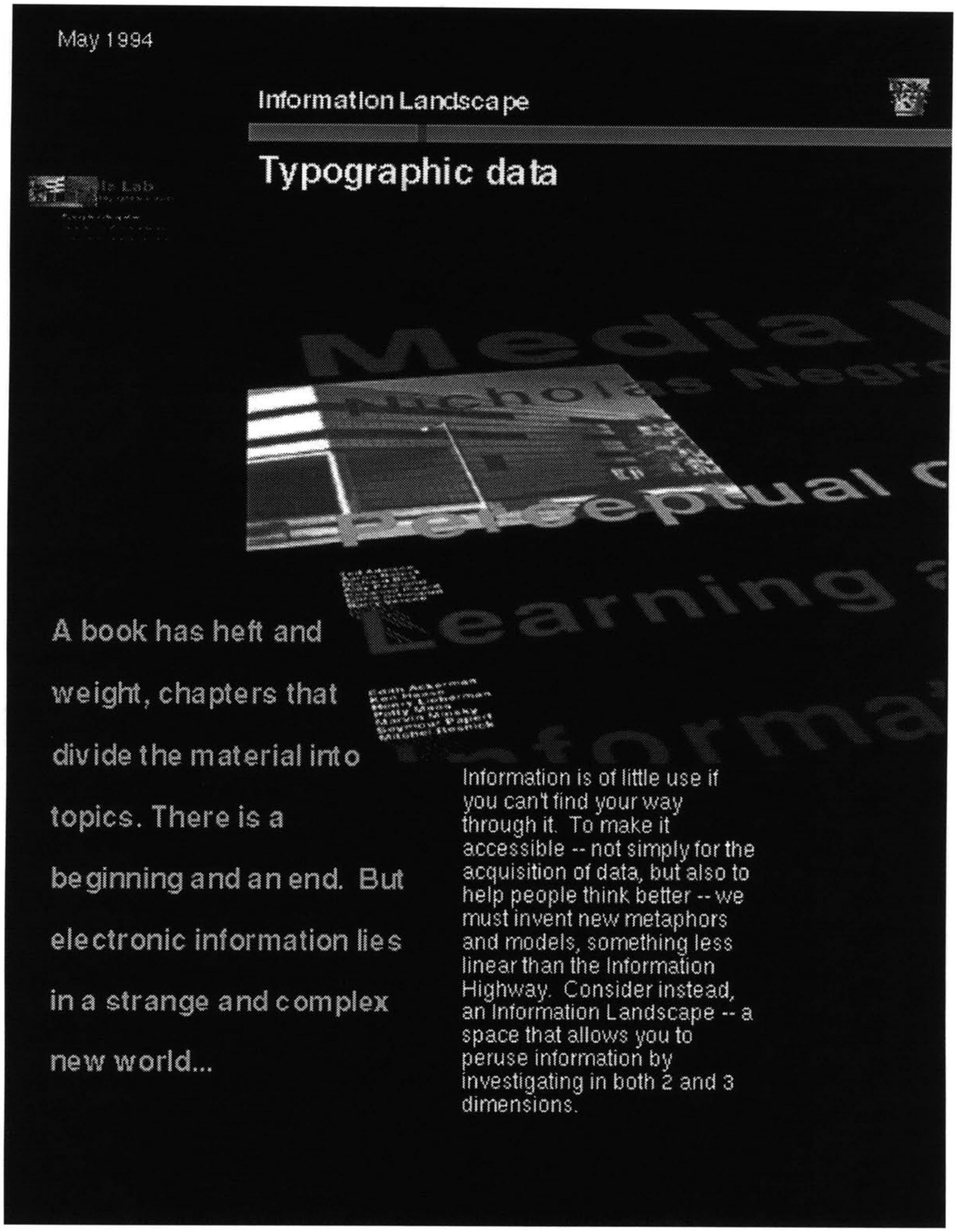
In general, the relationships in the input reflect the semantics of the data. In this example, the data is described as a multi-layered object. Using this description, the grammar can produce alternative presentations to reveal this aspect of the information being presented. In fact, multiple visualizations could result from this input such as exploding diagrams, storyboards, or transparent images. The grammar can maintain these different derivations to support multiple design alternatives.

Multi-page/multi-resolution presentations

The next example uses the demonstration, *Information Landscape*, which is comprised of work from the Visible Language Workshop that was shown at TED5.[4.5] The presentation consists of three separate pages describing visualizations of typographic, spatial and symbolic data. QuickTime movies and audio icons are included in the input. Within this grammar there are three discrete formats a presentation can take: either a large vertical display much like an 8.5 x 11 sheet of paper, a NTSC resolution display with horizontal aspect ratio, or a very small black and white display simulating a personal digital assistant (PDA). Figure 4.21 illustrates the PDA display while Figure 4.22 illustrates the high resolution, color display. A single page on the large display occupies multiple pages on the smaller PDA display. In VIA, resizing the background of a presentation to another format results in a reparse of the input. This produces a new presentation in one of the other sizes. When resizing within a format, elements maintain the constraints installed by the grammar without a reparse of the input. In addition, input control on the large display is through the mouse interacting with the horizontal slider while keyboard input is used to navigate through the pages on the PDA. This suggests the more limited input capabilites of some display environments, such as a settop box controlled by remote control. This same grammar is used to present advertising information on new cars shown in Figure 4.23.[4.6] The result is a similar three page color presentation or a six page PDA presentation.

4.5 Technology, Entertainment and Design conference in Monterey, 1994.

4.6 Buick Riviera, General Motors, 1995.



May 1994

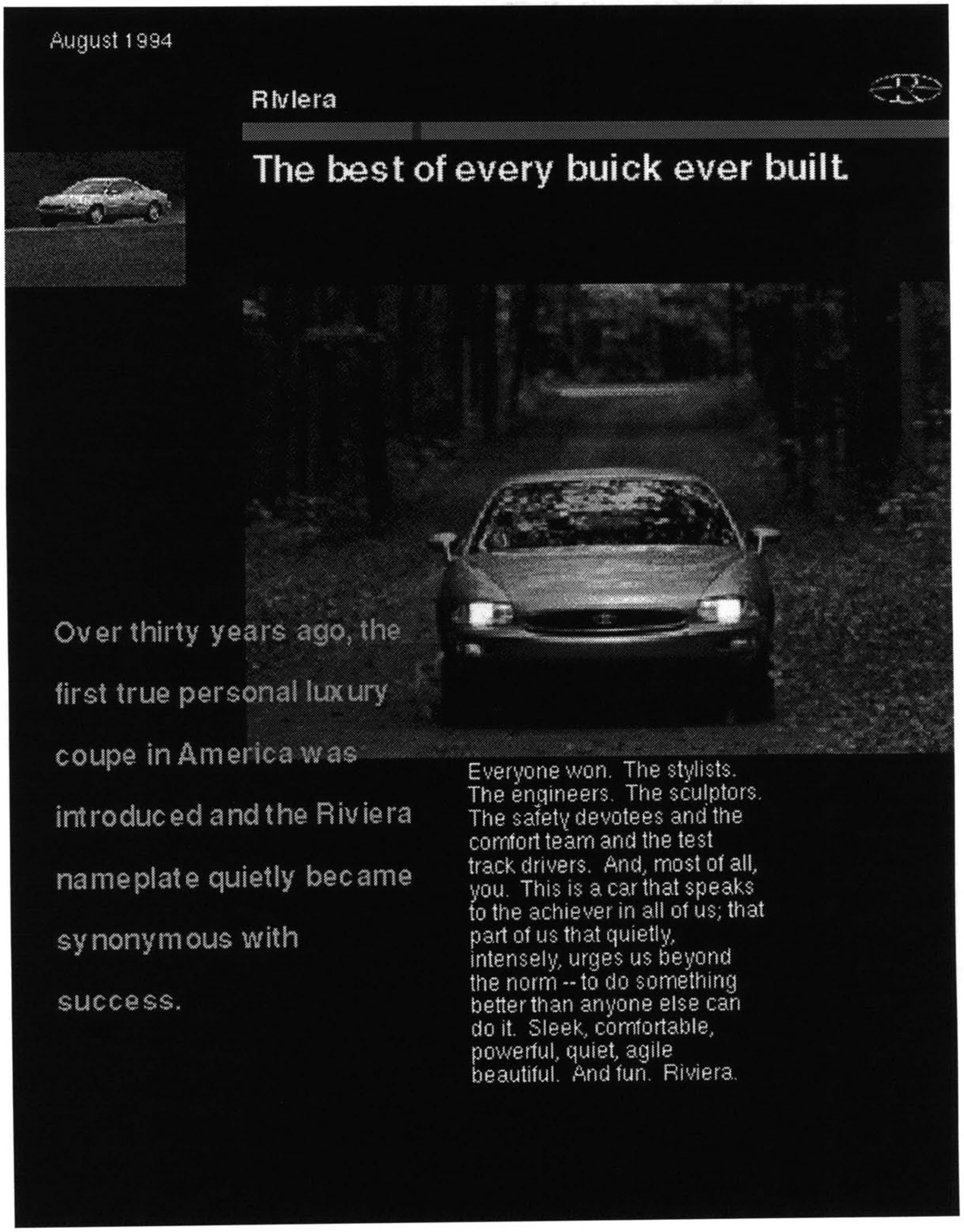
Information Landscape

Typographic data

A book has heft and weight, chapters that divide the material into topics. There is a beginning and an end. But electronic information lies in a strange and complex new world...

Information is of little use if you can't find your way through it. To make it accessible -- not simply for the acquisition of data, but also to help people think better -- we must invent new metaphors and models, something less linear than the Information Highway. Consider instead, an Information Landscape -- a space that allows you to peruse information by investigating in both 2 and 3 dimensions.

FIGURE 4.22
*The presentation of the **Information Landscape** research at the Visible Language Workshop. This first page of a three page presentation uses a full color, high-resolution display and includes QuickTime™ movies and audio icons.*



August 1994

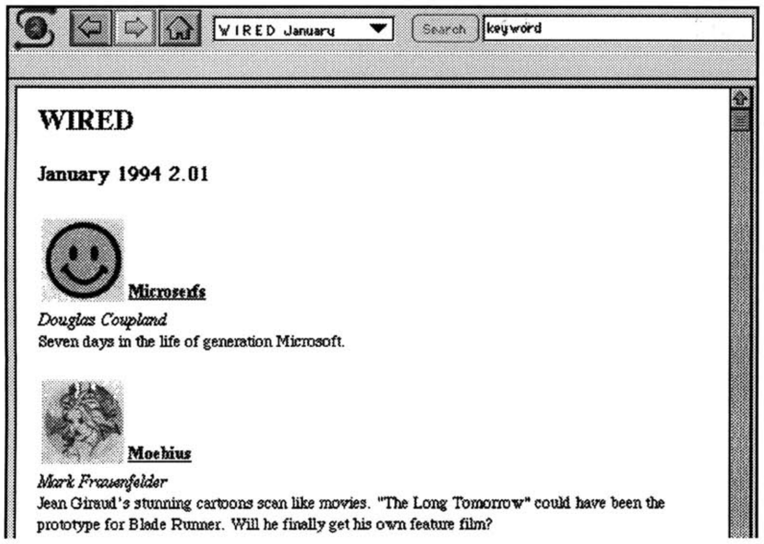
Riviera

The best of every buick ever built.

Over thirty years ago, the first true personal luxury coupe in America was introduced and the Riviera nameplate quietly became synonymous with success.

Everyone won. The stylists. The engineers. The sculptors. The safety devotees and the comfort team and the test track drivers. And, most of all, you. This is a car that speaks to the achiever in all of us; that part of us that quietly, intensely, urges us beyond the norm -- to do something better than anyone else can do it. Sleek, comfortable, powerful, quiet, agile beautiful. And fun. Riviera.

FIGURE 4.23
Using the same grammar as in Figure 4.21 and Figure 4.22, an advertisement of new car information is presented.



WIRED

January 1994 2.01

Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft.

Moebius
Mark Frauenfelder
Jean Giraud's stunning cartoons scan like movies. "The Long Tomorrow" could have been the prototype for Blade Runner. Will he finally get his own feature film?

FIGURE 4.24
Presentation of a table of contents from WIRED magazine with links in the Mosaic environment.

Dynamic hyperlinks

Another way to support interaction dynamics is through the use of hypermedia links. This class of documents provides easy access to related information by simple link traversal. A popular way to generate these documents is by creating HTML files and viewing them within a browser such as NCSA Mosaic [NCSA, 1993]. Figure 4.24 illustrates an on-line table of contents "home page" presented in this way. The titles of articles in this example are linked to the actual on-line article located on the World Wide Web.[4.7]

Input to this presentation is similar to the original table of contents examples in Figures 4.5-4.6. In this example, however, the grammars support the generation of an HTML file. The structural hierarchy of the information is supported in the hyperlinked articles. In addition, the grammar accommodates variation in the delivery environment based on its software capabilities. Here, the graphically more limited delivery environment of an HTML viewer is taken into account.

4.7 Accessed through WIRED's World Wide Web server at *http://www.wired.com*.

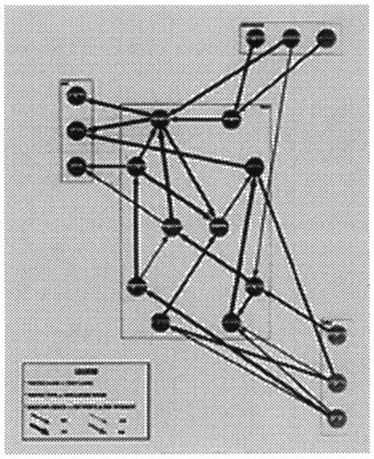


FIGURE 4.25
A diagram and legend automatically produced by the ANDD system.

Related work

VIA characterizes the generation problem as creating one or more instances of a complete design from a semantic representation of the data. This may, in fact, be done through parsing and VIA currently uses this *generation as parsing* paradigm to automatically create designs of dynamic documents. Other approaches typically use a rule-based approach for the creation of their presentations. In addition, a number of systems focus on the more general and comprehensive multimedia generation problem. Some typical examples of these systems are described next and compared to VIA.

Automatic presentation

At the fully automatic end of the scale of interaction, there are many different examples that create designs without intervention. A Presentation Tool, APT, [Mackinlay, 1986] was one of the first systems to do automatic generation based on a language-inspired paradigm. APT uses a rule-based system, MR, for its implementation. It automatically generates tables and charts from a relational database. APT uses criteria for different data types like nominal, ordinal, and quantitative, and rates their ability to express the data type and their relative effectiveness in doing so.

ANDD [Marks, 1991] automatically generates network diagrams to communicate information represented in arbitrary attributed graphs and is part of a multimedia explanation facility. It uses special pragmatic directives provided to the system. These directives convey the communicative intent to emphasize one or more structural properties, such as hub-like properties emphasizing both inputs and outputs, and nodes which are inputs relative to a specified node in a directed graph.

Two other related rule-based paradigms used for the automatic generation of graphic form include expert systems [Gero, 1985; Shapiro and Geller, 1987; Schmitt, 1987; Coyne, et.al., 1990] and shape grammars. Predikt applies expert-system technology for the preliminary synthesis and critique of kitchen designs [Oxman and Gero, 1987]. The system is able to convert between semantic and graphic representations to articulate the designs. Shape grammars [Stiny, 1980; Mitchell, 1990] are interesting in that they use the language of the designer, i.e., shapes, as the building blocks of domain rules. There have been shape grammars developed to explore a set of conventions [Knight 1989; Flemming, 1987], and others to extend an existing corpus of work such as Palladian villas [Stiny and Mitchell, 1978], Mughul gardens [Stiny and Mitchell, 1980], and Wright prairie homes [Koning and Eizenberg, 1981].

One characterization of these automatic systems is the fact that the domain can be expressed as a set of discrete rules that easily encode design knowledge. In architecture, this has been limited to applications in highly constrained specialties such as kitchen design [Oxman and Gero, 1987] or simple floor plan layout [Mitchell, et al., 1976]. Other domains, like information graphics [Mackinlay, 1986] and information design as described in this research, also lend themselves to this approach.

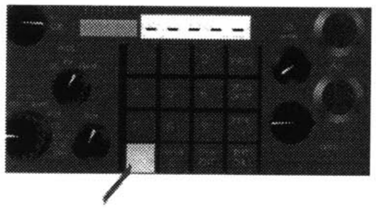


FIGURE 4.26
COMET illustrating the automatic presentation of the repair of an Army field radio.

Intelligent multimedia presentation systems

Intelligent multimedia presentation systems go beyond simple automatic generation systems by processing multimedia queries and generating multimedia answers in an intelligent, knowledge-based manner [Maybury, 1993]. An intelligent multimedia presentation must not just present the information, but must design it so that it communicates information using the techniques and capabilities of the various media to achieve communicative purposes and support users in performing their tasks.

As mentioned previously, the process of generation can be divided into four major categories: *content selection*, which determines what to say; *media allocation*, which identifies in what media to say it; *media realization* (or articulation), which describes how to say the elements in a particular media; and *media coordination*, which coordinates each of the separate media elements in a presentation. Various types of information must be represented in order to support each of these tasks. Models of the media characterize the strengths and weaknesses of the information presentation. Models of the user help to identify preferences and intentions. Models of the dialog history supports information from user's interaction with the system. Models of the situation track system parameters such as system load and media availability. This portion of VIA focuses on the problem of media realization.

COMET, COordinated Multimedia Explanation Testbed [Feiner, 1993; Karp and Feiner, 1990], is a knowledge-based system that produces interactive and coordinated explanations in the context of an intelligent multimedia presentation system. The system combines text with 3D graphics to explain the operation and maintenance of Army field radios. It uses "media generators" which can communicate with each other to produce a presentation. The underlying generator within COMET is called IBIS, Intent-Based Illustration System, [Seligmann and Feiner 1989; 1991].

IBIS illustrates the communicative goals specified from COMET's media planners. IBIS currently can show absolute and relative locations of objects; physical properties, such as size, shape, material and color; state, such as knob settings; change of state, such as changing a knob setting; and a variety of actions, such as pushing, pulling turning, and lifting. In creating a presentation, IBIS controls all aspects of the creation process, including the objects and their attributes, the lighting specifications, the rendering style, the viewing specification, and the structure of the presentation itself.

IBIS uses a generate-and-test approach to creating the presentations. The IBIS rule base contains at least one design rule for each communicative goal. Each of these rules invokes a set of stylistic strategies that specify high-level visual effects, such as highlighting an individual object. The lower-level rules create and manipulate the graphical representations of objects included in the presentation and modify lighting, viewing and rendering specifications. IBIS uses its rules to evaluate the success of each task. If a strategy doesn't succeed, it uses backtracking to try other visualizations. For example, maintaining visibility is a difficult task in three dimensions. Elements can obscure or be obscured by other objects. IBIS must determine the success of these incremental design decisions and if goals are violated, modify the illustration to accommodate the input requirements.

A major distinction of these larger multimedia presentation systems is the knowledge bases used in support of the generation problem. COMET uses three different knowledge bases in generating an explanation: a static representation of domain objects and actions, a diagnostic rule-base, and a detailed geometric knowledge used in graphics visualization. In addition, user models and models of previous discourses are maintained. COMET's content planner uses text plans or rhetorical schemas [McKeown 1985; McKeown et al. 1990] to determine which media-independent information from these knowledge sources should be used in the explanation for a particular request.

Another general multimedia platform for three dimensional generation, similar to COMET in its approach and scope, is WIP [Andre, et al., 1993]. WIP approaches the problem of multimedia design as a planning problem to achieve coherent multimodal presentations of text and graphics. The realization of the layout in WIP uses DeltaBlue, the same constraint satisfaction algorithm used in VIA. WIP encodes the graphical design knowledge through constraints which express semantic/pragmatic relations (alignment, grouping, symmetry or similarity) and geometrical/topological relations (absolute and relative position). In WIP each piece of information is assigned sequentially to its generators. The evaluation of potential success based on the knowledge of other elements generated so far, are sent back to the planner to determine media assignment.

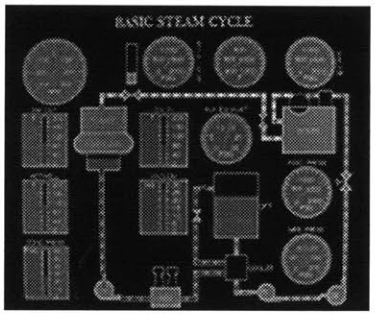


FIGURE 4.27
STEAMER's dynamic interface elements are tapped into a mathematical simulation of the steam process. This view illustrates the high-level view of the basic steam cycle.

The main difference between comparable parts of other automatic generation systems and the work described here has to do with the nature of the rules and the issue of control. Other systems employ some form of forward-chaining, rule-based system. These systems use rules and metarules to control the search of the design space and use the generate-and-test paradigm to determine the appropriateness of a solution. More often than not, this aspect of these systems receives little attention in the literature, perhaps because adequate control mechanisms can be difficult to specify. On the other hand, VIA uses an independently motivated parsing algorithm. The issue of control is thus folded into the more general problem of finding efficient parsing algorithms for higher-dimensional grammars, a continuing research topic. As progress is made on this front, the results can be incorporated into future versions of VIA. In the meantime, authors of the rules used in VIA need not concern themselves with issues of control.

Dynamic interface vs static layout

As documents migrate into the digital domain, they will evolve into a dynamic interface to multimedia information. In order to support this view, VIA allows each presentation element to be connected to a value within an application or simulation. In this way, each element in the interface has the potential to become dynamic by visualizing values within the application or simulation and/or modifying those values through user interaction. For example, in a multimedia training document, the mathematical simulation of a device to be repaired could control how the element is presented and how it behaves when the student interacts with the document. Some of this behavior can be encoded in the output of the grammar constraints while others will remain domain specific.

The separation of interface from application is common to the position developers of user interface management systems (UIMS) take. It enhances modular development and supports automatic construction and reuse of interface elements. These similar approaches separate the dynamics of the application from the specific presentation to the user. Early work in the visualization of dynamic interfaces separate from the application can be found in *Steamer* [Hollan, et al., 1984] and the Process Visualization System [Foley and McMath, 1986].

Steamer, an interactive, inspectable, simulation-based training system, was one of the first systems to support the capabilities of dynamic interfaces. It was used as an instructional aid in teaching the complex and potentially dangerous task of steam propulsion. Dynamic interface icons were used to monitor and control the simulation. Different views of the simulation illustrated different levels of abstraction of the process. Icons, tied to specific simulation variables, reflected the states of different components like pumps (e.g., *on*, *off*, or *standby*), valves (e.g., *open* or *closed*) and dials (e.g., continuous numeric values). The user could interact with the simulation by clicking on elements in the interface. An otherwise opaque process became visible and inspectable through the use of these dynamic interface icons. This framework was generalized to support the visualization of real-time data (e.g., UNIX operating systems), to control hardware devices (e.g., video switchers), and to visualize database information (e.g., data from files of a multiprocessor simulation).

[Rosenstein and Weitzman, 1990] extended the capabilities of these interface icons by creating an editor to construct new icons exhibiting new behaviors without coding. This dissertation incorporates these concepts by tapping presentation elements to a dynamic display process. Attributes of presentation elements are dynamically modified according to the state of this controlling process. Elements can change their graphic attributes such as visibility, position, size and color in response to the process.

Typical UIMS in fact support this same separation of application values and interface elements [Arens, et al., 1988; Wiecha, et al., 1990; Kim and Foley, 1993]. DON [Kim and Foley, 1993] consists of an application model (containing both data and control models), a design process model supporting top-down iterative design and graphic design knowledge to support the layout process. A rule-based expert system approach is used. ITS [Wiecha, et al., 1990] is similar but emphasizes the usability of the style language by the interface designer. By making the underlying rule-base more accessible, the designer has the ability to modify the rule-base itself.

Coordinating complex temporal relationships has been attempted in some of the larger multimedia systems. COMET has been extended to handle reasoning about temporal media for speech and animation. Using the temporal logic from [Allen, 1983], the same basic representation used in VIA, constraints can be maintained between events allowing for temporal presentations as well as special transitions between media presentations. [Karp and Feiner, 1990] examines these complex relationships in the context of building animations between elements of the presentation. This includes the temporal synchronization of special editing effects (e.g., cuts, wipes, fades, dissolves), camera control movements, and multiple views of the same information.

Summary

If design can be expressed as a formal language, the rule-based approach of generative grammars can be used to support the automatic presentation of information. In fact, this presentation can be designed for dynamic environments where the hardware, software, user and even data are likely to change. What has been described in this chapter is a vision of this automatic construction of documents based on a visual language.

Automatic presentation in VIA begins with data that has been classified and has relationships between the elements. The system then parses this information into a derivation tree which represents the structure of the document to be presented. A translation phase then produces a set of output forms, that when evaluated and solved, creates spatially and temporally constrained media objects. This scenario changes the role of the designer from designing individual documents into the role of designing languages for layout. This activity of meta-design creates visual languages, which can then create presentations automatically and without intervention. The following chapters complete the vision of document creation with interactive design paradigms and a description of the tools necessary for designers to build these languages interactively.

Interactive design of information

5

Introduction

To envision information ... is to work at the intersection of image, word, number, art. The instruments are those of writing and typography, of managing large data sets and statistical analysis, of line and layout and color. And the standards of quality are those derived from visual principles that tell us how to put the right mark in the right place.
EDWARD TUFTE,
ENVISIONING INFORMATION

Computers have provided access to tools for doing tasks that have traditionally only been done by design professionals. We should no longer expect that users of these tools are designers or have the necessary design expertise. In addition, professional designers need augmented tools to support their activities in the design process. Therefore, as design moves from traditional media to the electronic studio, representation of design knowledge becomes crucial in order to support a dialog between designer and machine. This chapter describes the application of the grammar formalism to support the interactive design process. Interactive support includes simple improvement of users' actions, top-down refinement of design solutions, graphic completion of partial design decisions, and verification of partially complete artifacts. Here, the focus is on interactive improver-based support.

Recently, the emergence of computational systems that embody languages of design to perform these tasks automatically [Feiner, et al., 1992]. Languages to support design generation are only one possible use of these formalisms. Supporting the general design process includes many other tasks such as parsing of design input [Myers and Buxton, 1986; Pavlidis and Van Wyk, 1985], supporting designers with critiques [Fischer, et al., 1988], prediction of design decisions and the exploration of design alternatives [Kochhar and Friedell, 1990]. The emphasis in this chapter is on interaction. As a designer builds up design alternatives, Relational Grammars can be used in various ways to support the process. Such a system can assist in the implementation of details during design exploration as well as suggest or enforce design requirements.

In this chapter, the general applicability of Relational Grammars for supporting the interactive design process is discussed. First, an overview of the architecture of VIA is presented focusing on the description of this interactive scenario. Then, an example interaction is described that uses this formalism to support the task of document creation. Design support takes the form of graphic inferences on partial input during design interaction with the user. Because of the nature of the rule definitions, design elements can be roughly "sketched" without concern for their specific details. During the interaction process, the system installs constraints, adds default attributes, and/or builds higher-level composite structures out of the original input. A generalized algorithm for the integration of grammar-based parsing in an interactive design environment is presented. The grammar used in the scenario is then described in detail. Additional examples are presented that illustrate generalized grammars to support the authoring of on-line documents, and the use of a predictive parser to enhance the interactive scenarios with design completion.

Architectural overview

Relational Grammars are central to the vision of document creation for both automatic and interactive design scenarios. The vision of document creation described in the previous chapter describes how Relational Grammars are used in the automatic production of documents. In addition, grammars can use this same formalism to also support the interactive authoring of the same class of multimedia documents.

In an interactive scenario, the designer carries out actions in a traditional direct-manipulation editor. VIA augments this editing process with the parsing action. When conditions arise in the input, in terms of the element types and their relationships, the parser produces actions to help improve the design. These actions can be taken immediately or placed on an agenda for later processing. Typically, these actions install constraints between the individual input and are used to clean up a roughly sketched design. These grammars can support a given style of layout or be more general and clean up the input based on alignment and sizing of input elements.

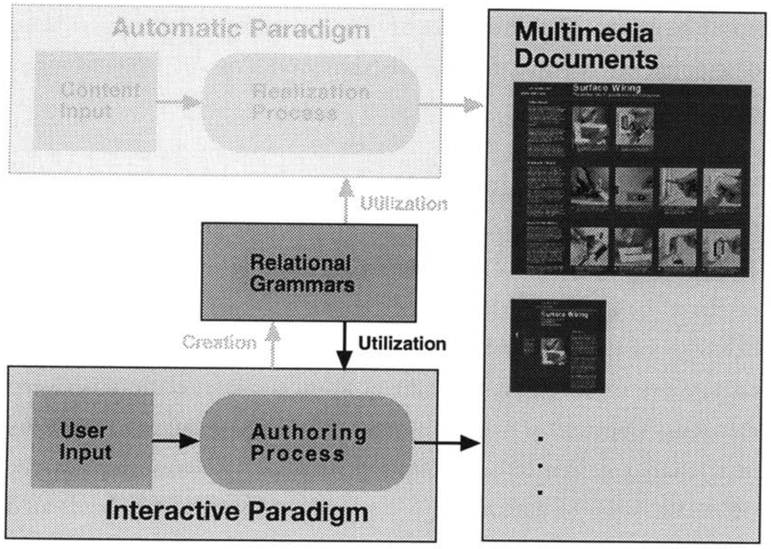


FIGURE 5.1
Vision of an integrated environment for interactive authoring and automatic articulation of multimedia documents. Here, the interactive scenario is highlighted.

As seen in Figure 5.1, the grammars play a central role in this vision of interactive design. The examples described in this chapter use classified elements (i.e., their type information) and the spatial relations between the elements to build composite structures. These composite groups incorporate constraints for layout specification. In addition, this interactive scenario can be used in a meta-fashion to assist in the authoring of other grammar rules which then can be incorporated back into other grammars. This later technique is the topic of the following chapter.

Example page design scenario

This section presents an example interaction with the system in supporting the design of the layout of a page from a book. The rules of this grammar may be inferred from the example page illustrated in Figure 5.2. The rules in this language capture this style and embody various layout articulation techniques such as graphic separators (or graphic rule bars) above chapter titles and section headings, relative font sizes, and the use of margins for images. This grammar will form the basis for this scenario and the following discussion.

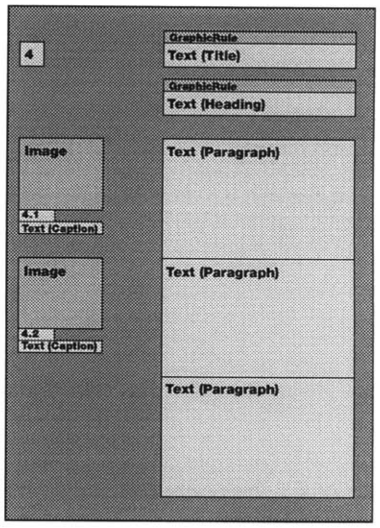


FIGURE 5.2
An example page layout based on the graphic design of [Mitchell, 1990] that is the model for the grammar described in the first example.

The interaction sequence begins by selecting a primitive lexical element from a menu and adding it to the layout. In this example, there are four basic categories of input which include: text, number, image and graphic rule. The elements are sized and placed within the workspace. As design input proceeds, the system parses the input and automatically identifies improvements to the design creating new composite structures. The creation of these composite structures is a form of graphic inferencing that installs constraints on the original input elements.

The constraint system uses the same substrate as the automatic paradigm and can be easily extended to include new graphic constraints. The graphic attributes of elements participating in an improvement triggered by a rule are constrained to act together as defined by the behavior of the new composite object. Typically, multiple graphical constraints are used to enforce the position and size relations between elements. Constraints may also make individual changes to elements (e.g., changing their color or font specification). Relationships can be defined so that the elements involved only roughly match the desired requirements. In this way, input can be loosely *sketched* and the application of the rules will clean up the input.

In this scenario, the user begins to create a *Figure* by adding a *Text* item and placing a *Number* roughly above and left aligned to it. This interaction automatically creates a *Caption* composite structure and in the process, the elements are left aligned exactly and placed immediately adjacent to one another. By continuing to add an *Image* element above the newly formed *Caption*, a *Figure* is then created and the *Image* is left aligned and placed above the *Number* element.

No assumptions were made as to the appropriate interaction paradigm and VIA allows the user to select between multiple interaction styles. The user can request automatic rule selection and invocation, as described above, or the user can postpone rule firing using an interactive graphical agenda as seen in Figure 5.3A-C. The interactive agenda is motivated by the desire to bring the user back into the toplevel loop. The agenda is provided to deal with ambiguous derivations in a nonobtrusive manner. This queuing mechanism puts the user in control and permits the exploration of the design space without interruption.

In the interactive mode, unfired actions accumulate on the agenda. The agenda is visualized as a set of buttons, each illustrating before and after states of the elements that participate in the pending inference. The user is free to ignore the suggestions offered by the system, or click on the most appropriate action. By selecting a button, the inference is performed and button is removed from the agenda. Figure 5.3 illustrates a sequence of interactions using the agenda to build the same *Figure* composite described above. In Figure 5.3A, the

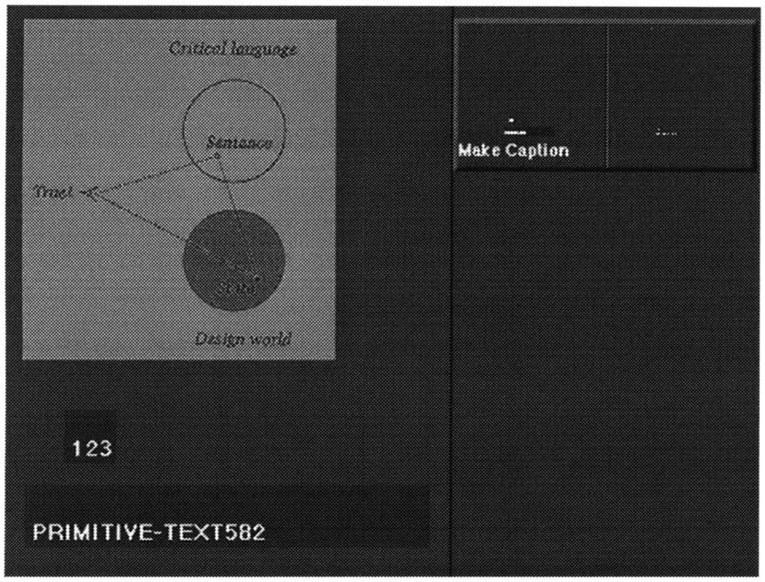


FIGURE 5.3A
*Interactive design support using bottom-up parsing of designer's actions while creating a **Figure** composite element. The agenda with one pending action, to make a **Caption**, is illustrated on the right with three design elements in the work space illustrated on the left.*

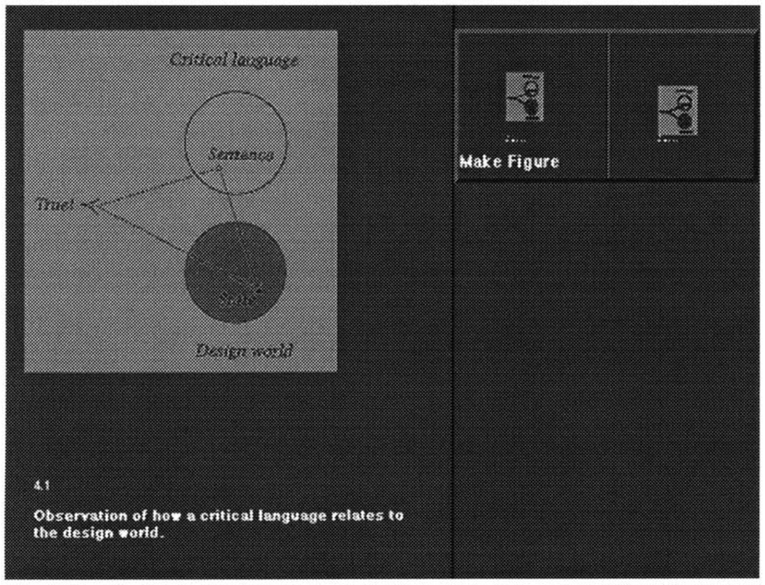

Critical language
Sentence
Truel
State
Design world
Make Figure
4.1
Observation of how a critical language relates to the design world.

FIGURE 5.3B
*The user selects the agenda item in Figure 5.3A and the system cleans up creating a **Caption**. The previous agenda item is replaced with one to make a **Figure**.*

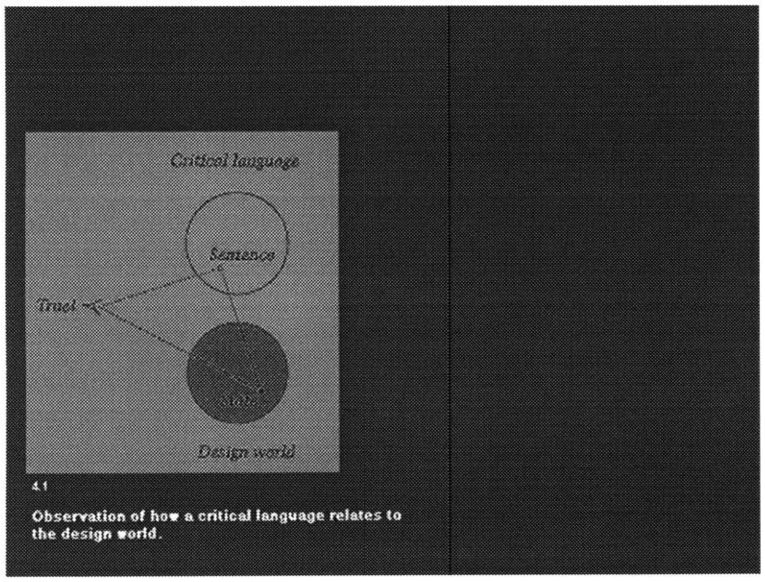

Critical language
Sentence
Truel
State
Design world
4.1
Observation of how a critical language relates to the design world.

FIGURE 5.3C
*The user selects the agenda item in Figure 5.3B and the system completes the design of a **Figure** element. No agenda items remain.*

user has created and placed three items, an *Image*, a *Number* and a *Text* item, and the agenda (on the right side of the figure) reflects one pending action, to create a *Caption*. After clicking on this "Make Caption" button, Figure 5.3B shows the resulting *Caption* element and a new agenda item to create a *Figure* composite element. Figure 5.3C shows the final *Figure* element with all constraints (position, size, and font specification) applied.

As described before, an interactive design environment must also support non-monotonic changes to the input. The user may move, resize, or delete any lexical input. This item may also be a composite element created by the parsing action. The parse table reflects these changes and pending derivations that are based on the modified or deleted element are removed from the agenda.

Toplevel interaction loop

The integration and maintenance of a working set of valid actions is a significant requirement for the system architecture. These actions are discovered through parsing in the toplevel interaction loop. There are actually three data structures which need to be maintained: the current *input state* (simple or composite objects that the user or system has created along with a data structure to maintain the object relationships); the *parse table* (the set of partially and completely matched rules indexed by input elements); and the system *action agenda* (actions stemming from the derivations admitted by the grammar that are headed by one of the root symbols of the grammar, and that terminate in a subset of the input).

Successful derivations can produce one or more system actions. Typically, such actions will involve building a composite object—for example, a *Figure* or *Heading*. The individual elements of the composite are typically "grouped," precisely aligned, and defaults such as font styles added. The introduction of a composite object in place of its parts entails the following bookkeeping with respect to the three data structures mentioned previously. First, the input elements making up the composite must be deleted from the current input set; second, any complete or partially matched rules in the parse table that refer to these now deleted input elements must be removed from the parse table; third, any derivations with their associated system actions that depend on these input elements must be removed from the action agenda as well. The effect of these operations may be to invalidate

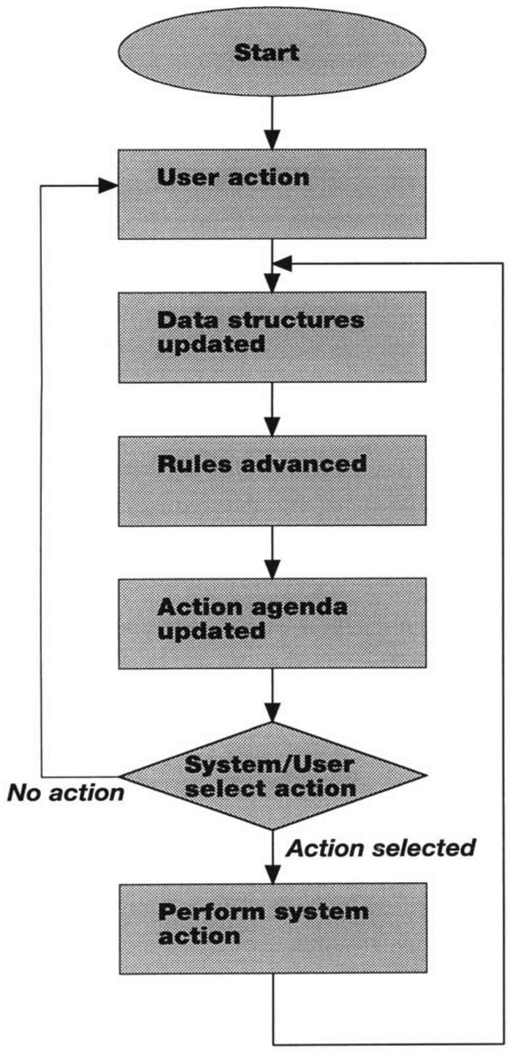


FIGURE 5.4
Flowchart of algorithm for top level user and system interaction.

previously valid actions on the agenda. For example, if two different derivations each use some of the same input elements but lead to different composites, the selection of one of the actions to produce one composite will eliminate the possibility of ever choosing the competing alternative. Finally, the new composite object is added to the current input set so that it now can be considered as input to further rule matches, derivations, and actions. Rules are advanced and the system agenda is updated. Currently, there are two modes of interaction with the agenda, automatically firing the first item on the agenda or interactively selecting an item from a graphic representation of the agenda. Sorting and thresholding of the pending inferences can provide additional support dealing with the agenda. If no action is selected, control is returned to the user who may once again add, move, delete, or resize objects. This is all summarized in Figure 5.4.

The bottom-up, incremental parsing algorithm of [Wittenburg, et al., 1991] is used here with only one change. Derivations are considered successful even if they do not cover all the input elements. This change has no effect on the internals of the parsing algorithm.[5.1]

It is typical for grammars used these design applications to have more than one goal, or *start*, symbol. The grammar for page layout that follows uses start symbols such as *Figure*, *Heading*, *Body*, etc., an indication that these are the goal categories that the parser will be seeking. If a larger grain size of interaction is desired (i.e., less frequent requests for graphic inferencing), the grammar would use fewer goal categories, giving the user the option to build a whole title page with one rule action rather than each of its structural parts step-by-step. The tradeoff here is between more interruptions and distractions caused by graphic inferencing and the feedback and confirmation of incremental design decisions.

Take for example the creation of a *Figure* composed of three elements: a *Number*, a *Text*, and an *Image*. A fine-grained grammar may initiate improvements after the *Number* and *Text* are added, creating a *Caption*. Then, after the *Image* is added, more improvements would occur completing the *Figure* element. Alternatively, a coarse-grained grammar may trigger an action only after all three elements have been created in the proper relationships. By controlling the way in which these derivations interact, different granularities can be used to suit the rule designer's intention.

A grammar for layout design

The basic components of the Relational Grammar utilized in the above scenario are described below. They include the lexical and compositional categories of the language, the relations between the design elements, and the rule specifications described in terms of these categories and relations. The supporting data structures are also described.

Lexical and compositional categories

The first step in defining a relational language is to identify the primitives of the language and the corresponding lexical mappings for those primitives in the grammar. In an interactive environment, the primitive lexical categories are available to the user from an input menu. In this example, they include:

5.1 Note, however, that such a change would affect predictive parsing algorithms such as [Wittenburg, 1993].

Text:	Text element for titles, headings, captions, and paragraphs.
Number:	Number element for figures and chapter numbers.
GraphicRule:	Graphic element for separating regions.
Image:	Bitmap image.

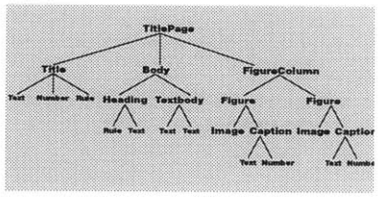


FIGURE 5.5
Structural hierarchy of a ***TitlePage*** *corresponding to a derivation in the grammar. An alternate visualization of this rule is shown in Figure 5.2.*

As the user selects a primitive category from an interface menu, an instance of the class is created. The user sizes and locates the newly created instance. These instances maintain local state information, such as size, position, color, and font specification. These primitive elements combine to form composite structures which may in turn be used to define other new structures. The exact details of the combination of these elements are encoded in the grammar rules and are described below. Composite elements are also treated as instances of classes. Composite icons maintain local information and information of their component elements. Some information, such as size and position, are calculated values based on these component elements.

The compositions for this language are:

```
TitlePage     = Title        + Body  + FigureColumn
Title         = GraphicRule  + Text  + Number
Body          = Heading      + TextBody
Heading       = GraphicRule  + Text
Textbody      = Text         + Text
Textbody      = Textbody     + Text
FigureColumn  = Figure       + Figure
Figure        = Image        + Caption
Caption       = Text         + Number
```

A hierarchy of objects representing a *TitlePage* is shown in Figure 5.5. The primitive lexical items form the leaves of the hierarchical tree. Each subtree of depth 1 corresponds to a rule application. From the bottom-up point of view, a rule application forms a composite element whose parts are its immediate daughters and whose type corresponds to the left-hand-side category of the rule.

What Figure 5.5 does not show is that there are additional relational requirements necessary for any rule to fire and subsequently introduce a composite element. For example, a *Caption* element will not combine with just any *Image* element to produce a *Figure*—an additional requirement is that the *Caption* be immediately below the *Figure* and almost equal in width. The relational constraints are the key to this parsing algorithm and the requirements for rule firing.

Domain relations and predicates

Domain relations determine rule applicability within the language. The relations may have any number of arguments but binary relations play a special role in parsing (see Wittenburg, et al., 1991; Wittenburg, 1992). For the domain of graphic design, relations are typically spatial, such as *Above* and *Equal-Width*. Other applications that use the grammars in an automatic paradigm, such as table of contents or graph layout, may describe information more semantic in nature, such as *Author-Of* or *Connected-To*, respectively. The standard graphic relations supported in this domain of layout include:

Above	**Below**
Left-Of	**Right-Of**
Top-Aligned	**Bottom-Aligned**
Left-Aligned	**Right-Aligned**
Equal-Width	**Equal-Height**

where directional relations such as *Above* have the additional requirement of adjacency. All relations can also be used as predicates. For the parser to operate, domain and range queries also need to be supported for some relations. A domain query answers questions such as "What is above X?" while a range query answers questions such as, "What is X above?" These queries for the *Above* relation are summarized below.

Category	Example	Query
Binary predicate	"Is X above Y?"	(query 'above X Y)
Domain query	"What is above X?"	(query 'above :? X)
Range query	"What is X above?"	(query 'above X :?)

A domain dependent object store is used to answer relational queries and predicates. These object stores differ depending on the application. In the 2D layout domain a KD-tree representation is used in order to optimize queries about spatial proximity. A KD-tree is a representation for storing objects based on K-dimensional attributes [Rosenberg, 1988]. In particular, they are useful for storing objects with rectangular bounding boxes, where each edge of the bounding box is a separate index into the tree, i.e. K=4. Subsequent queries are based on these rectangular regions. Elements are stored as nodes in a binary tree, using the bounding rectangle as keys for deciding whether an object should be stored on one branch or another of any given node. Once the icons are created within the interface, they are added to the KD-tree. Queries of the form "What is above X?" and "Is X above Y?" can then quickly be answered based on their rectangular bounding boxes.

Grammar rules

Grammar rules specify the conditions under which composite elements can be created. During the process of interpretation, rules support the designer by producing constraints on the input elements. Rules use the domain relations to create preconditions, that when satisfied, result in the rule firing and creation of composite objects. An example rule definition for creating the composite element, *Figure*, is illustrated below.

```
(Defrule (Make-Figure The-Grammar)
  (0 Figure)
  (1 Image)
  (2 Caption (Above     1 2)
             (Eq-Width  1 2))
  :Out
  (Make-Instance 'Figure-Element
    :Image-Element      1
    :Caption-Element    2))
```

FIGURE 5.6
The definition of the Make-Figure *rule adapted from the layout of the book, [Mitchell, 1990].*

This rule can be read as follows. Rule elements are numbered 0 to n. Element 0 is considered to be the left-hand-side (composite) category type. Elements 1 to n are the right-hand-side. For the *Make-Figure* rule to be applicable, it must first match an *Image* element. Next, it will seek a *Caption* element by trying to satisfy the query `(Above <image-element> :?)`. If these conditions are satisfied, then the predicate `(Eq-width <image-element> <caption-element>)` must return TRUE. If all these preconditions are satisfied, then a potential composite category *Figure* is introduced into the parse table. Assuming *Figure* is a root category of the grammar, an action to go ahead and create this composite is made available to the system action agenda. In addition, a rule may contain semantic output forms to be evaluated as part of the rule action. These forms follow the symbol :OUT in a rule body. In the case of the *Make-Figure* rule, the Make-Instance form creates a *Figure-Element* when the agenda item is invoked. This object references the input elements which satisfied this rule's preconditions. As part of running the agenda item, additional constraints are then installed on these input elements. These output constraints are different from the rule's preconditions. They are the reason input can be roughly sketched resulting in exactly aligned elements of the design. The rule is graphically depicted in Figure 5.7.

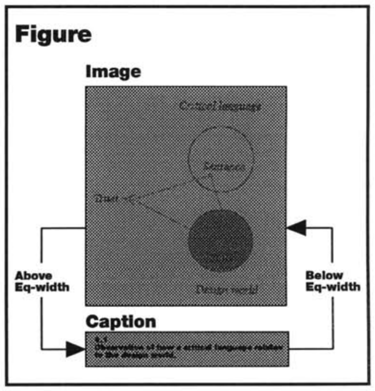


FIGURE 5.7
The graphic illustration of the ***Make-Figure*** *rule shown in Figure 5.6.*

Authoring on-line documents

In this scenario, the structured editing and incremental improvement of a document produces an HTML file which can be written out and

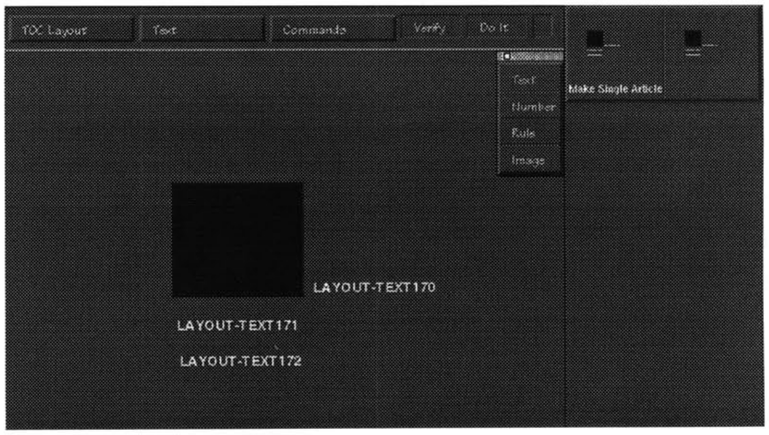


FIGURE 5.8A
*The first in a sequence of images illustrating the construction of an online table of contents for WIRED magazine. This first step shows three text elements and an image created and positioned in the workspace. The agenda on the right presents a button to suggest the creation of a single **Article** group.*

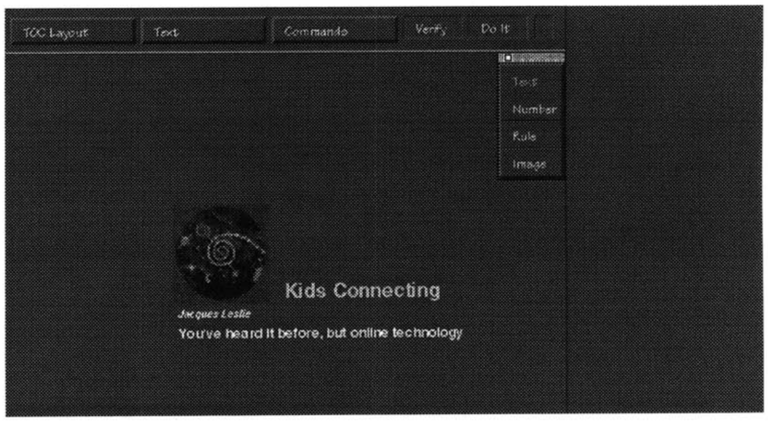


FIGURE 5.8B
*After selecting the agenda button in the previous step, the system aligns and constrains the elements. The user has edited the title of the **Article** and the system can then complete the remaining content from the local database.*

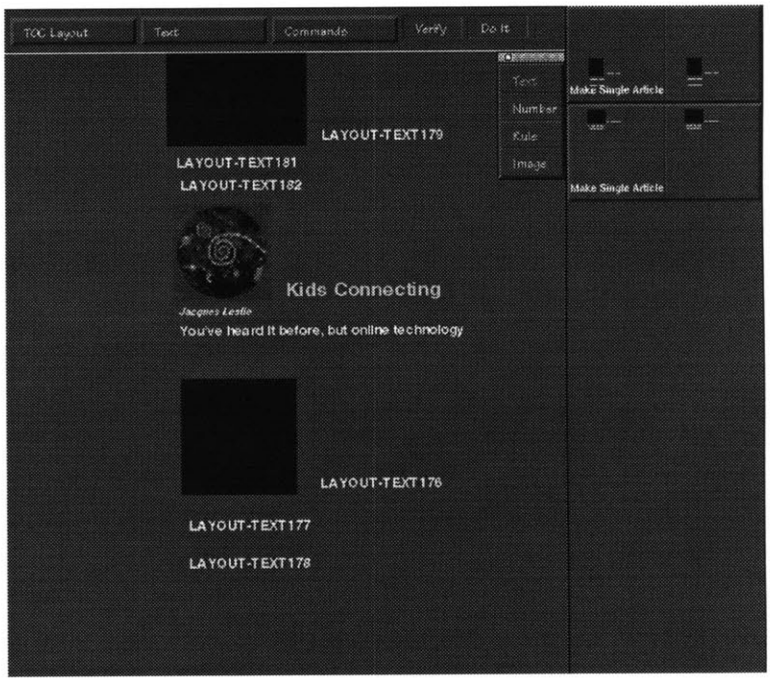
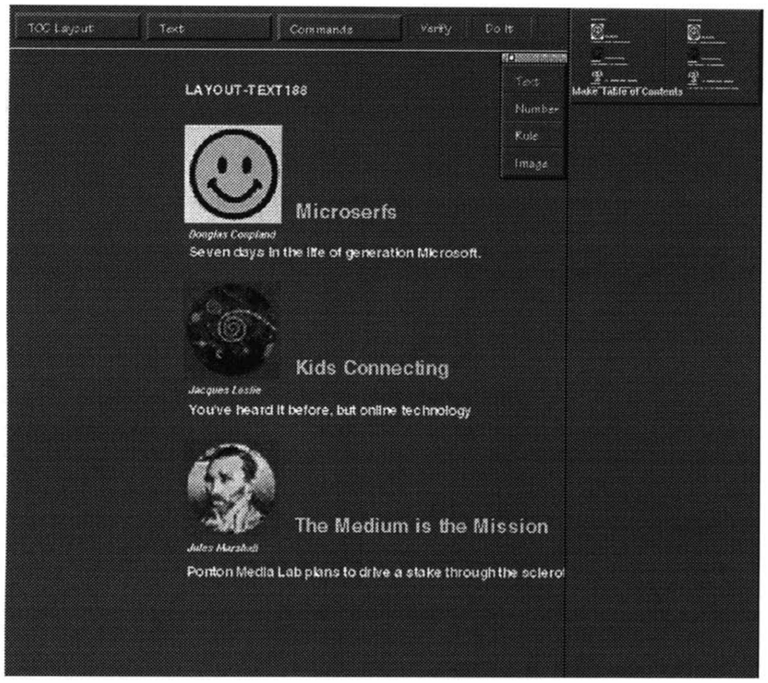


FIGURE 5.8C
*The beginning of two additional articles have been created in the work-space. The agenda reflects the new partial design state by suggesting the creation of these **Article** composites.*



LAYOUT-TEXT188

Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft.

Kids Connecting
Jacques Leslie
You've heard it before, but online technology

The Medium is the Mission
Jules Marshall
Ponton Media Lab plans to drive a stake through the sclero

FIGURE 5.8D
The two additional ***Articles*** *have been completed. Similar to Figure 5.8B, the user has edited one portion of the article and the system completes the editing process. The agenda reflects the new state suggesting to complete the table of contents by making the newly added text element at the top the title.*

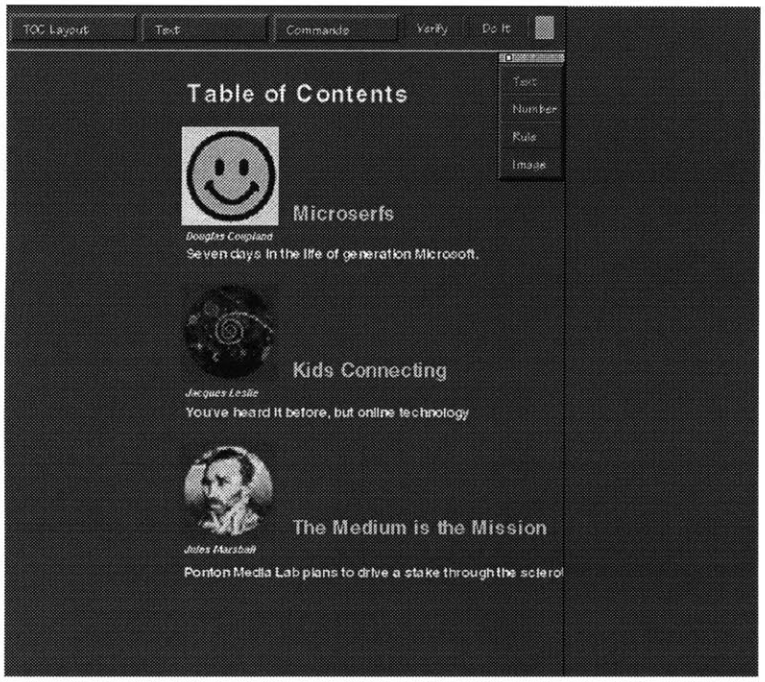


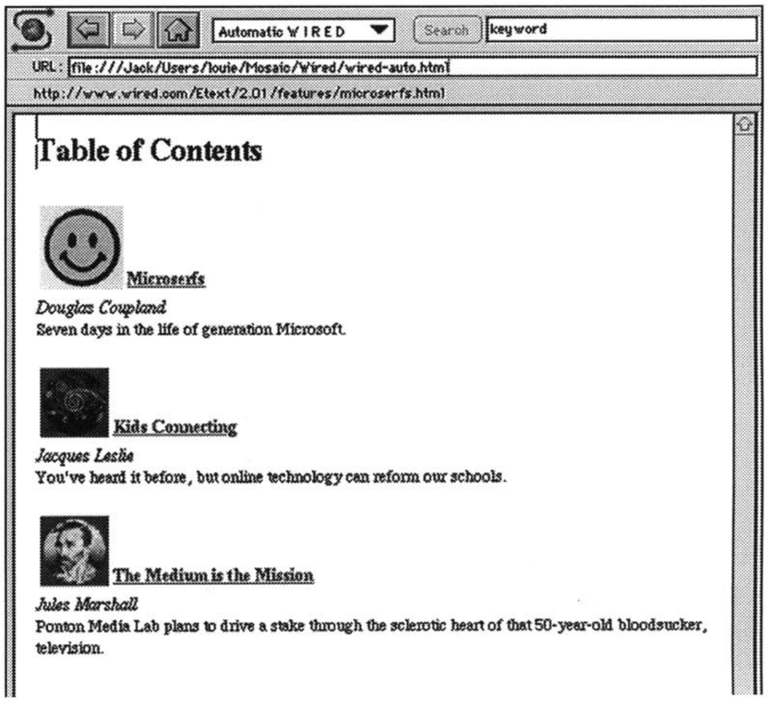
Table of Contents

Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft.

Kids Connecting
Jacques Leslie
You've heard it before, but online technology

The Medium is the Mission
Jules Marshall
Ponton Media Lab plans to drive a stake through the sclero

FIGURE 5.8E.
The button to the right of the "Do It" button highlights indicating the design is complete and correct according to the supporting visual language. The user selects the "Do It" button to write out an HTML file.



URL: file:///Jack/Users/louie/Mosaic/Wired/wired-auto.html

http://www.wired.com/Etext/2.01/features/microserfs.html

Table of Contents

Microserfs
Douglas Coupland
Seven days in the life of generation Microsoft.

Kids Connecting
Jacques Leslie
You've heard it before, but online technology can reform our schools.

The Medium is the Mission
Jules Marshall
Ponton Media Lab plans to drive a stake through the sclerotic heart of that 50-year-old bloodsucker, television.

FIGURE 5.8F
Mosiac reads the HTML file produced in Figure 5.8E. The underlined words are the links to the online articles at WIRED's World Wide Web server. Note that this example creates a customized table of contents from multiple issues of the magazine.

presented in an HTML viewer such as Mosaic.

Figures 5.8A-5.8F illustrates a sequence where the designer is constructing a table of contents of WIRED magazine similar to the presentation automatically constructed in the previous chapter. Each image in the sequence illustrates the workspace on the left and the agenda on the right. Here, the delivery environment will be Mosaic, a graphically more limiting display system than VIA so the grammar supports only simple layout constraints. The database of objects and relations can be used to help complete the editing of media objects because their relationships have helped to determine their role in the final layout. For example, titles and authors both start out as text elements but their spatial positioning helps disambiguate their final role iin the design.

In the process of creating the document, composite structures, such as *Articles*, are created. At any point during the design process, the author can request the generation of an HTML file describing the input. The system identifies the most general composite structure created thus far. This composite, in the traditional object-oriented programming style, can produce the HTML description of itself and the rest of the derivation. This is true for a single article, a sequence of articles, or the completed table of contents. Currently, the links themselves are automatically produced by a database look up, but could be a result of the parsing action.

Support for design completion

This section describes the use of Relational Grammars to support another interactive design scenario, that of graphic completion and verification. A variation of the parser used in VIA was constructed [Wittenburg, 1993], and then used to support both graphic completion and verification of partial designs.

Graphic completion is analogous to command completion commonly found in text editors. The difference is that graphic completion is defined in terms of the elements and their relationships to the existing elements of input. During the process of creating an artifact in a structured domain, the system guides the user by presenting only valid graphic continuations of the partially completed design. The system then builds higher level composite structures as new input is added. When the design is finished, the system notifies the user that a complete and correct solution has been achieved. In addition, this same parser can be used to verify partial or complete designs to indentify their conformity to a predetermined visual language. An example scenario of the use of this technique is presented in the structured domain of flowcharts.

The paradigm of completion

GNU Emacs [Stallman, 1993] supports command completion with a variety of interaction styles: a user can complete the next word (SPACE), complete as far as possible (TAB), complete and exit the command (RETURN), or list of all of the possible completions (?). In the graphic domain, it is useful to provide this same functionality by providing completion of the next lexical element, completion of the next higher level composite structure, completion of as much of the design as possible, and also, a menu of possible completions given the context of any partially completed artifact.

GNU also identifies and utilizes three levels of strictness for completion. These include: *permissive completion*, *cautious completion*, and *strict completion*. Permissive completion will permit any string to be used as the command argument (e.g., a name for a file can be created when running *find-file* if it does not exist). This in fact does no completion to the command or argument. Cautious completion only exits input to the command if the command is complete and is valid. Finally, there is strict completion which will complete the command only when a valid argument is used (e.g., an existing buffer name given to the *kill-buffer* command).

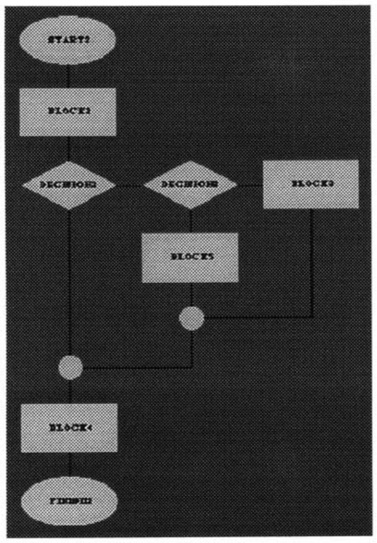


FIGURE 5.9
An example structured flowchart illustrates the type of structured design we are addressing in this design completion scenario.

Instead of keystrokes, VIA uses mouse clicks and menu items to provide access to this functionality. Rule priorities could make it possible to automatically select the most appropriate rule in order to complete partial designs.

Design completion scenario

This section presents an example interaction with the system in supporting the design of a flowchart using graphic completion.[5.2] The basic lexicon for this language includes rectangles (simple procedure blocks), diamonds (decision points), circles (junctions for decisions to reconnect), and ovals (either start or end blocks). Figure 5.9 shows a small flowchart defined by this language. This layout was actually produced automatically with the scenario presented in the previous chapter.

The interaction sequence begins with the user selecting an element from the menu of possible lexical types. The user selects, sizes, and positions a *Diamond* element to construct a decision block within the workspace. Colors provide feedback to the user to indicate which elements are continuable, which elements are complete (i.e., no further work is necessary) and which elements fall outside of the language definition. The user then clicks on the diamond and a menu of the possible continuations is presented. These continuations are presented geomentrically to indicate their proper geometric relationships within this design context. The user selects a *Rectangle* to place a procedure element to the right of the *Diamond* and places it in the design. Next, the user selects a circle element to complete the decision block. Now only the in and out elements of this group are continuable. The *Rectangle* procedure block is complete and cannot be continued. The design is completed by adding one *Oval* above the *Diamond* element (the decision block input) and below the *Circle* element (the decision block output). The system signals the user that the design is complete and correct given the current flowchart visual language. This sequence is presented in Figure 5.10. This figure shows both the current status of the design alternating with the completion menus at the various stages of the design process.

Design verification scenario

In this second scenario, a predictive parser can also be used to support design verification. The designer has created a partial flowchart design and would now like to verify the design as it stands. The designer selects a *Verify* button. This request accepts complete and correct elements but highlights those elements that are incomplete or

5.2 At the time of this writing this thesis, an initial predictive parser was utilized. There are more open research questions in making that parser complete and correct.

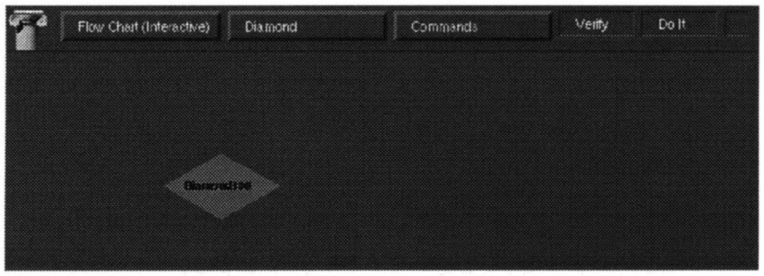


FIGURE 5.10A
*This sequence illustrates a scenario of creating a flowchart diagram using the paradigm of completion. The first menu allows any lexical item to be created. A **Diamond** is chosen and displayed in the workspace.*

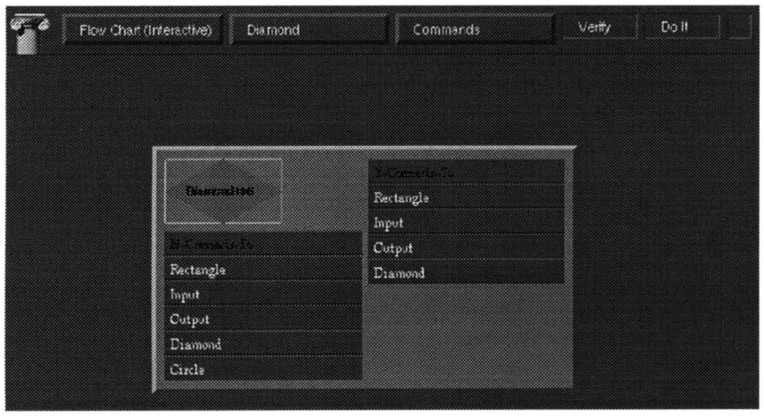


FIGURE 5.10B
*The user selects the **Diamond** and requests possible continuations from this element. A menu comes up over the element with a small set of options. These options stand in the relationship, **Y-Connects-To** or **N-Connects-To.***

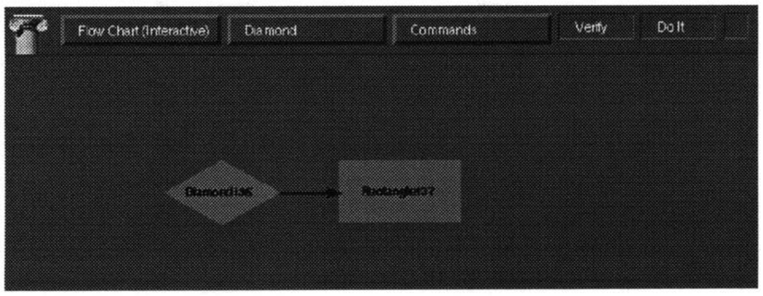


FIGURE 5.10C
*The user selects **Rectangle** and in the **Y-Connects-To** relationship in the menu in Figure 5.10B.*

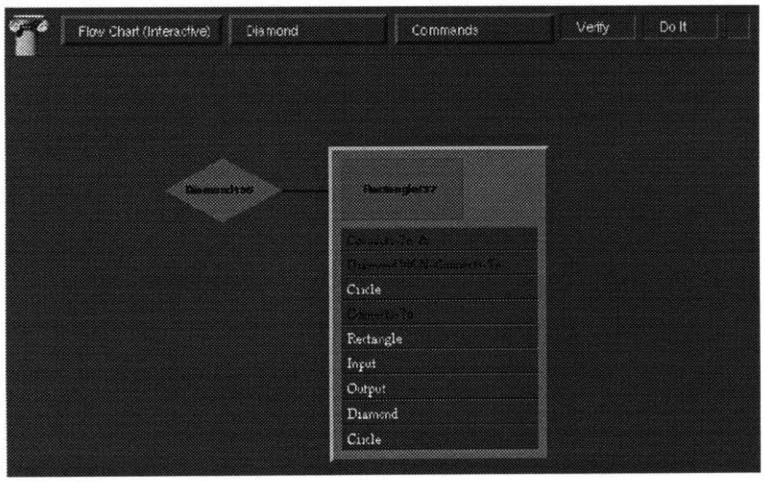


FIGURE 5.10D
*The user selects the **Rectangle** and requests possible continuations from this element. A menu comes up over the element with an appropriate set of options for continuation.*

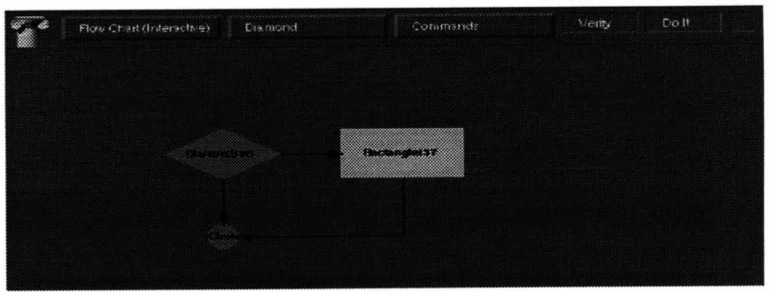


FIGURE 5.10E
The user selects the Circle in the menu in Figure 5.10D.

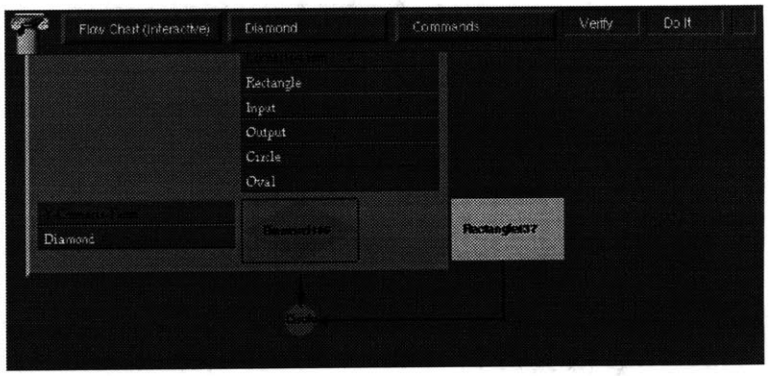


FIGURE 5.10F
The user requests continuations from the ***Diamond.***

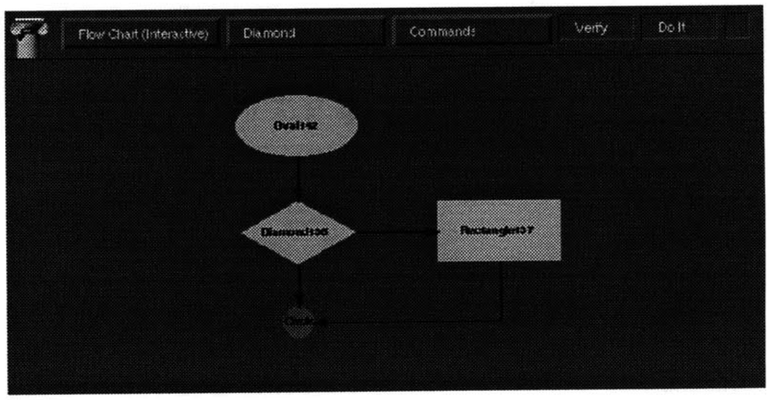


FIGURE 5.10G
The user selects the ***Oval*** *which represents the start of this procedure. The only element that can be continued is the* ***Circle*** *junction.*

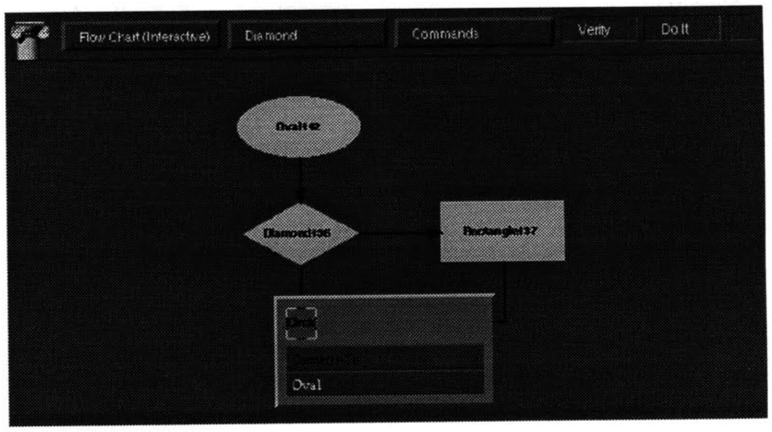


FIGURE 5.10H
The user selects the continuations from the Circle. (An early version of the parser produces a subset of all valid continuations.)

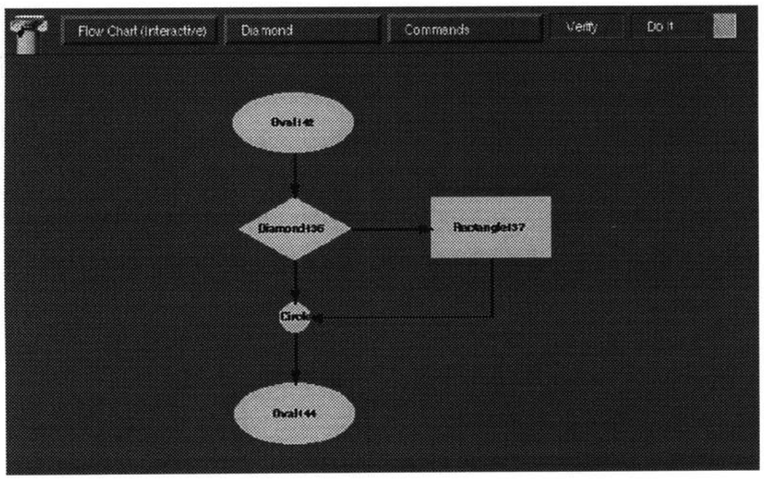


FIGURE 5.101
The final flowchart using design completion is shown here. The system highlights the box in the upper right to indicate a complete and correct design has been created.

violate any rules. For each element that still needs work, the designer can request alternative design solutions. An agenda displays these alternatives. This new agenda provides ways to bring the design into conformance, suggesting the deletion of some elements and the completion of others. The designer can select or ignore these suggestions while continuing towards completion of the design.

Verification and completion are related in their use of the predictive parser. Partial designs can be verified and then suggestions on how to create completed designs can be requested. This topic of completion and verification remains an open research topic. The discussion here is meant to illustrate possible directions for future work while providing a working example of the interaction scenarios.

Related work

Kochhar, Marks and Friedell have characterized the articulation of a designed artifact along an axis of automaticity [Kochhar, et al., 1991], from completely manual to completely automatic. This chapter spans a number of the techniques included in this characterization, including constraint-based systems, critic and improver-based systems and cooperative CAD. Related work in these areas will be described and compared to VIA.

Constraint-based systems

Some constraint-based systems make an effort to empower the user by giving them direct feedback and control of the inferencing process [Gross, et al., 1987; Leler, 1988; Sistare, 1991]. Rockit [Karsenty, et al., 1992], a system to apply graphic constraints in building an interface, provides not only visual, but audio, feedback to help the user in

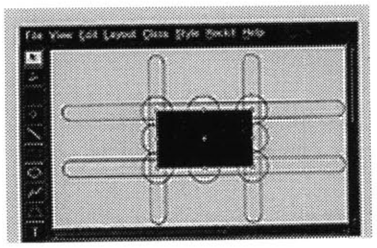


FIGURE 5.11
Rockit system showing gravity field regions for connectors and aligners of a rectangular object.

deciding the correct constraint application. The system showing *fields of gravity* to be used in constraint application is shown in Figure 5.11. Hudson and Yeatts [Hudson and Yeatts, 1991] explore a similar approach. In both these systems, the interaction and feedback is direct and immediate, i.e., it occurs over the elements being drawn. Somtimes it is inappropriate to interrupt the design process with immediate, "in your face" suggestions, while at other times it supports the more direct manipulation of interface and design components. VIA makes the compromise by providing control over the inference process with a visual agenda. The agenda queues suggestions for design improvement in a separate window. These delayed system suggestions can then be activated by the user at the appropriate time within the design process.

Interactive constraint-based systems have a long history in the research environment [Borning, 1981; Nelson, 1985] and only recently have appeared in commercial products like *Intellidraw* [Aldus, 1993]. This is due, in part, to the computational requirements of adequate implementations. This thesis incorporates a fast, interactive constraint-propagation algorithm, *DeltaBlue*, from the University of Washington. DeltaBlue is used to provide non-cyclic constraints between output elements in design articulation [Freeman-Benson, et al., 1990; Maloney, 1991]. Although this particular algorithm does not support cycles in its constraint networks, it has been sufficient for the construction of interactive design scenarios that embed constraints in the set of output forms.

A standard technique for controlling design articulation is the use of the grid [Hurlburt, 1978]. [Feiner, 1988] uses the grid for the articulation of automatic information layout. The system uses information about the kind of material to be displayed, the user, and display hardware. The system then creates displays using more information about the kinds of objects to be presented. This system first generates a grid and then uses it in the presentation of information. VIA characterizes the information but the notion of a grid is implicitly embedded within the output constraints of the grammar rules and the alignment of design elements. One future extension could derive different grids based on conditions in the input and display environment, and use that grid in further processing of design elements.

Design critic-based and improver-based systems

Critic-based systems [Fischer, et al., 1988] provide a commentary on the status of the design without providing the ability to change the design, while improver-based systems may or may not provide a critique but will modify an existing design. *CRACK* [Fischer and Morch, 1988] is a critic-based system to support the design of kitchens. The interface is illustrated in Figure 5.12. It uses domain knowledge to "look over the shoulder" of a user offering criticism, suggestions and explanations to improve the design. An example of an early improver-based system is Pavlidis and Van Wyk's beautifier system [Pavlidis and Van Wyk, 1985] for improving network diagrams. It first infers the relations between graphical objects in network diagrams and then modifies those objects based on the constraints of the domain. No effort was made to involve the user with information or control over the inferencing process. The incremental design assistance in VIA is very similar to this class of design-support system. It combines both the mechanism for describing design critiques with the opportunity to modify the original artifact.

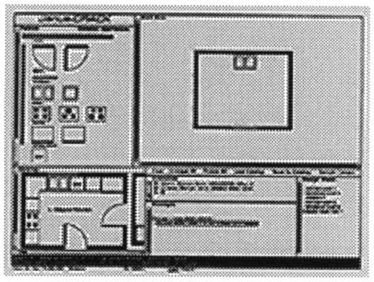


FIGURE 5.12
The CRACK system illustrating a palette of domain elements, a catalog of complete floor plans, the workspace, and the critique message window.

Other interactive design systems that use Relational Grammars in more structured design tasks have been created at Bellcore [Collier and Karlin, 1993]. One system uses a top-down algorithm (instead of bottom-up) to support engineering design tasks in telephone networks. The artifact being designed is represented as a network diagram where some of the nodes correspond to grammar nonterminals, representing an abstraction of parts of the design not yet filled in, and others correspond to terminals, representing completed decisions. By clicking on nonterminals, the user can explore choice points in the design represented in the grammar as alternative expansions of the nonterminal. The process is complete when all elements in the design have been replaced by terminal elements in the grammar. The top-down approach is well suited to this more structured domain, enforcing conformance to known design configurations. The bottom-up approach we have adopted, however, is well suited to the more unpredictable task of exploratory design.

Cooperative design

[Kochhar and Friedell, 1990] investigates the use of cooperative design that mixes interactive and automatic design paradigms. This exploration is in the context of the *FLATS* system shown in Figure 5.13. In this system the user directs search in the design space by first expressing initial design decisions and the properties of the final design via a set of guidelines. The system automatically produces partial developments of the design using rules defined in an object grammar. Alternatives are presented to the user in a spatial format based on their design attributes. The user selects and refines the most

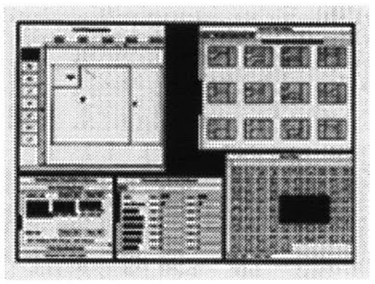


FIGURE 5.13
Cooperative CAD system, FLATS, exploring design alternatives of floor-plan layouts.

promising partial designs and then the system continues the design automatically. The process repeats itself with the user in control of the refinement process, restricting the activation of rules, controlling the focus of the development, and setting the resource allocation for the time and number of alternatives to explore. Browsing is supported by filtering the set of alternatives examined at any one time and structuring those alternatives displayed by design attributes. Cooperative design is a strong influence on this research. It is very important because it puts the user in direct control of the design process, like the scenarios described here, and supports the exploration of the design space. Future research to aid the exploration of the design space will combine the automatic and interactive scenarios in a more *cooperative* fashion.

Summary

This chapter presents a vision of how visual languages can be used to assist in the interactive design process. By providing the context of a visual language within which a designer works, VIA improves and cleans up incremental design decisions. This support also includes graphic completion and design verification of partially constructed artifacts.

In the interactive scenario, VIA uses a standard bottom-up parsing algorithm but the restriction of covering all of the input has been lifted. User interaction dictates a finer-grained parsing goal than is typical in most standard parsing applications. Local graphic inferencing can occur, cleaning up the design, while the parse table acts as a cache for rule matches. The next chapter applies this interactive scenario to author other visual language grammars by demonstration.

Grammar acquisition by demonstration

6

Introduction

We must now have sciences at the places where formerly intuition directed us.
FRANK LLOYD WRIGHT

New tools and techniques will be necessary in order to support the design process as presented in the previous two chapters. If design is truly going to be ubiquitous and accessible at all levels of information creation and consumption, these tools will unquestionably be central to the design process. By creating editors grounded in formal visual languages, designers can create and use their own visual languages to be used in both the automatic and interactive scenarios. In this chapter a vision of a grammar-based tool to support the creation of these languages is described.

As we have seen, there are many components to the full grammar definition. There are the lexical items and composite categories that the grammar uses in building a derivation of the input. There are the rule definitions, including the pre-conditions that first must be satisfied before the rule will fire, and the post-conditions that will create the composite categories and take actions in the output environment. In the interactive scenario, constraints will take effect upon rule firing, while in the automatic scenario, the constraints will only be applied to the media objects when a complete derivation is formed. A complete editor would provide a mechanism to create and manipulate each of these aspects of the rule formalism. In addition, it should be possible to enable and disable various sets of rules in order to direct design exploration.

This chapter describes a mechanism for supporting such an editor. What has been implemented is a system that supports the visualiza-

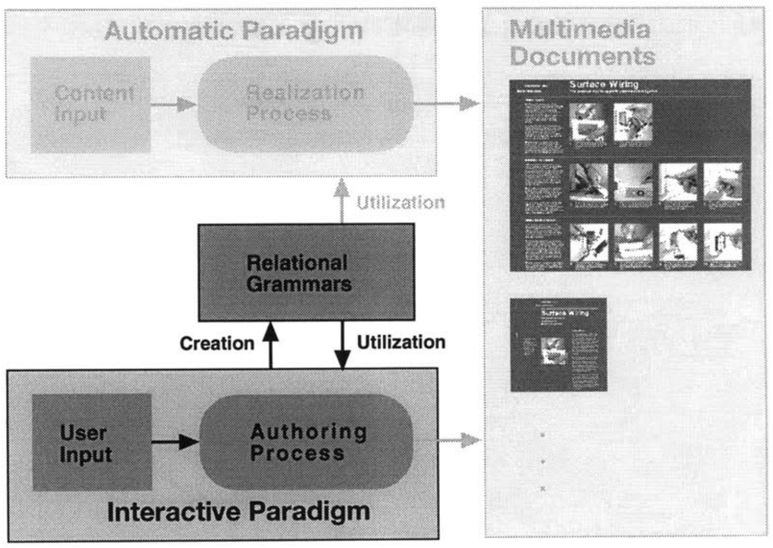


FIGURE 6.1
Architectural overview of rule editing showing how it fits into the vision of supporting the design process. A grammar supports the authoring of another grammar that can then be used in either the automatic or interactive paradigms.

tion and redefinition of existing rules. Creating new rules in some cases will be a straightforward extension of the techniques shown here. Other more difficult cases reflect issues that remain open research problems in the discipline of programming by demonstration.

First, an overview of how this paradigm fits into the overall architecture of VIA is illustrated. Then, an example scenario is demonstrated redefining a rule used in the automatic scenario described in Chapter 4. In this scenario, the definition of how the rule articulates the particular design (i.e., the constraints of the output) is modified. The improver-based grammar used to support the rule authoring process is then described. Various aspects of a rule editor necessary for a complete tool are then presented. This chapter concludes with a discussion of work related to the creation of code by demonstration.

Architectural overview

As we have seen in the other scenarios, Relational Grammars play a central role in design support. In this scenario, Relational Grammars are used in a meta-fashion to author other grammars that are then used in one of the other main paradigms. Figure 6.1 illustrates the use of the grammars for rule editing. The actions of the author produces code that is then evaluated and installed immediately into the underlying application grammar.

The grammar authoring process is supported by a separate grammar used in the interactive paradigm similar to the description in the pre-

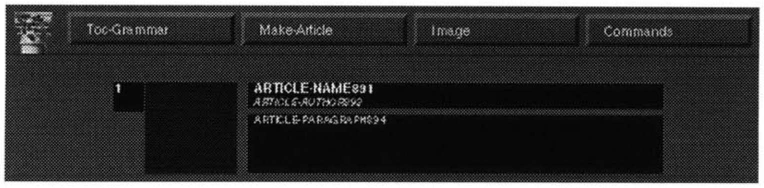


FIGURE 6.2
Visualization of the Make-Article rule's pre-conditions.

vious chapter. This supporting grammar is a general one and is described in more detail below. First, an example scenario is illustrated that redefines an existing grammar rule.

Example rule editing scenario

This section describes a scenario that uses programming by demonstration to redefine an existing rule within a grammar. In an effort to explore the design space, a variation of the existing rule is constructed. The rule that is modified here is our example rule that creates an *Article* group in the Scientific American grammar described in Chapter 4. Figure 4.3 illustrates the final realization of the complete table of contents. It uses the *Make-Article* rule of Figure 4.9 to form six separate article groupings. The example that follows will redefine the output constraints for this particular rule.

The designer begins by visualizing the *Make-Article* rule's output constraints illustrated in Figure 6.2. In this example, the designer will modify the output constraints and redefine how the rule presents an *Article* composite. The goal of this example scenario is to redefine the rule by moving the image and page number across the vertical axis to the right side of the group. Because of the currently defined constraints of the composite *Article* group in Figure 6.2, moving any one element will move all of the elements. Therefore, the first action the designer takes is to *Ungroup* the composite as it is currently formed. This has the effect of removing the constraints associated with that composite and placing the individual elements on the parse table. Rule actions to recombine these elements with new graphic constraints are placed on the visual agenda. These suggestions include rules for alignment and sizing. Mutually exclusive rules are initially placed on the agenda. Figure 6.3A and Figure 6.3B illustrate the agenda at different steps in this scenario.

As the designer moves and resizes the objects in the editor, action items come and go from the agenda. For example, if two elements are placed roughly left aligned, an agenda item appears that will *improve* these two elements by grouping them into a composite and aligning

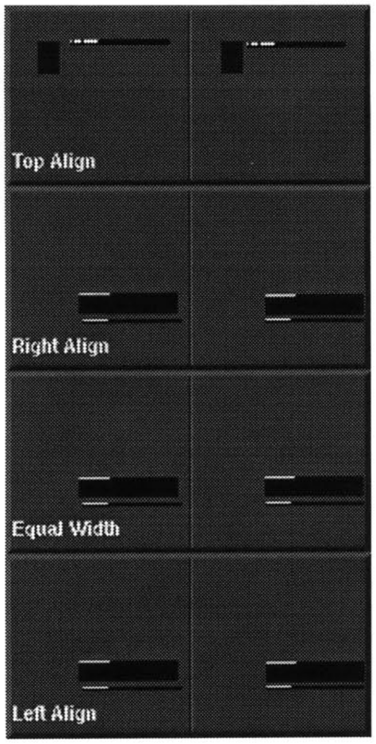


FIGURE 6.3A
Example agenda as the user moves, resizes and groups the elements of the Make-Article *definition. This agenda represents the state of the design shown in Figure* 6.4B.

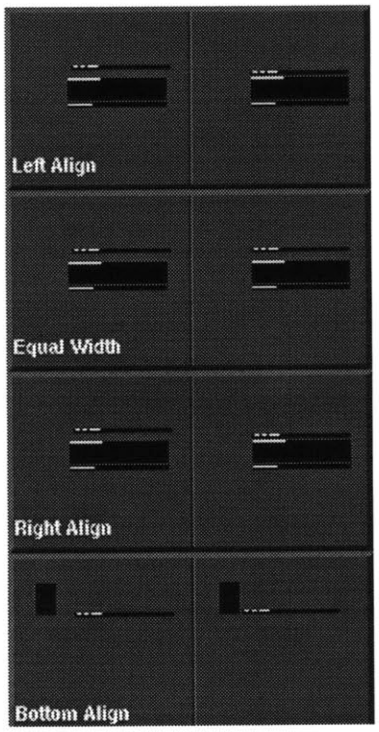


FIGURE 6.3B
Example agenda reflecting state of the agenda in Figure 6.4C.

them exactly. Elements can be quickly repositioned and the agenda items provide suggestions for improving and cleaning them up. After selecting an action, the two elements grouped together now are constrained as a single unit. Actions that are mutually exclusive with respect to the selected action are removed from the agenda, since they are now invalid. This group can then be constrained and merged with other elements and groups. Eventually, the elements are all positioned and sized together and act as a single unit. As one element is moved, all elements move together. Thus, all X and Y values have been constrained.

In this brief interaction, the designer has demonstrated the new version of the article definition. The sequence can be seen in Figure 6.4A-E. This new definition modifies the original Scientific American article layout by placing the image and number to the right and the author to the bottom of the new presentation. The system also provides a menu of primitive graphic items that may be used as additional elements to articulate the final presentation. For instance, the horizontal red lines in the WIRED table of contents could be added using this menu.

When the rule is complete, it can then be installed back into the grammar. The designer selects the *Evaluate Rule* button from the commands menu. The relationships that have been constrained by the improver-grammar rule firings have been recorded in the local object store. All the resulting constraints are then collected, the rule body is constructed, and evaluated into the application grammar. This redefines the original rule and makes it available in the Scientific American grammar.

The designer switches back to the workspace. This new rule is now an active part of the grammar and supports the automatic layout of the information. The results is a table of contents where all articles are reflected across the veritcal axis as shown in Figure 6.5.

A generic grammar for rule design

In the scenario above, the role of grammars is twofold. During rule creation, a grammar watches what the designer is doing and automatically suggests constraints to be incorporated into the rule definition. The interaction process for authoring documents described in Chapter 4 is the same one used here to author rules. The main dif-

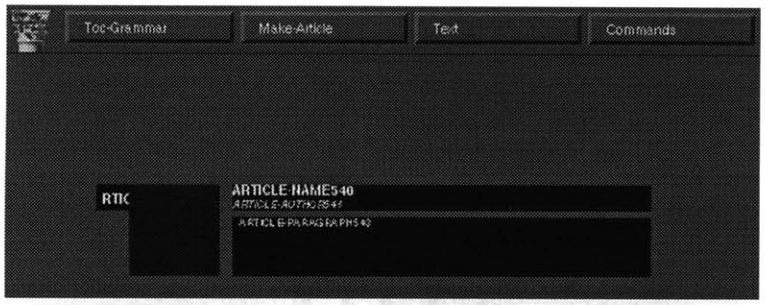


FIGURE 6.4A
*Sequence of steps redefining the **Make-Article** rule from the Scientific American table of contents grammar.*

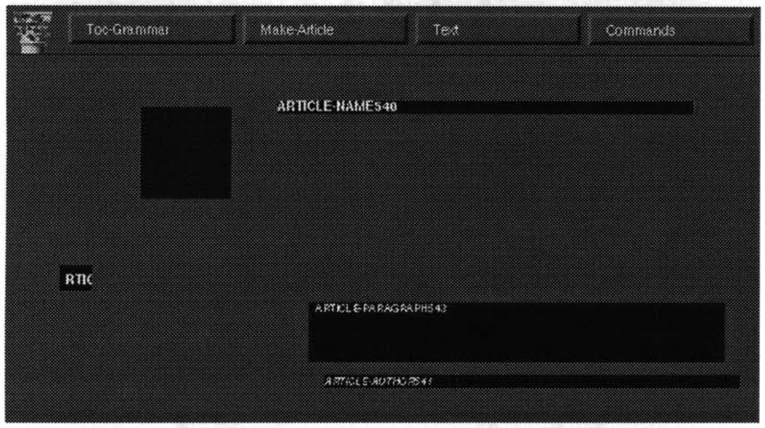


FIGURE 6.4B
The first step ungroups the composite removing all constraints. The rule author has separated the individual elements from one another. The agenda, shown in Figure 6.3A reflects the state of the pending rule firings derived from the partial design in this figure.

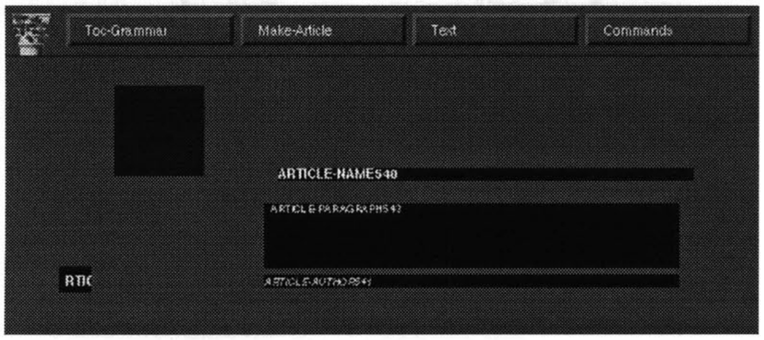


FIGURE 6.4C
The rule author has combined the paragraph and author and the agenda in Figure 6.3B reflects this new partial design.

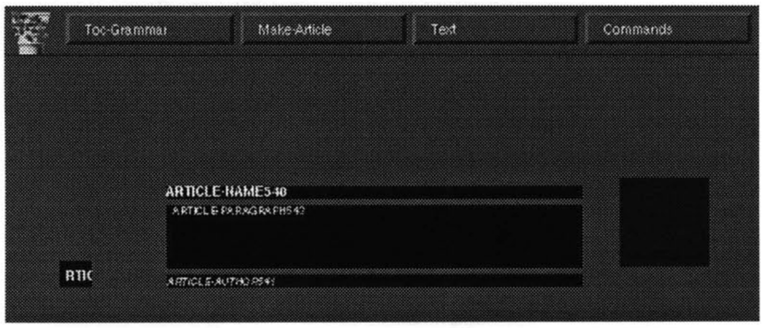


FIGURE 6.4D
*The title, **Article-Name**, is combined to form a new composite in the partial design.*

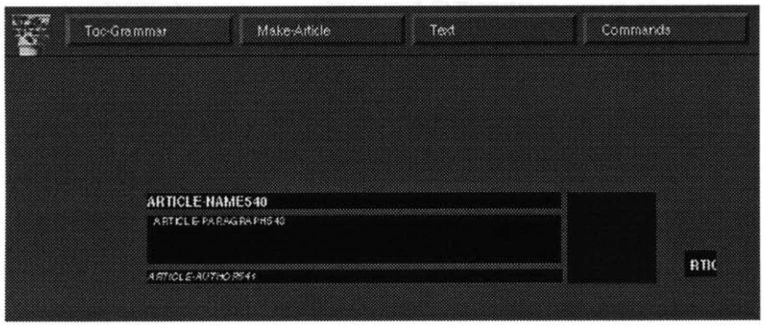
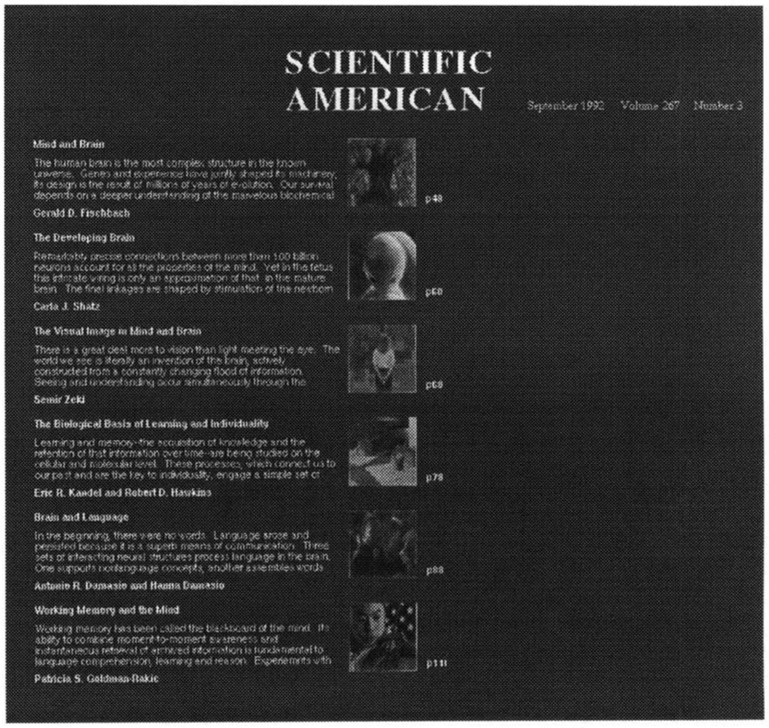


FIGURE 6.4E
The image is top-aligned with the growing composite. In the next step (not shown), the page number will be bottom-aligned to complete the rule description.


SCIENTIFIC AMERICAN September 1992 Volume 267 Number 3

Mind and Brain

The human brain is the most complex structure in the known universe. Genes and experience have jointly shaped its machinery. Its design is the result of millions of years of evolution. Our survival depends on a deeper understanding of the marvelous biochemical

Gerald D. Fischbach

p48

The Developing Brain

Remarkably precise connections between more than 100 billion neurons account for all the properties of the mind. Yet in the fetus this intricate wiring is only an approximation of that in the mature brain. The final linkages are shaped by stimulation of the newborn

Carla J. Shatz

p60

The Visual Image in Mind and Brain

There is a great deal more to vision than light meeting the eye. The world we see is literally an invention of the brain, actively constructed from a constantly changing flood of information. Seeing and understanding occur simultaneously through the

Semir Zeki

p68

The Biological Basis of Learning and Individuality

Learning and memory--the acquisition of knowledge and the retention of that information over time--are being studied on the cellular and molecular level. These processes, which connect us to our past and are the key to individuality, engage a simple set of

Eric R. Kandel and Robert D. Hawkins

p78

Brain and Language

In the beginning, there were no words. Language arose and persisted because it is a superb means of communication. Three sets of interacting neural structures process language in the brain. One supports nonlanguage concepts, another assembles words

Antonio R. Damasio and Hanna Damasio

p88

Working Memory and the Mind

Working memory has been called the blackboard of the mind. Its ability to combine moment-to-moment awareness and instantaneous retrieval of archived information is fundamental to language comprehension, learning and reason. Experiments with

Patricia S. Goldman-Rakic

p111

FIGURE 6.5
The information for the Scientific American table of contents is automatically presented using the redefined ***Make-Article*** *rule from the sequence illustrated in Figure 6.4. Note that only the Article composite has been redefined. Other rules within the grammar need to be modified in order to complete the redesign.*

ference is that this grammar is more general in nature and is based on principles of similar alignment and sizing of elements to form groupings within the input. The process is complete when all the X and Y values of the element have been constrained. This new rule is then installed in the application grammar to be used in the Scientific American application. In rule editing, the interaction between user and system produces rules, instead of producing documents.

Lexical and compositional categories

The grammar that supports this process consists of a simple recursive use of rules that are based on general graphic design principles. The primitive *Layout-Object* is the lexical type used in forming groups. When a composite group is formed, a side-effect creates a new *Layout-Object*. The original layout-objects are removed from the parse table and this newly created one is added.

Grammar rules

In this grammar there is a simple rule that directs the creation of composites. The context-free backbone of this rule is:

```
layout-group → layout-object layout-object
```

A typical rule in this general grammar is shown in Figure 6.6. This rule introduces an extended syntax to the rule definitions we have seen previously. That is, rules can now belong to different classes. In

Figure 6.6, the *Align-Left* rule, of the class *Alignment-Rules*, is being defined in the *Rule-Edit-Grammar.* In addition, a rule-label can be supplied which is used in constructing labels for the visual agenda. In this example, the label is *"Left Align."*

```
(Defrule (Align-Left Rule-Edit-Grammar Alignment-Rules)
 (0 Layout-Group (Setf (Object1 0) 1
                       (Object2 0) 2))
 (1 Layout-Object)
 (2 Layout-Object (Left-Aligned 1 2)
                  (Above        1 2))
 :Out (Make-Instance 'Layout-Object
      :Align-Function 'Left-Align-Objects
      :Item1 1
      :Item2 2)
 :Rule-Label "Left Align")
```

FIGURE 6.6
The definition of the Align-Left rule in the rule-edit-grammar to modify existing rules based on spatial layout.

In this scenario, we allow roughly "sketched" elements to satisfy the parsing preconditions. So the first predicate in the second rule element in Figure 6.6, *Left-Aligned*, is only looking for two elements *roughly* left aligned.

When all of the predicates are satisfied, the rule fires creating the composite *Layout-Group* and runs all the :OUT forms. In this case, a new *Layout-Object* is created. In the process of creating this new object three things happen. First, align-function is run, which left aligns the original objects. Second, these original objects are removed from the parse table. Third, the newly created *Layout-Object* is added to the parse table.

Rules in the grammar are currently divided into two categories, rules for alignment and rules for equal sizing. There are four alignment rules; *left*, *right*, *top* and *bottom*, and two sizing rules; for *equal width* and *equal height.* Ideally more rules will be incorporated into this grammar. Compared to the grammar described in Chapter 5, this grammar is potentially more useful because of the generic quality of the rules and the improvements that they support.

Currently, there is no error checking to ensure that all elements have their X and Y positions constrained. In the current implementation, there is the potential problem of an element being underspecified. Using predictive algorithms suggested in Chapter 5, the system can determine missing structure and suggest ways in which to improve the design as it stands to support the completion of the rule being edited.

Visualizing rule preconditions

When visualizing a rule's preconditions, the editor must show all elements and relationships that define a proper match for that rule. The interactive and automatic scenarios shown in previous chapters each require different support for the visualization of their preconditions. In the automatic scenario, symbolic domain relations (e.g., *author-of*, *image-of*) must be visualized, while for the interactive scenario the preconditions deal with spatial proximity and graphic relations (e.g., *above*, *left-of*). The visualization and modification of rule preconditions will be discussed in terms of the automatic scenario using the example rule, *Make-Article*, from the Scientific American grammar. This rule is presented in more detail in Figure 6.7.

```
(Defrule (Make-Article The-Grammar)
 (0 Article (setf  (article-name         0) 1
                   (article-author       0) 2
                   (article-description  0) 3
                   (article-page         0) 4
                   (article-image        0) 5))
 (1 Text)
 (2 Text    (Author-Of      2 1))
 (3 Text    (Description-Of 3 1))
 (4 Number  (Page-Of        4 1))
 (5 Image   (Image-Of       5 1))
 :OUT
 ((right-of        1 5)
  (right-of        2 5)
  (right-of        3 5)
  (right-of        5 4)
  (top-aligned     1 5)
  (top-aligned     5 4)
  (spaced-below    2 1)
  (spaced-below    3 2)
  (set-font 1 10pt :bold)
  (set-font 2 8pt  :italic)
  (set-font 3 8pt  :plain)
  (set-font 4 10pt :plain)))
```

FIGURE 6.7
*The **Make-Article** rule (also shown in Figure 4.9) used in the example scenario to produce an alternative presentation of the Scientific American table of contents.*

In the *Make-Article* rule, the preconditions are the elements numbered 1 through 5. The *Make-Article* rule's preconditions are visualized in Figure 6.8 while other rules in the same grammar are shown in Figures 6.9 and 6.10. The technique employed here is to use indentation to distinquish the head or left-hand side of the rule (e.g., the *Article* category), and the rule elements or right-hand side that must be matched in the input (e.g., *Text*, *Number* and *Image*). A second indentation is used to visualize the attributes of composite categories, as seen in Figures 6.9 and 6.10. Input elements are passed up the derivation hierarchy by attribute assignment in the rule body (i.e., the *Setf* forms in Figure 6.7). The attributes are subsequently used in rule predicates to test the appropriateness of rule application.

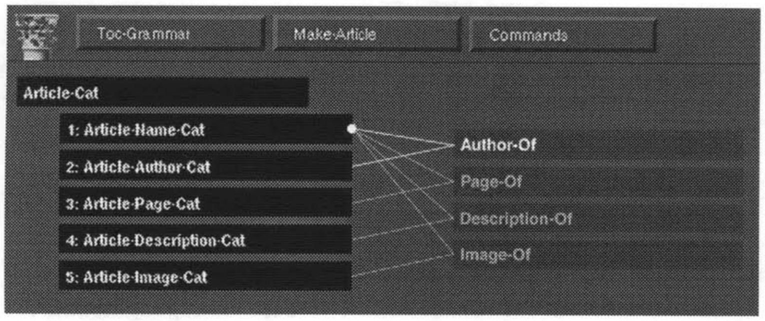


FIGURE 6.8
Visualization of the ***Make-Article*** *rule's precondition within the Scientific American grammar. The text description of the rule is shown in Figure 6.7.*

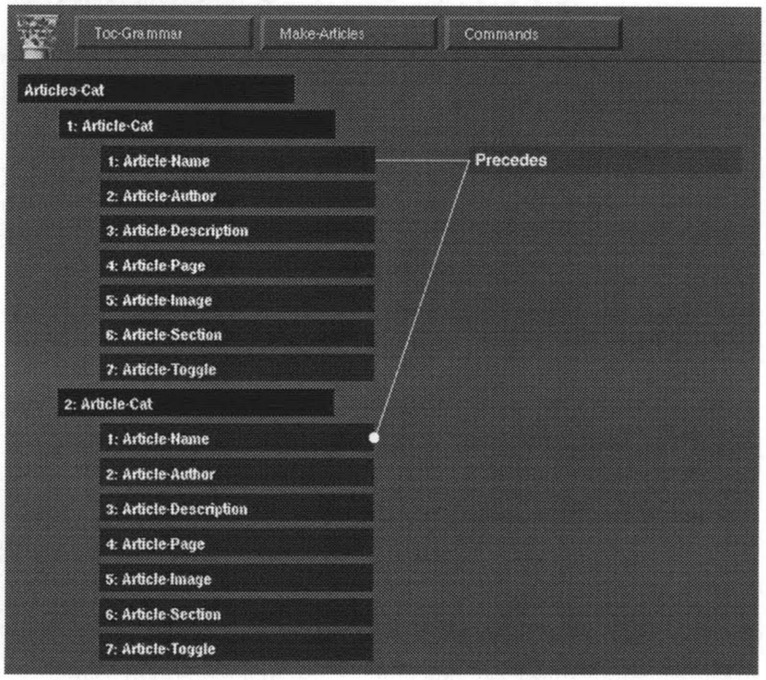


FIGURE 6.9
Visualization of the preconditions of the ***Make-Articles*** *rule within the Scientific American grammar. This rule combines two simple* ***Article-Cat*** *categories created by the rule shown in Figure 6.8. The* ***Article-Name*** *of one* ***Article-Cat*** *must stand in the* ***Precedes*** *relation to the second* ***Article-Cat.***

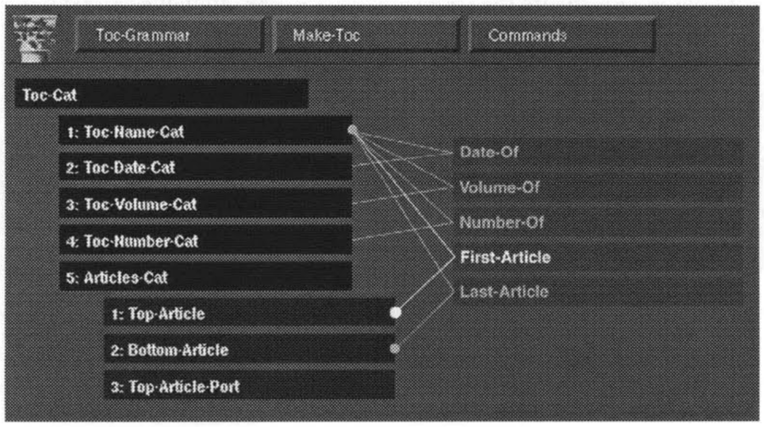


FIGURE 6.10
Visualization of the preconditions of the ***Make-Toc*** *rule within the Scientific American grammar. This rule produces a composite that represents the complete table of contents. The first indentation shows the rule elements and the second level of indentation shows the attributes of the* ***Articles-Cat*** *category. The* ***Top-Article*** *stands in the* ***First-Article*** *relation to the* ***Toc-Name-Cat*** *category.*

Alternatively, this presentation can be shortened by disabling the display of the attributes of rule elements. This option is available through the VIA command menu.

Relationships are visualized by labels with arcs. For example, the *Author-Of* relation is shown as a rectangular block with an arc coming from the second text element (the article's author) and going to the first text element (the article's title). These are highlighted as the user moves the cursor over the relation. In the more complicated rule definitions, the relations are between attributes of composites (i.e., *Precedes* and *First-Article* in Figures 6.9 and 6.10 respectively).

Modifications of existing preconditions or creation of new preconditions is currently not possible. However, a straightforward implementation to provide this functionality should be possible. Using menus, an author should be able to add elements to this rule body and link these elements through a menu of existing relationships. Also, creating completely new categories with attributes should also be straightforward. In the standard object-oriented fashion, these classes can easily be constructed dynamically and inherit the necessary slots and behaviors.

Another problem exists because of the complexity of the grammar derivations. An industrial strength editor must ensure valid derivations will be formed given a set of input. One possibility would be to use a structured editor, say one built on top of a structured graph editor, that would ensure the root categories of the language can be built out of the lexical inputs. Visualization of preconditions for the interactive scenario is more difficult and remains an open research topic in programming by demonstration.

Editing rule output constraints

The second phase of the rule creation process is the specification of output constraints to be used in subsequent realization. The goal here is to infer these constraints as a result of the actions of the author. The approach presented here utilizes the interactive paradigm of design support with another grammar disambiguating the author's actions. A generic grammar[6.1] suggests possible interpretations for grouping the elements and presents them to the author. An interative cycle records the author's decisions which are used later when generating the code for rule creation. This approach was illustrated in the sce-

6.1 This generic grammar is based on simple graphic principles of alignment and equal sizing to form groups.

nario at the beginning of this chapter in Figure 6.4A-6.4E.

For rules that only use lexical items in their input, output constraints are visualized by creating instances of media objects as indicated by the rule's lexical elements. These are then constrained relative to one another according to the rule body. Note, that this may be only one of many ways to derive a composite. For example, there could be multiple ways to create an *Article* composite, the normal article and one that reflects elements that are missing from the input.[6.2] Rules that use composite categories as input need to visualize the attributes and make an instance of that lexical item to be constrained. This has not been implemented yet.

To modify the rule output constraints, the rule editor uses the same interaction loop as the interactive design scenario but with some minor additions as seen in Figure 6.11. In both cases, the toplevel loop is used to maintain the valid system actions that are discovered through parsing. Unlike the interactive scenario, there is no option for automatic rule action selection. The successful derivations are always put on the interactive agenda. Because a number of conflicting derivations are always produced, any "intelligent" mechanism would eventually make the wrong inferences and choose inappropriate actions. By putting all derivations onto the interactive agenda, the designer is left in control of the decision process.

The first action taken in the above scenario is to ungroup the composite as it is currently described by the rule. The act of ungrouping places the individual elements on the parse table. In doing so, many system actions are added to the agenda to recombine these elements. Choosing any one exclusive action will remove all remaining actions that are no longer valid. For example, if there are multiple ways to align two elements, after one is chosen, all remaining alignment actions will be removed from the agenda.

The parser is order independent and allows recombination in any manner. Similar to the interactive scenario, the bottom-up parsing algorithm is considered successful even if the derivations cover only partial input. Here, the root symbols of the grammar are composites that forms graphic elements into new graphic groups. Multiple root categories used in the interactive scenario such as *Figure*, *Heading*, and *Body* are not needed. If another grammar is used to form the output constraints, other root categories may be employed.

6.2 Figure 4.7 illustrates such a case where the text elements *Author* and *Description* are missing from the input.

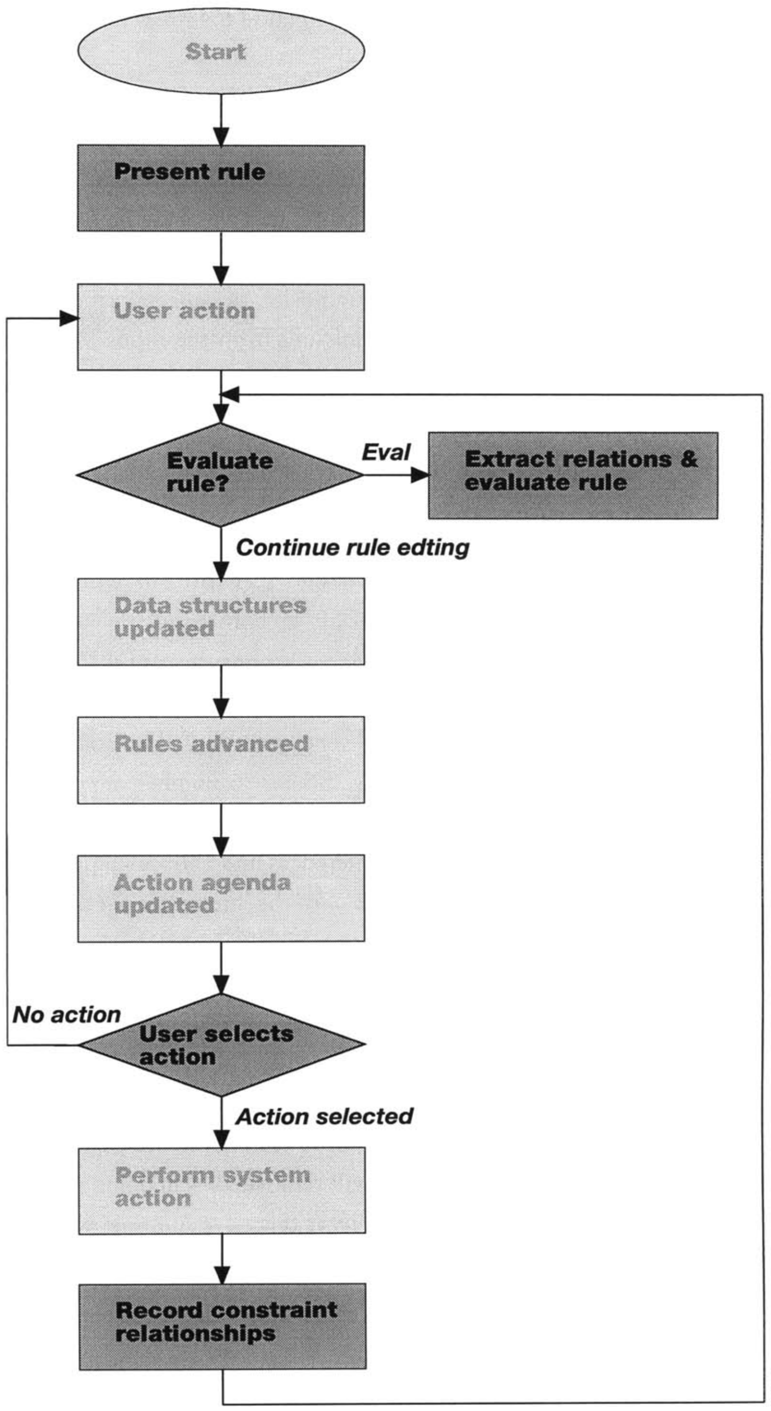
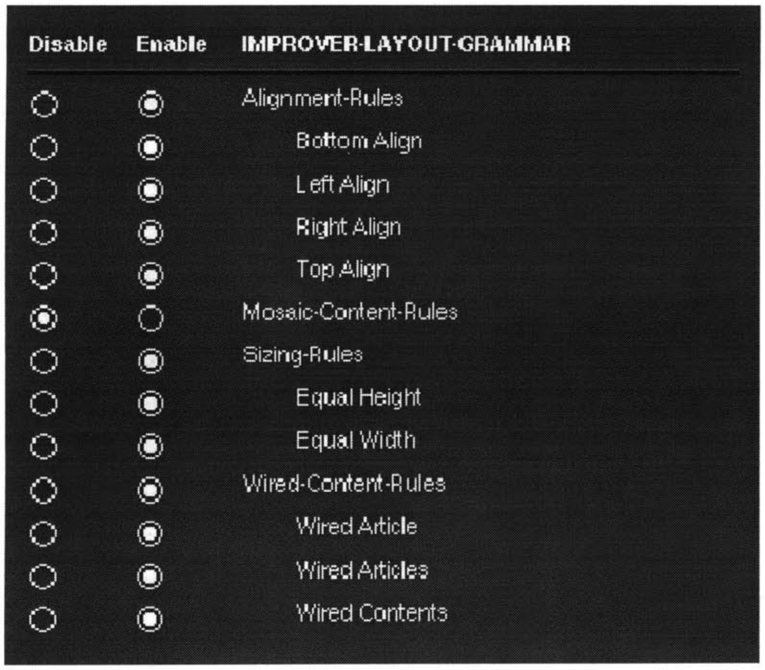


FIGURE 6.11
The toplevel loop for rule editing. This is similar to the toplevel loop for the interactive paradigm but here we record the relationships at the end of each iteration and extract these relationships at the end of the process to create the code for rule definition. The differences with the interactive paradigm are highlighted here.

This discussion illustrates the use of the rule editor to support the spatial constraints used in automatic presentation of documents. Producing rules for the interactive scenario is a bit more difficult. Here, an editor must infer constraints from the actions (possibly multiple examples) of the user. For instance, in the interactive scenario, the system will need to recognize when the position and size of elements are in the proper relationships. This means the rule editor has



Disable	Enable	IMPROVER-LAYOUT-GRAMMAR
○	●	Alignment-Rules
○	●	Bottom Align
○	●	Left Align
○	●	Right Align
○	●	Top Align
●	○	Mosaic-Content-Rules
○	●	Sizing-Rules
○	●	Equal Height
○	●	Equal Width
○	●	Wired-Content-Rules
○	●	Wired Article
○	●	Wired Articles
○	●	Wired Contents

FIGURE 6.12
*The rule set editor for VIA. By selecting (or deselecting) individual rules or classes of rules, these become activated (or deactivated) in parsing new input. Here, the **Mosaic-Content-Rules** have been disabled while **Alignment-Rules**, **Sizing-Rules** and **Wired-Content-Rules** are enabled.*

to recognize what it means to position two elements "almost left-aligned" with one another. This must include generalization techniques more commonly found in sophisticated programming by demonstration systems.

Rule set editor

As design progresses, a designer will modify and change the working assumptions of their language and its implementation. In order to support this dynamic interaction, a rule set editor is provided within VIA. This editor uses the class structure of the rules to enable or disable individual rules or sets of rules. This mechanism helps the support of explorationof design by forcing only certain rules to be active within the set of all possible rules within a language.

The rule set editor is shown in Figure 6.12. All rules within the grammar are organized by type and displayed in this list. For example, the **Align-Left** rule in Figure 6.6 is a member of the **Alignment-Rules** class. Each rule is given toggle buttons to enable and disable the rule. Each class can also be enabled or disabled. If an individual rule is toggled, only that rule is modified. When a class button is toggled, all rules within that class are either disabled, collapsed from view, or all the rules are enabled. After making the appropriate settings in this window, the design interaction can continue and only those rules specified will affect the outcome of the design.

Additional capabilities of rule editing

Any high-functionality tool should make simple things simple and difficult things possible. The tool must allow access to the smallest detail of the formalism but at the same time make the interaction seem straightforward and obvious. Using the interactive paradigm with Relational Grammars, we have shown how it is possible to create lexical definitions, composite categories with attributes, preconditions of the rule, attribute mapping within rules, and rule output constraints.

In a full functioning editor, there are a number of other capabilities that need to be supported in the specification of all aspects of the grammar formalism.

Initialize	Change the rule back to the last saved version
Create Rule	Begin the construction of a new rule
Eval Rule	Evaluate the rule placing it in the grammar
Delete Rule	Delete the rule from the grammar
Copy	Make a copy of a rule that can then be used in editing
Load	Load a grammar including lexicon, categories and rules
Save	Save a grammar including lexicon, categories and rules

In addition, the user should be able to add any number of elements to the output constraints to help articulate its presentation. These include general graphic elements and standardized objects for a particular domain (e.g., company logos) that might not be part of the input but need to be added for a complete presentation.

Related work

Since the first example of programming by demonstration, PYGMALION [Smith, 1993], research has been searching for techniques to provide simple, end-user customization of the environment. Currently, end-user programming can be categorized into four types: *preferences*, *scripting languages*, *macro recorders*, and *programming-by-demonstration* [Cypher, 1993a]. Preferences only provide customization to known situations. A fixed set of parameters and their alternatives are provided to the user. Preferences are only as good as the foresight of the application programmer. Scripting languages are a popular alternative because they provide more flexibility. However, they introduce the additional complexity of the scripting language.

Syntax and vocabulary of the language along with the concepts of *variables*, *loops* and *conditionals* must be understood by the user. Macro recorders allow the user to demonstrate actions for particular situations. Their deficiency is that they are too literal and cannot generalize to become useful in a wider range of situations.

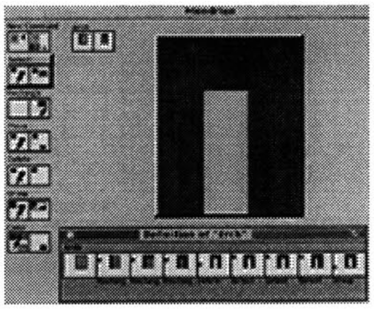


FIGURE 6.13
Mondrian is a graphical editor that learns new procedures through graphical demonstration.

Programming by demonstration uses various techniques to infer the intention of the user from example demonstrations. This problem of inferring intent makes the correct generalization very difficult. Some solutions to this problem have been: **Smallstar** [Halbert, 1993], allowing the user to select from a fixed list of alternatives; **Eager** [Cypher, 1993b] comparing multiple examples; **Turvy** [Maulsby, 1993], allowing the user to indicate relevant information; **Peridot** [Myers, 1993], requesting verification from the user; and **Mondrian** [Lieberman, 1993b], creating parameterized functions based on explicit user actions and implicit knowledge of graphical relationships within the interface (Figure 6.13).

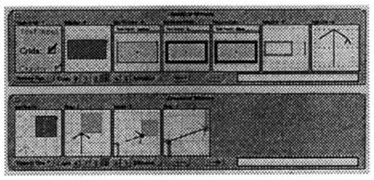
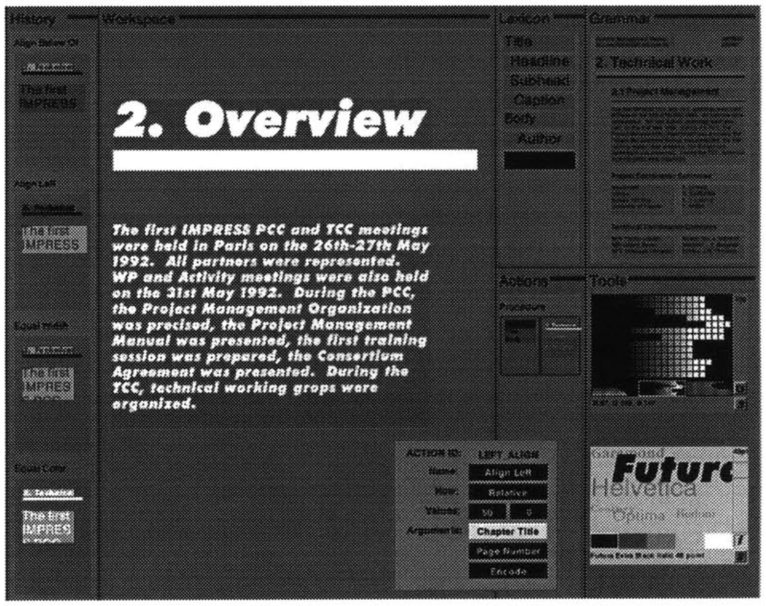


FIGURE 6.14
Chimera's editable graphical history to support the repetition of tasks found in most editors.

The problem of the flow of control is also a difficult one. **Tinker** [Lieberman, 1993a] has users write programming language expressions; **Chimera** [Kurlander, 1993; Kurlander and Feiner, 1993] presents graphical histories so users can select decision points; and **Metamouse** [Maulsby and Witten, 1993] infers branches automatically from multiple examples.

Programming by demonstration techniques aid the creation of new grammar rules. VIA does not use multiple examples but more closely resembles Peridot, using rules to identify possible inferences of groupings and constraints to apply. It also gives the user complete control in selecting the appropriate inference. Unlike Peridot, this inference is presented less obtrusively as a separate agenda to be selected when the designer feels it is appropriate. VIA also uses the technique of before and after *dominos* used in Mondrian to illustrate the effect a rule has in the context of the current design.

Abatan [Turransky, 1993] was an early system that could produce VIA grammar rules by demonstration. Abatan captured reusable graphic design knowledge from interactive user demonstrations. The system could then write out a file that contained a representation of this demonstration as a Relational Grammar rule. VIA could then read in this file (i.e. the two systems were developed on different platforms in two different languages) and transform the contents in an automatic presentation. An example scenario was demonstrated



2. Overview

The first IMPRESS PCC and TCC meetings were held in Paris on the 26th-27th May 1992. All partners were represented. WP and Activity meetings were also held on the 31st May 1992. During the PCC, the Project Management Organization was precised, the Project Management Manual was presented, the first training session was prepared, the Consertium Agreement was presented. During the TCC, technical working grops were organized.

FIGURE 6.15
Abatan, an editor developed by Alan Turransky, used to investigate the creation of rules by example.

modifying the layout of the Scientific American table of contents. Abatan's interface is shown in Figure 6.15.

In the scenario for creating grammar rules for the interactive paradigm, the use of multiple examples will most likely be necessary. In the tradition of programming by example, user involvement will ease the process of rule acquisition. By graphically demonstrating situations, new rules will be constructed and incorporated into the working grammar set.

Summary

In the larger vision of supporting the design process, this chapter begins to explore how these visual languages can be built by demonstration. This chapter highlights a number of the issues involved with the creation of rules by demonstration. In some cases this procedure is stratightforward. In other cases, it remains an open research question as to how to generalize a user's actions to infer the proper decisions to be included in the rule being constructed. By using the interactive scenario based on a generalized grammar, the rule author is supported in the creation of new rules for other visual languages. Making this a viable tool for designers and everyday users remains an open research question. By taking a unified approach to the problem of design support, we can leverage the strengths of visual languages to support this important aspect of the design process.

Conclusions

7

Introduction

I would like to see systems with enough intelligence and with enough rich vocabulary that a designer could interact with technology in an empowered way.
MURIEL COOPER
EYE MAGAZINE, AUTUMN 1994

This dissertation continues the investigation into interactive and automatic techniques to support the process of design. By providing a formal representation based on computational linguistics, new techniques and tools for the support of design are illustrated. This research has shown the effectiveness of this generalized framework for the design of interactive multimedia documents. This provides the opportunity to improve the presentation of information in a multimedia environment for users and design professionals alike. Interaction techniques afforded by this theoretical basis include automatic layout of documents, interactive paradigms for the support of authoring multimedia documents, and the use of programming-by-demonstration for the modification of grammar rules by example.

This framework is based on the use of Relational Grammars that exploit the structure of the information in creating and publishing dynamic documents. The visual languages created from Relational Grammars are analogous to natural languages but have additional requirements in order to process input. Input is in the form of domain objects and relationships between the elements. In the case of multimedia documents, this input is in the form of text, images, movies, audio, etc. Special processing requirements for visual languages are accommodated by the use of the Relational Grammar formalism. Input is not constrained to a particular order, and non-monotonic changes, e.g., moving, resizing, and deleting, are accounted for the parsing algorithm.

The structure of the information within the domain dictates how the construction of *visual sentences* are formed. This structure is the basis of an *Architecture of Information* which supports design in both interactive and automatic paradigms.

The automatic presentation of information is one of the main affordances of this visual language. In the larger problem of multimedia generation, this research illustrates a new techniques for solving the problem of media articulation. By processing tagged input that contains relationships between the elements, presentations are automatically created that are sensitive to their delivery environments. The same information can be dynamically reconfigured for different output media and different uses. This is a key technology in supporting a more agile model of publishing.

Visual languages, as described in this research, provide a rich source of interaction paradigms to support interactive design. These paradigms include the incremental improvement of design during construction, top-down refinement of design solutions, graphic completion of partial designs, and verification of partially complete artifacts. In the improver-based scenario, input can be quickly sketched without concern for the specific details the design. The system then installs constraints building up composite groups within the design. By using a language of design, the creation of these documents can be made within the context of a given style of layout.

In order to make these formal methods and new techniques accessible to designers, new tools need to be constructed. These tools need to support the process of *meta-design*, design that creates descriptions used at a time when the designer can not be in the production loop. By incorporating visual languages in the improver-based scenario, rules can be modified by demonstration. The system infers design decisions and creates grammars by the examples provided that can then be used offline.

This research was explored within the prototype environment *VIA: Visual Information Architecture*. VIA incorporates three key technologies developed at other research organizations: a constraint system to maintain both spatial and temporal constraints, the tapping mechanism to provide dynamic interfaces, and most importantly, the Relational Grammar system. This chapter continues with future work that is indicated by this research. A brief description of the implementation of VIA and its subsystems is also provided.

Future research

In support of the design process, Relational Grammars have proven useful as the basis for a number of different interaction paradigms. An open question is how much further higher dimensional grammars, like the ones used here, can be pushed to support the creative activity of design. This larger vision suggests a number of future research directions. Among these are the utilization of ambiguity during the design process, cooperative design with mixed initiative between system and user, grammar creation by demonstration, design by analogy, improved parsing algorithms, enhanced expressiveness, and new applications of Relational Grammars.

Ambiguity in design

Traditionally, parsing has attempted to reduce the amount of ambiguity in any interpretation. However, to support a creative process ambiguity should be viewed as beneficial and something that needs to be maintained until later design decisions have been able to disambiguate the designer's intention. These ambiguous interpretations can be viewed as alternative solutions in the larger design space. Traditionally, computers have made drawings look more like final presentations than the quick idea sketches they really are. Building an environment to explore this space and enable better solutions is a very important area of research.

The ability to search the design space and maintain multiple design alternatives is essential for any successful design system. A more interesting and rich design environment would track design decisions and provide support for backtracking of decisions found to be unproductive. The visual agenda described earlier begins to address this issue. The support of ambiguity in early design stages has parallels in natural language [Gazdar and Mellish, 1989], and is an important research direction.

Cooperative computer-aided design

Two distinct paradigms that use Relational Grammars for design support have been illustrated. Cooperative CAD [Kochhar, 1990] is a different approach that embraces both automatic and interactive methods within a single paradigm. This cooperative paradigm puts the user in control to manually articulate design decisions but also supports automatic design exploration by the system. Relational Grammars can provide a formalism for this approach to the integration of parsing and generation in the iterative cycle of design.

Extensions of this work could include the development of a more integrated environment where interpretation and generation are part of the same design process.

Grammars by demonstration

The very nature of design suggests that the solution to the problem is not known a priori. In fact, as the design progresses, initial assumptions and decisions may be redefined or dropped altogether. This suggests that the language of a design, i.e., the basic vocabulary and the rules for combination, evolves as the solution is explored. What is important is having an environment that will respond to this dynamic character of the design process.

A designer should have the ability to redefine rules and add new ones to support the design process. An area of research that can provide some insight is programming-by-demonstration [Cypher, 1993a]. With this approach, designers could modify an existing grammar or create new grammars without coding. This would help create a design environment in which nonprogrammers could modify the existing rule sets. A rule editor that seamlessly combines rule creation and modification into the design cycle will be necessary to empower designers with these new tools.

Design by analogy

The difficult problem of creative design is not directly addressed by this dissertation. However, one approach to this problem would be to analyze design based on the structure of the domain language and then use analogy to find similar structures in previously stored cases. Using the articulation methods of the newly found cases, alternative design solutions could automatically be created. For example, if a case library contained both the Scientific American and WIRED grammars, input could be analyzed and automatically matched against the grammars of these two cases. Alternative displays could then be created that would produce the layouts of Figures 4.4 and 4.6. Additionally, these examples might mutate structure, or layout constraints, to find more unique design solutions.

Parsing algorithms

Further research is necessary on parsing algorithms to support multimedia applications. For example, depending on properties of the content database, deterministic LR-style algorithms for Relational Grammars may be possible. This approach would be more efficient than the one currently being used. Research on such algorithms for multidimensional grammars is ongoing. On the other hand, nondeterminism in parsing, along with the possibility of ambiguity in derivations, may play the role of generating more than one possible presentation, which could in turn be critiqued by a higher-level control structure in more "intelligent" applications. Still another idea is to use predictive-style parsers to help do some of the content selection [Wittenburg, 1993]. Many issues remain open and this will be a rich area for future research.

New applications and enhanced expressiveness

Relational Grammars can support a number of languages of which only a small subset has been described. One interesting area of research is the use of this formalism to help disambiguate the structured input of user gestures from stylus, glove, or 3D spaceball. These alternative input devices provide new opportunities to use Relational Grammars to support the understanding of multimodal input. On the output side, new graphic articulations could include the creation of three dimensional artifacts. The creation of visual languages to support *Information Landscapes* is an obvious extension of this work.

Exploration of the use of Relational Grammars to support various applications within the design arena have been identified. On-line training manuals and the design of dynamic presentations is a continuing research focus. In addition, as interest in the Internet continues to grow, it is becoming more apparent that we need better techniques to support the automated design and presentation of information. One interesting application for Relational Grammars is in the production of timely, personalized newspapers. Intelligent agents would retrieve information over the network and submit it to the grammar for automatic presentation. Another application would be the creation of an HTML viewer that would be capable of using relations between elements in the markup file to parse and articulate a graphic presentation of the data. This presentation would be sensitive to the display environment but would maintain the *look and feel* desired by the document author. Relational Grammar Markup Languages, *RGML,* have been suggested and are being explored.

Today, many printed magazines are created in high functionality page layout systems, such as QuarkXpress. An interesting application of Relational Grammars would be to support the migration of these traditional print documents into their on-line counterpart. Relational Grammars can be used to translate document from one format to another. In this application, VIA could translate the original electronic document, parsing the individual pieces of text with spatial and graphic relationships to be viewed in an on-line Internet viewer, such as Mosaic.

In many professions, such as advertising or architecture, the same information needs to be presented in different contexts and for different purposes. In architecture, for example, if there was a common graphic database available (which is becoming more and more the norm), grammars could be used to help in the presentation of this information. Floorplans, sketches, and 3D renderings of buildings could be automatically constructed and presented to the client as the need arises.

Implementation

There are a number of subsystems on which this research is based and they are briefly described. VIA has been developed on the Macintosh family of computers in Macintosh Common Lisp (MCL) and the Common Lisp Object System (CLOS). It relies on a number of Apple's multimedia capabilities, including QuickTime movies, built-in audio, and enhanced imaging techniques of Quickdraw.

VIA incorporates the results from three independent research programs including the constraint solver, DeltaBlue, the tapping mechanism from the Steamer project, and of course, the parser and grammar formalism from the Relational Grammar system. Each of these provide necessary functionality in the VIA system. DeltaBlue [Freeman-Benson, Maloney and Borning, 1990; Maloney, 1991] from the University of Washington is being used to maintain both graphic and temporal constraints. DeltaBlue is designed to handle approximately ~20,000 interactive constriants in a non-cyclic propagation method. DeltaBlue has been very effective for this application within VIA. The Steamer system [Hollan, et al., 1984] was one of the first systems to employ interactive graphics to control and visualize simulations and complex processes. A derivation of this work has been incorporated to support the dynamic multimedia presentations created in VIA. The intellectual basis for this work is the Relational Grammar formalism and parser [Wittenburg, et al., 1991; Wittenburg, 1992; Wittenburg, 1993]. It is one of the higher-dimensional grammars that will help to analyze and produce multimedia and multimodal applications in the future.

Summary

The main contribution of this work is the application of an independently motivated parsing algorithm to support the process of design. This research focuses on three types of design support:

- **the automatic presentation of information,**
- **the interactive design of information,** and
- **the creation of grammars by demonstration**

In the digital domain, we need techniques that will support meta-design. This process produces descriptions, e.g., visual languages, that are used at a later time to support the automatic presentation of information. Similarly, we can incorporate visual languages to support the interactive design process. As the system *watches over* design actions, graphic inferencing causes incremental improvements to the artifact under construction. This process ensures that a design conforms to a predetermined style of layout. Using these same interactive techniques, grammars can be created by demonstration without users having to write code. Design and the domain of publishing are rapidly changing. The theories and techniques described in this dissertation provide insight into how to publish and present information in this more dynamic environment.

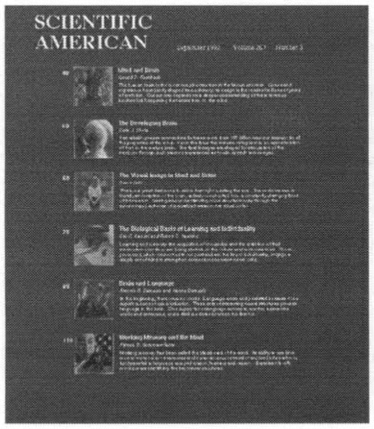
Appendix: table of contents grammar

8

Introduction

This appendix illustrates the grammar used for the layout of the Scientific American table of contents. Example outputs of this grammar are illustrated in Figures 4.3, 4.6 and 4.7 (two of which are reproduced here in Figures 8.1 and 8.2).

This appendix begins by describing the input to the parsing process. Then, the three major components of the grammar are described. These are the lexicon, the composite categories or groups, and the rules for combination. Note that this grammar not only produces the standard layout for the Scientific American table of contents (Figure 8.1) but also accommodates the situation where data is missing (as in Figure 8.2 and 8.3). Three separate derivations within the same grammar generate the examples in these figures.


SCIENTIFIC AMERICAN September 1992 Volume 267 Number 3

48 Mind and Brain
[illegible]

60 The Developing Brain
[illegible]

68 The Visual Image in Mind and Brain
[illegible]

78 The Biological Basis of Learning and Individuality
[illegible]

88 Brain and Language
[illegible]

110 Working Memory and the Mind
[illegible]

FIGURE 8.1
Presentation of complete table of contents.

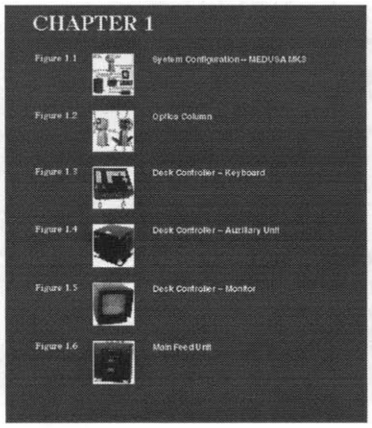

CHAPTER 1

Figure 1.1 System Configuration -- MEDUSA MK3

Figure 1.2 Optics Column

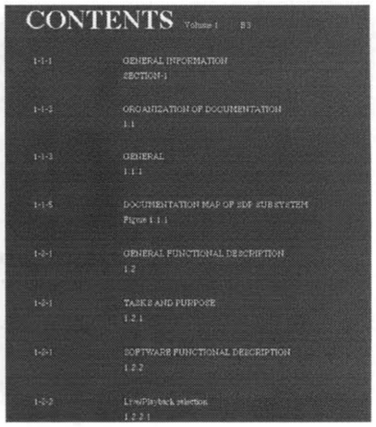
Figure 1.3 Desk Controller - Keyboard

Figure 1.4 Desk Controller - Auxiliary Unit

Figure 1.5 Desk Controller - Monitor

Figure 1.6 Main Feed Unit

FIGURE 8.2
Presentation of table of contents missing the author and description.


CONTENTS Volume 1 B3

1-1-1 GENERAL INFORMATION
SECTION-1

1-1-3 ORGANIZATION OF DOCUMENTATION
1.1

1-1-3 GENERAL
1.1.1

1-1-5 DOCUMENTATION MAP OF SDF SUBSYSTEM
Figure 1.1.1

1-2-1 GENERAL FUNCTIONAL DESCRIPTION
1.2

1-2-1 TASKS AND PURPOSE
1.2.1

1-2-1 SOFTWARE FUNCTIONAL DESCRIPTION
1.2.2

1-2-2 Live/Playback selection
1.2.2.1

FIGURE 8.3
Another variation within the grammar that accommodates missing data.

Input to parser

Sample input for this example is shown below. The system preprocesses this list of information and produces an object database that includes relationships between the elements. For instance, text objects are created for all *article-name*, *article-author*, and *article-description* elements. Likewise, image objects are created for all *article-image* elements. Other objects are similarly created for the rest of the input. Then, relationships are installed between the objects that have been created. For instance, the relationship *author-of* is installed between the two text elements "**Gerald D. Fischbach**" and "**Mind and Brain**." The relationship that "**Mind and Brain**" precedes "**The Developing Brain**" is similarly installed in the database.

```
;;; Description of input for the September 1992
;;; issue of Scientific American

(defparameter *toc-description*
  `(:toc-name                 "Scientific American"
    :toc-date                 "September 1992"
    :toc-volume               "Volume 267"
    :toc-number               "Number 3"
    :toc-articles
    ((:article-name           "Mind and Brain"
      :article-author         "Gerald D. Fischbach"
      :article-page           "48"
      :article-description    "The human brain is the most complex..."
      :article-image          "mind-and-brain")
     (:article-name           "The Developing Brain"
      :article-author         "Carla J. Shatz"
      :article-page           "60"
      :article-description    "Remarkably precise connections ..."
      :article-image          "developing-brain")
     (:article-name           "The Visual Image in Mind and Brain"
      :article-author         "Semir Zeki"
      :article-page           "68"
      :article-description    "There is a great deal more to vision..."
      :article-image          "image-in-mind")
     (:article-name           "The Biological Basis of Learning and
                              Individuality"
      :article-author         "Eric R. Kandel and Robert D. Hawkins"
      :article-page           "78"
      :article-description    "Learning and memory-the acquisition..."
      :article-image          "biological-basis")
     (:article-name           "Brain and Language"
      :article-author         "Antonio R. Damasio and Hanna Damasio"
      :article-page           "88"
      :article-description    "In the beginning, there were no words..."
      :article-image          "brain-and-language")
     (:article-name           "Working Memory and the Mind"
      :article-author         "Patricia S. Goldman-Rakic"
      :article-page           "110"
      :article-description    "Working memory has been called..."
      :article-image          "working-memory"))))
```

Primitive lexicon

The lexicon defines the the primitive elements used in the language. This grammar uses elements such as, text, numbers and images as its primitives. These are included below as *article-name-cat*, *article-page-cat* and *article-image-cat*. The following form first defines the lexicon used in this table of contents example.

```
(deflexicon toc-lexicon)
```

Then, entries into this lexicon are added. Each entry is identified by its name followed by the name of the lexicon into which it is being added. Each entry also defines the different alternative uses of that element (i.e., the different *senses*). This grammar only uses one sense for each entry. Similar to rules, entries also have `:OUT` forms. Typically, each sense provides `:OUT` forms to make an instance of a media object in the presentation environment.

```
;;; Table of contents entries
(defentry (toc-name-cat toc-lexicon)
  (:cat toc-name-cat
   :out ((make-instance 'article-paragraph :key (cover 0)))))

(defentry (toc-date-cat toc-lexicon)
  (:cat toc-date-cat
   :out ((make-instance 'toc-date :key (cover 0)))))

(defentry (toc-number-cat toc-lexicon)
  (:cat toc-number-cat
   :out ((make-instance 'toc-number :key (cover 0)))))

(defentry (toc-volume-cat toc-lexicon)
  (:cat toc-volume-cat
   :out ((make-instance 'toc-volume :key (cover 0)))))

;;; Article entries
(defentry (article-name-cat toc-lexicon)
  (:cat article-name-cat
   :out ((make-instance 'article-name :key (cover 0)))))

(defentry (article-author-cat toc-lexicon)
  (:cat article-author-cat
   :out ((make-instance 'article-author :key (cover 0)))))

(defentry (article-description-cat toc-lexicon)
  (:cat article-description-cat
   :out ((make-instance 'article-paragraph :key (cover 0)))))

(defentry (article-page-cat toc-lexicon)
  (:cat article-page-cat
   :out ((make-instance 'article-page :key (cover 0)))))

(defentry (article-image-cat toc-lexicon)
  (:cat article-image-cat
   :out ((make-instance 'article-image :key (cover 0)))))

(defentry (article-paragraph-cat toc-lexicon)
  (:cat article-description-cat
   :out ((make-instance 'article-paragraph :key (cover 0)))))

(defentry (article-section-cat toc-lexicon)
  (:cat article-section-cat
   :out ((make-instance 'article-section :key (cover 0)))))
```

Composite categories

When a rule fires, a composite group, or category, is formed. These categories are implemented as structures with attributes. As derivations are built, these categories can store and propagate information through their attributes. The complete derivations of the three alternatives this grammar produces are represented by the categories, *toc-cat*, *toc-simple-cat*, and *toc-wo-cat*.

These structures use inheritance in their definition. For example, the first category defined, *toc-internal-cat*, is used as a base type for other categories.

```
;;; Categories for complete table of contents
(defcategory (toc-internal-cat)      toc-name
                                     toc-date
                                     toc-volume
                                     toc-number
                                     toc-top-article
                                     toc-bottom-article)
(defcategory (toc-cat toc-internal-cat))
(defcategory (toc-simple-cat toc-internal-cat))
(defcategory (toc-wo-cat toc-internal-cat))

;;; Categories for articles
(defcategory (article-internal-cat)  article-name
                                     article-author
                                     article-description
                                     article-page
                                     article-image
                                     article-section
                                     article-toggle)
(defcategory (article-cat article-internal-cat))
(defcategory (article-wo-cat article-internal-cat))
(defcategory (article-simple-cat)    article-name
                                     article-page
                                     article-image)

;;; Categories for article groups
(defcategory (articles-internal-cat) top-article
                                     bottom-article
                                     top-article-port)
(defcategory (articles-cat articles-internal-cat))
```

Grammar rules

The next step is to define the grammar. The following form defines the table of contents grammar used in this example.

```
(defgrammar 'toc-grammar
  :startcat '(toc-cat toc-simple-cat toc-wo-cat))
```

This grammar produces three alternative presentations within the Scientific American style. The symbol *startcat* in the *defgrammar* form above, identifies the composite categories that represent a complete derivation. *Toc-cat* is the derivation which produces the complete presentation as illustrated in Figure 8.1. *Toc-simple-cat* creates a presentation when the author and description are missing, shown in Figure 8.2. *Toc-wo-cat* creates a presentation using only a name, section and page elements, as shown in Figure 8.3.

Each variation uses four rules to create the complete derivation: 1) a rule to form an article group, 2) a rule to combine two article groups together, 3) a rule to extend a set of article groups, and 4) a rule to complete the table of contents derivation. The rules listed below are grouped by derivation.

```
;;; Complete derivation of full table of contents

;;; Rule 1.1: Make an article group
(defrule (make-article toc-grammar via-rule)
  (0 article-cat (setf (article-name        0) 1
                       (article-author      0) 2
                       (article-page        0) 3
                       (article-description 0) 4
                       (article-image       0) 5))
  (1 article-name-cat)
  (2 article-author-cat      (author-of      2 1))
  (3 article-page-cat        (page-of        3 1))
  (4 article-description-cat (description-of 4 1))
  (5 article-image-cat       (image-of       5 1))
  :out
  (spaced-right-of  1 5 :spacing *toc-xspacing*)
  (top-align        1 5)
  (spaced-right-of  2 5 :spacing *toc-xspacing*)
  (spaced-below     2 1)
  (spaced-below     4 2 :spacing 5.0)
  (spaced-right-of  5 3)
  (top-align        5 3)
  (spaced-right-of  4 5 :spacing *toc-xspacing*)
  (setf (background-color 1) *layout-back-color*)
  (setf (background-color 2) *layout-back-color*)
  (setf (background-color 3) *layout-back-color*)
  (setf (background-color 4) *layout-back-color*)
  (set-font-spec 1 '(10 "Helvetica" :plain))
  (set-font-spec 2 '( 8 "Helvetica" :italic))
  (set-font-spec 4 '( 8 "Helvetica" :plain))
  :rule-constraints ;;; these constraints are for the rule editor
  (move-to 5 100 100)
  (setf (background-color 1) *black-color*)
  (setf (background-color 2) *black-color*)
  (setf (background-color 3) *black-color*)
  (setf (background-color 4) *black-color*)
  )

;;; Rule 1.2 Combine two articles together
(defrule (combine-articles toc-grammar via-rule)
  (0 articles-cat (setf (top-article      0) (article-name  1)
                        (bottom-article   0) (article-name  2)
                        (top-article-port 0) (article-image 1)))
  (1 article-cat)
  (2 article-cat (precedes (article-name 1) (article-name 2)))
  :out
  (spaced-below (article-image 2) (article-image 1)
                :spacing *spacing*))
  (left-align   (article-image 2) (article-image 1)))
```

```
;;; Rule 1.3 Extend article groups

(defrule (extend-articles toc-grammar via-rule)
  (0 articles-cat (setf (top-article      0) (article-name   1)
                        (bottom-article   0) (bottom-article 2)
                        (top-article-port 0) (article-image  1)))
  (1 article-cat)
  (2 articles-cat (precedes (article-name 1) (top-article 2)))
  :out
  (spaced-below (top-article-port 2) (article-image 1)
                :spacing *spacing*)
  (left-align   (top-article-port 2) (article-image 1))

;;; Rule 1.4, Final rule to complete the table of contents
(defrule (make-toc toc-grammar via-rule)
  (0 toc-cat (setf  (toc-name          0) 1
                    (toc-date          0) 2
                    (toc-volume        0) 3
                    (toc-number        0) 4
                    (toc-top-article   0) (top-article 5)
                    (toc-bottom-article 0) (bottom-article 5)))
  (1 toc-name-cat)
  (2 toc-date-cat   (date-of           2 1))
  (3 toc-volume-cat (volume-of         3 1))
  (4 toc-number-cat (number-of         4 1))
  (5 articles-cat   (first-article     1 (top-article 5))
                    (last-article      1 (bottom-article 5)))
  :out
  (spaced-right-of  2 1)
  (spaced-right-of  3 2)
  (spaced-right-of  4 3)
  (bottom-align     2 1)
  (bottom-align     3 2)
  (bottom-align     4 3)
  (setf (background-color 1) *layout-back-color*)
  (setf (background-color 2) *layout-back-color*)
  (setf (background-color 3) *layout-back-color*)
  (setf (background-color 4) *layout-back-color*)
  (set-font-spec 1 '("Times" 36 :bold))
  (set-font-spec 2 '("Times" 12 :plain))
  (set-font-spec 3 '("Times" 12 :plain))
  (set-font-spec 4 '("Times" 12 :plain))
  (spaced-below     (top-article-port 5) 1  :spacing *spacing*)
  (spaced-right-of  1 (top-article-port  5) :spacing *spacing*)
  (move-to 1 100 20))
;;; Derivation of table of contents missing author and description

;;; Rule 2.1: Make an article group
(defrule (make-article-simple toc-grammar via-rule)
  (0 article-simple-cat (setf   (article-name  0) 1
                                (article-page  0) 2
                                (article-image 0) 3))
  (1 article-name-cat)
  (2 article-page-cat  (page-of  2 1))
  (3 article-image-cat (image-of 3 1))
  :out
  (setf (background-color 1) *layout-back-color*)
  (setf (background-color 2) *layout-back-color*)
  (set-font-spec 1 '(14 "Helvetica" :bold))
  (set-font-spec 2 '(10 "Helvetica" :bold))
  (spaced-right-of  1 3 :spacing *spacing*)
  (top-align        1 3)
  (spaced-right-of  3 2)
  (top-align        3 2)
  :rule-constraints
  (move-to 2 100 100)
  (setf (background-color 1) *black-color*)
  (setf (background-color 2) *black-color*))

;;; Rule 2.2 Combine two articles together
(defrule (combine-articles-simple toc-grammar via-rule)
  (0 articles-cat (setf  (top-article     0) (article-name  1)
                         (bottom-article  0) (article-name  2)
                         (top-article-port0) (article-image 1)))
  (1 article-simple-cat)
  (2 article-simple-cat (precedes (article-name 1)
                                  (article-name 2)))
  :out
  (spaced-below (article-image 2) (article-image 1)
                :spacing *spacing*)
  (left-align   (article-image 2) (article-image 1)))

;;; Rule 2.3 Extend article groups
(defrule (extend-articles-simple toc-grammar via-rule)
  (0 articles-cat (setf (top-article      0) (article-name 1)
                        (bottom-article   0) (bottom-article 2)
                        (top-article-port 0) (article-image 1)))
  (1 article-simple-cat)
  (2 articles-cat (precedes (article-name 1)  (top-article 2)))
  :out
  (spaced-below (top-article-port 2) (article-image 1))
                :spacing 25)
  (left-align   (top-article-port 2) (article-image 1)))

;;; Rule 2.4 Make a complete table of contents in simplified form.
(defrule (make-toc-simple toc-grammar via-rule)
  (0 toc-simple-cat (setf (toc-name          0) 1
                          (toc-top-article    0) (top-article     2)
                          (toc-bottom-article 0) (bottom-article
2)))
  (1 toc-name-cat)
  (2 articles-cat (first-article 1 (top-article    2))
                  (last-article  1 (bottom-article 2)))
  :out
  (setf (background-color 1) *layout-back-color*)
  (set-font-spec 1 '(36 "Times" :bold))
  (spaced-below    (top-article-port 2) 1 :spacing 25)
  (spaced-right-of (top-article-port 2) 1 :spacing -100)
  (move-to 1 185 30))
```

```
;;; Derivation of table of contents with only a name, section and
page.

;;; Rule 3.1: Make an article group
(defrule (make-article-wo toc-grammar via-rule)
  (0 article-wo-cat (setf (article-name    0) 1
                          (article-section 0) 2
                          (article-page    0) 3))
  (1 article-name-cat)
  (2 article-section-cat  (section-of 2 1))
  (3 article-page-cat     (page-of    3 1))
  :out
  (setf (background-color 1) *layout-back-color*)
  (setf (background-color 3) *layout-back-color*)
  (setf (background-color 2) *layout-back-color*)
  (spaced-right-of 1 3 :spacing 70)
  (top-align       1 3)
  (spaced-below    2 1)
  (left-align      2 1)
  (set-font-spec 1 '(12 "Times" :plain))
  (set-font-spec 2 '(12 "Times" :plain))
  (set-font-spec 3 '(12 "Times" :plain))
  :rule-constraints
  (move-to 3 100 100))

;;; Rule 3.2 Combine two articles together
(defrule (combine-articles-wo toc-grammar via-rule)
  (0 articles-cat (setf (top-article      0) (article-name 1)
                        (bottom-article   0) (article-name 2)
                        (top-article-port 0) (article-name 1)))
  (1 article-wo-cat)
  (2 article-wo-cat (precedes (article-name 1) (article-name 2)))
  :out
  (spaced-below (article-name 2) (article-section 1)
                :spacing *spacing*)
  (left-align   (article-name 2) (article-name 1)))

;;; Rule 3.3 Extend article groups
(defrule (extend-articles-wo toc-grammar via-rule)
  (0 articles-cat (setf (top-article      0) (article-name   1)
                        (bottom-article   0) (bottom-article 2)
                        (top-article-port 0) (article-name   1)))
  (1 article-wo-cat)
  (2 articles-cat (precedes (article-name 1) (top-article 2)))
  :out
  (spaced-below (top-article-port 2) (article-section 1)
                :spacing *spacing*)
  (left-align   (top-article-port 2) (article-name 1)))

;;; Rule 3.4 Make a complete table of contents.
(defrule (make-toc-wo toc-grammar via-rule)
  (0 toc-wo-cat (setf (toc-name          0) 1
                      (toc-volume        0) 3
                      (toc-number        0) 4
                      (toc-top-article   0) (top-article    4)
                      (toc-bottom-article0) (bottom-article 4)))
  (1 toc-name-cat)
  (2 toc-volume-cat (volume-of 2 1))
  (3 toc-number-cat (number-of 3 1))
  (4 articles-cat   (first-article 1 (top-article    4))
                    (last-article  1 (bottom-article 4)))
  :out
  (setf (background-color 1) *layout-back-color*)
  (setf (background-color 2) *layout-back-color*)
  (setf (background-color 3) *layout-back-color*)
  (bottom-align 1 3)
  (bottom-align 1 2)
  (spaced-right-of 3  2)
  (spaced-right-of 2  1)
  (spaced-below    (top-article-port 4) 1 :spacing *spacing*)
  (spaced-right-of (top-article-port 4) 2 :spacing -100)
  (set-view-font 1 '(36 "Times" :bold))
  (set-view-font 2 '(12 "Times" :plain))
  (set-view-font 3 '(12 "Times" :plain))
  (move-to 1 70 20))
```

References

9

[Abelson, et al., 1985]	Abelson, H., Sussman, G., Sussman, J. *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, Massachusetts, 1985.
[Aho, 1986]	Aho, A.V., R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, Massachusetts., 1986.
[Aldus, 1993]	Aldus. Intellidraw, 1993.
[Allen, 1983]	Allen, J. Maintaining Knowledge About Temporal Intervals, *Communications of the ACM*, 26, 11 (1983), pp. 832-843.
[Andre, et al., 1993]	Andre, E., Finkler, W., Graf, W., Rist, T., Schauder, A. and Wahlster, W. WIP: The Automatic Synthesis of Multimodal Presentations, *Intelligent Multimedia Interfaces* (Maybury, M. ed.), AAAI Press and MIT Press, Cambridge, Massachusetts, 1993, pp. 75-93.
[Arens, et al., 1988]	Arens, Y., Miller, L., Shapiro, S. and Sondheimer, N. Automatic Construction of User-Interface Displays, In *Proceedings of the American Association of Artificial Intelligence*, Vol. 2 (August 21-26, St. Paul, Minnesota). 1988, pp. 808-813.
[Arens, Hovy, and Vossers, 1993]	Arens, Y., Hovy, E. and Vossers, M. On the Knowledge Underlying Multimedia Presentations, *Intelligent Multimedia Interfaces* (Maybury, M. ed.), AAAI Press and MIT Press, Cambridge, Massachusetts, 1993, pp. 280-306.
[Bertin, 1983]	Bertin, J. *Semiology of Graphics*, University of Wisconsin Press, Madison, Wisconsin, 1983.

[Bordwell and Thompson, 1990]	Bordwell, D. and Thompson, K. *Film Art: An Introduction*, 3rd Edition, McGraw-Hill, Inc., New York, New York, 1990.
[Borning, 1981]	Borning, A. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory, *ACM Transactions on Programming Languages and Systems* 3, 4 (October, 1981), pp. 353-387.
[Braun, 1992]	Braun, M. *Picturing Time*, University of Chicago Press. Chicago, Illinois, 1992.
[Bryan, 1988]	Bryan, M. *SGML An Author's Guide to the Standard Generalized Markup Language*, Addison-Wesley, New York, New York, 1988.
[Chakravarthy, 1993]	Chakravarthy, A. Toward Semantic Retrieval of Pictures and Video, In *Intelligent Multimedia Information Retrieval Systems and Management, RIAO'94*, (October 11-13, Rockefeller University, New York). 1994, pp. 676-686.
[Chomsky, 1957]	Chomsky, N. *Syntactic Structures*, Mouton, The Hague, 1957.
[Collier and Karlin, 1993]	Collier, G. and Karlin, P. Grammar of Design: Applying Grammatical Techniques to Phone Engineering, *IEEE International Conference on Communications* '93, (Geneva, Switzerland). 1993.
[Cooper, 1989]	Cooper, M. Computers and Design, *Design Quarterly*, 142, Warlker Art Center and MIT, 1989.
[Coyne, et al., 1990]	Coyne, R., Rosenman, M., Radford, A., Balachandran, M. and Gero, J. *Knowledge-Based Design Systems*, Addison-Wesley Publishing Company, New York, New York, 1990.
[Crimi, et al., 1989]	Crimi, C., Guerico, A., Tortora, G., and Tucci, M. An Intelligent Iconic System to Generate and to Interpret Visual Languages, In *IEEE Workshop on Visual Languages*, (October 4-6, Rome, Italy). 1989.
[Cypher, 1993a]	Cypher, A., (ed.) *Watch What I Do: Programming by Demonstration*, MIT Press, Cambridge, Massachusetts, 1993.
[Cypher, 1993b]	Cypher, A. Bringing Programming to End Users, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 1-11.

[Cypher, 1993c]	Cypher, A. Eager: Programming Repetitive Tasks by Example, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. *205-217.*
[Dondis, 1973]	Dondis, D. A Primer of Visual Literacy, MIT Press, Cambridge, Massachusetts, 1973.
[Earley, 1970]	Earley, J. An Efficient Context-Free Parsing Algorithm, *Communications of the ACM*, 13 (1970), pp. 94-102.
[Ehrig, Nagl and Rozenberg, 1986]	Ehrig, H., Nagl, M. and Rozenberg, (eds.) Graph Grammars and their Application to Computer Science, *3rd International Workshop*, Warrenton, Virginia, Springer-Verlag, New York, 1986.
[Eisner, 1989]	Eisner, W. *Comics and Sequential Art*, Poorhouse Press, Tamarac, Florida, 1989.
[Feiner, 1988]	Feiner, S. A grid-based approach to automating display layout, In *Proceedings of Graphics Interface '88* (June 6-10, Edmonton, Canada). 1988, pp. 192-197.
[Feiner, et al., 1992]	Feiner, S., Mackinlay, J. and Marks, J. Automating the Design of Effective Graphics for Intelligent User Interfaces, Tutorial Notes 30, *Conference on Human Factors in Computing Systems*, (May 3-7, Monterey, California). 1992.
[Feiner and McKeown, 1993]	Feiner, S. and McKeown, K. Automating the Generation of Coordinated Multimedia Explanations, *Intelligent Multimedia Interfaces* (Maybury, M. ed.), AAAI Press and MIT Press, Cambridge, Massachusetts, 1993, pp. 117-138.
[Fischer, et al., 1988]	Fischer, G., Lemke, A, Mastaglio, T. and Morch, A. *Critics: An Emerging Approach to Knowledge-Based Human Computer Interaction*, University of Colorado, Bolder, Colorado, 1988.
[Fischer and Morch, 1988]	Fischer, G. and Morch, A. CRACK: A Critiquing Approach to Cooperative Kitchen Design, *International Conference on Intelligent Tutoring Systems*, (Montreal, Canada). 1988.
[Flemming, 1987]	Flemming, U. The Role of Shape Grammars in the Analysis and Creation of Designs, *Computability of Design* (Kalay, Y. and Majkowski, B. eds.), John Wiley and Sons, New York, New York, 1987.

[Freeman-Benson, et al., 1990] Freeman-Benson, B., Maloney, J. and Borning, A. An Incremental Constraint Solver, *Communications of the ACM* 33, 1 (January, 1990), pp. 54-63.

[Foley and McMath, 1986] Foley, J. and McMath, C. Dynamic Process Visualization, *IEEE Computer Graphics and Application* 6, 3 (March, 1986), pp. 16-25.

[Foley, et al., 1991] Foley, J., Kim, W., Kovacevic, S. and Murray, K. UIDE - An Intelligent User Interface Design Environment, *Architectures for Intelligent Interfaces: Elements and Prototypes* (Sullivan, J. and Tyler, S. eds.), Addison-Wesley, Reading, Massachusetts, 1991, pp. 339-384.

[Gazdar and Mellish, 1989] Gazdar, G. and Mellish, C. *Natural Language Processing in Lisp*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[Gero, 1985] Gero, J. (ed.) *Knowledge Engineering in Computer-Aided Design*, Elsevier Science Publishing Company, 1985.

[Gerstner, 1968] Gerstner, K. *Designing Programmes*, Arthur Niggli Ltd., Teufen, Switzerland, 1968.

[Golin and Reiss, 1989] Golin, E. and Reiss, S. The Specification of Visual Language Syntax, In *IEEE Workshop on Visual Languages*, (October 4-6, Rome, Italy). 1989.

[Gore, 1991] Gore, A. Infrastructure for the Global Village, *Scientific American* 265, 3 (September, 1991), pp. 150-153.

[Grolier, 1992] Grolier Inc. *The New Grolier Multimedia Encyclopedia*, CD-ROM, 1992.

[Gross, et al., 1987] Gross, M., Ervin, S., Anderson, J. and Fleisher, A. Designing with Constraints, *Computability of Design* (Kalay, Y. and Majkowski, B. eds.), John Wiley and Sons, New York, New York, 1987.

[Grosz, et al, 1986] Grosz, B. Jones, K. and Webber, B (eds). *Readings in Natural Language Processing*, Morgan Kaufman Publishers, Inc. Los Altos, California. 1986.

[Haimes, 1994a] Haimes, R. Managing Workflow and content for agile publishing, *Color Publishing*, (January/February, 1994), pp. 24-33.

[Haimes, 1994b] Haimes, R. Document Interface, *interactions,* (October, 1994), pp. 15-18.

[Halbert, 1993] Halbert, D. Programming by Demonstration within the Desktop Metaphor, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 103-123.

[Hearst, 1993] Hearst Corporation, *Popular Mechanics*, December 1993.

[Helm and Marriott, 1986] Helm, R. and Marriott, K. Declarative graphics, In *Proceedings of the Third International Conference on Logic Programming*, 1986, pp. 513-527.

[Helm and Marriott, 1991] Helm, R., and Marriott, K. A Declarative Specification and Semantics for Visual Languages, *Journal of Visual Languages and Computing* 2 (1991), pp. 311-331.

[Henkenius, 1993] Henkenius, Merle. Surface Wiring, *Popular Mechanics* 170, 12 (December, 1993), pp. 63-66.

[Herwijnen, 1994] Herwijnen, E. van. *Practical SGML*, Second Edition, Kluwer Academic Publishers, Boston, Massachusetts, 1994.

[Hiebert, 1992] Hiebert, K. *Graphic Design Processes*, Van Nostrand Reinhold, New York, New York, 1992.

[Hollan, et al., 1984] Hollan, J., Hutchins, E. and Weitzman, L. Steamer: An Interactive Inspectable Simulation-Based Training System, *AI Magazine* 5, 2 (1984), pp. 15-28.

[Holtzman, 1994] Holtzman, S. *Digital Mantras, The Languages of Abstract and Virtual Worlds*. MIT Press, Cambridge, Massachusetts, 1994.

[Hudson and Yeatts, 1991] Hudson, S. and Yeatts, A. Smoothly Integrating Rule-Based Techniques into a Direct Manipulation Interface Builder, In *Symposium on User Interface Software and Technology (UIST'91)*, (November 11-13, Hilton Head, South Carolina). 1991.

[Hurlburt 1978] Hurlburt, A. *The Grid*, Van Nostrand Reinhold Co., New York, 1978.

[Ishizaki, 1993] Ishizaki, S. *Dynamics in Visual Design*. Visible Language Workshop, MIT Media Lab, internal document, 1993.

[Kalay and Majkowski, 1987] Kalay, Y. and Majkowski, B. (eds.) *Computability of Design* , John Wiley and Sons, New York, New York, 1987.

[Kandinsky, 1979] Kandinsky, W. *Point and Line to Plane*, Dover, New York, New York, 1979.

[Kanarick, 1993] Kanarick, C., *AIDE, A Case-Based Approach for Designing Graphics from Locative and Temporal Data*, Master of Science in Visual Studies, Visible Language Workshop, MIT Media Lab, September, 1993.

[Karp and Feiner, 1990] Karp, P. and Feiner, S. Issues in the Automated Generation of Animated Presentations, In *Proceedings of Graphics Interface*, (May 14-18, Halifax, Canada). 1990, pp. 39-48.

[Karsenty, et al., 1992] Karsenty, S., Landay, J. and Weikart, C. Inferring graphical constraints with Rockit, In *Human Computer Interaction: Proceedings of HCI'92*, (September, University of York, U.K.). 1992.

[Kepes, 1961] Kepes, G. *Language of Vision*, Paul Theobald and Co., Chicago, Illinois, 1961, pp. 15-63.

[Kim and Foley, 1993] Kim, W. and Foley, J. Providing High-level Control and Expert Assistance in the User Interface Presentation Design, In *INTERCHI'93 Proceedings*, (April 24-29, Amsterdam, The Netherlands). 1993. pp. 430-437.

[Klee, 1953] Klee, P. *Pedagogical Sketchbook*, Frederick Praeger, New York, New York, 1953.

[Knuth, 1968] Knuth, D. Semantics of context-free languages, *Mathematical Systems Theory* 2 (1968), pp. 127-146.

[Knight, 1989] Knight, T. Color grammars: designing with lines and colors, *Environment and Planning B* 16 (1989) pp. 417-449.

[Kochhar and Friedell, 1990] Kochhar, S. and Friedell, M. User control in cooperative computer-aided design, In *Symposium on User Interface Software and Technology '90*, (October 3-5, Snowbird, Utah). 1990, pp. 143-151.

[Kochhar, et al., 1991] Kochhar, S., Marks, J., and Friedell, M. Interaction Paradigms for Human-Computer Cooperation in Graphical-Object Modeling, in *Proceedings of Graphics Interface '91* (June, Calgary, Canada). 1991, pp. 180-191.

[Koning and Eizenberg, 1981] Koning, H., and Eizenberg, J. The Language of the Prairie: Frank Lloyd Wright's Prairie Houses, *Environment and Planning B* 8 (1981), pp. 295-323.

[Kosslyn, 1989] Kosslyn, S. Understanding Charts and Graphs, *Applied Cognitive Psychology* 3 (1989), pp. 185-226.

[Kosslyn, 1994] Kosslyn, S. *Elements of Graph Design*, W.H. Freeman and Company, New York, New York, 1994.

[Kurlander, 1993] Kurlander, D. Example-Based Graphic Editing in CHIMERA, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 271-290.

[Kurlander and Feiner, 1993] Kurlander, D. and Feiner, S. A History of Editable Graphical Histories, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 405-413.

[Lakin, 1986] Lakin, F. Spatial Parsing for Visual Languages, *Visual Languages* (Chang, S., Ichikawa, T. and Ligomenides, P. eds.), New York, New York, 1986, pp 35-85.

[Leler, 1988] Leler,W. *Constraint Programming Languages*, Addison-Wesley, 1988.

[Lieberman, 1993a] Lieberman, H. Tinker: A Programming by Demonstration System for Beginning Programmers, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 49-64.

[Lieberman, 1993b] Lieberman, H. Mondrian: A Teachable Graphical Editor, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 341-358.

[Lupton and Miller, 1993] Lupton, E. and Miller, J. *The ABC's of▲■●: The Bauhaus and Design Theory*, The Cooper Union for the Advancement of Science and Art, New York, New York, 1993.

[Mackinlay, 1986] Mackinlay, J. Automating the design of graphical presentations of relational information, *ACM Transactions on Graphics* 5, 2 (1986). pp. 110-141.

[MacNeil, 1989a] MacNeil, R. TYRO, a Constraint-Based Graphic Designer's Assistant, In *Proceedings of the IEEE Workshop on Visual Languages* (1989).

[MacNeil, 1990b] MacNeil, R. Adaptive Perspectives: Case-Based Reasoning with TYRO, The Graphic Designer's Apprentice, *IEEE Workshop on Visual Languages*, (October 4-6, Skokie, Illinois). 1990, pp. 138-142.

[Makela, L. and Lupton E., 1994] Makela, L.H. and Lupton, E. Underground Matriarchy, *Eye Magazine*, Autumn, 1994.

[Maloney, 1991] Maloney, J. *Using constraints for user Interface Construction*, PhD Thesis, University of Washington, Technical Report 91-08-12, Seattle, Wa., 1991.

[Marks, 1990] Marks, J., A syntax and semantics for network diagrams, In *Proceedings of the IEEE Workshop on Visual Languages* (October 4-6, Skokie, Illinois). 1990, pp. 104-110.

[Marks, 1991] Marks, J. A Formal Specification Scheme for Network Diagrams that Facilitates Automated Design, *Journal of Visual Languages and Computing* 2 (1991), Academic Press Ltd., pp. 395-414.

[Maulsby, 1993a] Maulsby, D. and Witten, I. Metamouse: A programming by example system and its experimental evaluation, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 155-181

[Maulsby, 1993b] Maulsby, D. The Turvy Experience: Simulating an Instructible Interface, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 239-269.

[Maybury, 1993] Maybury, M. (ed.) *Intelligent Multimedia Interfaces*, AAAI Press and MIT Press, Cambridge, Massachusetts, 1993.

[McCloud, 1993] McCloud, S. *Understanding Comics: The Invisible Art*, Tundra Publishing Ltd., Northhampton, Massachusetts, 1993.

[Michalski, J, 1994] Michalski, J. Smart Page and Screen Layout, *Release 1.0*, 11-94, November 22, 1994, pp. 1-16.

[Mitchell, et al., 1976] Mitchell, W., Steadman, J. and Liggett, R. Synthesis and optimization of small rectangular floor plans, *Environment and Planning B: Planning and Design* 3, 1 (1976), pp. 37-70.

[Mitchell, 1990] Mitchell, W. *The Logic of Architecture*, MIT Press, Cambridge, Massachusetts, 1990.

[Moholy-Nagy, 1947] Moholy-Nagy, L. *Vision in Motion*, Paul Theobald & Co., Chicago, Illinois, 1947.

[Myers and Buxton, 1986] Myers, B., Buxton, W., Creating Highly-Interactive and Graphical User Interfaces by Demonstration, In *Siggraph Proceedings* 20, 4 (August 18-22, Dallas, Texas). 1986, pp. 249-258.

[Myers, 1993] Myers, B. Peridot: Creating User Interfaces by Demonstration, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 125-153.

[NCSA, 1993] NCSA. *NCSA Internet Manuscript* from mosaic@ncsa.uiuc.edu, 1993.

[NIF, 1994] News in the Future Consortium, MIT Media Lab, 1994.

[Nelson, 1985] Nelson, G. Juno, a constraint-based graphics system, In *Siggraph Proceedings* 19, 3 (July 22-26, San Francisco, California). 1985, pp. 235-243.

[Norman, 1991] Norman, D. Cognitive Artifacts, *Designing Interaction*, (Carroll, J. ed.), Cambridge University Press, Cambridge, England, 1991, pp. 17-38.

[Norvig, 1992] Norvig, P. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*, Morgan Kaufmann Publishers, San Mateo, California. 1992.

[Oxman and Gero, 1987] Oxman, R. and Gero, J. Using an expert system for design diagnosis and design synthesis, *Expert Systems* 4, 1 (February, 1987), pp. 4-15.

[Pavlidis and Van Wyk, 1985] Pavlidis, T. and Van Wyk, C. An automatic beautifier for drawings and illustrations, In *Siggraph Proceedings* 19, 3 (July 22-26, San Francisco, California). 1985, pp. 225-234.

[Pereria and Warren, 1980] Pereria, F. and Warren, D. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks, *Artificial Intelligence* 13 (1980), pp. 231-278.

[Rosenberg, 1988] Rosenberg, J. Geographical Data Structures, *IEEE Transactions on CAD*, CAD-4, 1 (January 1988).

[Rosenberg, et al., 1992] Rosenberg, J., Kraut, R., Gomez, L., Buzzard, C., Multimedia Communications for Users, *IEEE Communications Magazine*, May 1992.

[Rosenfeld, 1990]	Rosenfeld, A. Array, Tree, and Graph Grammars, *Syntactic and Structural Pattern Recognition: Theory and Applications* (Bunke, H., and Sanfeliu, A., eds.), World Scientific, Singapore, 1990.
[Rosenstein and Weitzman, 1990]	Rosenstein, M. and Weitzman, L. Icon Editor: The Specification of Graphic Behavior without Coding, In *Proceedings of the 23rd Annual Hawaii International Conference on Systems Sciences* (January 2-5, Kailua-Kona, Hawaii.). 1990, pp. 523-530.
[Roth and Mattis, 1990]	Roth, S. and Mattis, J. Data Characterization for Intelligent Graphics Presentation, In *Proceedings of the Conference on Human Factors in Computing Systems*, (April 1-5, Seattle, Washington). 1990, pp193-200.
[de Saussure, 1956]	de Saussure, F., *Course in General Linguistics*, McGraw Hill Book Company, New York, New York, 1956.
[Schmitt, 1987]	Schmitt, G. Expert Systems in Design Abstraction and Evaluation, *Computability of Design* (Kalay, Y. and Majkowski, B. eds.), John Wiley and Sons, New York, New York, 1987.
[Seligmann and Feiner, 1991]	Seligmann, D. and Feiner, S. Automated Generation of Intent-Based 3D Illustrations, In *Siggraph Proceedings* (July 28-Aug. 2, Las Vegas, NV.). 1991, pp. 123-132.
[Senay and Ignatius, 1991]	Senay, H. and Ignatius, E. Compositional Analysis and Synthesis of Scientific Data Visualization Techniques, Scientific Visualization of Physical Phenomena, In *Proceedings of Computer Graphics International '91*, (Tokyo, Japan). 1991, pp. 269-281.
[Shapiro and Geller, 1987]	Shapiro, S. and Geller, J. Artificial Intelligence and Automated Design, *Computability of Design* (Kalay, Y. and Majkowski, B. eds.), John Wiley and Sons, New York, New York, 1987.
[Simon, 1981]	Simon, H. The Science of Design in *The Sciences of the Artificial*, 2nd Edition, MIT Press, Cambridge, Massachusetts, 1981.
[Sistare, 1991]	Sistare, S. Graphical interaction techniques in constraint-based geometric modeling, In *Proceedings of Graphics Interface '91*, (June 3-7, Calgary, Canada). 1991.
[Smith and Alexander, 1988]	Smith, D. and Alexander, R. *Fumbling the Future*, Quill William Morrow, New York, New York, 1988.

[Smith, 1993]	Smith, D. Pygmalion: An Executable Electronic Blackboard, *Watch What I Do: Programming by Demonstration*, (Cypher, A. ed.), MIT Press, Cambridge, Massachusetts 1993, pp. 20-47.
[Source, Inc.]	Source, Inc. Images of advertising, Chicago, Illinois.
[Stallman, 1993]	Stallman, R., *GNU Emancs Manual*, Eighth Edition, Version 19, Free Software Foundation, June, 1993.
[Stiny and Mitchell, 1978]	Stiny, G. and Mitchell, W. The Palladian grammar, *Environment and Planning B* 5 (1978), pp. 5-18.
[Stiny, 1980]	Stiny, G. Introduction to Shape and Shape Grammars, *Environment and Planning B* 7 (1980), pp. 343-351.
[Stiny and Mitchell, 1980]	Stiny, G., and Mitchell, W., The grammar of paradise: on the generation of Mughul gardens, *Environment and Planning B* 7 (1980), pp. 209-226.
[Thomas and Johnston, 1981]	Thomas, F. and Johnston, O. *Disney Animation, The Illusion of Life*, Walt Disney Productions, Burbank, California, 1981.
[Tufte, 1983]	Tufte, E. *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, 1983.
[Tufte, 1990]	Tufte, E. *Envisioning Information*, Graphics Press, Cheshire, Connecticut, 1990.
[Tufte, 1992]	Tufte, E. *When Design Makes a Difference*, presentation at the Media Lab, MIT, 1992.
[Turansky, 1993]	Turansky, A. *Capturing Graphic Design Knowledge from Interactive User Demonstrations*, Master of Science in Media Arts and Science, Visible Language Workshop, MIT Media Lab, September, 1993.
[Wiecha, et al., 1990]	Wiecha, C., Bennett, W., Boies, S., Gould, J. and Greene, S. ITS: A tool for rapidly developing interactive applications, *ACM Transactions on Information Systems* 8, 3 (July, 1990), pp. 204-236.
[Weiser, 1991]	Weiser, M. The Computer for the 21st Century, *Scientific American* 265, 3 (September, 1991), pp. 94-104.

[Weitzman, 1992] Weitzman, L. Designer: A knowledge-based graphic design assistant, *Artificial Intelligence in Engineering Design*, (Tong, C. and Sriram, D. eds.), Academic Press, Inc., New York, New York, 1992, pp. 433-463.

[Weitzman and Wittenburg, 1993] Weitzman, L. and Wittenburg, K. Relational Grammars for Interactive Design, In *IEEE Symposium on Visual Languages* (August 24-27, Bergen, Norway). 1993, pp. 4-11.

[Weitzman and Wittenburg, 1994] Weitzman, L. and Wittenburg, K. Automatic Presentation of Multimedia Documents Using Relational Grammars, *ACM Multimedia'94*, (October 15-20, San Francisco, California). 1994, pp. 443-451.

[Weizenbaum, 1966] Weizenbaum, J. ELIZA–A Computer Program for the Study of Natural Communication between Man and Machine, *Communications of the ACM* 9, 1 (January, 1966), pp. 36-44.

[Winograd, 1972] Winograd, T. *Understanding Natural Language*, Academic Press, New York, New York, 1972.

[Winston, 1984] Winston, P. *Artificial Intelligence*, Second Edition. Addison-Wesley Publishing Company, 1984.

[Wired, 1993] Wired Ventures, Ltd., *Wired Magazine* 1.6 (December 1993).

[Wired, 1994] Wired Ventures, Ltd., *Wired Magazine* 2.01 (January 1994).

[Wittenburg and Weitzman, 1990] Wittenburg, K. and Weitzman, L. Visual Grammars and Incremental Parsing for Interface Languages, In *IEEE Workshop on Visual Languages* (October 4-6, Skokie, Illinois).1990, pp. 111-118.

[Wittenburg, et al., 1991] Wittenburg, K., Weitzman, L. and Talley, J. Unification-Based Grammars and Tabular Parsing for Graphical Languages, *Journal of Visual Languages and Computing* 2 (1991), pp. 347-370.

[Wittenburg, 1992] Wittenburg, K. Earley-style Parsing for Relational *Grammars,* In *Proceedings of IEEE Workshop on Visual Languages* (September 15-18, Seattle, Washington). 1992, pp. 192-199.

[Wittenburg, 1993]	Wittenburg, K. Adventures in Multidimensional Parsing: Cycles and Disorders. In *Proceedings of the Third International Workshop on Parsing Technology* (August, Tilburg, Netherlands and Durbuy, Belgium). Association for Computational Linguistics and Tilburg University, 1993, pp. 333-348.
[Wurman, 1990]	Wurman, R., *Information Anxiety*, Bantam Books, New York, 1990.
[Zipf, 1935]	Zipf, G., *The Psycho-biology of Language*, Houghton-Mifflin, Boston, Massachusetts 1935.

Index

10

4003