VISUAL LANGUAGE THEORY: TOWARDS A HUMAN-COMPUTER INTERACTION PERSPECTIVE [1]

N. Hari Narayanan[2]
Roland Hübscher

Technical Report CSE97-05

September 30, 1997

Visual Information, Intelligence & Interaction Research Group
Department of Computer Science & Engineering
Auburn University
Alabama 36849-5347
USA
http://www.eng.auburn.edu/department/cse/research/vi3rg/vi3rg.html

[1] A revised version of this technical report appears as Chapter 3 (pp. 85-127) in K. Marriott & B. Meyer (Eds.), *Visual Language Theory*, Springer-Verlag Publishers, Berlin, 1997.
[2] Author to whom correspondence should be addressed. Email: narayan@eng.auburn.edu

VISUAL LANGUAGE THEORY: TOWARDS A HUMAN-COMPUTER INTERACTION PERSPECTIVE

N. Hari Narayanan

Roland Hübscher

Abstract

The main reason for using visual languages is that they are often far more convenient to the user than traditional textual languages. Therefore, visual languages intended for use by both computers and humans ought to be designed and analyzed not only from the perspective of computational resource requirements, but equally importantly, also from the perspective of languages that are cognitively usable and useful. Theoretical and practical research on visual languages need to take into account the full context of a coupled human-computer system in which the visual language facilitates interactions between the computational and the cognitive parts. This implies that theoretical analyses ought to address issues of comprehension, reasoning and interaction in the cognitive realm as well as issues of visual program parsing, execution and feedback in the computational realm. The human aspect is crucial to visual languages, and therefore we advocate a correspondingly broadened scope of inquiry for visual language research. In this chapter we describe aspects of human use of visual languages that ought to be important considerations in visual language research and design, and summarize research from related fields such as software visualization and diagrammatic reasoning that addresses these issues. A framework consistent with the broadened scope of visual language research is proposed and used to categorize and discuss several formalizations and implemented systems. In the course of showing how a sample of current work fits into this framework, open issues and fruitful directions for future research are also identified.

I. Why Visual Languages?

In the various disciplines of human inquiry, be it science, arts or humanities, the answer to this question is obvious. Our visual systems have evolved over eons to become fine tuned in being able to perceive patterns, sometimes even where none exist, and our brains can make sense of them through recognition and interpretation. Conversely, our brains are adept at converting thoughts into visual representations, some of which are constrained by conventions, while others are brilliantly executed depictions governed by nothing other than the imaginations of an artist. In fact, visual thinking (Arnheim, 1969) is deeply embedded in the psyche of every culture, and visual languages predate the development of textual ones (Tversky, 1995).

The ease with which we create and comprehend visual representations is surely a significant factor motivating research in the area of visual languages, and this is articulated in many different ways (Blackwell, 1996). Most of this research has been on visual programming languages in the past. However, the adeptness of the human visual system masks the computational complexity of parsing, interpreting, and operationalizing the semantics carried by visual representations - from the simplest of diagrams to complex real world scenes. Computers are not yet capable of vision at the human scale. Neither are these machines capable of generating visual representations that depict abstract thoughts and ideas except under the direction of a human (McCorduck, 1991). Chapters of this book demonstrate the difficulty of formalizing visual languages so that their computational properties can be analyzed and corresponding parsers, interpreters, or compilers can be built with the assurance that they will run in reasonable time and consume reasonable resources.

So why visual languages usable by computers and humans? After all, computers can deal very well with textual languages generated by grammars with attractive formal properties. Programming languages is a vibrant research area, and it seems that new programming languages sprout, become fashionable and fade away, only to be replaced with newer and fancier ones at regular intervals. It is our belief that the primary reasons for visual language research are the benefits such languages can provide to the human user by enabling humans and computers to communicate and interact through visual representations depicted on the computer screen. Some have even suggested that an important goal of most visual programming languages is that these provide a medium for human-human communication; the fact that a program can be executed on a computer is "just" a nice side effect (Abelson, Sussman & Sussman, 1996).

While cognitive issues are equally relevant to conventional programming languages since the programming task is carried out by humans, design and analysis of programming languages have always been driven by issues of tractability and efficiency, not usability or comprehensibility. One of the exceptions is COBOL, but the result speaks for itself. Cognitive properties of a language have typically been investigated only after its design and widespread use. This has slightly changed with scripting languages, once end users were targeted as the new group of programmers. But programming languages are primarily intended for computers, whereas visual languages are primarily intended for humans. So the central role of visual languages usable by both humans and computers may turn out to be as tools for communication and interaction. This implies that visual language theory requires a broader and interdisciplinary basis, compared to programming language theory. As Mahling and Fisher (1990) state: "Only by acknowledging the cognitive reality in which the language is employed by programmers and end users, can we recognize that visual programming does introduce a new paradigm."

However, despite the pervasiveness of the cognitive benefit assumption in the visual languages community (Blackwell, 1996), there is a lack of research to back it up. On the other side, there is literature that question the validity of the belief that graphical syntax alone will increase comprehensibility or programmability (Badre & Allen, 1989; Green & Petre, 1992; Green, Petre & Bellamy, 1992; Petre, 1995). A visual language whose graphical syntax is as incomprehensible as that of a terse textual language can provide no justification for the additional complexity that such a language introduces on the computational side. In other words, since visual languages in general are more difficult to parse and interpret, these need to demonstrate a concomitant payoff on the cognitive side. Without theoretical and practical research on this issue, executable visual languages may remain confined to narrow application areas where the use of graphics "obviously" makes sense. This leads us to argue that both visual language theory and design must progress in the human-computer interaction context; neither can afford to confine itself to the computational or cognitive aspects alone. It should be noted that research in visual languages is currently broadening beyond visual programming languages, and so it is indeed an opportune time for visual language theory researchers to address issues relating to the cognitive utility of visual languages as well as their computational properties.

II. Which Visual Languages?

When we look at the main reason for visual languages - enabling visual communication and interaction - it is evident that from the perspective of human-computer interaction visual languages encompass a broad class of entities that have been subjects of research in a variety of research communities with much in common. Two examples are the interdisciplinary areas of software visualization and diagrammatic reasoning. Research in software visualization and algorithm animation strives to define visual languages that computers can generate, in order to depict the static structure and dynamic semantics of programs or algorithms in a meaningful way.

Researchers in diagrammatic reasoning have looked at formalizing the language of diagrams, characterizing how humans represent and reason with diagrams, and programming computers to understand and reason with diagrams.

A variety of related terms, all containing the root word “visual” appear in the literature with subtly different meanings. Many authors (for example, Price, Baecker & Small, 1993 and Petre, 1995) rightly observe that it is not correct to restrict terms such as “visual” or “visualization” to visible images only. These should instead mean “facilitating the formation of a mental picture”. For the purposes of discussion in this chapter, we will constrain “visual” to mean having to do with a representation, internal or external, that conveys information by virtue of its topological, geometric or other visual properties that humans can see, and explicitly exclude purely textual representations. Since we will be using similar terms in this chapter, a few short definitions that explicate their meanings are in order.

Visual languages: Languages with alphabets consisting of visual representations and used for human-human or human-computer communication.

Visual programming: The use of visual representations to communicate data and operations to a computer. The underlying programming language may or may not be visual.

Visual programming language: A programming language with an alphabet made up of visual representations.

Software visualization: The generation and display of visual representations to convey static and dynamic aspects of software including data structures, code and algorithms to humans.

Algorithm animation: The generation and display of animations in which dynamic visual representations convey to humans how algorithms operate on data.

Diagrammatic representations: Visual representations that encode and convey information about their referents without being true analogs of the entities being represented.

Diagrammatic reasoning: The processes of comprehending and making inferences from diagrammatic representations.

In this chapter, we use the term “visual languages” to include programming languages whose syntax is based on visual representations (as in visual programming), computer visual languages designed to convey aspects of an underlying computation or its declarative specification (as in software visualization and algorithm animation), and human visual languages that seem amenable to formalization and computer implementation (as in diagrammatic representation and reasoning).

III. Scope of Visual Language Research

This book is about visual language theory viewed from multiple perspectives. This expansion of scope is both necessary and timely. It is necessary because of the human dimension. We believe that the central issue in visual language theory is the design and characterization of computationally tractable and cognitively effective visual languages. Tools of logic and language theory can be applied to characterize computational properties of visual languages. But how can one measure cognitive effectiveness? It is here that expanding the scope of inquiry to include work in other areas can help visual language research, since cognitive questions about the design of effective visual representations have indeed been raised, if not yet answered completely, by researchers in other areas. There is much work in these areas that is of potential use to visual language research.

The expansion of scope is also timely. An informal survey of papers related to visual language theory appearing in the previous five volumes of the Proceedings of the IEEE Symposia on Visual Languages indicates an emphasis on logic and computation instead of a balance between computation and cognition. Cognitive questions have generally been ignored, with the implicit assumption that others (e.g., perhaps researchers in areas such as psychology of programming) will address those. The result is that it is not clear whether the investment of the visual language community as a whole in designing and characterizing a multitude of visual programming languages over the last decade has indeed made the area progress toward one of its major goals of liberating programmers and end users from the shackles of tedious textual programming, and providing them instead with intuitive and useful visual tools for programming. This perceived lack of progress led one prominent researcher to conclude, prematurely in our opinion, that "nothing even convincing, much less exciting, has yet emerged from such efforts [on visual programming]...I am persuaded that nothing will" (Brooks, 1987). Nine years after this criticism, a more optimistic note is struck by Green and Petre (1996): "Overall, we believe that in many respects visual programming languages offer substantial gains over conventional textual languages, but at present their human-computer interaction aspects are still under-developed."

Visual languages abound in human endeavors and enjoy prominence comparable to that of non-visual languages. One can indeed find many visual languages - perhaps not formally defined in terms of a grammar, but languages nevertheless - that have been immensely successful in facilitating human-human and human-computer communication. Many research areas have developed their own visual languages, for instance, mathematicians use commutative diagrams, physicists use Feinman diagrams, and computer scientists describe data structures with boxes and arrows. The desktop metaphor is a highly successful visual language for interacting with an operating system. Once familiar with such a language, it becomes an extremely efficient tool for communication and reasoning. While visual languages are extremely useful for people for certain tasks, and to some degree are also used by computers, it is important to note that these do not replace textual or propositional languages. Not even comic strips get away completely without text. There is no need to choose either visual or textual languages because they are the two extremes of a spectrum spanning from text to illustrated text, to annotated pictures, and to purely visual representations.

Most current literature on visual language theory is about those visual languages which are visualizations of originally textual languages. Logic programming languages (Puigsegur, Agusti & Robertson, 1996) and finite state automata (Dinesh & Üsküdarli, 1997) are some examples. Although formalizing such languages is useful and their visual forms are sometimes advantageous over the textual ones, such languages may not help us understand those properties that make human visual languages so powerful. In this chapter, we will look at visual languages from a human-computer interaction perspective. We will restrict ourselves to visual languages that are interesting in the context of computers, and will therefore ignore a large class of visual languages, especially those originating from the visual arts (Gombrich, 1968; Arnheim, 1969; Goodman, 1976). These languages are cognitively inspired and are not necessarily the result of visualizing a formal language. We will discuss some of their characteristics to argue that current theories of visual languages need to be extended to include a larger, more interesting class of languages. One aim of this chapter is to provide a sample of ideas from other related areas that we believe will be useful in furthering visual language theory by encompassing both computational and cognitive aspects. Another goal is to sketch the beginnings of a corresponding integrated framework.

IV. Cognitive Utility of Visual Languages

Visual languages are interesting because they are used by people. Thus, to understand these languages better, we need to take a brief look at a few commonplace ways in which people and computers utilize them for communication, interaction, and reasoning.

IV.1 Direct Manipulation

Direct manipulation allows users to execute actions by directly interacting with visually displayed objects instead of having to describe the action. For instance, marking a file for deletion in a graphical user interface can often be done by dragging the file's icon into an icon depicting a trash can. This executes the command `(move <file> trash-folder)`. With direct manipulation users can “depict” the operations they desire; without it they would have to describe the command to an interpreter which then translates the command into the actual action (Frank & Foley, 1994). The cognitive benefits of direct manipulation arise from the elimination of the need to move between two vastly different representations: the natural visual representation (e.g., metaphorical desktop) which permits interactive gestures and the textual program that specifies the objects and behaviors visible on the screen (Hübscher, 1996). Thus, graphical user interfaces for direct manipulation may be viewed as visual languages that humans use to interact with computers.

IV.2 Visualization of Information

Visual representations are often used to make higher order relations more accessible to the human intuition (Tufte, 1990). A graph of a function makes its characteristics much more explicit than a table of values of the same function even though informationally both are equivalent. Printed textual descriptions are frequently illustrated with pictures that serve to exemplify the ideas contained in the text, to provide different representations of the same information, or to complement what the text describes. In these cases, visual views of descriptive representations allow the viewer to discover relations and characteristics that are hidden in the textual representation. This is particularly true of complex hypermedia information structures such as the World Wide Web for which various graphical visualizations (e.g., Cone Trees, Robertson, Card & Mackinlay, 1993) have been developed. Thus, information visualization is an area concerned with developing visual languages for communicating information structures to humans and allowing them to interact with these structures through direct manipulation.

IV.3 Visualization of Software

Visualizing static and dynamic characteristics of programs can facilitate comprehension and debugging for the user (Price, Baecker & Small, 1993). Data structures and program control flows are often easier to understand when visualized using graphs and flowcharts. The hierarchical file structure of an operating system is easily visualized either with a directory tree or folders. Three dimensional spatial arrangement of nested transparent cubes have been proposed recently (Freeman, Gelernter & Jagannathan, 1995) as a means of visualizing program structure and execution. Program visualization is essentially the inverse of parsing a visual programming language. It generates a visual language that depicts the same information as the syntactic and semantic specifications of a program, but in a way that facilitates the human observer’s comprehension.

IV.4 Diagrammatic Representation and Reasoning

Most of current work in visual language theory can be characterized as specifying the syntax and semantics of a visual language using propositional descriptions. But one reason visual formalisms are so pervasive in human enterprise is because these facilitate easy and compact representations of information that is cumbersome to adequately describe using propositional descriptions. Many visual relations and properties are implicit in propositional representations and require extensive reasoning to be deduced, whereas visual representations display them explicitly at all times (Larkin & Simon, 1987). Visual representations such as diagrams facilitate situated reasoning because these make serendipitous inferences, inferences by recognition, prediction by mental visualization, and cueing prior knowledge possible (Narayanan, Suwa & Motoda, 1994a). This makes it easy for humans to make inferences from pictorial displays (Larkin, 1989). Furthermore, since visual representations typically represent spatial and geometric properties of their referents in an isomorphic way, modifications of these properties of the constituents of such a representation can be used to reason about the effects of real world processes in which the represented objects participate (Narayanan, Suwa & Motoda, 1995b). This makes such representations especially suitable for reasoning by mental simulation. Most propositional descriptions tend to loose these advantages. Therefore, one kind of visual representation - diagrammatic representation - and its role in reasoning have been the subject of recent research interest (Narayanan, 1992; Glasgow, Narayanan & Chandrasekaran, 1995). Diagrammatic representations are different from propositional representations because visual properties of the constituent units of the representation contribute to the semantics of a diagrammatic representation as much as the semantics of the constituent units themselves. Therefore, researchers in this area have developed formal characterizations of diagrammatic representations using a heterogeneous logic that exploits propositional and diagrammatic representations (Barwise & Etchemendy, 1995; Hammer, 1995; Shin, 1994).

IV.5 Graphical Simulations

The notion of diagrammatic reasoning is combined with direct manipulation in interactive graphical simulations. These have become an important tool in educational research (CACM, 1996). Many user friendly tools for creating such simulations are becoming available. Some examples are KidSim^{TM} from Apple Computer (Smith, Cypher & Spohrer, 1994), Star Logo from MIT Media Laboratory (Resnick, 1996) and Agentsheets from the University of Colorado (Repenning & Sumner, 1995). Such simulations employ direct manipulation techniques to specify the model to be simulated and its initial conditions, and produce visualizations of time-varying processes (e.g., predator-prey relationships in an ecosystem). Users can directly manipulate objects in the simulation without having to access their internal representations. Dynamic visual representations such as animations are especially well suited to display temporal information because students are quite used to observing sequences of moving pictures. The cognitive benefits of such interactive graphical simulations arise from the elimination of the need to move between two vastly different representations: the natural dynamic visual representations of processes being simulated and the underlying descriptive specifications of them (Hübscher, 1995).

V. A Framework for Analysis and Synthesis of Visual Languages

In this section we propose a framework, which consists of a model of visual language use and a corresponding taxonomy of issues, that is expected to be useful for the discourse on design of new visual languages as well as characterizations and comparisons of existing ones. Our aim is to propose a preliminary taxonomy with the breadth that we believe visual language research requires, and open it to further discussion and extension. In fact, the most important feature of any taxonomy is that it will be extended as research progresses. A well founded taxonomy can serve at least two purposes. It can provide a concise picture of the central issues of a field by facilitating the characterization of existing work, and spur additional research by pointing out areas that current research has not addressed well. A taxonomy needs to be derived in a principled way, so that later modifications do not require a complete reordering. This section describes the systematic approach we took in deriving it, expands on each element, and discusses a sample of pertaining literature. It should be noted that our purpose is to illuminate the distinctions rather than to present a complete survey. Open issues are pointed out as fruitful avenues for future research in the field.

Consider visual languages usable by both computers and humans. There are three objects of interest to any theoretical or practical investigation of such languages. A *computational system*, a *cognitive system*, and the *visual language* itself. The language may be specified formally by a grammar or other means. It is embodied by the visual representations used for communication. Visual representations that encode and convey information appear on the interface, the visual display. Communication requires comprehension, inference and feedback. However, differences in architecture make these processes on both sides of the display quite different from each other. On the computational side, this implies processes such as visual parsing, interpretation or compilation, and program execution. On the cognitive side, this implies processes of visual perception, comprehension, and reasoning with the information. Furthermore, both systems construct and manipulate visual representations on the display to convey results of these processes to the other. The success of a visual language in this model rests on two criteria: its *computational tractability* and *cognitive effectiveness*. This simple but complete model of the use of visual languages for human-computer interaction, illustrated in Figure 1, provides a conceptual basis for defining the top level structure of the taxonomy of issues we propose.

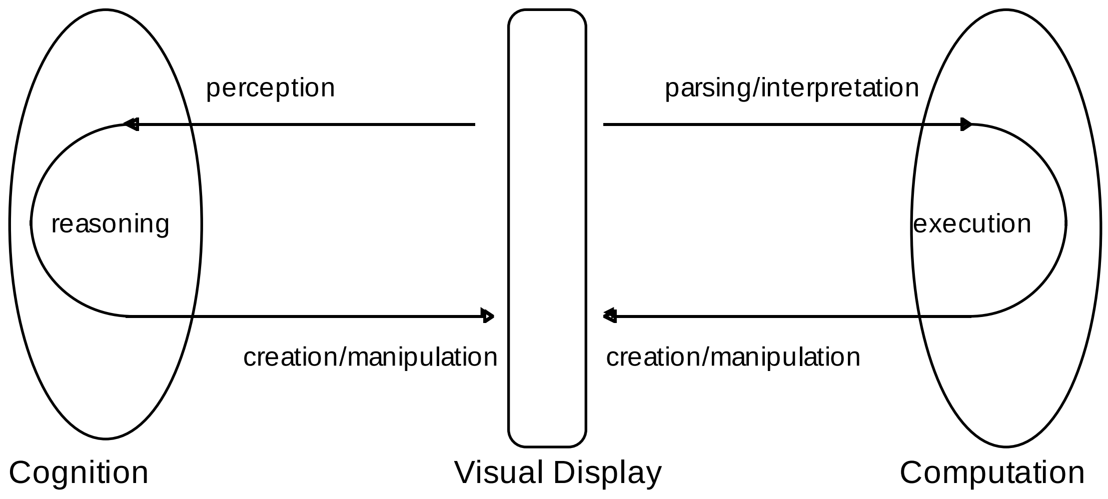

Figure 1. Model of Visual Language

The taxonomy consists of three major categories - *representation of information*, *cycle of interaction* and *evaluation* - and a number of minor categories into which the major ones can be subdivided (see Figure 2). Below we present each category in more detail with some additional subdivisions. This is a useful starting point for providing a shared vocabulary for discussions and relative comparisons of work in visual languages. There is the additional benefit of revealing areas that are sparsely covered indicating potentially fruitful avenues for future research. Clearly, this is only a beginning; a more complete taxonomy and a more formal system for characterizing and comparing research efforts using the taxonomy is required, which is an excellent avenue for future research.

1. Representation of Information
    - 1.1 Application Domain
    - 1.2 Static Syntax
    - 1.3 Static Semantics
    - 1.4 Dynamic Syntax
    - 1.5 Dynamic Semantics

2. Usage: Cycle of Interaction
    - 2.1 Granularity
    - 2.2 Visual Communication
    - 2.3 Computational Aspects
    - 2.4 Cognitive Aspects

3. Evaluation
    - 3.1 Computational
    - 3.2 Cognitive

Figure 2. Top Levels of a Taxonomy for Visual Language Research

This is not the only taxonomy that has been proposed. There have been many attempts at classification, but these prior approaches generally emphasize only a few measures. Myers (1986) describes an operational approach to classification based on two programming styles (programming by example and visual programming), three kinds of visualizations (program visualization, visualizing static data and visualizing dynamic data) and two interaction styles (batch mode and interactive). Shu (1986) uses three dimensions - level of the language, its scope or generality, and the extent of meaningful visual expressions it utilizes - to characterize a number of languages. Menzies (1995) uses the dimensions of purpose, visual expressions, and design of the visual programming system to assess the languages. Chang (1987) categorizes visual languages in terms of the type of activity they support - visual interaction, visual programming, visual information processing or iconic visual information processing. Selker and Koved (1988) present a taxonomy of issues for analyzing and designing visual languages, with five major categories: visual alphabet, visual syntax, interaction, structure, and coverage. In a more recent and rigorous approach, Marriott and Meyer (1997) use the measures of expressiveness and parsing complexity to classify visual language grammars. Our hope is that the past and present efforts will stimulate additional research along these lines, resulting in the development of other taxonomies, revisions of existing ones, or the integration of multiple taxonomies. All of these can contribute to not only developing a better understanding of where we are in terms of visual language theory and practice, but also where we ought to be going.

1. Representation of Information

The first top level category, representation of information (see Figure 3), describes the nature of representations that the language utilizes - characterizing the contents of the visual display in Figure 1. The elements of a visual language are visual representations, sometimes called visual sentences. In general, visual representations need to convey dynamic or time-varying information such as how data changes as a program executes, as well as static information. Three central issues of information representation are *what* is to be represented, *how* to represent it, and how to *associate* the representation with the represented.

1. Representation of Information

    1.1 Application Domain
        1.1.1 Type: Discrete, Continuous, Generality, etc.
        1.1.2 States: Objects, Attributes, Relations
        1.1.3 State Transitions: Objects, Attributes, Relations

    1.2 Static Syntax
        1.2.1 Visual Sentences: Primitives, Dimensions, Relations
        1.2.2 Primitive Type: Graphical, Mixed
            1.2.2.1 Mixed: Textual Primary and Graphical Secondary Notations
            1.2.2.2 Mixed: Graphical Primary and Textual Secondary Notations

    1.3 Static Semantics
        1.3.1 Mapping: Domain, Range, Type
        1.3.2 Specification

    1.4 Dynamic Syntax
        1.4.1 Visual Sentence Transformations: Discrete, Continuous

    1.5 Dynamic Semantics
        1.5.1 Mapping: Domain, Range, Type
        1.5.2 Specification

Figure 3. Subcategories of Information Representation

1.1 Application Domain

The key aspect of any representation is what is being represented. Every visual language has at least one application domain (AD), its domain of discourse. Such domains typically contain objects O, attributes A and relationships R^n with which any state of the domain can be described, i.e., $S = \{ O, A, R^n\}$. Most ADs of interest are dynamic. That is, besides states there are state transitions of interest in such domains as well. Dynamic processes in the domain result in changes of state, i.e., creation, deletion or modification of objects - ΔO, of attributes - ΔA, and of relationships - ΔR^n, i.e., $\Delta S = \{\Delta O, \Delta A, \Delta R^n\}$. ADs may be discrete or continuous. Elements and dynamics of a discrete domain are characterizable in terms of discrete states and transitions. An example is a file system in which the objects, files, may have several attributes such as author, creation date, access rights etc. and relations. Every state of the file system is describable as a finite set S, and the system changes state through discrete transitions describable by ΔS. A continuous domain, on the other hand, is one in which state changes are fluid, making it difficult to distinguish discrete states and transitions. The ecosystem of a pond is an example. Such systems are semantically dense (Goodman, 1976), and require explicit abstraction and quantization in order to be describable in terms of S and ΔS.

Generality of AD is an important dimension. Some visual languages are tailored to one specific AD. Weather maps used by meteorologists and circuit diagrams that electrical engineers use are examples. Algorithm animation systems such as Balsa (Brown & Sedgewick, 1985) can be used to automatically create and display a visual language that depicts the operation of a single algorithm. Others can represent a class of domains. Pictorial Janus (Kahn & Saraswat, 1990) and Prograph (Steinmann & Carver, 1995) are examples of visual programming languages that can represent a variety of concurrent and dataflow computations respectively. The American Sign Language is a visual language that is very general.

The next issues are how to represent the information (syntax) and how to associate the representation with the represented (semantics). Different visual languages use different graphical alphabets and rules of combination to represent information about entities in the AD (static syntax and semantics), and allow different kinds of transformations on these to convey changing information (dynamic syntax and semantics).

1.2 Static Syntax

A visual language may be viewed as a set of valid visual sentences, i.e., VL = {VS}. The vocabulary V using which visual sentences are constructed consists of a set of visual primitives P, a set D of visual dimensions such as position, orientation, size, shape, color, texture, etc. and a set of visual relations V^n that can exist between two or more graphical primitives. Thus, V = {P, D, V^n}. A visual sentence then consists of a set of primitives drawn from P with each primitive having a set of meaningful visual dimensions drawn from D and a set of n-ary visual relations that exist among these primitives. This is a broad characterization of the static syntax of a visual language. It may be formally specified with a grammar that generates all valid visual sentences in a language or with rules that can be used to make an accept/reject decision on a given visual sentence.

Visual languages may be classified in terms of the visual primitives, dimensions and relations they employ. Primitives used in a language may be purely graphical, mixed (graphical primitives and textual symbols) or purely textual. On one end of this spectrum lie completely visual languages and at the other end are completely textual languages. Languages that use symbols of one kind or the other as secondary notations (Price, Baecker & Small, 1993; Green & Petre, 1996) fall in the middle. Textual languages that also utilize graphical secondary notations - e.g., text that uses color, boldfacing etc. to convey additional information - and graphical languages that utilize textual secondary notations - e.g., map with labels - are two intermediate points of this spectrum. The kind of primitives of the language syntax - purely graphical, graphical with textual secondary notations, or textual with graphical secondary notations - can thus be used to characterize visual languages. Drawings by our cave-dwelling ancestors were purely graphical. Most current human visual languages and visual programming environments use mixed notations. Bitpict (Furnas, 1991;1992) and Cartoonist (Hübscher, 1996) are visual programming languages that use purely graphical notations.

There are a variety of dimensions such as shape, geometry, color, texture, 2D and 3D locations, etc., and a variety of visual relations that can be used to carry meaning. There are a number of studies that have attempted to enumerate and classify visual representations (Lohse *et al*, 1994), dimensions and relations (Bertin, 1967; Goel, 1995; Goodman, 1976). Nevertheless, a systematic categorization and the use of such a categorization to characterize and compare visual languages in terms of their visual syntax have not yet been done. This, we believe, is a very fruitful area for visual language research.

1.3 Static Semantics

Sentences of the visual language represent states of an AD. In other words, any particular state of the world in this domain, describable in terms of a set of objects, attributes and relations, can be mapped to a corresponding visual sentence consisting of graphical primitives, visual dimensions and visual relations. The domain, range and type of this mapping are important. It may be one-to-one, one-to-many, many-to-one or many-to-many. One-to-one $O \leftrightarrow P$, $A \leftrightarrow D$, and $R^n \leftrightarrow V^n$ mappings are common. For example, algorithm animation systems map data items to geometric primitives such as circles, size of data items to areas of the primitives, and relative positions of data items in a data structure to relative spatial locations of the corresponding graphical primitives. Other kinds of mappings may be difficult to comprehend without training unless based on established conventions. Venn diagrams represent a many-to-one mapping since a single circle represents many elements in the AD. Network and tree diagrams use a $R^n \leftrightarrow P$ mapping: the "connected" relation is represented by a line. Different kinds of $S \leftrightarrow VS$ mapping are examined by Gurr (1997), who provides a characterization of visual representations in terms of properties such as lucidity, laconicity, soundness and completeness. Andries et al. (1997) propose graph data structures for representing VS and S, and a "represents/represented-by" relation that maps between the two. Tufte (1983; 1990; 1997) and Engelhardt and colleagues (1996) suggest principles for mapping from different kinds of information (e.g., nominal, categorical, ordinal, quantitative, topological and spatial) to visual primitives, dimensions and relations.

This mapping captures the static semantics of the language. From its formal specification a cognitive or computational system can generate a visual sentence to depict a state in the AD, or interpret a visual sentence in terms of the state in the AD it represents. A number of researchers have addressed the issue of formally specifying the static semantics of a visual language. Meyer (1993) describes a specification scheme that itself uses a mixed vocabulary consisting of textual symbols and partially specified pictures, suitable for human comprehension. Haarslev (1997) presents a theoretical framework that combines topology, spatial relations, and description logics for developing syntactic and semantic specifications of static visual sentences. An advantage of this approach is that it allows the construction of visual language editors that can both verify the soundness of specifications of a visual language developed in this framework and parse visual sentences of the language. Gooday and Cohn (1997) show how spatial logic can be used to describe syntax and procedural semantics of a visual language without having to first translate the visual language into a textual one before specifications can be written. A complete treatment of various approaches can be found in (Marriott, Meyer & Wittenburg, 1997).

One interesting issue to consider is the development of declarative specifications of syntactically and semantically dense (Goodman, 1976) visual languages. This requires deeming certain aspects of visual sentences as irrelevant and discarding them while explicitly describing certain others. To see this, consider developing a set of assertions to describe Mona Lisa. This process results in certain information loss (from a cognitive perspective) since what is discarded as irrelevant by the developer of the specification may very well be interpreted as relevant and meaningful by another. The other side of this coin is that if a computational system has to rely on declarative specifications of semantics, it becomes necessary, during visual language design, to restrict what can be depicted by visual sentences of the language to what is describable by symbols of the specification language. This implies that visual sentences of the language will form a notational system rather than a much richer analog system (Goodman, 1976).

1.4 Dynamic Syntax

The dynamic syntax of a visual language specifies how visual sentences may be transformed. Such transformations may include creating, deleting or modifying graphical primitives - ΔG, changing the attributes of graphical objects - ΔD, or changing the spatial relations between graphical objects - ΔV^n, in a visual sentence.

One important issue of dynamic syntax is the specification of how the changes are to be displayed in the visual display. The visual display update may be discrete (a new visual representation replaces the old one) or continuous (the old representation smoothly transforms to the new one). Animations are examples of the latter. The former is suitable when the visual language is notational and the latter is suitable when the visual language is analog (Raymond, 1991), and the corresponding AD is continuous. In practice, animations are employed even when the corresponding AD is not continuous (e.g., algorithm animations), but whether this contributes to cognitive effectiveness or is merely an irrelevant and potentially distracting feature is yet to be determined.

Rewrite rules that systems like Agentsheets (Repenning & Sumner, 1995) and Bitpict (Furnas, 1991; 1992) utilize are examples of dynamic syntax specifications. Often the dynamic syntax is implicitly specified via graphical transformation procedures, as is the case in computer animations produced by visualization systems and visual feedback in graphical user interfaces.

1.5 Dynamic Semantics

Transformations of visual sentences can represent state changes in the AD. The conditions under which domain state transitions occur and how these affect the objects, their attributes and relations may be mapped to conditions under which a visual sentence can be transformed to another and the nature of this transformation. In other words, ΔO, ΔA and ΔR^n are mapped to ΔP, ΔD and ΔV^n. The domain, range and type of this mapping are important. It may be one-to-one, one-to-many, many-to-one or many-to-many. The dynamic semantics of the language is captured by this mapping. From its formal specification a cognitive or computational system can transform a visual sentence to depict a dynamic process in the AD, or interpret a visual sentence transformation in terms of the corresponding state change in the AD. There is not much research yet on developing specifications of the dynamic semantics of visual languages intended for human comprehension and computer execution. Typically this semantics is implicitly captured in the graphical procedures or rewrite rules employed by the system. Repenning (1995) discusses extending rewrite rules to explicitly capture dynamic semantics.

For continuous ADs, this mapping requires either quantization of the continuous state changes so that a continuous domain can be represented by a discrete visual language, or specifying the dynamic syntax so that the visual display is continuously changed. An example of the former is the time display employed by digital watches that simulate analog dials using liquid crystal displays so that the second hand has only 60 meaningful locations on the dial. The latter option is preferable for cognitive effectiveness because the visual language then becomes both syntactically and semantically dense (Goodman, 1976), faithfully reflecting the continuous nature of the AD. However, this can create computational difficulties. Computational properties of such visual languages are yet to be explored.

While typically ΔO is mapped to ΔP, ΔA is mapped to ΔD, and ΔR^n is mapped to ΔV^n, other mappings are possible. There is no a priori reason for mapping the domain statics or dynamics to static or dynamic visual language syntax. The static syntax of a visual language includes its visual sentences. Its dynamic syntax captures how visual sentences may be transformed. This describes the representations. The represented, the AD, includes both states of the domain in terms of objects, attributes and relations, the static semantics of the domain, and state transitions, the dynamic semantics of the domain. The central issue for a visual language designer then is how to represent the static and dynamic semantics of the AD using the static and dynamic syntax of the visual language. There are four possible mappings: (1) $S \leftrightarrow VS$, (2) $\Delta S \leftrightarrow \Delta VS$, (3) $S \leftrightarrow \Delta VS$, and (4) $\Delta S \leftrightarrow VS$. (1) and (2) are commonly occurring mappings. An example is a visual language representation of a microworld consisting of moving balls and stationary walls (Hübscher, 1996) that maps balls and walls to circles and rectangles on a display, and maps the motions of balls to the corresponding motions of circles on the display. One example of (3) is the use of blinking or flashing graphical objects on the display representing some objects in the AD to which the user's attention needs to be drawn. An example of (4) is the use of arrows, static graphical objects, to denote AD dynamics such as motion.

Each of the above four possibilities include nine possible mappings, since S, ΔS, VS, and ΔVS are describable by three types of entities each. Each of these mapping may in turn be one-to-one, one-to-many, many-to-one or many-to-many, providing a total of 144 possibilities. Thus, the static and dynamic syntax and semantics of visual languages may be characterized in terms of the following:

Application Domain - O, A, R^n, ΔO, ΔA and ΔR^n.

Visual Sentences - P (purely graphical or mixed), D, V^n, ΔP, ΔD and ΔV^n.

Semantic Mappings - one or more of the 144 kinds of mappings that define language semantics. One of these, $S \leftrightarrow VS$: ($O \leftrightarrow P$, $A \leftrightarrow D$, $R^n \leftrightarrow V^n$) and $\Delta S \leftrightarrow \Delta VS$: ($\Delta O \leftrightarrow \Delta P$, $\Delta A \leftrightarrow \Delta D$, $\Delta R^n \leftrightarrow \Delta V^n$) where every map is one-to-one, is called a consistent mapping.

Consider, for example, Hyperproof (Barwise & Etchemendy, 1994) which is a mixed visual language consisting of geometric diagrams and logic sentences. It is an interactive language using which students solve logic problems assisted by the computer. Its AD is configurations of 3D geometric objects in 3-space. Each state in the domain is one such configuration describable by a set of objects. Their relevant attributes are shape and spatial location. Inter-object relationships of interest are in-front-of, left-of, etc. For this language, ΔO is null. Objects' attributes may change however. For instance, the shape attribute may be initially "unknown" but can be determined via inference. Similarly spatial locations can change. Such changes constitute ΔA. Location changes results in inter-object relation changes - ΔR^n. P consists of 3D geometric shapes drawn on a plane on the display and symbols constituting logic sentences displayed on the screen. It uses a mixed vocabulary. D includes shape and location. V^n includes the spatial relations stated above. Both D and V^n may be visually depicted and textually specified. The semantic mapping in Hyperproof is consistent.

Just as Hyperproof is an implementation of the notion of a heterogeneous logic of symbols and pictures, Erwig and Meyer (1995) describe the design of a programming environment that implements the notion of heterogeneous programming that integrates visual and textual aspects. A heterogeneous visual programming language (HVPL) is an extension of a textual programming language that allows the programmer to replace parts of a textual program with diagrams. This system deals with only static programs. In this case, the AD is the original textual programming language, and consists only of O - symbols of the language. A visual sentence consists of P, D and V^n, where $P = G \cup O$ and G is a set of domain-specific visual primitives. The elements of G, D, V^n and the $S \leftrightarrow VS$: ($O \leftrightarrow \{P, D, V^n\}$) mapping are determined by the designer of an HVPL for a textual programming language. The mapping is implemented as a set of productions that translate diagrams into their textual equivalents. The translated text is merged with the text of the rest of the program, which can then be compiled and executed.

Signature morphism, proposed by Wang and colleagues (Wang, Lee & Zeevat, 1995; Wang & Zeevat, 1997), is a formally derived one-to-one mapping between signatures of a visual sentence and a state of the AD, specifying the meaning of the sentence. A signature consists of a set of types with a partial order, a set of functions including instances of types and their attributes, and a set of predicates that specifies relations between instances. The signature morphism is a consistent mapping. While the authors propose it only for the static semantics of pictures, this approach appears suitable for dynamic semantics as well. On the other hand, Douglas et al (Douglas, et al., 1995) describe a technique called visualization storyboarding by which a mapping from the application domain of algorithms to a software visualization language can be derived empirically.

2. Cycle of Interaction

So far we have discussed the structure and meaning of visual languages - syntax and semantics of visual sentences and their transformations. The second central issue is the *usage* of visual languages. This requires specification of how a visual language serves as a medium of communication between a computational agent and a cognitive agent. One way of modeling this is to consider the cycles of interaction involved in visual language use. The cycle of interaction is a term proposed by Norman (1986) to describe the cognitive processes that take place in one cycle of activity during an ongoing episode of human-computer interaction. We extend this notion to include computational as well as cognitive processes in the use of visual languages. Figure 4 provides a detailed view of this category.

2. Usage: Cycle of Interaction

- 2.1 Granularity
- 2.2 Visual Communication: User to Computer, Computer to User, Both Ways
- 2.3 Computational Aspects
  - 2.3.1 Parsing: Grammars, Rewrite Rules, Detecting Emergent Visual Objects, Context Sensitivity, Ambiguity, etc.
  - 2.3.2 Interpretation/Compilation: Type/Token Referentiality, Semantic Density, Coreference Resolution, etc.
  - 2.3.3 Execution: Executable Specifications, etc.
  - 2.3.4 Generation/Transformation
- 2.4 Cognitive Aspects
  - 2.4.1 Perception: Detecting Emergent Visual Objects, Ambiguity, etc.
  - 2.4.2 Comprehension: Realism, Visual Cues, Coreference Resolution, etc.
  - 2.4.3 Inference: Prior Knowledge, Mental Visualization, etc.
  - 2.4.4 Creation/Manipulation

Figure 4. Subcategories of Interaction Cycle

Bottoni and colleagues (Bottoni *et al*, 1995; 1997) have developed a theoretical characterization of visual languages from the interaction and communication perspective illustrated by Figure 1. They consider visual languages as consisting of visual sentences that are specified in terms of an image on the display, a description of what the image means (i.e., a description of its programmatic implication), an interpretation function from image to description, and a materialization function in the reverse direction. Given this framework, visual languages are characterized in terms of whether components of every image in every visual sentence can be interpreted in terms of programmatic components, whether every programmatic component has an associated image component visible on the display, and whether the user can interact with, and receive feedback from, every image component that is visible.

Wittenburg and Weitzman (1997) describe a practical application of visual language theory that supports the cycle of interaction. When users of their system ShowBiz desire to aggregate and hide parts of process flow diagrams, they select parts that they wish to include in the aggregation. The system uses a relational grammar to expand the user selection to a well-structured component that can be aggregated, and highlights this component. Users are then free to operate upon this sub-flow to hide it, view it in another window, etc.

2.1 Granularity

Using even a non-interactive visual language requires one cycle of interaction: the user specifies the program to be executed in the visual language, the computer interprets or compiles the program, executes it and displays the results, and the user views and comprehends the results. On the other extreme, use of highly interactive visual languages may involve many cycles of interaction, each occurring at a much finer granularity than that of an entire program as in the former case. Thus, the granularity of the cycle of interaction is a criterion for comparing various visual languages from an HCI perspective. The highly successful Macintosh™ Desktop depicts the state of the file system to the user, and the user can communicate actions to the operating system through gestures such as dragging a file icon to the trash can, which are then executed by the computer and the resulting state is displayed graphically. This is a visual language with a fine granularity of interaction.

2.2 Communication

A visual language may be used for one-way or two-way communication within an interaction cycle. A visual programming language may allow the user to create and communicate a program to the computer using visual representations. But the program itself may be non-visual and the results of its execution may be conveyed back to the user textually. Software visualization systems such as algorithm animators, on the other hand, require users to use textual means for specifying visualizations, the result of executing which is a visual language that depicts program execution for the benefit of the user. Other visual languages such as ShowBiz (Wittenburg & Weitzman, 1997) or those used in virtual reality systems are designed for two-way communication.

Given the model in Figure 1, we can see that two additional aspects of the interaction cycle that are important are computation and cognition, each of which can be further subdivided in terms of computational and cognitive processes involved in a cycle.

2.3 Computational Aspects

The computational processes involved are parsing the visual sentence, interpreting it or compiling it into executable code, executing it, and transforming the visual sentence on the visual display based on execution results. For visual languages intended for one-way communication, either only generation is relevant, or only parsing, interpretation and execution are relevant.

Two properties of visual representations make them difficult to parse and interpret. One is that they are multidimensional unlike text which can be linearly read and decomposed. The second is that visual primitives are typically token-referential unlike textual symbols which are generally type-referential (Barwise & Etchemendy, 1995). Since visual primitives that are of the same type may not necessarily have common semantics because of token-referentiality, interpretation of such primitives is more difficult than that of type-referential primitives. This difficulty is overcome in human visual languages through the use of accepted conventions. Another approach is enforcing type-referentiality at the cost of expressiveness in computer visual languages (e.g., Hyperproof) and in formalisms (e.g., the partial order over types in signature morphisms). Other obstacles to efficient parsing and interpretation of general purpose visual representations are detecting emergent[3] objects (Koedinger, 1992; Gero & Yan, 1994), context sensitivity of their interpretations (Figure 6 provides an example), ambiguity, and semantic density (see 3.2 below). Another chapter in this volume (Marriott, Meyer & Wittenburg, 1997) provides more information on visual language parsing.

[3] Emergence is a term used to describe the perception of objects that a visual sentence does not explicitly represent. This can occur due to gestalt processes (e.g., seeing a circle when the representation contains only dots) or when explicitly represented objects give rise to additional implicit objects (i.e., as in two overlapping circles being perceived as three closed contours).

Generative grammars allow computers to generate or recognize valid visual sentences. Such grammars are useful for implementing interpreters or compilers for visual languages. But grammars are not useful for transforming one valid visual sentence to another in order to capture AD dynamics. Rewrite rules (Repenning, 1995) allow computers to transform existing visual sentences to represent some dynamic process. Rewrite rules are similar in nature to operators that allow a computation to move from one valid state to another in a problem space (Newell & Simon, 1972). Visual languages with a fine granularity of interaction are typically interpreted whereas those with a coarse granularity may be compiled. Some commercial mixed languages allow both. Researchers have addressed the issue of specifying meanings of visual sentences in such a way that the specifications are executable (Haarslev, 1997).

Whether grammars and rewrite rules help human users understand computational processes in the interaction cycle has not yet been studied in the context of any visual language. Computer interpretation of diagrams used in human visual languages remains a difficult problem as well. Electronic Cocktail Napkin (Gross, 1994), Beatrix (Novak & Bulko, 1993), Redraw (Tessler, Iwasaki & Law, 1995), and Anon (Joseph & Pridmore, 1992) are examples of computer programs that use artificial intelligence techniques to understand diagrams commonly employed in human endeavors. Electronic Cocktail Napkin interprets architectural sketches, Beatrix interprets physics problems stated using English and labeled diagrams, Redraw interprets diagrams of load bearing structures that civil engineers draw, and Anon interprets mechanical engineering drawings. These systems illustrate the difficulty of interpreting even highly stylized visual representations used in their respective ADs. When visual representations use a mixed vocabulary of diagrams and text, there is an additional source of difficulty - that of resolving coreferences (Novak & Bulko, 1993). This is the computationally and cognitively difficult problem of resolving references of multiple textual symbols and visual primitives that have the same referent.

2.4 Cognitive Aspects

The cognitive processes involved are those of perceiving the displayed visual sentences, comprehending their meaning, making inferences about the state of affairs depicted by the visual sentences, and interacting with the computer in order to communicate the next actions by manipulating elements of the visual sentence. Research on perceptual processes (e.g., detecting emergent objects, disambiguating pictures, etc.) has traditionally taken place in disciplines far removed from visual languages. However, ways in which humans comprehend and reason with static and dynamic pictures have been investigated by researchers in the diagrammatic representation and mental imagery communities. A number of findings from this research are relevant to visual languages.

It has been found that parsing and semantic interpretation is not easy for humans (Lowe, 1993) and that realism facilitates semantic interpretation (Schwartz & Black, 1996). This supports the argument that isomorphism (Gurr, 1977) is an important property. Resolving coreference when multiple elements of the visual sentence have the same referent, as in the case of a mixed visual sentence consisting of text and picture in which many words and graphical units refer to same things, and constructing a unified model of information from text and diagrams is difficult both computationally (Novak & Bulko, 1993) and cognitively (Hegarty & Just, 1993). This is relevant to the type of mapping a visual language utilizes, indicating that one-to-many mappings from AD to visual sentences may not be a good idea. Prior knowledge, such as that of diagramming conventions, has been found to be critical to correct interpretation (Lowe, 1994), and visual primitives help cue and retrieve relevant prior knowledge from long term memory (Narayanan, Suwa & Motoda, 1994b). This supports the use of knowledge intensive parsing and interpretation techniques that systems like Beatrix (Novak & Bulko, 1993) and Anon (Joseph & Pridmore, 1992) employ. It has been found that people employ mental visualization (Antonietti, 1991;

Hegarty, 1992; Narayanan, Suwa & Motoda, 1994b) to make inferences when static pictures are used to represent dynamic ADs, and that this process creates a mismatch between the external and internal representations resulting in increased cognitive load (Narayanan, Suwa & Motoda, 1995b). This provides a strong argument for additional research on the dynamic syntax and semantics of visual languages. It has been observed that diagram generation and manipulation activities play an important role in creative thought processes such as architectural design (Goel, 1995). This points to the potential utility of developing advanced knowledge-based visual languages to assist such human endeavors.

Results from studies on how humans comprehend diagrammatic representations can serve to illuminate the cognitive processes. Narayanan and Hegarty (Narayanan & Hegarty, 1997) have developed a cognitive model of comprehension of diagrams and mixed representations. This model postulates that comprehending the functioning of dynamic systems from static diagrams, as in inferring how mechanical devices work from cross sectional schematic diagrams, involves five stages - decomposition, recognition, recomposition, determination of activity propagation paths, and mental animation. This suggests that the relevant visual primitives, dimensions and relations in a visual sentence should be explicit and recognizable. In addition, configurations of visual primitives in visual sentences must be amenable to easy and unambiguous parsing. The reverse mapping from P, D and V^n to O, A and R^n and the compositional semantics of visual sentences (i.e., how the meaning of a visual sentence can be derived from composing the meanings of P, D and R^n elements present in the sentence) should be obvious or easily learnable. Furthermore, this model indicates the potential cognitive utility of meaningful animations or transformations of visual sentences in a visual language. However, cognitive researchers have generally investigated human visual languages such as engineering diagrams, weather maps and architectural sketches, not visual languages that both computers and humans can use. Cognitive issues in the cycle of interaction for visual languages have not yet been thoroughly explored (Petre, Blackwell & Green, in press).

3. Evaluation

The third important issue and corresponding top level category is evaluation (Figure 5). Evaluation involves the *development* of specific measures, *characterization* of individual languages using these measures, and *comparison* of multiple visual languages on the basis of these characterizations. Visual languages have to fulfill two objectives: that of being computationally efficient and cognitively effective. The former objective is essential for implementing the language, whereas the latter one is essential for justifying the language. What is the guarantee that any visual language fulfills these objectives? In order to evaluate this, we require both computational and cognitive analyses, which form the next level subdivision.

3. Evaluation

- 3.1 Computational
  - 3.1.1 Measures: Expressiveness, Time Complexity, Space Complexity, Graphic Token Count, Diagram Class Complexity, Graphic Token Density, Syntactic Differentiation/Density, Semantic Differentiation/Density, etc.
  - 3.1.2 Characterization of Individuals: Notational, Analog, etc.
  - 3.1.3 Comparative Analysis
- 3.2 Cognitive
  - 3.2.1 Measures
    - 3.2.1.1 Comprehensibility Factors: Explicitness, Homomorphic, Injective, Surjective, or Isomorphic Semantic Mapping, Salience, Confusion Count, etc.
    - 3.2.1.2 Usability Factors: Learnability, Retainability, Performance Efficiency, Modifiability, Maintainability, Error Rate, Initial Impression, Long Term Satisfaction, Cognitive Fit, etc.
    - 3.2.1.3 Cognitive Dimensions: Visible Dependencies, Secondary Notations (e.g., Spatial Clustering), Viscosity, etc.
  - 3.2.2 Analysis of Individuals
  - 3.2.3 Comparative Analysis

Figure 5. Subcategories of Evaluation

3.1 Computational Evaluation

Computational analysis requires the specification of a set of measures such as expressiveness, time complexity, and space complexity. The analysis can be done at an individual level, to characterize the tractability of a specific visual language. Multiple languages may be compared using these measures. An example is the complexity hierarchies developed by Marriott and Meyer (1997). Nickerson (1994) proposes two integer-valued measures, graphic token count and diagram class complexity, and one real-valued measure, graphic token density, that are useful for comparing common diagrammatic notations such as graphs and trees. How these syntactic measures of visual notations relate to complexity measures of the corresponding visual languages, and whether there are other useful measures, need to be further researched. Another interesting possibility is to examine computational implications of Goodman's (1976) notions of syntactic and semantic differentiation and density, and the corresponding notions of notational and analog systems (see Raymond, 1991 for definitions) for visual languages.

3.2 Cognitive Evaluation

Conducting a cognitive evaluation requires experimental research with human participants, which is outside the scope of visual language theory. However, theorists can contribute by identifying relevant cognitive measures to characterize individual languages or to comparatively analyze multiple ones. Excellent beginnings along these lines are presented in (Raymond, 1991) and Nickerson (1994). Both authors come to similar conclusions regarding the kind of visual language that is likely to be most effective:

> "The visual character of languages is rooted in their ability to use dense representations to describe dense domains" (Raymond, 1991). "Application of these metrics to current visual programming languages does not paint an optimistic future for the use of fully general, fully diagrammatic visual programming languages due to their low density" (Nickerson, 1994).

Unfortunately this line of research, judging by the literature, appears not to have been pursued further.

One useful cognitive measure is comprehensibility - how easy is it for a user to understand and learn the syntax and semantics of the language. A number of factors influence comprehensibility. It has been noted that visual representations are easier to understand because they make information explicit (Larkin & Simon, 1987). One way of achieving explicitness is through representations that are similar or analogous to the represented, and which preserve properties of the represented in an explicit way. Such representations have been called isomorphic representations. It has been experimentally observed that when static visual representations depict dynamic situations, people tend to mentally simulate the dynamic processes by internally manipulating the visual representations (Hegarty, 1992; Narayanan, Suwa & Motoda, 1995a; Shepard & Cooper, 1986; Schwartz & Black, 1996). Isomorphic visual representations can aid this, and thus increase comprehensibility, if transformations of the visual representations are isomorphic to the domain processes being simulated. Gurr (1997) analyzes properties that contribute to isomorphism, and develops characterizations of different levels of similarity. Using Gurr's terminology, it appears that at least a homomorphic mapping between the AD and visual sentences is required for similarity. Injectivity, surjectivity and bijectivity or isomorphism of the mapping further increase the level of similarity. This research on categorizing visual representations based on the extent to which they are similar to the represented has begun to provide a formal framework for what has hitherto been an intuitive notion. Salience is another factor that can influence comprehensibility. This is a measure of semantically salient aspects of elements of a visual representation. Consider, for example, a visual language that uses circles as one of its visual primitives. A circle can have a number of visual dimensions: its location, radius, thickness of its boundary, color of its boundary, color of its inside area, etc. Out of *n* possible dimensions in D if the language employs *m* dimensions, of which only *l* are meaningful, then *l/m* provides a measure of the salience of D for circles. A low salience indicates a high number of irrelevant and potentially distracting visual features in the representation, implying lower comprehensibility. A visual language with low salience needs to make the relevant primitives, dimensions or relations visually explicit in order to aid comprehension. A related measure is confusion count, developed by Nickerson (1994) by modifying a metric originally proposed by Glinert (1990). It is the sum of the number of crossings and elbows in a diagrammatic notation, with the assumption that a high number of these adversely affect comprehensibility.

The broad notion of usability, borrowed from interactive system and user interface design literature (Hix & Hartson, 1993; Newman & Lamming, 1995), is a useful measure to apply. Usability is typically measured using its constituent factors such as learnability or ease and speed of learning the visual language, retainability - whether the visual language helps programmers better remember the programming constructs provided, performance efficiency (i.e., productivity improvement resulting from use of the visual language), maintainability and modifiability of programs constructed using the visual language, error rate indicating a decrease in number of errors committed when using the visual language, initial impression and long term satisfaction (two qualitative measures of how satisfied programmers are in using the visual language), and several others. One usability study (Badre & Allen, 1989) appears to indicate that graphical notations do not provide an advantage for procedural languages. Cognitive fit (Sinha & Vessey, 1992), or the match between the forms of information users seek in the context of a problem solving task or application domain and the forms in which the visual language presents information (Petre, Blackwell & Green, in press), is another factor affecting the usability of visual languages.

The cognitive dimensions approach of Green (Green, 1990; Green & Petre, 1996) provides additional measures. For example, the dimension of hidden dependencies indicates how much or how little of the local and global dependencies between variables, subroutines, etc. are made explicitly visible by the visual language syntax. Another dimension is secondary notations - the use of D and V^n to convey meaning. One way to accomplish this is by spatial clustering. Graphical representations can spatially organize and localize relevant information together (Larkin & Simon, 1987) so that a visual search can locate information quickly. Very little research has been done on how secondary notations might improve cognitive effectiveness. Another cognitive dimension called viscosity, the effort required to transform a visual representation from one valid state to another, may also be useful for evaluation and comparison.

These notions need to be further developed and formalized so that the measures can be applied to characterize and compare current and future visual languages. A research project intended to rigorously define the scientific basis for cognitive advantages of visual programming, an initial result from which is an explicit enumeration of metacognitive beliefs motivating the design of visual languages, holds considerable promise for achieving this goal (Blackwell, 1996).

There is no significant research yet that addresses *both* computational and cognitive evaluation of the same visual language. Hyperproof (Barwise & Etchemendy, 1994) is a mixed visual language system that was designed to teach logic to students. While its authors have studied theoretical and computational issues (Barwise & Etchemendy, 1995), others have looked at the cognitive effectiveness of the language - i.e., how well it helps students learn logic in comparison with traditional teaching methods. The results (Stenning, Cox & Oberlander, 1995) have been mixed in that while advanced students benefited, students with less knowledge or visual capability did not show improved learning. Researchers in software visualization in general (Price, Baecker & Small, 1993), and algorithm animation in particular (Brown & Sedgewick, 1985), strive to create visual languages using which computers can communicate the inner workings of programs to humans. These are not visual languages that support the full cycle of interaction - instead, they are intended for one-way communication from the computational to the cognitive. Most visualizations that have been designed so far are special purpose languages designed for a particular algorithm or program, or a class of algorithms. Recent research on the cognitive effectiveness of such visualizations has also unearthed mixed results (Byrne, Catrambone & Stasko, 1996). It should be noted that in all these cases the visual languages have already been designed and implemented in the computational realm before cognitive studies were undertaken.

In contrast, Mahling and Fisher (1990) apply cognitive theory to design rather than evaluation of a visual language. But the question of how to design a visual language to be simultaneously, and provably by empirical or theoretical evaluations, efficient computationally and effective cognitively is still open, and in our opinion provides a fertile avenue for interdisciplinary research in visual language theory. Opposing computational and cognitive requirements (e.g., realism aids human comprehension but realistic pictures are much harder for the computer to parse) make this difficult, especially for purely visual languages. In this case, how to trade off computational complexity and cognitive effectiveness is also an open issue. But the development of visual simulative languages - mixed languages that allow a user to specify some state of the world and actions to execute in that state, and have the computer execute the actions and display the results - indicate that finding a good balance between computational efficiency and cognitive effectiveness is not impossible for mixed visual languages in specific domains. The Alternate Reality Kit (Smith, 1986), which is an environment for users to create and experiment with the dynamics of moving particles, is an early example. KidSim™ (Smith, Cypher & Spohrer, 1994) and Star Logo (Resnick, 1996) are currently popular mixed visual languages using which one can build simulations of microworlds and processes occurring inside them. The computer then executes the simulations and depicts results graphically through interactive animations of the microworlds using pictures created by the user. These are meta visual languages in a sense, because these allow users to build graphical interactive simulations which are visual languages tailored to specific ADs.

Another approach that shows promise as an answer to this question is heterogeneous visual programming languages that allow the incorporation of pictures into traditional textual programming language syntax. This improves cognitive effectiveness since complex code can be replaced with self-evident pictures. For an example, see the illustrated code for AVL tree insertion in Erwig & Meyer (1995). Computational efficiency is not compromised since the language is translated back to the source textual language before being compiled and executed.

VI. Issues for Visual Language Research

In this section we discuss a few significant characteristics of human visual languages that enable these to support and enhance the cognitive abilities of people. These indicate additional avenues for visual language research from a human-computer interaction perspective.

VI.1 Affordances of Visual Representations

There are many different ways to define what a visual language is. In the introduction of this book, Marriott and Meyer define a visual language as consisting of visual sentences constructed from arrangements of symbols in two or three dimensional space. Current theories typically translate such visual sentences of a language into a propositional representation which is then ascribed with meaning. A picture can, in principle, be described propositionally in an informationally equivalent fashion. One can, for instance, write a set of propositions describing each bit of a bitmap of the picture. Although informationally equivalent, such descriptions may be too reductionistic from a cognitive perspective because the propositional representation into which a visual sentence is translated may not provide the same cognitive affordances[4] as the original pictorial representation of the visual sentence. This does not just apply to the picture as a static object, but also to changing or moving pictures, to operations applied to pictures, and to transformations of pictures.

The following four characteristics of visual languages afford effective comprehension and reasoning:

- explicit representation of salient relations of the domain through visually perceivable properties,
- spatially localized and visually cued organization of relevant information about domain objects and relations,
- facilitation of visual transformations to simulate domain processes, and
- reduction of complexity through recognition and guidance.

These characteristics make visual sentences of human visual languages quite different from corresponding propositional representations. These properties are only partially captured by current theories of visual languages. In fact, for many human visual languages it is not clear how one might construct informationally equivalent propositional representations of visual sentences in the language, even when such representations are based on spatial formalisms. Consider, for example, the difficulty of developing specifications of architects' sketches using the spatial logic approach of Gooday and Cohn (1997) without loosing the aesthetic properties of the sketches.

Explicit Representation

Visual representations are designed by people to explicitly depict those relations that are important in the domain or for the task at hand. This allows a viewer to recognize relevant patterns and relationships instead of having to deduce them through complex reasoning. This advantage does not carry over to propositional representations since visual properties of constituent symbols do not carry meaning in such representations. Thus, even an informationally equivalent propositional representation may not explicitly represent the same properties as the corresponding picture. It does not, of course, mean that all visual representations are good at explicating information they carry.

[4] Affordance is a term that Norman introduced (Norman, 1988). He defines it as "the perceived and actual fundamental properties of something that allow a perceiver to determine just how it could possibly be used". In this sense, visual representations afford certain kinds of interpretations and inferences.

This requires careful design of representations. One informal yardstick to apply might be "guessability" of the signature morphism (Wang, Lee & Zeevat, 1995) of the visual representation.

Consider the configuration of disks in Figure 6, which can easily be described in propositional form using their absolute coordinates. Figure 6A is represented by the following assertions:

```
white-circle(d1)
white-circle(d2)
location(d1,10,1)
location(d2,13,1)
```

Are the two white circles close to each other in Figures 6A, 6B, 6C and 6D? Given the visual representation it becomes immediately clear that the answer must take into consideration the location of other disks in the figure. The white circles are close in Figure 6B but not in 6D. In 6A and 6C the question cannot be answered satisfactorily. This dependency of closeness judgment on the surrounding context is not evident in the propositional representation. While regular and context free pictures may be easy to deal with computationally, sentences of human visual languages, we suspect, are generally context sensitive.

Another problem with propositional descriptions is that because of the intentional or unintentional information loss that occurs when a picture gets translated into propositions, the depiction-to-description mapping is many-to-one unless one describes every visual aspect of the picture in excruciating detail. Once the translation is done in which information is discarded or lost, the reverse translation cannot be done uniquely. As Meyer (1993) states: "If we want visual languages to be more than standard languages in disguise, we have to learn how to describe pictures by pictures".

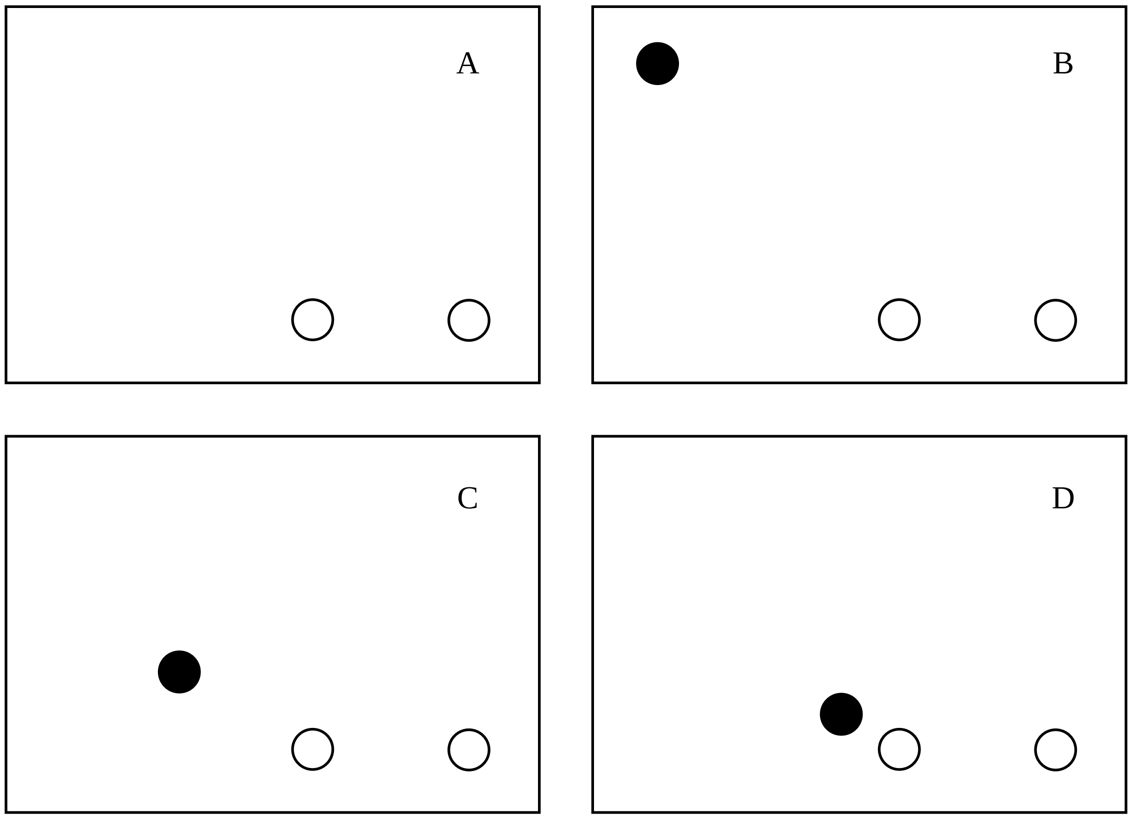

Figure 6. Are the two white circles close to each other?

Spatial and Visual Organization of Information

Visual representations help guide reasoning because they permit related information to be spatially organized. They allow the use of visual cues such as color or texture to convey information and to draw the reasoner's attention to relevant objects and properties. Larkin and Simon (1987) provide an analysis of how spatial adjacency and connectedness can be used to reduce the complexity of reasoning in solving physics problems when the problem descriptions are accompanied by diagrams. A series of experiments on how people reason about mechanical devices from cross-sectional schematic diagrams (Narayanan, Suwa & Motoda, 1994b; 1995a) show that diagrams guide the reasoning process along the lines of causal propagation in the operation of the device. People use an incremental reasoning strategy of predicting behaviors of local components and propagating these to other components by exploiting spatial cues of adjacency and connectedness. This strategy works most of the time because diagrams organize components spatially with component depictions reflecting their spatial organization; thus the deduced order of events in the operation of the machine corresponds to paths of causal propagation. Scientific visualizations go one step further in using not only spatial organization but also visual cues such as color and density to convey considerable information in a concise and spatially localized manner.

Transformation of Images

Changing a picture may or may not change the meaning of it, depending on whether the changed visual aspects are semantically significant or not. Changing a propositional representation generally implies that the current state of affairs has changed since all its constituent symbols carry meaning. Visual representations support the generation of hypotheses because they represent relevant relations explicitly. We often test these hypotheses by slightly changing an image mentally to test whether a certain relationship holds even if other things have changed. Propositional representations do not support reasoning about continuous change, especially the simulation of spatial processes, as well as corresponding visual representations can. Visual representations support such reasoning by facilitating the process of transforming the image under the domain's constraints, generating new hypotheses, and testing them against the state of affairs depicted by the transformed image (Narayanan, Suwa & Motoda, 1994b; Hegarty, 1992).

The examples in Figure 7 show situations where the visual representation affords reasoning by image transformation more than a corresponding propositional one. Figure 7A shows a round hole and a cube. The question to be answered is whether the cube can be pushed through the hole. Nontrivial reasoning is required to answer this question given a propositional representation of the geometries involved. However, using an interactive visual representation similar to Figure 7B the cube can be moved over the hole and it becomes immediately clear that it will not fit through the hole.

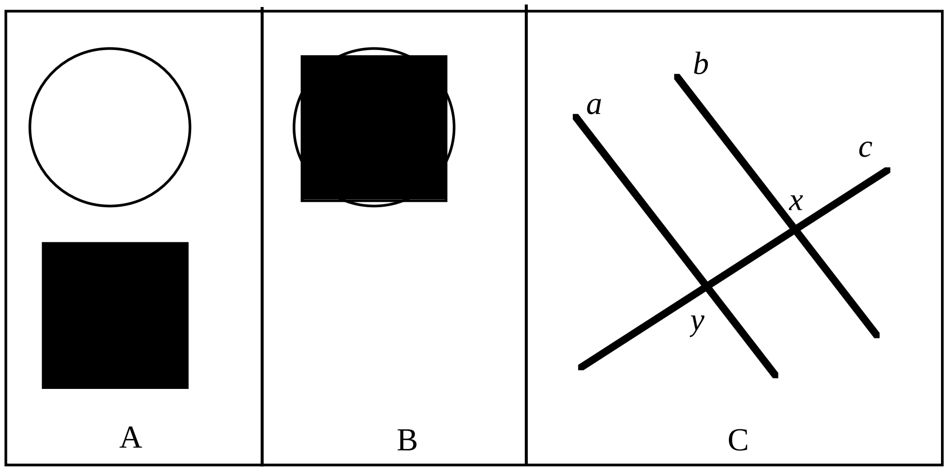

Figure 7. Reasoning by image transformations

Figure 7C shows two angles *x* and *y* formed by two parallel lines *a* and *b* and an intersecting line *c*. It is easy to read out from the diagram that the two angles appear to be equal. Furthermore, it is also evident that translating or rotating line *c* does not affect this equality. The angles *x* and *y* remain equal independently of the exact position or orientation of line *c* relative to *a* and *b* as long as it intersects both. We can come to these conclusions quickly because of our ability to mentally simulate transformations of visual representations. Starting with informationally equivalent propositional representations of the situations in Figure 7, the same inferences can be made, but not as easily and directly. In these kinds of problems, even when given a descriptive propositional or natural language representation people tend to draw or mentally image a corresponding visual representations in order to solve the problems (Huttenlocher, 1968).

Computational Complexity

Visual representations make characteristics of the represented situation explicit, group related information spatially or by other visual cues, and facilitate inferences based on direct manipulation. In comparison, propositional representations contain considerable implicit information. This means that for information processors that are attuned to the perceptual modality, visual representations can be more efficient than propositional ones, and that the opposite situation holds for processors that are attuned to propositional representations. Thus, informationally equivalent representations can have different computational complexity depending on the operations performed on them and the nature of the underlying information processing architecture that performs the operations (Larkin & Simon, 1987). The implication for visual language theories is that making visual languages easier for humans may in general result in making them harder for computers and vice versa. There are, however, exceptions to this. Figure 7 showed that visual representations afford certain inferences, thereby reducing reasoning to recognition of a spatial relationship. Visual and non-visual reasoning are often intertwined (Barwise & Etchemendy, 1995; Stenning & Oberlander, 1995), and in such cases visual representations can provide guidance to the logical reasoning process. Such guidance can result in a lower number of alternatives that the inference process needs to evaluate, thereby making it more efficient. Computer programs that use diagrams that humans find equally useful for guiding and controlling inferences, such as Redraw (Tessler, Iwasaki & Law, 1995) and Hyperproof (Barwise & Etchemendy, 1994), illustrate this point. Characterizing this role of visual languages is an open problem in visual language theory.

VI.2 Other Issues

Deriving Meaning from Multiple Pictures

Many current theories specify pictures using text, such that for each meaningful visual sentence there is a corresponding propositional sentence. In addition, for each transition between two visual sentences there is a similar transition between two propositional sentences with an equivalent meaning. However, this latter assumption generally works only because meaningful changes are initially defined in terms of rewriting propositional sentences from which corresponding visual sentences can be generated.

Consider going the other way instead: developing formalisms for rewriting propositional sentences from the way the corresponding visual sentences change while depicting some dynamics of the represented domain (e.g., a temporal sequence of visual sentences that are snapshots of an animation of a physical process). Since such visual representations evolve in a continuous fashion and are syntactically and semantically dense (Goodman, 1976), one can find many intermediate pictures between two that represent a qualitative change of state that can propositionally be asserted. For example, consider the set of pictures one can draw depicting states between `¬over(cube,hole)` and `over(cube,hole)` in Figure 7 above. Regardless of how many discrete states a propositional representation is designed to assert between the two states above, one can always create visual sentences that depict states between any two adjacent states asserted by the propositional representation. Thus, the transition from visual to propositional one is not loss-free when considering dynamic visual representations.

It is sometimes suggested that the issue of sequences of temporally related visual sentences can be addressed by building on current theories that deal with single visual sentences. It is not clear from the current state of research on visual languages whether semantic dependencies among visual sentences in a temporally ordered sequence can be simply added to a current theory of static visual language. It seems an open question, ripe for research.

Emergent Properties

Even when a continuous transformation is not involved, as in Figure 6, it is unclear when an informationally equivalent propositional representation should assert that the two white circles are close. The visual representation does not have to explicitly assert this because closeness is intrinsically represented in the representation. One way to “solve” this problem would be to say that the visual representation says nothing about whether the circles are close to each other, that it is left to the judgment of the observer. But this is exactly the point. In the model of visual language use that Figure 1 presents, the cognitive system is capable of such inferences whereas the computational system, to the extent it relies on propositional translations of visual sentences on the display for inferences, is not capable of the same inference. In other words, the cognitive system is capable of detecting context sensitive emergent properties of visual representations whereas the computational system requires explicit specification of such properties.

VII. Concluding Discussion

Visual languages are interesting because they allow humans to represent, comprehend, modify, and make inferences about the represented via direct manipulation of visual representations. If the human element was unimportant, there would be no need for visual languages executable by computers since computers are far better at parsing, compiling and executing textual languages. Therefore, visual languages ought to be studied not only from the perspective of specifications that can be compiled and executed by computers, but, equally importantly, also from the perspective of specifications that can be understood and modified by humans. Thus one needs to take into account the full context of human-machine coupled systems, and the role of the visual language in facilitating communication and interaction between computational and cognitive agents in this system, in both implementing useful and usable visual languages and investigating their formal properties.

This implies that theoretical analyses ought to address issues of comprehension, reasoning, and interaction in the cognitive realm as well as issues of visual program parsing, execution, and feedback in the computational realm. This is not to deny the utility of formal specifications — namely, allowing precise definitions of meanings of the syntactic elements of a visual language, implementation of a correct parser, generator or reasoning algorithm, and comparative analyses of different visual languages — but to argue for the necessity of complementing this with theoretical investigations of the other aspects as well. This is a lacuna in the current state of visual language research. Most current theories of visual languages are almost entirely computer centered. However, since we are dealing with a concept that has been originally invented by humans for effective communication, a computer centered theory of visual languages can at most be a start. Indeed, chapters in this book point to the beginnings of a broader scope for visual language research.

In this chapter we have looked at characteristics of visual languages that support the cognitive abilities of people. To a large degree these are the very characteristics that make visual languages so interesting. Unfortunately, these tend not to be easily amenable to formalization. Nevertheless, we should strive for theories that help us better understand visual languages, their properties, and their role in human-computer interaction. As a beginning, a framework consistent with a broader scope of inquiry for visual language research is proposed and used to develop a preliminary taxonomy for characterizing theoretical and practical aspects of visual languages. In the process of showing how a sample of current work fits into this framework, open issues and fruitful directions for future research are pointed out. We propose the taxonomy not as an end, but as a means to encourage further research on the issues that it explicitly addresses. We agree with Raymond (1991) when he asserts:

> "Practical languages contain both visual and notational elements, however, and we must evaluate a given language for its suitability both visually and as a notation. Thus many dimensions and taxonomies are needed to determine a language's overall effectiveness".

Theories of human visual languages have existed for quite some time: visual thinking (Arnheim, 1969), semiotics of visual languages (Saint-Martin, 1990), etc. But new types of theories are necessary for visual languages in the context of human-computer interaction. A theory of visual languages must be formal enough to derive its computational properties. So most authors of visual theories take the approach of converting a visual representation into a textual one in order to define its semantics. But if we are interested in the characteristics that distinguish visual languages from textual ones, can we expect much help from textual translations? It may be sensible to translate visual sentences into propositional ones because we understand formal languages well and can ascribe semantic interpretations to them. However, what do we really learn about visual languages if the translation does not capture characteristics that the cognitive system exploits in comprehending visual sentences? We do not know the answer. But we will never find out if we don't ask this question seriously.

REFERENCES

Abelson H., Sussman G. J. & Sussman J. (1996). *Structure and Interpretation of Computer Programs*, 2nd ed., MIT Press, Cambridge, MA.

Andries, M., Engels, G. & Rekers, J. (1997). Using graph grammars to represent visual programs. Chapter in this volume.

Antonietti, A. (1991). Why does mental visualization facilitate problem solving? In R. H. Logie & M. Denis, (Eds.), *Mental Images in Human Cognition*, Elsevier Science Publishers B. V., Amsterdam, Netherlands, pp. 211-227.

Arnheim, R. (1969). *Visual Thinking*, University of California Press, Berkeley, CA.

Badre, A. & Allen, J. (1989). Graphic language representation and programming behavior. In S. Salvendy & M. J. Smith, (Eds.), *Designing and Using Human-Computer Interfaces and Knowledge Based Systems*, Elsevier Science Publishers, Amsterdam, Netherlands, pp. 59-65.

Barwise, J. & Etchemendy, J. (1994). *Hyperproof*, Cambridge University Press, Cambridge, England.

Barwise, J. & Etchemendy, J. (1995). Heterogeneous logic. In *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, J. Glasgow, N. H. Narayanan, & B. Chandrasekaran, (Eds.), AAAI Press, Menlo Park, CA and MIT Press, Cambridge, MA. pp. 211-234.

Bertin, J. (1967). *Semiology of Graphics*, English translation by W. J. Berg, University of Wisconsin Press, Madison, WI, 1983.

Blackwell, A. F. (1996). Metacognitive theories of visual programming: What do we think we are doing? *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 240-246.

Bottoni, P., Costabile, M. F., Levialdi, S. & Mussio, P. (1995). Formalizing visual languages. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 45-52.

Bottoni, P., Costabile, M. F., Levialdi, S. & Mussio, P. (1997). Specification of visual languages as means for interaction. Chapter in this volume.

Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4), pp. 10-19.

Brown, M. H. & Sedgewick, R. (1985). Techniques for algorithm animation. *IEEE Software*, 2(1), pp. 28-38.

Byrne, M. D., Catrambone, R. & Stasko, J. T. (1996). Do algorithm animations aid learning? Tech. Rep. GIT-GVU-96-18, GVU Center, Georgia Institute of Technology, Atlanta, GA, August 1996.

CACM, (1996). Special section on educational technology, *Communications of the ACM*, 39(4), April.

Chang, S- K. (1987). Visual languages: A tutorial and survey. *IEEE Software*, 4, pp. 29-39.

Dinesh, T. B. & Üsküdarli, S. (1997). Specifying input and output of visual languages. Chapter in this volume.

Douglas, S., Hundhausen, C. & McKeown, D. (1995). Toward empirically-based software visualization languages. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 342-349.

Engelhardt, Y., Bruin, J., Janssen, T. & Scha, R. (1996). The visual grammar of information graphics. In N. H. Narayanan & J. Damski, (Eds.), *Proc. AID'96 Workshop on Visual Representation, Reasoning and Interaction in Design*, Key Center for Design Computing, University of Sydney.

Erwig, M. & Meyer, B. (1995). Heterogeneous visual languages: Integrating visual and textual programming. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 318-325.

Frank, M. R. & Foley, J. D. (1994). A pure reasoning engine for programming by example. Tech. Rep. GIT-GVU-94-11, GVU Center, Georgia Institute of Technology, Atlanta, GA, April 1994.

Freeman, E., Gelernter, D. & Jagannathan, S. (1995). In search of a simple visual vocabulary. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 302-309.

Furnas, G. (1991). New graphical reasoning models for understanding graphical interfaces. *Proc. Human Factors in Computing Systems Conference (CHI'91),* ACM Press, pp. 71-78.

Furnas, G. (1992). Reasoning with diagrams only. *Proc. AAAI Spring Symposium on Reasoning with Diagrammatic Representations,* AAAI Technical Report SS-92-02, AAAI Press, Menlo Park, CA, pp. 118-123.

Gero, J. S. & Yan, M. (1994). Shape emergence by symbolic reasoning. *Environment and Planning B: Planning and Design,* 21, pp. 191-218.

Glasgow, J., Narayanan, N. H. & Chandrasekaran, B. (Eds.) (1995). *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press, Menlo Park, CA and MIT Press, Cambridge, MA.

Glinert, E. P. (1990). Nontextual programming environments. In S- K Chang (Ed.), *Principles of Visual Programming Systems,* Prentice-Hall, New York, pp. 144-230.

Goel, V. (1995). *Sketches of Thought,* MIT Press, Cambridge, MA.

Gombrich, E. H. (1968). *Art and Illusion: A Study in the Psychology of Pictorial Representations,* Phaidon, London.

Gooday, J. M. & Cohn, A. G. (1997). Visual language syntax and semantics: A spatial logic approach. Chapter in this volume.

Goodman, N. (1976). *Languages of Art: An Approach to a Theory of Symbols*, Hackett Publishing Company, Indianapolis, Indiana.

Green, T. R. G. (1990). Cognitive dimensions of notations. *Proc. HCI'90 Conference.*

Green, T. R. G. & Petre, M. (1992). When visual programs are harder to read than textual programs. In G. C. van der Veer, M. J. Tauber, S. Bagnarola & M. Antavolits, (Eds.), *Human-Computer Interaction: Tasks and Organization, Proc. 6th European Conference on Cognitive Ergonomics,* pp.167-180.

Green, T. R. G. & Petre, M. (1996). Usability analysis of visual programming environments: A cognitive dimensions framework. *Journal of Visual Languages and Computing,* 7(2), pp. 131-174.

Green, T. R. G., Petre, M. & Bellamy, R. K. E. (1992). Comprehensibility of visual and textual programs: A test of superlativism against the match-mismatch conjecture. In J. Koenemann-Belliveau, T. G. Moher, S. P. Robertson (Eds.), *Proc. Fourth Workshop on Empirical Studies of Programmers,* Ablex Publishers.

Gross, M. (1994). The fat pencil, the cocktail napkin, and the slide library. In A. Harfmann & M. Fraser, (Eds.), *Proc. ACADIA 94,* Association for Computer Aided Design in Architecture, pp.103-113.

Gurr, C. A. (1997). On the isomorphism (or otherwise) of representations. Chapter in this volume.

Haarslev, V. (1997). A fully formalized theory for describing visual notations. Chapter in this volume.

Hammer, E. (1995). Logic and visual information. In *Studies in Logic, Language & Computation,* CSLI Press, Stanford University.

Hegarty, M. (1992). Mental animation: Inferring motion from static displays of mechanical systems. *Journal of Experimental Psychology: Learning, Memory & Cognition,* 18(5), pp. 1084-1102.

Hegarty, M. & Just, M. A. (1993). Constructing mental models of machines from text and diagrams. *Journal of Memory and Language*, 32, pp. 717-742.

Hix, D. & Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product & Process,* John Wiley & Sons, Inc., New York.

Hübscher, R. (1995). Rewriting interaction. *Proc. Human Factors in Computing Systems Conference (CHI'95),* ACM Press.

Hübscher, R. (1996). Composing complex behavior from simple visual descriptions. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 88-94.

Huttenlocher, J. (1968). Constructing spatial images: A strategy in reasoning. *Psychological Review,* 75(6), pp. 550-560.

Joseph, S. H. & Pridmore, T. P. (1992). Knowledge-directed interpretation of mechanical engineering drawings. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 14(9), pp. 928-940.

Kahn, K. M. & Saraswat, V. A. (1990). Complete visualizations of concurrent programs and their executions. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 7-14.

Koedinger, K. R. (1992). Emergent properties and structural constraints: Advantages of diagrammatic representations for reasoning and learning. *Proc. AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, AAAI Technical Report SS-92-02, AAAI Press, Menlo Park, CA, pp. 154-169.

Larkin, J. (1989). Display based problem solving. In D. Klahr & K. Kotovsky, (Eds.), *Complex Information Processing*, Lawrence Erlbaum Publishers, Hillsdale, NJ.

Larkin, J. H. & Simon, H. A. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, pp. 65-99.

Lohse, G. I., Biolsi, K., Walker, N. & Rueler, H. H. (1994). A classification of visual representations. *Communications of the ACM,* 37(12), pp. 36-49.

Lowe, R. K. (1993). Constructing a mental representation from an abstract technical diagram. *Learning and Instruction,* 3, pp. 157-179.

Lowe, R. K. (1994). Selectivity in diagrams: Reading beyond the lines. *Educational Psychology*, 14, pp. 467-491.

Mahling, D. E. & Fisher, D. L. (1990). The cognitive engineering of visual languages. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 22-28.

Marriott, K. & Meyer, B. (1997). Towards a hierarchy of visual languages. Chapter in this volume.

Marriott, K. Meyer, B. & Wittenburg, K. (1997). A survey of visual language specification and recognition. Chapter in this volume.

Menzies, T. (1995). Frameworks for Assessing Visual Languages. Technical Report TR95-35, Department of Software Development, Monash University.

Meyer, B. (1993). Pictures depicting pictures: On the specification of visual languages by visual grammars. Technical Report No. 139, Informatik Berichte, FernUniversitat, Hagen, Germany. (A shorter version appears in *Proc. 1992 IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 41-47.)

McCorduck, P. (1991). *Aaron's Code*, Freeman, San Francisco, CA.

Myers, B. (1986). Visual programming, programming by example and program visualization: A taxonomy. *Proc. Human Factors in Computing Systems Conference (CHI'86),* ACM Press, pp. 59-66.

Narayanan, N. H. (Ed.) (1992). *Proc. AAAI Spring Symposium on Reasoning with Diagrammatic Representations,* AAAI Technical Report SS-92-02, AAAI Press, Menlo Park, CA.

Narayanan, N. H. & Hegarty, M. (in press). On designing comprehensible interactive hypermedia manuals. *International Journal of Human-Computer Studies*.

Narayanan, N. H., Suwa, M. & Motoda, H. (1994a). How things appear to work: Predicting behaviors from device diagrams. *Proc. 12th National Conference on Artificial Intelligence*, AAAI Press, pp. 1161-1167.

Narayanan, N. H., Suwa, M. & Motoda, H. (1994b). A study of diagrammatic reasoning from verbal and gestural protocols. *Proc. 16th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 652-657.

Narayanan, N. H., Suwa, M. & Motoda, H. (1995a). Diagram-based problem solving: The case of an impossible problem. *Proc. 17th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 206-211.

Narayanan, N. H., Suwa, M. & Motoda, H. (1995b). Behavior hypothesis from schematic diagrams. In *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, J. Glasgow, N. H. Narayanan, & B. Chandrasekaran, (Eds.) AAAI Press and MIT Press, pp. 501 -534.

Newell, A. & Simon, H. A. (1972). *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ.

Newman, W. M. & Lamming, M. G. (1995). *Interactive System Design*, Addison-Wesley, Wokingham, England.

Nickerson, J. V. (1994). Visual programming: Limits of graphic representation. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 178-179.

Norman, D. A. (1986). Cognitive engineering. In D. A. Norman & S. W. Draper, (Eds.), *User Centered System Design*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 31-65.

Norman, D. A. (1988). *The Psychology of Everyday Things*, Basic Books, New York.

Novak, G. S. & Bulko, W. C. (1993). Diagrams and text as computer input. *Journal of Visual Languages and Computing*, 4, pp. 161-175.

Petre, M. (1995). Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38, pp. 33-44.

Petre, M., Blackwell, A. F. & Green, T. R. G. (in press). Cognitive questions in software visualization. In J. Stasko, J. Domingue, B. Price & M. Brown, (Eds.), *Software Visualization: Programming as a Multi-Media Experience*, MIT Press, to appear.

Price, B. A., Baecker, R. M. & Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages and Computing*, 4(3), pp. 211-266.

Puigsegur, J., Agusti, J. & Robertson, D. (1996). A visual programming language. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 214-215.

Raymond, D. R. (1991). Characterizing visual languages. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 176-182.

Repenning, A. (1995). Bending the rules: Steps toward semantically enriched graphical rewrite rules. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 226-233.

Repenning, A. & Sumner, T. (1995). Agentsheets: A medium for creating domain-oriented visual languages. *IEEE Computer*, 28, pp. 17-25.

Resnick, M. (1996). Beyond the centralized mindset. *Journal of the Learning Sciences*, 5(1), pp. 1-22.

Robertson, G. G., Card, S. K. & Mackinlay, J. D. (1993). Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4), pp. 57-71.

Saint-Martin, F. (1990). *Semiotics of Visual Language*, Indiana University Press, Bloomington, IN.

Schwartz, D. L. & Black, J. B. (1996). Analog imagery in mental model reasoning: Depictive models. *Cognitive Psychology*, 30, pp. 154-219.

Selker, T. & Koved, L. (1988). Elements of visual language. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 38-44.

Shepard, R. N. & Cooper, L. A. (Eds.) (1986). *Mental Images and Their Transformations,* MIT Press, Cambridge, MA.

Shin, S-J. (1994). *The Logical Status of Diagrams*, Cambridge University Press, Cambridge, England.

Shu, N. C. (1986). Visual programming languages: A perspective and a dimensional analysis. In S. K. Chang, T. Ichikawa & P. A. Ligomenides (Eds.), *Visual Languages,* Plenum Publishing Corporation, New York, pp. 11-34.

Sinha, A. & Vessey, I. (1992). Cognitive fit in recursion and iteration: An empirical study. *IEEE Trans. on Software Engineering,* SE-10(5), pp. 386-379.

Smith, D. C., Cypher, A. & Spohrer, J. (1994). Kidsim: Programming agents without a programming language. *Communications of the ACM,* 37, pp. 54-68.

Smith, R. B. (1986). The alternate reality kit: An animated environment for creating interactive simulations. *Proc. IEEE Symposium on Visual Languages*, IEEE Computer Society Press, pp. 99-106.

Steinman, S. & Carver, K. (1995). Visual programming with Prograph CPX, Manning Publications/Prentice Hall, Englewood Cliffs, NJ.

Stenning, K., Cox, R. & Oberlander, J. (1995). Contrasting the cognitive effects of graphical and sentential logic teaching: Reasoning, representation and individual differences. *Language and Cognitive Processes,* 10(3/4), pp. 333-354.

Stenning, K. & Oberlander, J. (1995). A cognitive theory of graphical and linguistic reasoning: Logic and implementation. *Cognitive Science,* 19, pp. 97-140.

Tessler, S., Iwasaki, Y. & Law, K. (1995). Qualitative structural analysis using diagrammatic reasoning. In *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, J. Glasgow, N. H. Narayanan, & B. Chandrasekaran, (Eds.), AAAI Press and MIT Press, pp. 711-730.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information,* Graphics Press, Cheshire, CT.

Tufte, E. R. (1990). *Envisioning Information,* Graphics Press, Cheshire, CT.

Tufte, E. R. (1997). *Visual Explanations,* Graphics Press, Cheshire, CT.

Tversky, B. (1995). Cognitive origins of graphic productions. In F. T. Marchese (Ed.), *Understanding Images: Finding Meaning in Digital Imagery,* Springer-Verlag, New York, pp. 29-53.

Wang, D. & Lee, J. R. & Zeevat, H. (1995). Reasoning with diagrammatic representations. In *Diagrammatic Reasoning: Cognitive and Computational Perspectives,* J. Glasgow, N. H. Narayanan, & B. Chandrasekaran, (Eds.), AAAI Press and MIT Press, pp. 339-396.

Wang, D. & Zeevat, H. (1997). A syntax directed approach to picture semantics. Chapter in this volume.

Wittenburg, K. & Weitzman, L. (1997). Relational grammars: Theory and practice in a visual language interface for process modeling. Chapter in this volume.