Using Perceptual Syntax to Enhance Semantic Content in Diagrams

Pourang Irani and Maureen Tingley
University of New Brunswick, Canada

Colin Ware
University of New Hampshire

We use structured object recognition to map problem semantics to 3D diagram structures. This makes the diagram easier to read with minimal training.

Diagrams are essential in documenting large information systems. They capture, communicate, and leverage knowledge indispensable for solving problems and act as *cognitive externalizations* (intertwining internal and external processes to extract information from the external world to enhance thought).[1] A diagram provides a mapping from the problem domain to the visual representation by supporting cognitive processes that involve perceptual pattern finding and cognitive symbolic operations.[2] However, not all mappings are equal, and for effectiveness we must embed a diagram's representation with characteristics, which lets users easily perceive meaningful patterns. Consequently, a diagram's effectiveness depends to some extent on how well we construct it as an input to our visual system.[3,4]

In our research, we focus on a class of diagrams commonly referred to as graphs or node-link diagrams. Nodes representing entities, objects, or processes, and links or edges representing relationships between the nodes characterize them. Their most common form is outline circles or boxes denoting nodes and lines of different types representing links between the nodes. Entity-relationship diagrams, software structure diagrams, and data-flow models are examples of node-link diagrams used to model the structure of processes, software, or data.

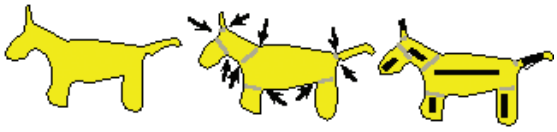
Currently, the most widely used graphical language for modeling complex systems is the Unified Modeling Language. UML contains a suite of diagramming techniques that lets you model various aspects of a software system,[5] a real-time application,[6] or an enterprise structure.[7] Its versatility in several application areas results from the rich semantics it seeks to model. For example, class diagrams in UML model software structures and include methods for depicting inheritance and composition. When we use these semantics in the enterprise modeling realm, UML can capture relationships between organizations or relationships between corporations and their employees. However, although developers have made UML notations general and complete, the actual choice of graphical notations appears to be somewhat arbitrary; only an expert in the field can easily read them.

Structured object perception

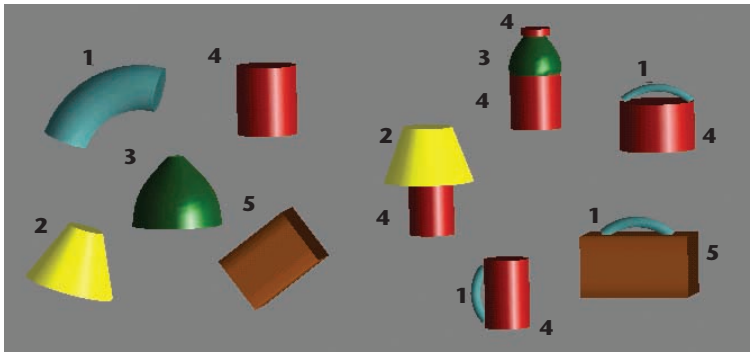
We can roughly divide theories of object perception into two general approaches: image- and structure-based perception. Image-based theories propose that human perception stores multiple views of an object in memory, specifying processes for matching the stored views to what's being perceived.[8] Structure-based theories place emphasis on the extracting 3D structural information of objects for recognition.[9] This extraction results in decomposing the image into perceptual primitives consisting of 3D solids such as cones, cylinders, and ellipsoids, along with information about how they're interconnected. Each theory has strong evidence supporting its validity and we believe that the visual system possibly uses both mechanisms in a hybrid manner at several layers of the recognition process. However, for our purposes the structure-based theories are more interesting because they suggest that if we can map information structures into structured objects then the structure will be extracted automatically as part of normal perception.

As our starting point, we take the theories of structural object recognition. Marr and Nishihara proposed a model in which our visual system extracts information from the viewed object's 2D contour or silhouette structure.[10] The silhouette decomposes into regions of concavity that facilitate the extraction of the image's subparts. Transformations within our visual system let us translate the sillhouett'es subparts into a set of 3D generalized cones. Figure 1 illustrates a crudely drawn animal that we nevertheless readily perceive as having distinct head, legs, torso, and tail parts. Marr and Nishihara also proposed a mechanism that cognitively connects the parts' axes to draw a structural skeleton.

0272-1716/01/$10.00 © 2001 IEEE



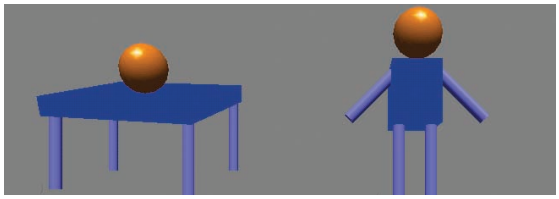
1 According to Marr and Nishihara,[9] concave sections of the silhouette define the object's subparts. These points are critical in defining a structural skeleton. (Adapted from Marr and Nishihara.[9])



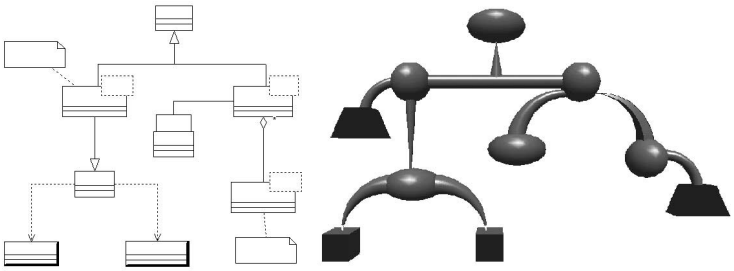
2 Geons are object primitives in Biederman's theory. When connected in a particular structural relationship, they can define an object. Different connections of the same geons can result in different objects (geons 1 and 4).

Biederman elaborated Marr's initial theories in two significant ways.[11] He extended the set of generalized cones defined by Marr by describing them based on geometrical properties of the silhouette in the 2D plane including colinearity, symmetry, parallelism, curvature, and cotermination (the contours meet at a point—such as a cone). He devised a set of 36 primitives he termed *geons* (geometrical ions). We depict a sample set of geons and objects constructed with them in Figure 2. A second significant contribution by Biederman is his description of the structural composition of the decomposed geons from the image. The decomposition of an object results in a geon structural description (GSD), consisting of geons, their attributes, and their relations with adjacent geons. The structural description contributes to object constancy. For example, if two views of an object result in a similar GSD, then our perceptual system recognizes them as equivalent objects.

The structural description isn't purely topological. As Figure 3 illustrates, two objects—such as a human figure and a table—can have identical geons and an identical skeleton in terms of its topology but still be identified as different objects. In describing the representational capacity of his set of geons, Biederman suggests looking at the number of readily discriminable relations between any pair of geons. These relations are largely viewpoint independent, preserve their 2D silhouette structure, and are categorical.[12] We base the following set of relational rules on the set Biederman proposed. We have added an additional containment rule.



3 The same geons and topological arrangement result in different objects. The geon structural description (GSD) is important for identification.

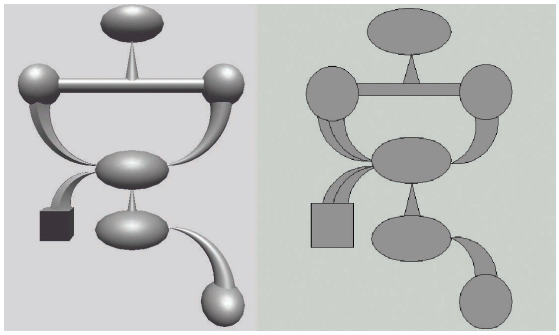


4 UML and Geon equivalent of a structured diagram. The relationships between nodes didn't represent any actual system.

- *RR1*. Color and texture are surface properties of geons that play a secondary role in perceptual object classification. These properties may aid in the recognition process, but don't play a major role in entry-level classification.
- *RR2*. Verticality. Geon A can be above, below, or beside geon B.
- *RR3*. Centering. Objects can be connected on- or off-center. For example, human legs connect to the right and left of the bottom of the torso. Human arms are at the top of the torso.
- *RR4*. Connection relative to elongation. Most geons are elongated, and connecting to the long face versus the short face has important perceptual semantics. We differentiate humans and four-legged animals in this way.
- *RR5*. Relative size. One geon is larger or smaller than another.
- *RR6*. Containment. An important perceptual task is identifying objects enclosed within larger components. This relationship is inherently hierarchical. However, we can only display strict containment using transparency.

Geon diagrams

If we identify structured objects through these mechanisms then we should be able to apply this theory to making more effective diagrams. We call such diagrams geon diagrams because they are loosely based on Biederman's geon theory. To evaluate this concept we did a series of studies that compare diagrams constructed with 3D geon primitives to various types of 2D diagrams (Figure 4).[13] Subjects in our experiments identified substructures 40 percent faster and about twice as accurately using the geon diagrams compared to corresponding UML diagrams. They also recognized geon diagrams they had seen briefly with far greater accuracy (18 percent versus 39 percent error rate). As an additional



5 A geon diagram and an equivalent 2D-silhouette diagram.

test, we repeated the experiments without surface attributes (color and texture) for the geon diagrams and without the corresponding labels on the UML diagrams. The result was the same. We found that subjects made half as many errors recognizing the geon diagrams compared to their equivalent 2D UML diagrams.

However, because many differences between UML and geon diagrams exist other than the use of 3D shape primitives, we conducted a further set of studies comparing shaded diagrams with the same diagrams as flat outline shapes (Figure 5). These results were also highly significant. Using 3D shaded primitives resulted in much more accurate substructure identification (11.4 percent versus 21 percent errors) and shorter times (4.1 seconds versus 5.2 seconds). The subjects also accurately recognized more 3D diagrams than 2D silhouette diagrams (20 percent versus 34 percent errors).[14]

The results of our experiments strongly suggest that 3D shaded primitives facilitate diagram structures' visual parsing and recognition compared to box and line diagrams (such as UML) and 2D silhouette equivalents. Thus, using 3D shaded components for diagram elements can facilitate interpretation and recognition of diagrams.

Perceptual semantics

The work of Biederman and others suggests that certain spatial relationships, such as on top of or unbalanced, may have a kind of immediately understandable perceptual meaning. We reasoned that if we could map the semantics of systems modeling into this perceptual semantics we could make diagrams that are easier to read.

UML class diagrams present a view of software structure. In particular, they depict the objects that exist, their internal structure, and their relationships with one another. They don't show temporal or causal information.[5] The notation was derived from three main sources—Booch, Rambaugh, and the Object Modeling Technique (OMT)—through the efforts of the Object Management Group (OMG) to standardize software modeling semantics and notations. UML offers a rich semantic base that we use for deriving our visual representations. We aren't augmenting UML notation by suggesting the use of differently shaped boxes or different types of edges. Instead, we're exploring the use of certain visual constructs to enhance the understanding and intuitiveness of semantics such as those available through UML. We hope that our readers can generalize our findings to other diagramming applications.

We carried out our investigation in a three-stage process. In the first stage, we constructed several different visual representations for each of the following modeling concepts:

- generalization—(a) is a (b);
- dependency—(a) depends on (b);
- strength of relationship—some relationships are stronger or weaker than others;
- multiplicity of relationship—for example, from one to many; and
- aggregation—(a) has a (b).

In each case, we used a perceptual principle to construct at least one of the instances. The other members of the set were made up of what we thought were reasonable alternatives. In the second stage, we conducted a multipart evaluation study to find out if subjects agreed on which mappings were the best. This experiment had made five parts, each evaluating the representation that best fit the five semantics. We conducted the experiment on 40 volunteer students, 20 of whom were familiar with software diagramming notation, in particular UML (experts). The remaining 20 students hadn't been exposed to any form of diagram modeling and hadn't been trained in understanding software semantics (novices). The experiment's interface was a Web browser. We used HTML and Javascript to resize and randomize the images' appearances. In the third stage, we created diagrams using the best mappings and evaluated how well users perceptually understood them.

For clarity, we separated each of the modeling concept mapping subexperiments, together with the evaluation results, in the following sections.

Generalization

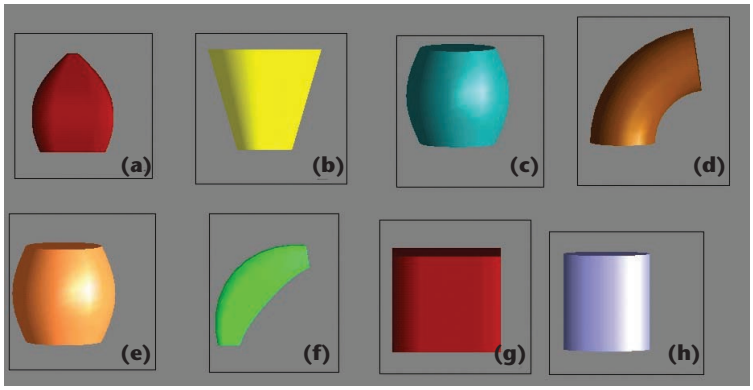
In abstract terms, *generalization* is the task of grouping concepts that fit a given pattern under a common header by moving from the particular to the general. This capability of the human mind has led to the concretization and refinement of ideas over the centuries. In software modeling, we use generalization to classify objects based on their common functionality. The UML notation guide defines generalization as being the relationship between a general element and a specific element that adds additional information to it.[15] It's also commonly referred to as inheritance (the specific object inherits properties of the general object) and is casually referred to as an *is a* relationship. Thus, objects can belong to a common class—for example, rottweilers, boxers, and retrievers are all dogs. The perceptual principle is based on Biederman's claim that the primitive shapes play a primary role in object classification, whereas surface properties such as color and texture play a secondary role (see our RR1 definition).[1]

Representing generalization

The purpose of this part of the experiment was to determine whether shape had a stronger influence than color in classifying objects of the same kind. If we determined shape was a better cue, then we could suggest using same-shaped primitives to denote objects of the

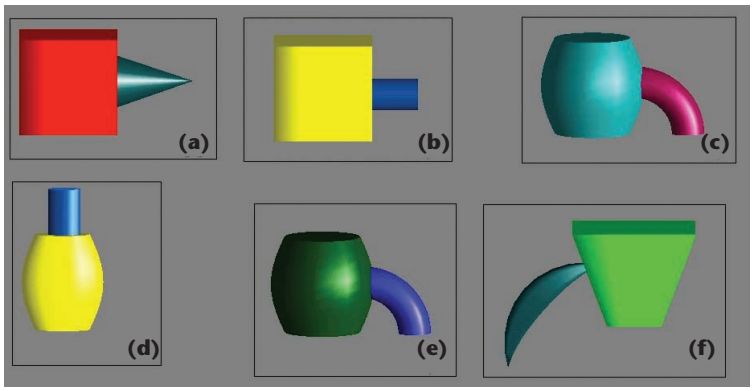
same kind. This is radically different than what is available in UML, where objects of the same kind connect by a solid line arrow with a closed arrowhead pointing to the general class.

We considered two cases, one containing objects made of a single geon primitive (a one-component case) and the other consisting of images containing two primitive shapes (a two-component case). The procedure used for each case was similar, so they only differed in their representations.



6 A sample set of representations containing only one component. This contains a matching color pair (a) and (g) and a matching pair of shapes (c) and (e).

The one-component case. We constructed three sets of images. Each set contained eight shaped primitives labeled (a) to (h) (Figure 6). The types of primitives were different from one set to another. Each set contained two primitives with the same color but different shapes, and two others that had the same shape but different surface color. For example in the set depicted in Figure 6, (a) and (g) are the same color (red) while (e) and (c) are the same shape (barrels). All the other shapes in the set have either a different color or different shape than these two pairs.



7 A sample set of two components contains a pair with matching major- and minor-shaped components (c) and (e), and matching color coding on major and minor components (b) and (d).

The two-component case. In this case, we created three sets of pictures, each containing six images. Each image was comprised of two geon primitives (Figure 7) labeled (a) to (f). For the images in all the sets we used one major (bigger) and one minor (smaller) geon primitive to construct the image. We created each set with two images containing major and minor components of the same color and different shapes—color pair (b) and (d)—and two others whose minor and major components were made of the same shaped geons with different colors—shape pair (c) and (e). All the other images in the set contained primitives whose major and minor components weren't colored as our selected color pair and didn't have the same arrangement of primitives as our shape pair.

Evaluating generalization

We showed subjects the three sets of images for the one-component case and the three sets of images for the two-component case. We asked them to select using their best judgment the pair of images that are of the same kind, which they recorded on handouts. For each of the two cases, we show the three sets in different order for all the subjects.

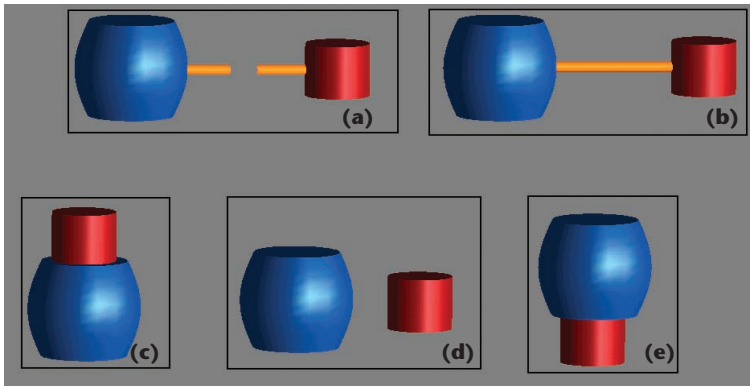
We didn't observe any statistically significant differences between responses from novice and expert subjects (a test for null hypothesis of agreement between novices and experts yielded *P*-values of 0.403 for the one-component case and 0.3091 for the two-component case). We therefore combined the results. Overall, 92 percent of the responses favored same shaped objects, whereas 8 percent of the responses favored the same color. This suggests that shape is a better stimulus for identifying objects of the same kind than color.

The small, representative set of 36 geon primitives[11] limits a large system in the number of different distinct objects that can be represented. By using shape to identify objects of the same kind, a diagram couldn't be formed by more than 36 distinct objects. We might resolve this limitation by creating objects with pairs of primitives, as we did for the two-component case of this part of the experiment. However, designing compound objects that express single-object identity might be challenging.

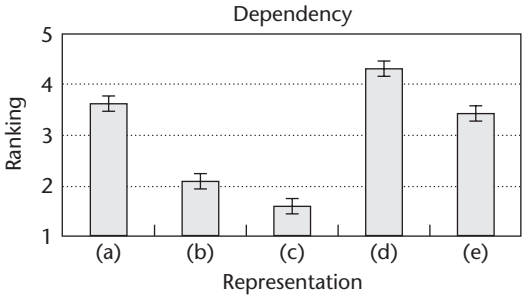
Using same shaped objects for generalization will also be problematic when modeling a system using multiple inheritance. Two objects of different shapes generalize into a third object whose representation may be a combination of the earlier two. On the other hand, some regard multiple inheritance as an improper structure in software modeling.[16] This has led to the exclusion of this feature in Java.

Dependency

Dependency commonly means a relationship in which an entity is supported by another. In software modeling terms, dependency describes a relationship



8 A sample set containing representations ranked for showing dependency. The red cylinder depends on the blue barrel.



9 Average rankings for dependency (1 equals best and 5 equals worst).

in which changes to one component can cause changes to the state of the dependent component. Therefore, the dependent module is unstable when changes are made to the entity it depends on. A goal in proper software modeling is to reduce the number of dependencies.[5]

Certain spatial representations are easily visually recognized.[17] These visual properties include such relations as longer or shorter, thinner or thicker, and above or below. Biederman suggests that the spatial property of verticality (see RR2) makes up for more than 80 percent of arrangements between visual components during object perception.[11] Glasgow and Papadias use such spatial properties to construct models that lower the computational costs for AI systems to retrieve and understand the representations of visual images.[18] Petre's study concluded that secondary notation, such as relative positioning of nodes (for example, placing two linked or unlinked nodes near each other), plays an important role in conveying meaning.[19]

Representing dependency

The purpose of this part of the experiment was to derive a representation best suited for understanding the semantics of dependency. If our perceptual system has certain spatial properties deeply ingrained to describe and classify objects, then perhaps we could use such properties to depict dependency.

We constructed five different ways of representing the dependency relationship, as Figure 8 illustrates. To show that the red cylinder depends on the blue barrel, these representations consisted of a broken tube (a), a connected tube (b), the cylinder on top of the barrel (c), disconnected objects (d), and the cylinder on the bottom of the barrel (e). Representation (a) most closely resembles the dependency representation in UML, which consists of a dashed line with an open arrowhead going from the dependent to the depended on.[5]

Evaluating dependency

We asked subjects to rank from one to five (best to worst) the representation denoting that one object depends on another. In Figure 8, the red cylinder depends on the blue barrel. They recorded their rankings for all three sets of representations. We showed each subject the representations in different random orders.

A χ^2 test on the results shows that there were no statistically significant differences in selecting the best representations between novices and experts (a P-value of 0.49 for null hypothesis of agreement between novices and experts). Therefore, we combined the results, and Figure 9 shows the average rankings of all 40 subjects. A top-down test of correlation on the average rankings shows a strong agreement between all 40 subjects for the best-ranked representations (P-value < 0.0001 for null hypothesis of no correlation between rankings chosen by 40 subjects). As Figure 9 shows, dependency is best depicted using the on top of representation (c). This representation is significantly better than the second best representation of a connected tube (b). With 95 percent confidence, the probability that any subject—novice or expert—would choose (c) over (b) is between 0.58 and 0.86.

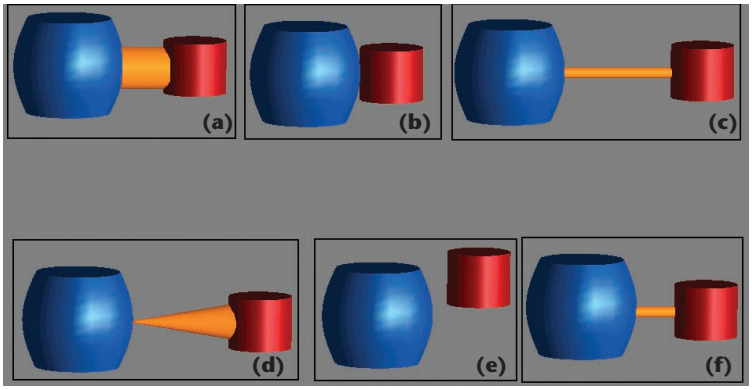
You don't commonly see dependency represented using the spatial property on top of in software structure diagrams, but you can find it in other types of visual representations. Organizational charts, which stack different parts of the corporation on a pyramid, use such a spatial organization between the represented entities. In such diagrams, an implicit assumption of the dependency exists between objects on the top of and on the bottom of the pyramid. A drawback of such a representation is the amount of space required to show several entities on top of any given object.

Relationship strength

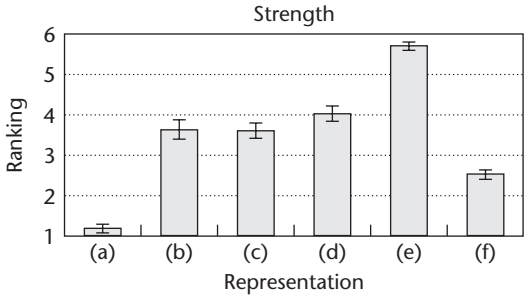
A common semantic in structured diagrams is strong and weak relationships. This semantic isn't directly modeled using UML but is common in other types of structured diagrams, such as entity-relationship diagrams.[20] In UML class diagrams, we refer to a strong aggregation as a composition, and this denotes a strong relationship between two entities. Biederman suggests that during object recognition our perceptual system differentiates parts of an object based on their relative sizes (see RR5).

Representing relationship strength

The purpose of this subexperiment was to examine whether we can apply the perceptual mechanism of discriminating based on relative ratios toward the semantic of relationship strength.



10 One of three sets being ranked for showing relationship strength. The red cylinder and blue barrel are strongly related.



11 Average rankings for the six representations for relationship strength (1 equals best and 6 equals worst).

We created six different ways of representing the strength of a relationship (Figure 10). These representations consisted of a thick connected tube (a), adjacent entities (b), a long connected tube (c), a conic connection (d), disconnected entities (e), and a short connected tube (f).

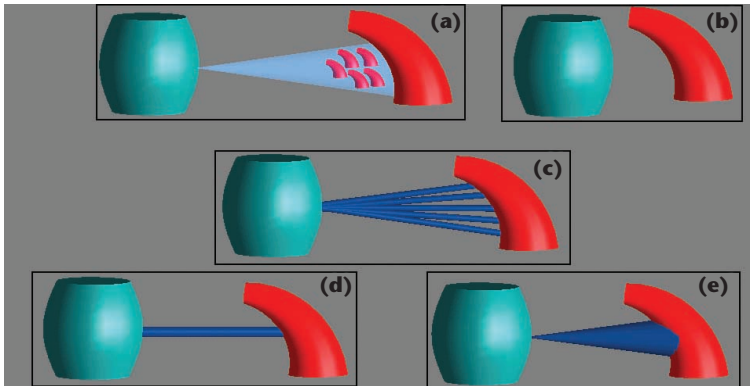
Evaluating relationship strength

We asked subjects to rank from one to six (best to worst) the representation denoting that one object is strongly related to another. For the example in Figure 10, we asked subjects to rank the representations according to how effectively each denoted a strong relationship between the blue barrel and red cylinder. For each subject, the representations appeared in a different random order.

A χ^2 test on the results showed that novices and experts were in perfect agreement on their choice of rankings (a test of null hypothesis of agreement yields *P*-value of 1.0). We thus combined the results and summarized the average rankings for 40 subjects in Figure 11. A top-down test of correlation shows that all 40 subjects were in strong agreement in selecting the top best rankings (*P*-value < 0.0001 for null hypothesis of no correlation between rankings chosen by 40 subjects), and Figure 11 depicts the best representation for showing that two entities are strongly related is a thick connection between the objects (a). This representation is significantly better than its alternative representation (f). With 95 percent confidence, the probability that any subject—novice or expert—chooses (a) over (f) is greater than 0.99. These results strongly suggest that using a relatively larger size for a connection can depict strength of relationship.

Multiplicity (or cardinality)

In many instances, skills involving counting or knowing the exact quantity isn't essential in providing a stable image. For example, a shepherd need not count to know whether his group is complete.[20] When children are asked to copy a figure made with counters, they don't use the exact number of counters (even if they know to count) but do justice to the shape of the figure.[21] A common method for representing this semantic attribute is the use of an asterisk (*) or an exact numeral (such as 1, 2, and so forth) over the link and beside the entity which is associated in multiples. Other common notations are 1..*, 0..*, or *. However, numbers are learned symbols and aren't cognitively immediate.



12 A sample set used in ranking representations of multiplicity. The blue barrel is associated with multiple instances of the red horn.

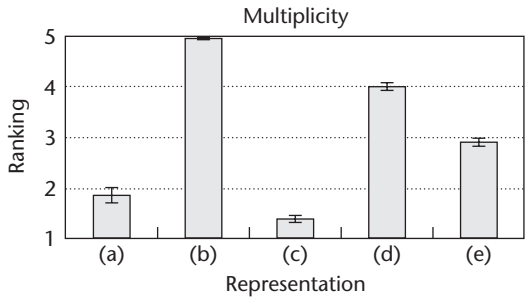
Representing multiplicity

The purpose of this part of the experiment was to derive a representation that would represent that an entity is associated with multiple copies of another object.

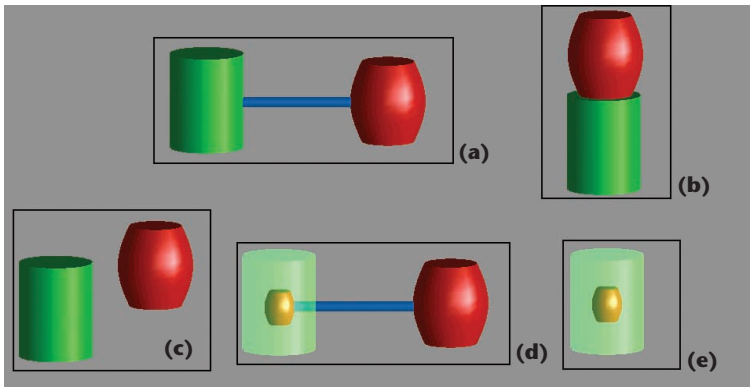
We constructed five different ways of representing the multiplicity attribute of a relationship as Figure 12 illustrates. These representations consist of a conic connection with multiple glyphs (a), disjoint objects (b), multiple connecting tubes (c), single connecting tube (d), and simple conic connection (e). In this illustration, the representations depict the blue barrel being associated to multiple instances of the red horn.

Evaluating multiplicity

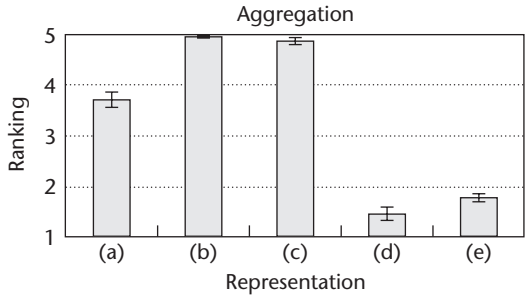
We asked subjects to rank from one to five (best to worst) the representation denoting that one object is associated to multiple copies of another object. We presented them three sets of images. For the set in Figure 12, we asked subjects to rank the representation that



13 Average rankings for the six representations for relationship strength (1 equals best and 5 equals worst).



14 A sample set used in ranking aggregation. The red barrel is contained within the green cylinder.



15 Average rankings for the five representations for aggregation (1 equals best and 5 equals worst).

best denotes the blue barrel is associated to multiple copies of the red horn. The order of the representations was random for each subject.

A χ^2 test on the results shows that both groups of subjects strongly agree on rankings (a test of null hypothesis of agreement yields *P*-value of 0.18). We therefore average the results, which we summarized in Figure 13. A top-down test of correlation on the average rankings shows strong agreement between all 40 subjects on their rankings (*P*-value < 0.0001 for null hypothesis of no correlation between rankings chosen by 40 subjects). As Figure 13 shows, multiple connecting tubes (c) best represent multiplicity. This representation is significantly better than its alternative (a). With 95 percent confidence, the probability that any subject—novice or expert—chooses (c) over (a) is between 0.52 and 0.82.

This representation of multiplicity has obvious problems. People are likely to interpret multiple representations in an overly literal sense. For example, if three branches exist, exactly three instances will be inferred. On the other hand, there's a theory that humans and animals have a perceptual sense of numbers, but only in a limited amount. We may be only able to naturally separate one, two, and possibly three objects. If there are more objects, we simply perceive them as a lot.[22]

Aggregation

In UML methodology, aggregation describes a special form of association in which an object contains another. In software engineering terms, we also refer to aggregation as a *has a* relationship. For example, an organization has a president. We refer to a strong aggregation as a *composition*. Evidence suggests that our perceptual system can separate an object when it's seen as being contained within other objects.

Representing aggregation

The purpose of this part of the experiment was to establish a representation suitable to denote an aggregation, part of, or containment relation.

We constructed five different ways of representing aggregation (Figure 14). These representations consist of a connected tube (a), disjoint objects (b), an object on-top-of (c), connection with containment (d), and containment (e).

Evaluating aggregation

We asked the subjects to rank from one to five (best to worst) the representation denoting that one object is contained within another. In the case of the representations given in the set in Figure 14, we asked subjects to rank the representation that best denotes that the red barrel is contained within the green cylinder.

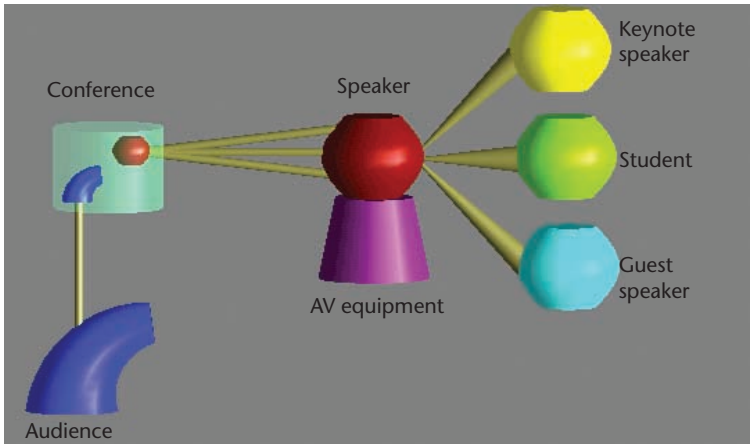
A χ^2 test on the results shows that both groups of subjects strongly agree on rankings (a test of null hypothesis of agreement yields *P*-value of 0.74). This result lets us average the rankings of all 40 subjects, which we summarize in Figure 15. A top-down test of correlation on the average rankings shows that subjects are consistent in their ranking (a *P*-value < 0.0001 for null hypothesis of no correlation between rankings chosen by 40 subjects). The results show that the best depiction for aggregation is a connection with containment (d). This representation isn't significantly better than its alternative (e). With 95 percent confidence, the probability that any subject—novice or expert—chooses (d) over (e) is between 0.47 and 0.76.

Applying theories of perception to drawing diagrams

Together with results from our previous studies, the results we describe here help us define rules for the geon diagram. We've used Biederman's term (geon), as it nicely describes the idea of a 3D shape, although we don't necessarily endorse the particular set of 3D shape primitives in Biederman's theory. We define five rules relating to the use of geons as 3D primitives, three rules that relate to the layout of the geon structure, and nine

additional rules for portraying certain semantics.

- *G1.* Present a system's major entities using simple 3D shape primitives (geons).
- *G2.* Represent links between entities by the connections between geons. Thus the geon structural skeleton represents the data structure.
- *G3.* Represent minor subcomponents as geon appendices—small geon components attached to larger geons.
- *G4.* Shade geons to make their 3D shape clearly visible.
- *G5.* Represent secondary attributes of entities and relationships by geon color and texture and by symbols mapped onto the surfaces of geons.



16 The geon representation for communicating structural information between representable entities in a conference.

Although geons are 3D shape primitives, theories of shape extraction rely heavily on a clear silhouette. For this reason, a good 2D layout will also be important in determining how easily users identify a geon structural description. Thus we add the following layout rules:

- *L1.* All geons should be visible from the chosen viewpoint.
- *L2.* Lay out the geon diagram predominantly in the plane orthogonal to the view direction.
- *L3.* Make junctions between geons clearly visible.

To provide rules for constructing effective diagrams, we need to extend the syntax of structural description to include mapping of semantics to data elements. The results of our experiments suggest that we can use certain types of naturally occurring semantic rules:

- *SM1.* Similarity and generality. We can use geons with the same structural geometrical composition (or shape) to denote objects of the same kind.
- *SM2.* Gravity. If geon (a) is on top of geon (b), this suggests that geon (b) supports geon (a). In addition, gravity determines that structures are perceived as either being stable or unstable.
- *SM3.* Enclosure. This shows that geon (b) contains geon (a). Syntactically, we can show this as an internal component attached to the same primitive geon on the outside.
- *SM4.* Ordinality. To show multiple associations between two entities, a series of attachments can best denote such a relationship.
- *SM5.* Strength of connection. Using a thicker connection as opposed to a thinner one can denote a stronger relationship between two entities.
- *SM6.* Sequence. Geons arranged in a line become a metaphor for a chain of operations or some other linear structure.
- *SM7.* Symmetry. Some information structures have symmetry, and we should use a symmetrical arrangement of geons to show this.
- *SM8.* Central and peripheral. If a component is of central importance to a structure, we can illustrate this through its position and by the location of the interconnections.
- *SM9.* Size. We can map larger components to an understanding of superiority in areas such as finances, economics, and politics.

Validating the perceptual syntax

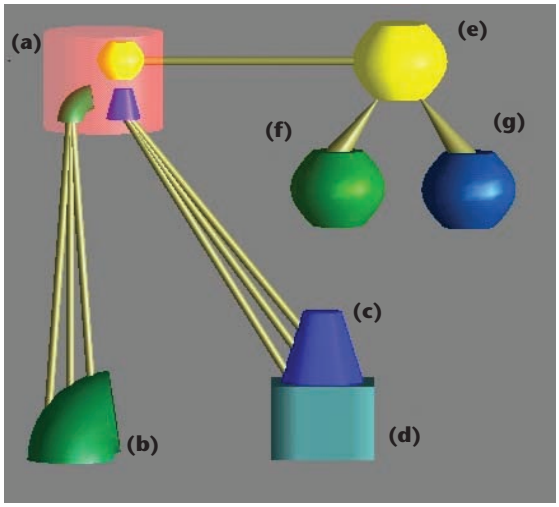
To evaluate the geon diagram syntax and semantics, we created geon diagrams to model real-world examples. For example, Figure 16 illustrates one of the diagrams modeling an academic conference whose components are attendees, speakers, A/V equipment, and so on. Using this diagram and others like it, we conducted an experiment to evaluate whether the notations of the geon diagrams were intuitive and easily describable. A group of 35 students (different from the 40 subjects we used earlier) who were unfamiliar with the software modeling semantics in UML participated in the experiment.

We created a set of three UML diagrams and equivalent geon diagrams for the experiment. We arbitrarily labeled the entities in the diagrams as we were primarily validating the syntax and didn't want to influence the choice of semantic based on the labeling (Figures 17 and 18, next page). In a classroom setting, we presented the subjects with the UML diagrams and equivalent set of geon diagrams. They had no previous experience with either type of diagramming convention. We asked them to describe the relationships between several pairs of entities in the diagram using multiple-choice answers. The choices included is the same kind as, depends on, has many, and has one.

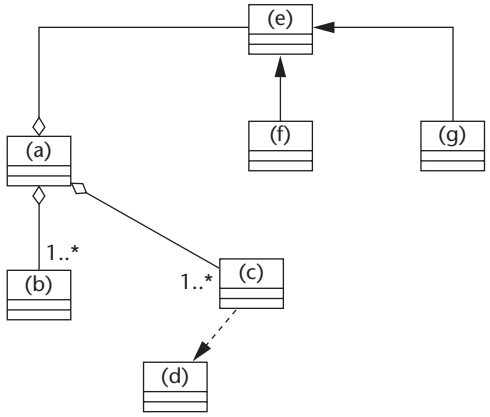
We summarize the error rates in identifying relationships as follows: the geon error rate was 11.5 percent and the UML error rate was 53.6 percent. These results show that there were almost five times as many errors deciphering relationships using UML notation than using the perceptual syntax.

Conclusion

Perception researchers have theorized that complex structured objects are automatically parsed by the visual system into component parts, together with a structural skeleton. We've shown that applying this theory



17 A geon diagram representation used in validating the syntax.



18 A UML representation equivalent to Figure 17's geon representation.

leads to a perceptual syntax with which we can construct diagrams that are easier to interpret and remember. The results suggest that mapping information structures into connected structures built with 3D geon primitives will make the information easy to read.

We recognize that our semantic mappings are far from complete. For example, studying the perceived meaning of different kinds of linking objects—broken tubes, dashed tubes, cone-shaped tubes, and transparent tubes—might further enrich our graphical vocabulary.

There are tradeoffs inherent in creating geon diagrams. Due to the amount of real estate they can consume, the complexity of what can be represented using these kinds of primitives may be less than what is possible using more cryptic line and box diagramming techniques. A possible tradeoff also exists between the concrete nature of geon diagrams and the representation of certain abstractions. If, for example, we use size to represent magnitude, then it becomes difficult to represent objects or concepts that have arbitrary magnitude. Labeling is an important element of diagrams and is another issue that we must address when using geon primitives. It may be difficult to show text as clearly wrapped on a 3D-shaded object. A textured object is especially likely to interfere with the readability unless the texture is subtle.

Our new notation can possibly help experts and nonexperts interact. In many projects, interaction between the client, manager, and programmer is essential for the proper system development, so a diagramming system that is more easily accessible to nonexperts could be useful. We of course aren't advocating the abandonment of UML notations; they're the most highly evolved graphical modeling tools we have. However, we hope that for new applications we can start to see the development of diagrams that take advantage of diagrams made with 3D shape primitives. ■

Acknowledgment

We gratefully acknowledge the support of an NSERC PGS-B award to Pourang Irani and an NSERC research grant to Colin Ware. Discussions with Jane Fritz have helped focus some of the ideas. We also thank Richard Spacek, who helped with the manuscript, and the volunteers who participated in our experiments.

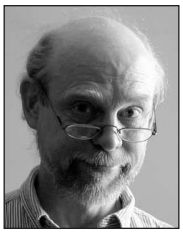
References

1. J. Zhang, "The Nature of External Representation in Problem Solving," *Cognitive Science*, vol. 21, no. 2, Apr. 1997, pp. 179-217.
2. J.H. Larkin and H.A. Simon, "Why a Diagram is (Sometimes) Worth Ten Thousand Words," *Cognitive Science,* vol. 11, no. 1, Jan.–Mar. 1987, pp. 65-99.
3. C. Ware, *Information Visualization: Perception for Design*, Morgan Kaufman, San Francisco, 2000.
4. M. Fowler, *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley, Reading, Mass., 1996.
5. B.P. Douglass, *Real-Time UML*, Addison-Wesley, Reading, Mass., 1998.
6. C. Marshall, *Enterprise Modeling with UML: Designing Successful Software through Business Analysis*, Addision-Wesley, Reading, Mass., 1999.
7. S. Ullman, "Aligning Pictorial Descriptions: An Approach to Object Recognition," *Cognition*, vol. 32, no. 3, Aug. 1989, pp. 193-254.
8. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Freeman, San Fransisco, 1982.
9. D. Marr and H.K. Nishihara, "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes," *Proc. Royal Society of London*, Series B, Biological Sciences, vol. 200, no. 1140, Feb. 1978, pp. 269-294.
10. I. Biederman, "Recognition-by-Components: A Theory of Human Image Understanding," *Psychological Review*, vol. 94, no. 2, Apr. 1987, pp. 115-147.
11. I. Biederman and P.C. Gerhardstein, "Recognizing Depth-Rotated Objects: Evidence and Conditions for Three-Dimensional Viewpoint Invariance," *J. Experimental Psychology: Human Perception and Performance,* vol. 19, no. 6, Dec. 1993, pp. 1162-1182.
12. P. Irani and C. Ware, "Diagrams Based on Structured Object Perception," *Proc. Advanced Visual Interfaces* (AVI 2000), Palermo, Italy, May 2000, pp. 61-67.

13. P. Irani and C. Ware, "Should the Elements of Diagrams Be Rendered in 3D?" *Late Breaking Hot Topics, IEEE Information Visualization 2000 Proc.*, CD-ROM, IEEE Computer Society, Los Alamitos, Calif., Oct. 2000.
14. *UML Notation Guide*, v. 1.1, Rational Software, Lexington, Mass., 1997.
15. C.S. Horstmann, *Computing Concepts with Java 2 Essentials*, John Wiley & Sons, New York, 1999.
16. J.I. Glasgow, N. Hari Narayanan, and B. Chandrasekaran, *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, MIT Press, Cambridge, Mass., 1995.
17. J.I. Glasgow and D. Papadias, "Computational Imagery," *Cognitive Science*, vol. 16, no. 3, July–Sept. 1992, pp. 355-394.
18. M. Petre, "Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming," *Comm. ACM*, vol. 38, no. 6, June 1995, pp. 33-44.
19. P.P. Chen, "The Entity-Relationship Model—Toward a Unified View of Data," *ACM Trans. Database Systems*, vol. 1, no. 1, Mar. 1976, pp. 9-36.
20. R. Arnheim, *Visual Thinking*, Univ. of California Press, Berkeley, 1969.
21. J. Piaget, *The Child's Conception of Numbers*, W.W. Norton, New York, 1952.
22. D. Stanislas, *The Number Sense: How the Mind Creates Mathematics*, Oxford Univ. Press, New York, 1998.

Pourang Irani *is a PhD candidate at the University of New Brunswick, where he received his BS in computer science in 1997. His research interests include the application of theories of perception to diagramming, to user interfaces, and in general to visualization systems. He is currently working on a diagramming model for English grammatical structures as well as on a visualization system for bioinformatics.*

Maureen Tingley *is a professor of statistics and director of the Applied Statistics Center at the University of New Brunswick. She holds a BA from the University of Adelaide; an MAT and MSc from Michigan State University; and an MA and PhD from Dalhousie University. Her research interests include small sample techniques (bootstrap), information and experimental design, and regression. Her primary areas of consultation and collaboration are in biology, engineering, ergonomics, ethics, forestry, and mathematics education.*



Colin Ware *is the director of the Data Visualization Research Laboratory, which is part of the Center for Coastal and Ocean Mapping at the University of New Hampshire. He is a professor of computer science and is also cross-appointed to the Department of Ocean Engineering. He holds an MMath in computer science from the University of Waterloo and a PhD in psychology (perception) from the University of Toronto. His research interests include applying perceptual principles to information display, building visualization systems, and methods for manipulating virtual objects.*

Readers may contact Pourang Irani at the Visual Interfaces Laboratory, Univ. of New Brunswick, Box 4400 Fredericton, NB, Canada E3B 2A3, email pirani@nbnet.nb.ca.