The Effect of a Perceptual Syntax on the Learnability of Novel Concepts

Pourang Irani
Department of Computer Science
University of Manitoba
Winnipeg, MB - R3T 2N2
Canada
irani@cs.umanitoba.ca

Colin Ware
Data Visualization Research Lab
University of New Hampshire
Durham, NH - 03824
USA
colinw@cisunix.unh.edu

Abstract

Language theorists argue that the reason why spoken language is acquired so rapidly is that we have an innate predisposition for understanding linguistic structures. Theories of perception also hold that there may be deeply seated mechanisms for decomposing visual objects and analyzing them into both component parts and the structural interrelationships of those parts. We propose the theory that diagrams that activate the mechanisms for structural object perception should be similarly easy to learn. This builds on previous work in which we have developed diagramming principles based on the theory of structural object perception. We call these geon diagrams. We have previously shown that such diagrams are easy to remember and to analyze. To evaluate our hypothesis that geon diagrams should also be easy to understand we carried out an empirical study to evaluate the learnability of geon diagram semantics in comparison with the well-established UML convention. The results support our theory of learnability. Both "novices" and "experts" found the geon diagram syntax easier to apply in a diagram-to-textual description matching task than the equivalent UML syntax.

Keywords *-- Pedagogy, software engineering, software engineering visualization, UML diagrams, perception, geon theory, semantic learning, object recognition.*

1. Introduction

In software engineering, conceptualizing the design of a system is an important element of the entire development process. This activity is supported by the use of sketches and diagrams to capture various aspects and semantics of the system being modeled. These visualizations mainly depict entities and their inter-relationships in any given system. The diagrams are characterized by nodes representing entities, objects, or units, and by interconnected links representing relationships of various kinds. Nodes are represented using outline forms such as boxes and circles, and links are depicted with lines of different characteristics. Many forms of diagramming techniques have been developed for modeling software engineering problems such as those available through the Unified Modeling Language (UML) [10]. The wide spread use of UML for modeling software systems is a result of the rich set of semantics it is capable of modeling through its diagrams. These diagrams help software architects, programmers, project managers, and, more recently, end-users communicate with one another. Although the semantics provided in these diagrams are general and complete, the choice of graphical notations appear to be somewhat arbitrary so that only an expert in the field can easily read and understand them. As a result, these notations may be even less effective as teaching aids.

To some extent, learning and using software engineering semantics is analogous to learning semantics in a natural language. Chomsky's theory that language understanding is based on innate deep cognitive structures is now widely, if not universally, held [2][11]. One of the main pieces of evidence for the theory is the very rapid acquisition of language exhibited by young infants in the second year of life. This accelerated learning has been attributed to the idea that the infant is predisposed to extract meaning from utterances because deep grammatical structure of those utterances is innately understood [4].

It has also been argued that there is a similar deep structure in vision, although the purpose of this structure is not communication but perception of the environment. The perceptual theory of Marr contains visual primitives such as "blobs", "bars", and "terminations" [9]. These are interpreted according to a kind of visual syntax thereby enabling us to understand 3D structured objects. Jackendoff [7] argues that the rules of visual structure are similar to verbal language rules. He further proposes that there are cognitive "correspondence rules" between the visual meaning of a 3D structure and linguistic structure. This provides a natural link between visual structure and linguistic structures that may help explain why certain kinds of diagrams are easy to understand. Others have also argued that mapping information representations into the appropriate perceptual representations should make the information easier to understand [8][14][15][16].

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

What has not been previously addressed is the implications from Jackendoffs theory that *perception based visual syntax and semantics* should also support accelerated learning. In our previous work [5][6] we have used the theories of Biederman and Marr to create rules for the construction of diagrams. We have shown that such diagrams are easier to remember and analyze than diagrams with a non-perception based syntax. Here we argue that such diagrams should be easier to learn and apply to matching of verbal descriptions to corresponding diagrams. Note, this is different from our previous work in which we showed *a)* that such diagrams could be easily recognized as having been seen before [5] and *b)* that the syntax of the diagrams could be remembered [6]. Since our argument is based on the perceptual validity of our diagrams we review structural object recognition theory and the set of diagramming conventions that we have derived, based on the theory.

2. Structural Object Recognition Theory

Biederman and his co-workers [1] elaborated the theories of structural object recognition proposed by Marr [9]. According to Biederman's theory, object recognition is accomplished in a hierarchy of processing stages. Objects are initially decomposed into edges, then into component axes, oriented blobs, and vertices. Extracting the three-dimensional primitives such as cones, cylinders and boxes (geons) follows this. Biederman defines a family of 36 geons that are derived from image properties on the silhouette contours in the 2D plane, by symmetry, parallelism, and curvature.

The extraction of the structure that specifies how the geon components interconnect follows the decomposition stage. This structure (or skeleton), known as the geon structural description (GSD), consists of geons, their attributes, and their relations with adjacent geons. It is primarily the structural description that contributes to viewpoint invariance, i.e. if two views of an object result in a similar GSD, then they should be treated as equivalent by the object recognition system. Finally, object recognition is achieved. Figure 1 illustrates a subset of geons and some simple objects constructed with them.

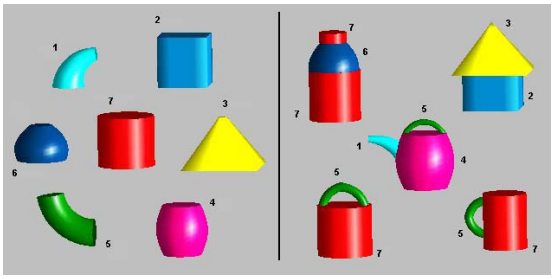


Figure 1. (a) Geons are object primitives in Biederman's theory. (b) When connected in a particular structural relationship they can define an object. (c) Different connections of the same geons can result in different objects as the figure shows geons 5 & 7 can give two different objects.

3. Perceptual Semantics

The structural description does not only consist of topology but includes a set of connection rules between geons. As illustrated in Figure 1, identical geons with different connectivity between them can define two very distinct objects (Figure 1.c). Thus in defining diagrams we need perceptual rules for connection that extend beyond topology. If we can understand these rules then we can map the semantics of diagrams onto geon structures in ways that will make diagrams more easily interpreted.

Biederman suggests that any given GSD is composed of a modest set of readily discriminable relations among geons. These relations can be determined from any viewpoint, preserve their two-dimensional silhouette structure, and are categorical [1]. The following set of relational rules is based on the set proposed by Biederman:

- **RR1:** Similarity - Shape of primitives plays a primary role while color and texture are surface properties of geons that play a secondary role in entry-level classification.
- **RR2:** Verticality - Geon A can be ON-TOP-OF, BOTTOM-OF or BESIDE geon B.
- **RR3:** Centering - Objects can be connected on or off-center. For example, human arms are off-centered while a human head is connected at the center-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 semantic implications.
- **RR5:** Relative size - One geon is larger or smaller than another, e.g. legs vs. arms.
- **RR6:** Containment - An important perceptual task is identifying objects enclosed within larger components or PART-OF relationships.
- **RR7:** Multiples – An exact amount of counters is not necessary to identify multiples.

We identify the image in Figure 2 with that of a human body by recognizing the connections and relationships defined by rules RR1 – RR7. For example, the head is on-top-of (RR2) the torso. The arms are connected to the long face of the elongated geon representing the torso (RR4) and the legs are connected to the short face of the torso geon. Legs are relatively larger than arms (RR5). A belly is part-of the torso (RR6). Multiple fingers are identifiable by means of showing several (RR7).

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

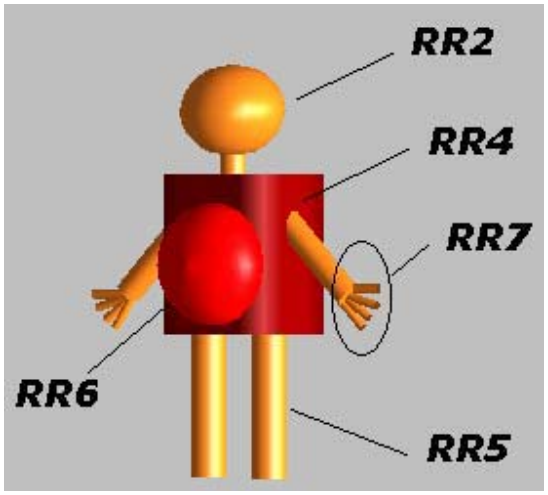


Figure 2. Relational rules extracted by the visual system for identifying a human 'object'.

4. PERCEPTUAL SYNTAX

Reviewing our argument, Geon Diagrams enable us to map both data structures and certain kinds of semantics into a kind of $2^{1/2}$ D diagramming convention. Figure 3 illustrates the relationships between deep linguistic and visual structures to diagram semantics. Similarly to Jackendoff's definition of "correspondence rules" between deep innate linguistic structures and semantics, we created in previous work [6] a set of "geon correspondence rules" that link Biederman's deep visual structures to diagramming semantics. According to Jackendoff's theory these kinds of diagramming structures should make accelerated learning possible, similar to the accelerated learning that occurs with verbal language.

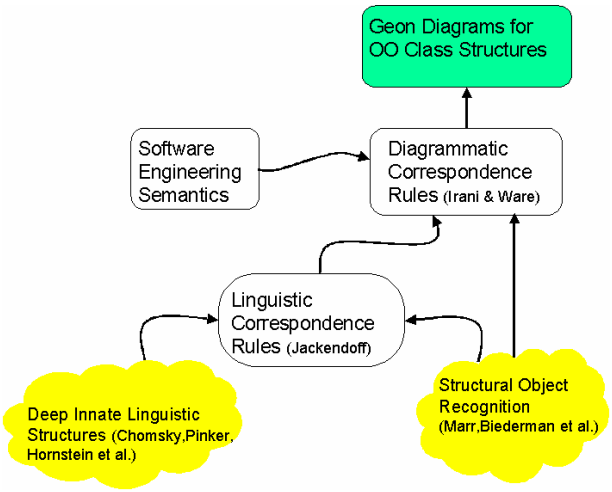


Figure 3. Theoretical concepts contributing to the correspondence rules for geon diagrams.

In our previous work, the syntax of Biederman's relational description rules were extended to include a mapping of semantics to data elements [6]. We focused on semantics found in software modeling diagrams specifically those in the category of UML class diagrams: 1-generalization ("is-a"), 2-aggregation ("has-a"), 3-dependency, 4-multiplicity ("many-of"), and 5-relationship strength. We applied the relational rules (RR1-RR7) defined above to construct a set of representations (or "geon correspondence rules") for each of the five semantics.

We suggest that certain types of "naturally" occurring "geon correspondence rules" can be used to map diagram semantics that will facilitate their comprehension. Here we describe the subset relevant to software engineering semantics:

- **GCR1:** Generality, Inheritance - Geons with same structural geometrical composition (or shape) can be used to denote objects of the same kind (derived from RR1).
- **GCR2:** Support, Dependency - If geon A is on-top-of geon B this suggests that geon A is supported by geon B. In addition gravity determines that structures are perceived as either being stable or unstable (derived from RR2).
- **GCR3:** Enclosure, Aggregation - shows that Geon A is contained within Geon B. Syntactically this can be shown as an internal component attached to the same primitive geon on the outside (derived from RR6).
- **GCR4:** Ordinality, Multiplicity - to show multiple associations between two entities a series of attachments can best denote such a relationship (derived from RR7).
- **GCR5:** Strength of connection - Using a thicker connection as opposed to a thinner one can denote a stronger relationship between two entities (derived from RR5).

We evaluated the "geon correspondence rules" defined above and from these produced the following perceptual syntax for each of the five semantics. We summarize the resulting syntax in Figure 4 (details are provided in [6]).

Figure 5 illustrates an example of how these rules describe related entities of a Space Center. The Space Center has-many Buildings (containment with multiple connecting lines), and has-many Spacecraft. A Gas Station and a Lab are two different types of buildings (same shape primitive as Building). The Gas Station has Fuel (containment with connection). Shuttles are also a type of Spacecraft (same shape primitive). Shuttles have-many Wings, and has-one Engine. The Engine depends on Fuel (depicted on-top-of the Fuel entity).

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

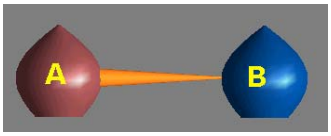
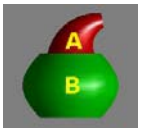
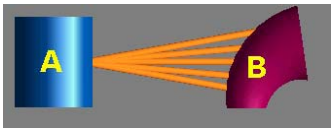
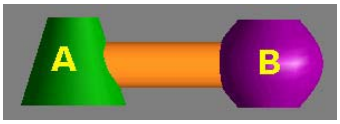
Generalization 	Objects of the same kind can be represented using ***same shaped primitives***. A ***is-a*** B (GCR1).
Dependency 	This relationship is best represented by showing the dependent ***on-top-of*** the depended. A ***depends-on*** B (GCR2).
Aggregation 	To show that one entity is composed of another can be shown using ***containment.*** B ***is contained in*** A (GCR3).
Multiplicity 	This attribute of a relationship can be depicted using ***multiple connecting lines*** (GCR4).
Relationship Strength 	***Relatively thicker connections*** can be used to show stronger relationships (GCR5).

Figure 4. Perceptual notation for software modeling semantics based on "geon correspondence rules".

We should note here that the emphasis of the geon diagram syntax is mostly placed on the "correspondence rules" described above and not simply on its $2^{1/2}$D representation. The additional dimension of the geon diagram is not exploited for enriching the display by superimposing 2D diagrams such as in [3] or for combining dynamic and static representation such as in [13], but instead for highlighting the connectivity between the different entities in a diagram.

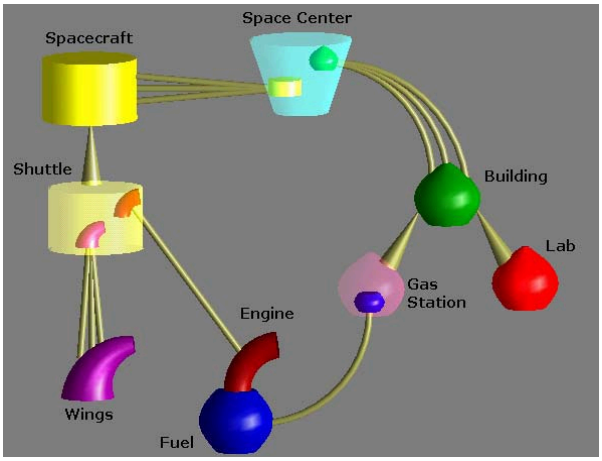


Figure 5. Representing some related entities in a system describing a Space Center.

In our previous study we compared the comprehension of the perceptual syntax to that of UML class diagrams containing the chosen semantics. Subjects unfamiliar with either notation were asked to describe geon and UML diagrams based on multiple-choice answers. The subjects made almost five times as many errors in identifying the relationships in the UML diagram (53.6%) in comparison to the geon diagram (11.5%) [6].

While our previous results showed that the geon correspondence rules were easier to recall and more intuitive than UML correspondence rules, they did not say anything about their ability to help people match a diagram to a problem domain. The remainder of this paper describes the experiment that we conducted to compare how easily students could learn and apply our diagram semantics compared to UML diagram semantics. To find out if our correspondence rules helped at this level, we required subjects to match diagrams to informally written descriptions of sample problem models.

5. Experiment

We hypothesized that it should be possible to learn, recall and apply diagram semantics, more accurately to diagrams created with the perceptual notation presented above in comparison with an equivalent UML graphical notation. We measured the error rate for subjects attempting to match diagrams to informal written descriptions of a set of entities and their relationships modeling various real-world problem domains.

5.1 Method for the Experiment

Diagrams. Five problem descriptions incorporating the semantics of generalization, dependency, one-to-many, and aggregation were constructed. The semantics in the problems were clearly presented using their common terminology. For example, to describe related entities of a neighborhood the following text was provided: *"The neighborhood **depends** on the city for clean water, sewage and garbage disposal. **Many** families live in this neighborhood. The neighborhood **has a** school and a pharmacy. It also has several **types** of stores: a grocery store, a convenience store, and a bakery."* The problem descriptions were created with a comparable number of relationships. For each problem description, a set of four UML diagrams and a set of their four equivalent geon counterparts were created. Only one of the four diagrams accurately depicted the relationships in the corresponding problem description. The remaining three diagrams mis-represented several relationships. All four diagrams in each set were randomly numbered from 1 to 4. The geon and UML diagrams corresponding to the problem description given above are shown in Figure 6.

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

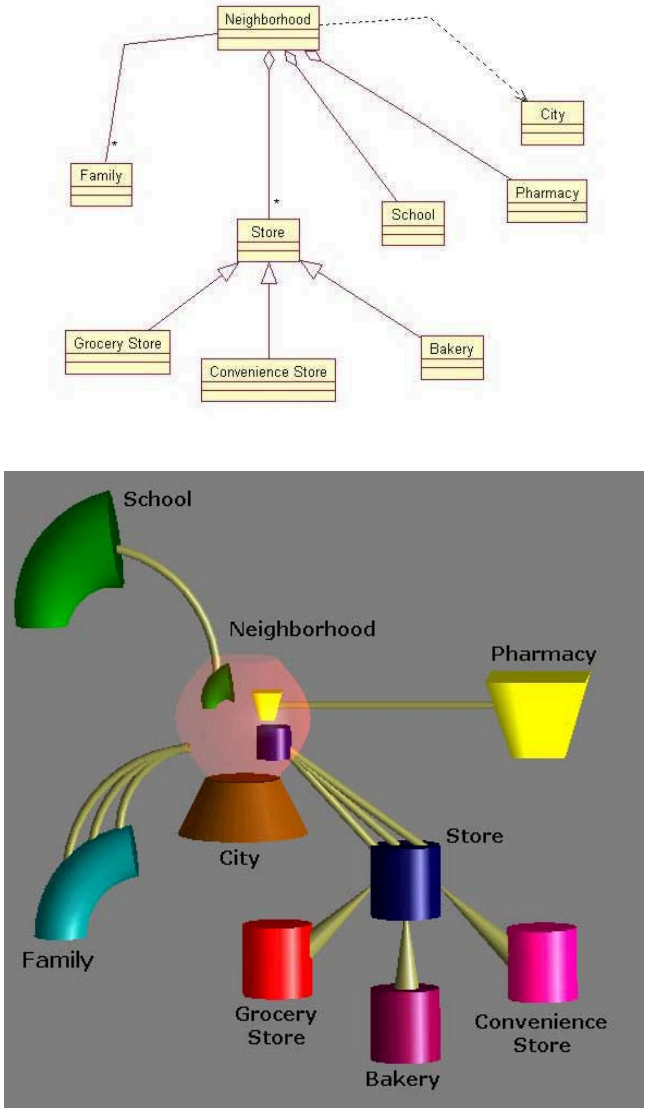


Figure 6. Sample UML and equivalent Geon diagram for representing entities of a neighborhood.

Training. All the subjects were collectively given an hour-long instruction on the various semantics and their respective notations. The training included an introduction to object oriented modeling, a description of each semantic with its UML and geon diagram notation, and sample UML and geon diagrams of complete systems with objects and their relationships. The emphasis during the training was placed on the concept underlying each semantic. One slide was dedicated to the description of each semantic and included the notation of both diagramming systems, first in UML and then in geon. Equal amount of time was spent describing each type of notation. There were 13 slides all together. Figure 7 is a sample slide used for describing the concept of inheritance. After describing each individual semantic and its associated notation, complete UML and geon diagrams illustrating the use of the semantics were presented. The subjects were asked to return a week later for the experiment. At the testing stage they were tested individually.

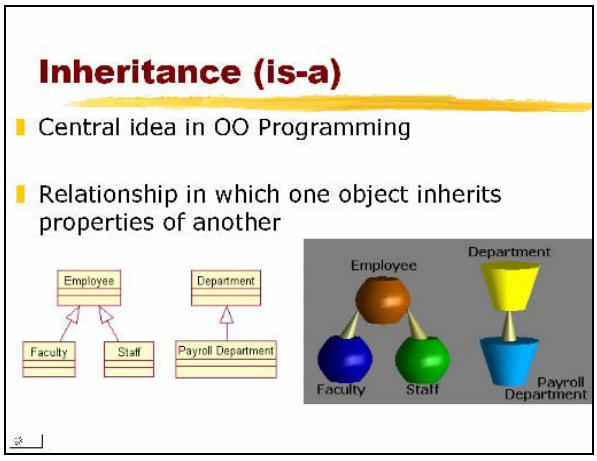


Figure 7. Sample slide used for explaining the concept of inheritance.

Task. For this experiment we used a diagram-to-problem matching paradigm. After reading each problem description, the subject was asked to match one of the four diagrams created for that problem. The subject marked on the hand-out sheet the number of the matching diagram. The problem descriptions were available to the subjects while reading the diagrams, and so they could occasionally consult the description. Therefore we were not testing subject memory of a given problem text.

Subjects were restricted to two minutes for matching a diagram to a problem description. A within-subject design was used where half the subjects matched the UML diagrams first and the other half matched the geon diagrams first.

Twenty-six paid volunteers participated in the experiment. Twelve subjects had previous exposure to UML diagrams through a third year software engineering course (experts) offered by the Faculty of Computer Science at the University of New Brunswick. The other fourteen subjects had never been exposed to UML or software engineering semantics (novices).

5.2 Results of the Experiment

Results are summarized in Table 1, which reports error rates by level of experience and type of diagram. The results are obtained by averaging each subject's scores. A One-Sample T-Test (or Sign Test) statistically shows that subjects performed better with the geon diagrams ($p < 0.0001$). A Two-Sample T-Test (or Mann-Whitney Test) shows that subjects' level of experience is not a relevant factor in comparing the performance between the two types of diagramming notation (p-values are 0.704 and 0.425 respectively).

Proceedings of the Eighth International Conference on Information Visualisation (IV'04)
1093-9547/04 $ 20.00 IEEE

	Geon	UML
Novices	22.3%	44.3%
Experts	2.9%	25.9%
Combined	14.6%	36.2%

Table 1. Average error rates of matching Geon and UML diagrams to problem descriptions.

The most striking result is the performance of expert subjects with respect to the interpretation of Geon diagrams (error rate 2.9%). This suggests that although subjects may be well versed in UML, they still preferred and performed better in interpreting the Geon diagrams. These subjects also performed better than the novice subjects with respect to both the Geon and UML scores. Overall, combining the results we can say that there were more than twice as many errors in analyzing and matching the UML diagrams than the Geon diagrams.

6. Conclusion

In this paper, we described an experiment that evaluated the learnability of software engineering (in particular object-oriented) concepts with diagrams created using perceptual structures. These diagrams were created using a set of "geon correspondence rules" that were constructed from deeply ingrained perceptual structures used in structural object recognition [ ]. In this paper we mapped the semantics used for object-oriented class modeling onto the set of "geon correspondence rules" and compared their learning expressiveness to that of the *de facto* standard diagramming notations provided in UML.

The results obtained from the experiment described here, show that the mapping of software engineering semantics onto "geon correspondence rules" can be used as guidelines for making effective diagrams. In particular we see that the geon diagrams are well suited for learning a subset of object-oriented concepts such as those necessary for modeling class structures. In comparing the learnability of matching problems to diagrams, we found that subjects, regardless of their experience in software modeling, were capable of learning and applying the perceptual syntax with fewer errors. The results were particularly significant in showing that with very little training, experts (subjects experienced only with UML diagrams and semantics through a software engineering course at the university) performed better with the geon diagrams. The use of a diagrammatic notation that requires minimal training may be particularly useful in instances where end users are involved in the development process and therefore need to quickly learn the diagrammatic notations.

One apparent drawback of the geon diagrams is the amount of space required by the geon notation compared to that required by the more conventional UML diagrams. However, in pedagogic environments, the educator has control over the size of diagrams, and typically a diagram used for elucidating a given concept does not require a high level of complexity. In certain cases the more complex the diagram, the harder it might be for grasping the particular concept being taught. Therefore the geon notation is well suited for small sized problems, such as those used for teaching a specific set of concepts, for example, design patterns.

It is worth noting, that we cannot claim to have shown that the learnability of geon diagrams is necessarily due to an innate grammar of perception (although we regard this as plausible). Alternatively, it may be that we have acquired an ability to interpret structured objects through years of perceptual interpretation of the world. The ease of learning the geon diagramming notation might come from transfer of this learned visual syntax and semantics, not from anything that is innate. In either case, our results provide a strong argument that perception-based diagrams are easier to learn. The experiment described in this paper focused on only one aspect of learning, i.e. matching problem descriptions to diagrams. While this constitutes a justifiable starting point for this line of research, further experimentation needs to be conducted in order to determine whether the geon notation can facilitate the process of software modeling by allowing the user to create proper abstractions of a problem. Our effort to elucidate a larger set of geon correspondence rules for other semantic concepts and discover which are most effective is part of an ongoing project to find better ways of integrating diagrams into the process of problem solving.

Finally, beyond investigating the use of geon diagrams for modeling problems for deriving the proper software class structures, the methods applied in this study for developing the notations and evaluating the learnability of their concepts will be extended to other software engineering semantics. In particular, we are interested in developing perceptual notations for the class of behavioral diagrams in UML. This will involve extending the geon notation to include dynamic (animated) representations and developing notations for semantics such as sequences, state transitions, and flow of activity. We hope that these notations can then be integrated into existing modeling environments that can then be used for facilitating pedagogy in software engineering.

7. Acknowledgements

The first author gratefully acknowledges the support of an NSERC PGS-B award. Discussions with Jane Fritz have helped focus some of the ideas. The authors are thankful to Maureen Tingley for her help with the statistical analysis of the data.

8. References

[1] Biederman, I., Recognition-by-Components: A Theory of Human Image Understanding, Psychological Review, 94:2, 115-147, 1987.

[2] Chomsky, N., Aspects of the theory of syntax, Cambridge, Mass: MIT Press, 1965.

[3] Gil, Y. and Kent, S., Three Dimensional Software Modelling, In Proceedings of ICSE98, IEEE Press, September 1998.

[4] Hornstein, N., and Lightfoot, D., Explanation in Linguistics, Longmans, London, 9-31, 1981.

[5] Irani, P., and Ware, C., Diagramming Information Structures using 3D Perceptual Primitives, ACM Transactions on Computer Human-Interaction, 10:1, 2003.

[6] Irani, P., Ware, C. and Tingley, M., Using Perceptual Syntax to Enhance Semantic Content in Diagrams, IEEE Computer Graphics & Applications, 21:5, 76-85, 2001.

[7] Jackendoff, R., On Beyond Zebra: The relation of linguistic and visual information, Cognition, 26, 89-114, 1987.

[8] Kosslyn, S. M., Elements of Graph Design, W.H. Freeman, New York, 1994.

[9] Marr, D., Vision: A computational investigation into the human representation and processing of visual information, San Fransisco, CA: Freeman, 1982.

[10] Object Management Group, Unified Modeling Language (UML™), version 1.4, September 2001.

[11] Pinker, S., The Language Instinct, New York: William Morrow, 1994.

[12] Pullum, G., Learnability, hyperlearning, and the poverty of the stimulus. Jan Johnson, Matthew L. Juge, and Jeri L. Moxley (eds.), Proceedings of the 22nd Annual Meeting: General Session and Parasession on the Role of Learnability in Grammatical Theory, 498-513. Berkeley, California. 1996.

[13] Radfelder, O. and Gogolla, M., On better understanding UML diagrams through interactive three-dimensional visualization and animation, Advanced Visual Interfaces, 292-295, Palermo, Italy, May 2000.

[14] Tversky, B., Kugelmass, S. and Winter, A., Cross-cultural and developmental trends in graphic productions. Cognitive Psychology. 23, 515-557, 1991.

[15] Ware, C., Information Visualization: Perception for Design, Morgan Kaufmann, 2000.

[16] Zacks, J. and Tversky, B., Bars and lines: A study of graphic communication, Memory and Cognition, 27, 1073-1079, 1999.