Transforming Descriptions and Diagrams to Sketches in Information System Design

Barbara Tversky[1], James E. Corter[1], Jeffrey V. Nickerson[2], Doris Zahner[2], and Yun Jin Rho[1]

[1] Teachers College, Columbia University
[2] Stevens Institute of Technology
bt2158@columbia.edu, corter@tc.edu, jnickerson@stevens.edu, dzahner@stevens.edu, yjr2101@columbia.edu

Abstract. Sketching is integral to information systems design. Designers need to become fluent in translating verbal descriptions of systems to a variety of kinds of sketches, notably sequential and logical, and to translate among the kinds. Here, we investigated these cognitive skills in design students, asking them to design a system configuration starting from either a sequential diagram or a sequential description. Although the two source descriptions were logically equivalent, the diagram led to designs that corresponded more closely to the source description – that is, designs with fewer omissions of crucial components and links. Text descriptions led to more variable and less accurate designs, most likely because they require more cognitive steps from problem representation to problem solution.

Keywords: Diagrammatic reasoning, sketches, descriptions, problem representation, information systems design, topological diagrams.

1 Introduction

Sketching is critical to design, whether for design of the concrete, products or buildings, or for design of the abstract, information systems or corporate structures. In designing, formative sketches show the components, walls, for example, or computers, and their relations, spatial, functional, or temporal. In emphasizing the components and their relations, sketches abstract out information not needed in early stages of design such as the surface of the walls or the specifics of the servers. One common task of designers, especially when working with clients, is translating a verbal description of the requirements for a building or an information management task into a sketch. On other occasions, designers need to translate one kind of sketch into another, for example, transforming a plan into an elevation in the case of architecture, changing perspective, or transforming a temporal sketch of information transmission into a structural sketch of the components in the case of information design, changing time to space. For design of information systems, the focus here, the design task is made more challenging by the fact that it is the topology of the system that is critical, not the Euclidean properties. The topology reflects the functional organization of the system. This fact causes difficulty for novices, since sketches are inherently Euclidean, and beginning designers bring Euclidean assumptions and habits to the situation.

Designers must master many types of external representations. Often a client will give the requirements to the designer verbally, and the designer will present the initial design as a sketch. More conversation occurs, followed by more sketches. Designers must be facile translators to and from diagrammatic representations. For the design of information systems, there are two general types of requirements, sequential and structural. Sequential requirements are step-by-step constraints, for example, the steps from a customer's request to the shipment of an item or from placing an order with a supplier to the arrival of the goods. Structural requirements fulfill a large set of sequential requirements, insuring that the right information gets to the right sources efficiently, and that the wrong information does not get to the wrong sources.

Because information systems involve links that function near the speed of light, the Euclidean distance between components is usually not important. Instead, the number of connections between system components and the nature of those connections are critical. The components – computers, servers, and the like – and their connections or links represent the functional structure of the system. Systems are represented topologically; the components, called actors, as nodes and the connections or links as edges. The placement of the nodes is of little importance, but the patterns of edge connection are of high importance because they carry the functional structure. Because information systems can have many interconnections, diagrams of them have developed conventions to increase transparency by lowering clutter. One convention in particular, the logical bus, reduces the number of edges that need to be drawn, yielding a connectivity pattern that is often misinterpreted by novices. This convention is widely used to represent Local Area Networks (LANs). Within a LAN all system components are interconnected, however those interconnections are not explicitly shown, but rather indicated by a superordinate line connecting all of them, as shown in Fig. 5c.

These abstractions are not always well understood by the users of technology. Nor are they well understood by novice designers [1]. Experts are expected to be able to understand the deep functional structure of a system, that is, its underlying topology, from many different forms of surface representations, including sequential diagrams, structural diagrams, natural language descriptions, and source code. They are also expected to be adept at going from one surface representation to another.

What are the consequences of translating from one external representation to another? It is well known that the form of an external representation affects interpretation and inference from it; think of doing multiplication with Roman numerals (e. g., [2][3]). The present focus is a common task faced by designers, translating one form of external representation to another, notably text to sketch and diagram to sketch.

Here, students in a course in system design were asked to sketch the configuration of two systems. In one case, the source for the design of the system structure was a diagram of a sequence of operations the system should support. In the other case, the source for the design of the structure of a system was text, a verbal description of the same sequence of operations the system should support. Two questions are of interest: Which source leads to better design sketches, and are there qualitative differences between sketches produced from diagrams and sketches produced from text? Especially for novices, there should be advantages to producing a sketch from a diagram, that is, going from a sequential diagram to a structural design sketch. The source sequential diagram will have already abstracted all the critical components of the system. In addition, because the source sequential diagram shows one possible temporal route through the information system, it also shows at least part of the connectivity, the topological relations. A verbal description of the sequence requires the design student to abstract the components and the links. Novice designers are likely to miss some of them. Because verbal descriptions have fewer explicit components and spatial constraints, they are likely to lead to more variable design sketches.

The source, whether diagram or text, specifies one sequence of operations to be accomplished by the system, but not all of them. Discussions with clients often begin that way; the client outlines the major task the system is expected to accomplish. That sequence does not completely specify the configuration. In order to construct a complete system, the designer must bring in other considerations deriving from general knowledge of information systems. For example, experienced designers might realize that certain groups of components should be grouped as a LAN, with restricted access. There are two ways that student designs could deviate from the minimal constraints of the source: they could omit nodes or edges, or they could add them. Omissions are always errors. Because the source problem does not completely stipulate the design, additions could be errors or they could be creative or wise design considerations.

The source representation is also expected to affect the actual spatial layout of the design sketch, which is not specified by the source. Because systems design depends on connectivity, the locations of components in the Euclidean world are not relevant; consequently, the spatial locations of components in the sketches need not reflect their locations in the world. One default for determining sketch location in the absence of Euclidean constraints is reading order (e. g., [1][4]). Reading order predicts that the layout of components in sketches where the source is text should correspond to the order of mention in the text, and that the layout of components in sketches where the source is a sequence diagram should correspond to the left-right order of the sequence diagram.

A challenge for research on sketches is to develop a coding system, because there is so much variability in spontaneous productions. Coding sketches is especially challenging for topological sketches, where it is desirable to extract the underlying logical structure from sketches that may differ in only superficial ways. Once this logical structure is abstracted, it becomes possible to establish the equivalence of diagrams, and to separately measure the surface and the logical dissimilarity of diagrams. To these ends, we developed a technique for establishing logical equivalences as well as similarities between different systems sketches and diagrams. This involved the construction of a *canonical graph*, the nodes and links that correspond to the minimal solution that incorporates the source constraints. The student solutions can be compared to the canonical graph to assess omissions and additions.

Since topological diagrams are used not only in information systems, but also in such diverse fields as electrical systems, transportation systems, systems biology, and geography (c.f. [6][7][8]), the coding system and insights into the production and understanding of information systems sketches will have broader implications.

2 Methods

The predictions were tested in a Master's level course in the design of information systems. The thirty-six student participants had varying levels of expertise: some were relative newcomers to information systems, and others were working professionals with many years of system design experience. During the course, students solve a series of increasingly complex design problems and bring them in for critique every week [9].

Two problems were given to students in the present experiment. Each student sketched the configuration design for two information systems, using a sequential diagram as the source for one problem and using a text description as the source for the other problem. The problems were chosen to involve the same number of nodes (actors in UML terminology [5]), but to be in different domains. One problem, called "FastStuff," asked students to design a system that delivers purchased products to the customer's door. The other problem, called "HedgeFund," asked students to design a new Internet presence for hedge fund investors (see Figures 1 and 2). A source diagram and a source text were developed for each of the problems. These different

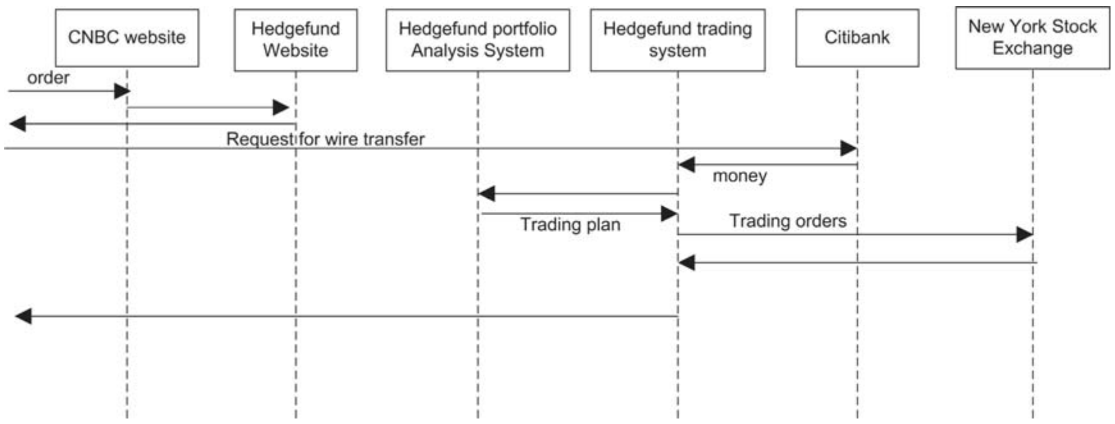


Fig. 1. HedgeFund Diagram, given to half the participants, and HedgeFund text, given to the other half. The diagram is in the Unified Modeling Language (UML) sequence diagram format [5].

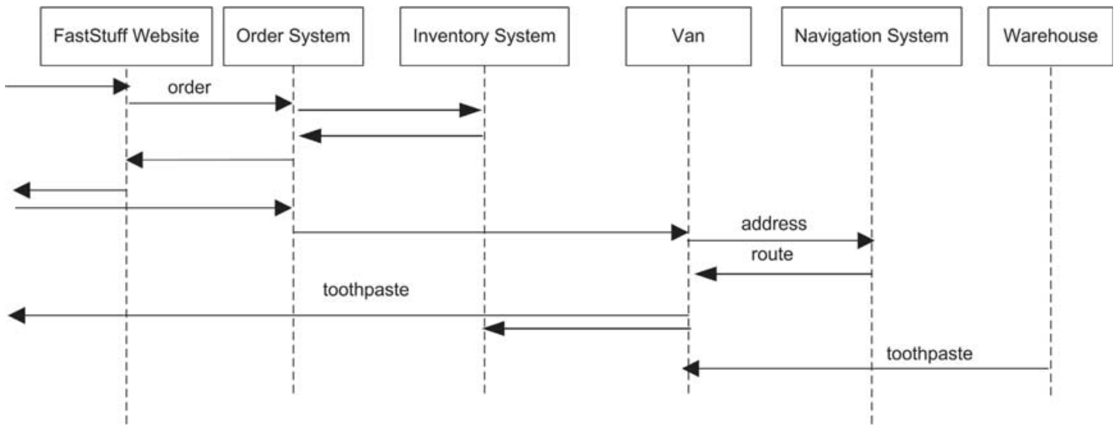


Fig. 2. FastStuff Diagram, given to half the participants, and FastStuff text, given to the other half

problem representations are only approximately equivalent. Complete equivalence would have resulted in either stilted text or unconventional diagrams, because the two types of representation have differing conventions. For example, sequence diagrams have an implied user on the left of the diagram that is never labeled, while text descriptions of design problems usually explicitly mention the user. Textual representations will often provide semantic information on the nature of an interaction (describing the type of message transmitted), but may not say explicitly where the interaction originates. However, this implicit information can usually be derived from the context. In contrast, sequence diagrams show all interactions explicitly, but do not give semantically meaningfully labels to the type of message transmitted. Rather than try to exactly equate the types of information provided by the two problem formats, we chose to use naturalistic problem descriptions that resemble those that students would encounter in a design course or in actual work settings.

Half the students were given the text source for FastStuff and the diagram source for HedgeFund; the other half were given the diagram source for FastStuff and the text source for HedgeFund. Because of the small sample size, all students first used a diagram source and then a text source. If there is any consequential bias on the results from solving a problem presented as a diagram first and a problem presented as text second, the additional experience should favor text as source.

3 Results

3.1 Omissions and Additions

Both the sequence diagram and the text versions of each problem explicitly define a set of actors (nodes in the diagram) and a set of interactions (edges). Thus, adequate performance in each problem involves constructing a labeled graph to represent these actors and interactions. The graph that represents all and only these explicitly defined actors and relationships will be termed the *canonical graph*. The distance between two graphs can be defined as the edits (additions or deletions) that will transform one graph into another [10]. In this paper, cases of leaving out any explicitly mentioned actors or communication links are termed *omissions*, and regarded as errors.

However, these design problems also offer opportunities to be creative, to go beyond the problem information by envisioning additional actors and relationships, additional system capabilities that might be beneficial, and by optimizing system performance in other ways. Thus, additional nodes and edges not explicitly stipulated in the problem statement are not necessarily errors. Rather they may reflect more or less relevant elaborations to the conception of the problem or the solution. However, irrelevant additions may be considered to be a form of error, because they have costs (time, expense, security concerns) and do not bring benefits.

In addition to omissions and additions, the spatial layout of the produced sketch is of interest: does the left-right organization of its components correspond to the order of presentation in the text or the left-right organization of the sequence diagram?

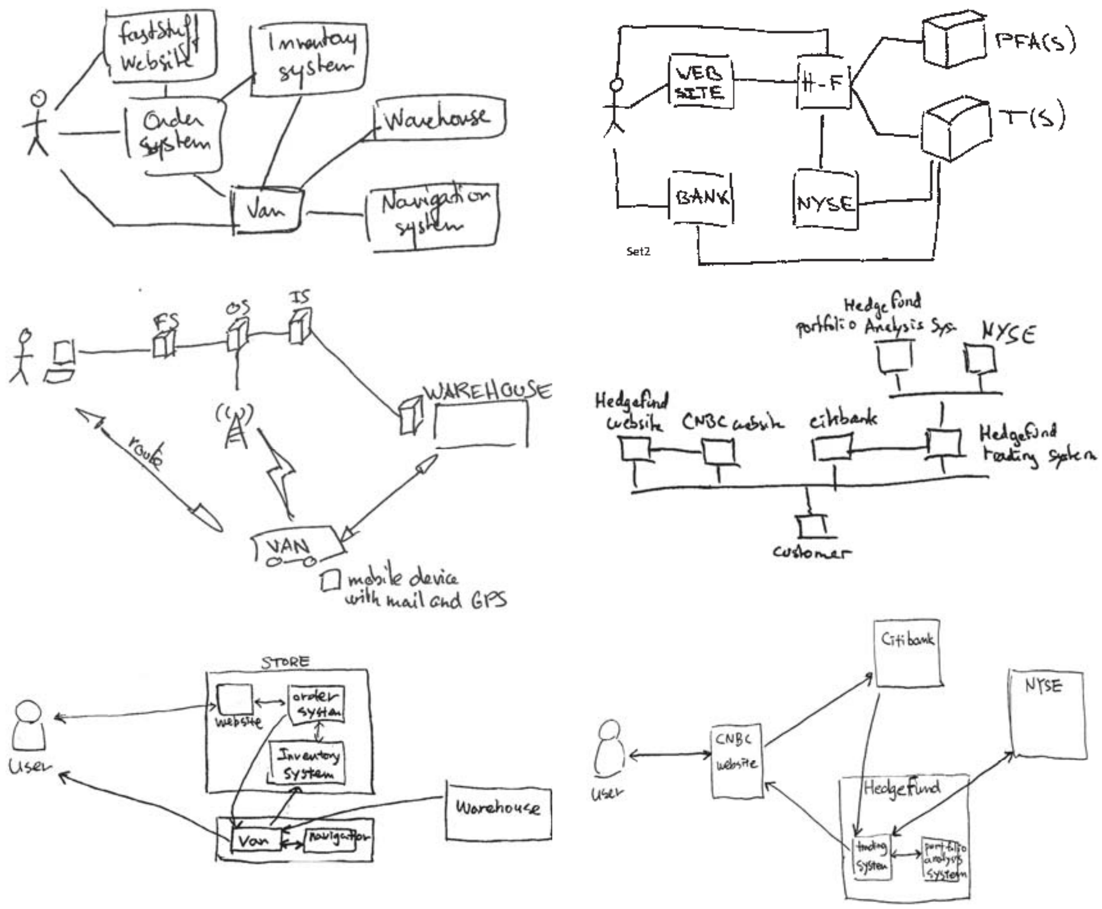


Fig. 3. Example student sketches for the FastStuff (left) and HedgeFund (right) problems

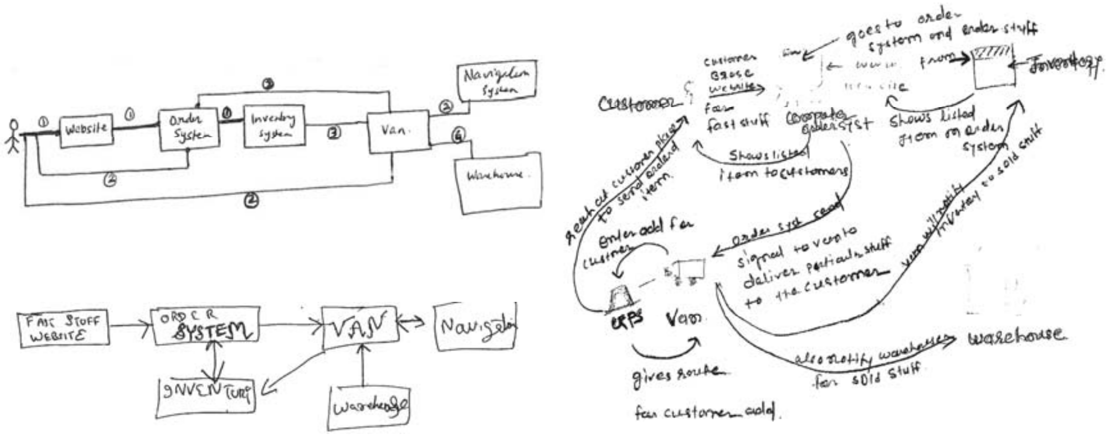

Website
Order System
Inventory System
Van.
Navigation System
Warehouse
FAST STUFF WEBSITE
ORDER SYSTEM
VAN
Navigation
INVENTORY
Warehouse
Customer
goes to order system and order stuff
Inventory
Shows listed item to customers
shows listed item on order system
order syst send signal to van to deliver particular stuff to the customer
ERPS
Van
gives route for customer add
warehouse
also notify warehouses for sold stuff

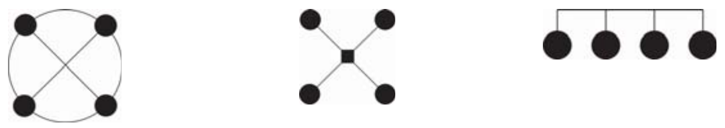
Fig. 4. Sketches generated for the FastStuff problem that are highly similar with respect to Euclidean node positions and other surface characteristics (the two on the left), with a third sketch (right) that is very dissimilar. In contrast, with respect to logical structure the sketch on the right is almost identical to the sketch at top left.

To evaluate produced designs, we first analyzed the topological information provided to students in the form of a sequence diagram or text for both problems. For both problems, the text and diagram versions provide identical information about actors (nodes) and their pattern of connectivity. We call the logical graph that can be drawn for each of the problems, shown in Figures 1 and 2, the *canonical graph* of the problem, because this standardized representation of the problem is analogous to canonical data models used in systems integration (e. g. [11]).

We then developed a scheme for coding the student-produced sketches. Some examples of student-produced sketches appear in Fig. 3. It is obvious that the diagrams vary in their surface details, but without a formal analysis, it is less obvious if they differ in their logical structure. In order to test the predictions, we need a method to compare the student-produced sketches at the logical level. This requires coding the sketches in terms of their graph topology.

As an illustration of the coding issues, we show three more sketches in Fig. 4. In terms of surface structure, the two sketches on the left are very similar to each other. In fact, they are the two closest sketches in the data set as measured by the Euclidean distances between corresponding nodes in the sketches. For example, the position of the node *Van* on one sketch is compared with the position of the node *Van* on the other sketch. The sketches on the left are very dissimilar to the sketch on the right. But by analyzing the logical connections between nodes (the *edges* in the terminology of graph theory), we find that the two sketches on the left are quite dissimilar to each other with respect to logical structure, and the sketch on the top left is very similar to the sketch on the right.

However, analyzing the logical structure of the sketched diagrams involves more than simply coding the student-produced diagrams for their graph topology. Specifically, the use of diagrammatic conventions, such as sketching a logical bus to represent a LAN, means that logical connectivity in the system and graph topology do not have a one-to-one correspondence. Thus, we recoded various diagrammatic



a) Complete Graph b) Hub and Spokes c) Logical bus

Fig. 5. Alternative network representations topologically equivalent at the logical level

devices used to represent networks, such as logical buses, the Internet, and satellite links, in terms of the network connectivity implied by the device and its constituents. This was done in the following way. First, all examples of use of LANs and other types of networks were identified. Two people coded these network types, and discussed any discrepancies until consensus was achieved regarding the presence and type of network connections in each student solution. The next step in the process was automatic – the graph representation of the problem was transformed by software we wrote into a standardized form. All devices directly connected to LANs were assumed to be directly connected to each other, and the resulting connections were represented in the *logical form* of the graph for each student solution.

For example, Fig. 5c shows the standard convention for representing a LAN. The convention means that all nodes can communicate to each other, and so is logically equivalent to the canonical form shown in Fig. 5a. To code this case, we formed a new graph, G', in which the edges implied by the networks were added, and the depicted node (if any) representing the network was deleted. We call G' the logical form of the graph G, or the *logical graph* for short. In this way, we can compare the logical connections in the diagrams produced for a particular problem, regardless of the student's choice of convention.

Several students created sketches whose logical graph exactly matched the canonical graph. Most students, however, either omitted edges or added new edges (sometimes as a result of adding new nodes). We analyzed the differences between the students' logical graphs and the canonical graph.

For the FastStuff problem, five graphs were produced that matched exactly the canonical graph. One of these solutions is depicted in Fig. 3 (top left), the other four

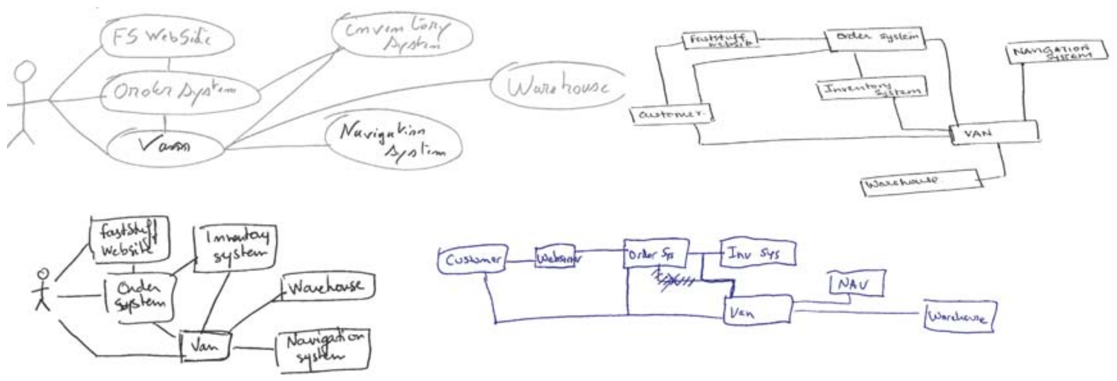


Fig. 6. The four additional graphs that (along with the top left graph shown in Fig. 3) are topologically identical to the canonical graph for the FastStuff question

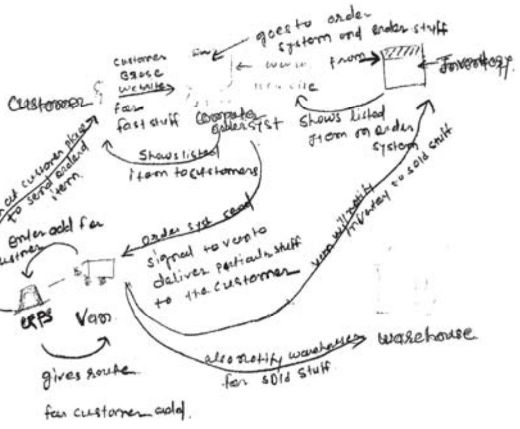

Customer
customer Gross website
for fast stuff
Computer
order syst
goes to order system and order stuff from
Inventory
shows listed item to customers
shows listed item on order system
order syst send signal to van to deliver particular stuff to the customer
enter add for customer
GPS
Van
gives route for customer add.
also notify warehouses for sold stuff
warehouse

Fig. 7. A graph that is logically almost identical to those in Fig. 6 – an edge has been added between the GPS device and the customer

are shown in Fig. 6. Remarkably, these five matching graphs were all drawn in the Diagram condition. Notice that very different styles are used to represent the same underlying structure. The graphs exhibit different degrees of linearity, with the bottom right graph mapping closely to the positions of the nodes shown in the problem diagram, and the top right graph exhibiting a weaker correlation with the positions of nodes in the problem diagram.

Fig. 7 shows a diagram that is almost identical to those in Fig. 6. An edge has been added from the GPS system to the customer. This addition is a creative idea: the customer could then know exactly where the truck was at all times. The student did not consider a better alternative, that the information could be transmitted through the truck's network back to the website so that customers could track the truck's location without overloading the GPS system.

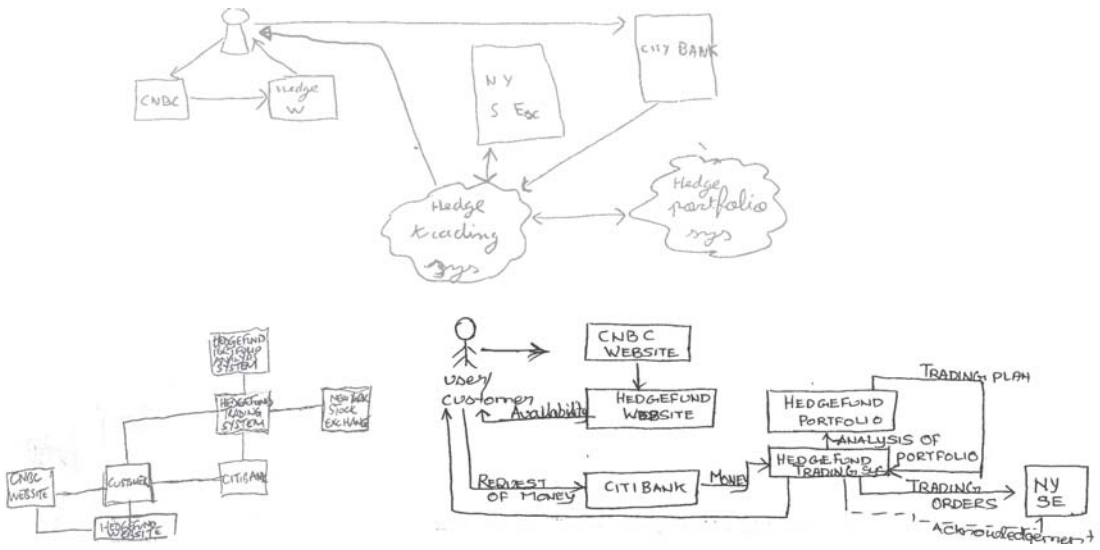


Fig. 8. Sketches that match the canonical graph for the HedgeFund problem

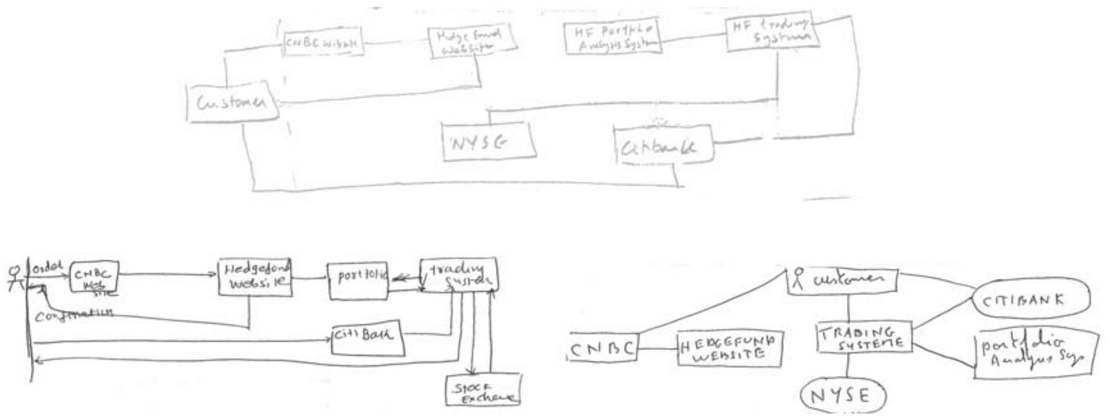


Fig. 9. Graphs that are one edit distant from the HedgeFund canonical graph. The graphs are not identical to each other. There is an extra edge on the diagram on the left, and missing edges in the two other diagrams.

For the HedgeFund problem, students produced three sketches whose logical graphs matched the canonical graph; these are shown in Fig. 8. Again, these all were drawn in the Diagram condition. Thus, for this problem too, no student with text as source created a graph that matched the canonical graph. There were three figures that differed by one edge from the canonical graph; these are shown in Fig. 9.

The sketch coding scheme just described allows comparisons of sketches produced from diagram and text sources for omissions and additions. The number of omitted edges (relative to the canonical graph) was analyzed in a replicated 2 x 2 Latin Square

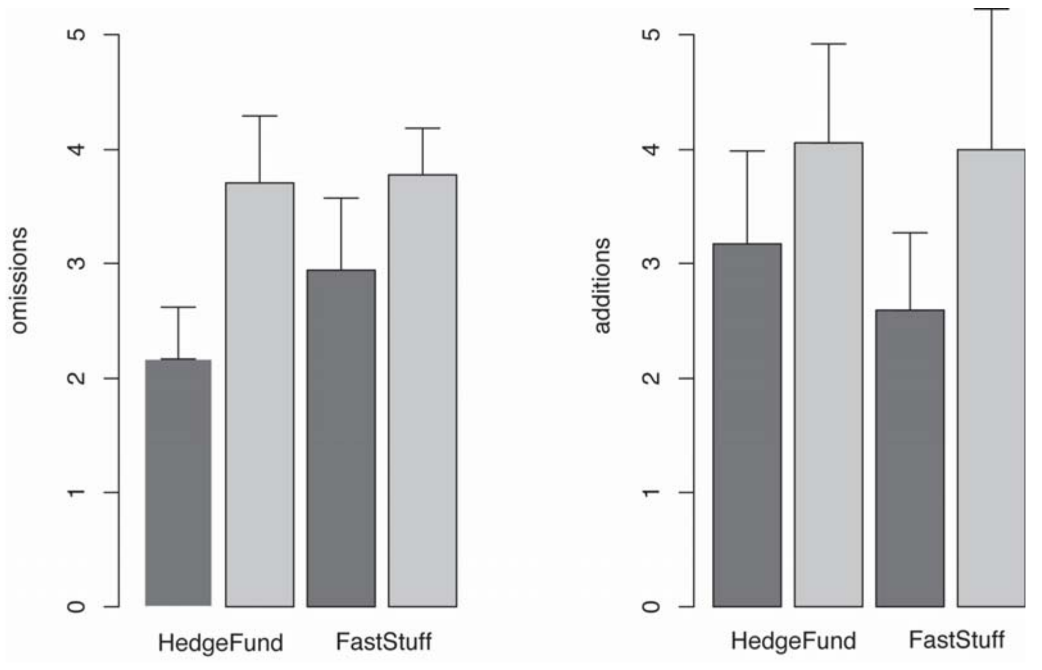
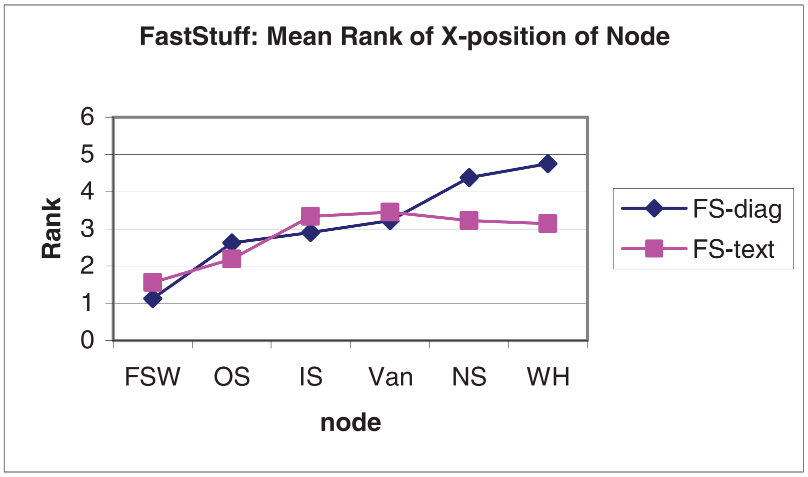


Fig. 10. Number of omissions and additions by Modality and Problem. The dark bars represent sketches drawn from diagrams, the light bars sketches drawn from text. The error bars show standard error.



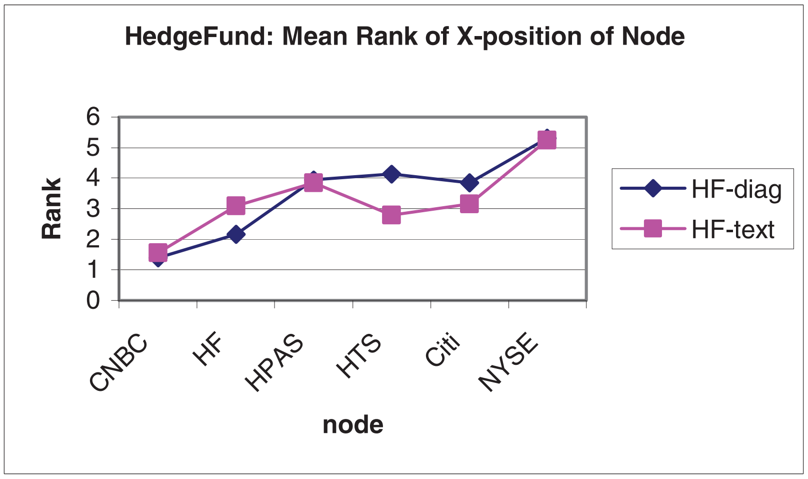


Fig. 11. Rank X-axis positions of the objects of Figures 1 and 2 for the two problem scenarios

design, with Condition as the between-subjects factor. The within-subjects factors were Problem (FastStuff or HedgeFund) and problem Format (diagram or text). Means and standard errors for the four conditions are shown in Fig. 10.

There were fewer sketch errors with a diagram source than a text source. For omissions, this was significant by a repeated-measures ANOVA, F(1,33) = 14.37, p=.001. The effects of Problem and Condition were not significant, F(1,33) = 1.82, p=.186 and F(1,33) = 0.27, p=.605. For additions, the differences due to Source were not significant, F(1,33) = 2.64, p=.114, nor were the effects of Problem and Condition, F(1,33) = 2.64, p=.656 and F(1,33) = 0.05, p=.816.

3.2 Reading Order Bias

Because there were no constraints on the positions of objects in the sketches, the left-right organization of components was expected to correspond to the reading order of the source text or the left-right organization of the source diagram. To test for reading order bias, we first coded the horizontal locations of all nodes in the source diagrams shown in Figures 1 and 2, and coded the locations in the problem texts of the same objects, treating the text as a continuous string. Next, we coded the vertical and horizontal locations of each node in the sketches produced by students. Then we compared the positions of objects in each sketch to the reading order of the source, whether diagram or text (Fig. 11).

As is evident from Fig. 11, there was a reading order bias. The X-axis of the figure shows the rank-order left-right position of each node in the problem description and the Y-axis shows the rank-order horizontal position of each node in the student's sketch for those students who depicted all six actors explicitly mentioned in the problem text. For the FastStuff problem, when the source was a diagram the order of nodes in the students' sketches was exactly the same as the order in the source diagram. By a permutation test [12], this ordering can be shown to deviate significantly from random in the direction of the reading order, p=.001. When the source was text, the correspondence between the order of nodes in the sketch and the order of mention in the source text was nearly as strong, with only one pairwise inversion of node order compared to the reading order. This too differs from random ordering, p=.008.

The pattern was similar for the HedgeFund problem (Fig. 11), showing closer correspondence of source ordering to sketch ordering when the source was a diagram compared to when it was text. For the diagram presentation, the mean rank order of nodes in the sketch differed from the diagram order by only a single inversion, which again represents a significant degree of deviation from random ordering, p=.008. For the text condition, the mean rank order differed from the order of nodes in the text by three pair inversions. This correspondence with the reading order was only marginally greater than chance, p=.068. In both conditions, the deviation of node order in the design sketches differed from the source ordering mainly in the location of "Citibank." The source diagram located it to the right of HedgeFund, HTS (the HedgeFund trading system) and HPAS (HedgeFund's portfolio analysis system), but students' sketches locate it close to HedgeFund and to the left of HPAS. It could be argued that this is a better placement, because the HTS and perhaps the HPAS node need to communicate frequently with the New York Stock Exchange, the rightmost node, while the customer (at the extreme left in the sequence diagram) needs to communicate directly with Citibank.

4 Conclusions and Implications

Students in a class in information systems were presented two comparable design tasks and asked to produce sketches of their solutions. One problem was presented in the form of a diagram of a sequence of operations the system should perform. Another problem was presented as a description of the identical sequence. Thus, students were presented with the same information, but in different modes. We sought to investigate if the mode of the problem source, diagram or text, would affect students' designs. The results showed that the mode of the source for the design problem, diagram or sketch, in fact affected students' design sketches in two major ways. First, a diagram source led to designs that were more accurate in the sense of more closely matching the specified problem structure. That is, students omitted fewer explicitly mentioned actors and connections when designing from a diagram, probably because the source diagram preprocesses that information for the student. There is some indication that students also made fewer additions when the source was a diagram than when the source was text. Additions, however, are not necessarily erroneous; they may be creative and wise elaborations. Next, both the left-right spatial arrangement of actors in the source diagram and the order of mention of actors in the source text affected the left-right order of actors in students' design sketches. However, that correspondence was stronger when the source was a diagram than when the source was text. Together the findings indicate that students' design representations are more variable and more flexible when created from text than when created from diagrams. There are more mental steps to translate text to a design sketch than to translate a diagram to a design sketch. Each mental step provides an opportunity for error but also an opportunity for creativity. Thus, diagrammatic representations of design problems constrain design solutions more than verbal representations of design problems.

Students' design sketches are wonderfully variable, posing problems for data analysis. What's more, what is critical in system design is not the surface connections of the diagrams but the underlying functional connections, which are logical and topological, not Euclidean. In order to analyze and compare students' sketches, we developed a coding system that captured the underlying topology That coding system allowed us to count and characterize omissions and additions. This coding system should prove useful for many other cases of diagrams whose structure is topological, including electrical systems, transportation systems, systems biology, and geography.

Diagrammatic representations of design problems provide a scaffold for designers by selecting the relevant actors and specifying some of the relevant links. This scaffold serves as training wheels for beginning designers, making sure that they stay upright. However, the scaffold, like training wheels, also places limits and constraints. Verbal descriptions of design problems are freer of constraints, and are likely to lead to more flexible and creative performance in the long run. In any case, just as a bicyclist must eventually abandon the training wheels, designers must eventually become expert at translating diagrammatic and verbal representations of problems to good designs. And, in fact, it seems likely that the problem representations with fewer constraints will eventually lead to more flexible and creative designs. Ambiguity allows invention (e. g., [13]), but successful invention requires expertise, skills and knowledge.

Sketches are integral to design, of products, of buildings, and even of abstract information systems. They are used to translate clients' desires into initial designs, they are used by designers to articulate and revise design ideas, and they are used to present the design ideas as they progress for discussions with colleagues and clients. In information design, clients often specify a series of temporal steps; from that, the designer must configure a set of components. This is one of the challenges novice designers face in order to become expert, a challenge examined here. For systems designers, another challenge is appropriate use and interpretation of sketches and diagrams that are manifest in space but that do not always support Euclidean assumptions. The spatial configuration of components and connections, though highly salient in a sketch, does not carry useful information about functional relations. Rather, the information on functional relations is carried by the network connections among the components. In typical information systems, those connections are dense; expressing each and every one would quickly clutter a sketch, making it difficult to follow. To cope, systems designers have developed conventions that summarize a set of connections, notably, a logical bus to represent a LAN. A LAN is drawn as a single line from which a set of components hang like clothes on a clothesline, but its meaning is that all the hanging components are interconnected. Such sketches may bear similarities to familiar route maps, but their interpretation is quite different. In a LAN, to get from the left-most component to the right-most component, information does not have to go through the middle components; it goes directly. Using this and related conventions also causes problems for design students, because the conventions cannot be approached with the Euclidean assumptions people usually bring to bear in interpreting sketches. The difficulty this causes for design students was revealed in a previous study [1]. The difficulty was echoed in the current study: for example, one student had problems sketching a LAN connected to the Internet when working from text, even though the student drew a correct network when working from a diagram. Thus, working from diagrams can provide support for beginning designers, but also can mask conceptual difficulties from instructors.

Designers of information systems must learn to resist Euclidean interpretations of sketches and become fluent in the conventions. They also must become adept at going back and forth between ideas and sketches, and between two kinds of sketches, temporal and logical. Design, of products, buildings, and systems, is a cognitive activity, and successful design entails honing cognitive skills. Some of these skills require learning to benefit from different external representations, to translate among them, and to use each to its advantage.

Acknowledgments. We are grateful for the support of the National Science Foundation under grant IIS-0725223 as well as to NSF REC-0440103 and the Stanford Regional Visualization and Analysis Center.

References

1. Nickerson, J.V., Corter, J.E., Tversky, B., Zahner, D., Rho, Y.: Diagrams as Tools in the Design of Information Systems. In: Third International Conference on Design Computing and Cognition (DCC 2008). LNCS, Springer, Berlin (2008)
2. Zhang, J., Norman, D.A.: A Representational Analysis of Numeration Systems. Cognition 57, 271–295 (1995)
3. Hayes, J.R., Simon, H.A.: Psychological Differences Among Problem Isomorphs. In: Castellan, N.J., Pisoni, D.B., Potts, G.R. (eds.) Cognitive Theory, vol. 2, pp. 21–41. Erlbaum, Hillsdale (1977)
4. Taylor, H.A., Tversky, B.: Descriptions and Depictions of Environments. Memory and Cognition 20, 483–496 (1992)
5. Fowler, M.: UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, Reading (2004)
6. Egenhofer, M., Franzosa, R.: Point-Set Topological Spatial Relations. International Journal of Geographical Information Systems 5, 161–174 (1991)
7. Egenhofer, M., Mark, D.: Naive Geography. In: Frank, A., Kuhn, W. (eds.) COSIT 1995. LNCS, vol. 988, pp. 1–15. Springer, Heidelberg (1995)
8. Stevens, A., Coupe, P.: Distortions in Judged Spatial Relations. Cognitive Psychology 10, 422–437 (1978)

9. Nickerson, J.V.: Teaching the Integration of Information Systems Technologies. IEEE Transactions on Education 49, 1–7 (2006)
10. Sanfeliu, A., Fu, K.S.: A Distance Measure Between Attributed Relational Graphs for Pattern Recognition. IEEE Trans. Systems, Man, and Cybernetics 13, 353–362 (1983)
11. Bergamaschi, S., Castano, S., Vincini, M.: Semantic Integration of Semistructured and Structured Data Sources. SIGMOD Rec. 28, 54–59 (1999)
12. Rosander, A.C.: The Use of Inversions as a Test of Random Order. Journal of the American Statistical Association 37, 352–358 (1942)
13. Suwa, M., Tversky, B.: What do Architects and Students Perceive in their Design Sketches? A Protocol Analysis. Design Studies 18, 385–403 (1997)