Sketching with Conceptual Metaphors to Explain Computational Processes

David G. Hendry
Information School
University of Washington, Seattle, WA 98195
dhendry@u.washington.edu

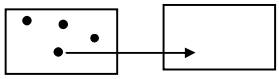
Abstract

To explore how people conceptualize a complex system, 232 university students were asked to sketch how a search engine works. While the sketches reveal a diverse range of visual and conceptual approaches, a subset of the sketches exhibit an underlying regularity for describing algorithmic processes. To explain this regularity, I propose the conceptual metaphor: A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS and describe a set of mappings from sketchable graphic markings to abstractions in the search engine domain. I believe that this metaphor can be applied to enable people to more effectively conceptualize, describe, and explore complex systems.

1. Introduction

Sketching with pencil and paper can promote various forms of communication with one's self and others. Sketching can be used to externalize working memory, giving the designer a greater capacity for solving problems [18]. It can promote rapid exploration, allowing the implications of a disciplined approach to solving a problem to be concretely examined for reflection and discussion [25]. Sketching can be used to capture one's evolving understanding of a system or can be used to outline a plan for implementing an algorithm [2, 24]. When informal or provisional in nature, sketches can enlist comments and they can support creative discussion [11, 12, 26]. When relatively more 'finished', sketches can be used to systematically examine the 'correctness' of a process or structure under study. A particular style of sketch can become a genre, carrying tacit information that is held by a group [6, 7]. Finally, sketches can be used to communicate with computational systems for improved design processes [5, 20, 23]. Sketches, in short, can play many roles within design.

Nevertheless, all of these examples depend, at least to some degree, on convention. Conventions for sketching and modeling can help establish and maintain the identity of a community of practice [10], and it seems that design teams often adapt conventions to their particular needs and proclivities. How do conventions emerge and become established? In balance with cultural and socio-technical factors are people's "natural" inclinations for how to represent concepts in sketches. Certainly, conventions for sketching and modeling are not arbitrary [27]. Consider, for example, this sketch:



The circles might represent *documents* and the rectangle might represent a *folder* to hold the documents. When a directed line is added, with its source touching a document and its destination lying within a second folder, we might ask: What is to happen to the document?

If this is a transformation of some kind, should it happen to all documents in the source folder or to only some of them? If the answer is only some of them, then how are they identified? Once the transformation is complete, what next happens to the document? How is the transformation specified? Do the documents from the source folder remain after the transformation is completed? And so on.

In this way, sketches allow us to reason in a relatively concrete fashion so that inferences can be made about a more abstract domain. In this case, the perceptual elements—dots, rectangles and a directed line—provide a conceptual metaphor [16, 17] for the relatively more abstract domain of documents, folders, and transformation.

How do people naturally use such externalizations to represent plans or outline computational processes? Rather than introduce a visual notation and study how it was learned and used, as many empirical studies have done [4], this study used an open-ended task [8, 21] where participants were prompted to "Draw a sketch that explains how a search engine, such as Google, works." The original aim of the study was diagnostic: To access students' understandings for search engines so that instruction for the fundamentals of Information Retrieval systems could be improved. Analysis of the data, however, showed that a significant number of sketches attempted to represent algorithmic processes, and in doing so these sketches seem to reveal interesting, and potentially useful, regularities. In this paper, I report on these regularities and give a cognitive

Visual Languages and Human-Centric Computing (VL-HCC'06)
0-7695-2586-5/06 $20.00

account of their "naturalness". This analysis leads to a conceptual metaphor that maps graphic markings, which are easily created in sketches, to concepts in a problem domain. This metaphor, in turn, might be used by people to conceptualize computational processes and plan for their implementation.

2. Background

2.1. Conceptual metaphor

A conceptual metaphor is a set of mappings from a relatively concrete domain to a more abstract domain [16]. Through these mappings, the more abstract domain becomes more readily understood. For example, here are three metaphors for making things:

1. "I made the database *out of* the document titles." (i.e., THE OBJECT [database] COMES OUT OF THE SUBSTANCE [document titles] metaphor).
2. "I put the document titles *into* the database." (i.e., THE SUBSTANCE [document titles] GOES INTO THE OBJECT [database] metaphor).
3. "The importance of the documents *came out* of their linking structure." (i.e., EMERGENCE metaphor).

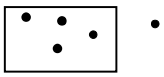
Conceptual metaphors have been found to be a fundamental cognitive mechanism and their use is unconscious and pervasive [16].

More recently, Lakoff and Núñez [17] have applied this approach to analyzing the conceptual structure of mathematics, aiming to decompose the layers of conceptual metaphor that enable mathematical reasoning. For example, the metaphor NUMBERS ARE POINTS ON A LINE is central to Euclidian geometry and trigonometry. But, one can also consider NUMBERS TO BE SETS with the empty set being 0, 1 being the set containing the empty set, 2 being the set containing 0 and 1, and so on. By identifying such metaphors and the specific mappings that exist between source and target domains, they claim that mathematics becomes more comprehensible. Ambitiously, their project proposes nothing less than an empirical basis for an embodied-mind theory of mathematics.

The analysis that follows relies on two concepts from this theory that are now briefly and informally introduced. The first concept is the Container schema, which consists of three distinctions: An inside, an outside, and a boundary. Conceptually, we can say, for example, "The document is *in* the database" or "The word is *in* the document". If we assume that these two statements are true, then by the spatial relation, *in*, we know that, all things equal, the word is in the database. The Container schema is a place to hold things.

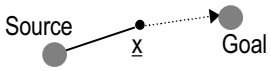
The second concept is the Source-Path-Goal schema, which consists of a source and goal locations, a trajectory that joins the source location with its goal, and a moving entity that is located on the trajectory. With these elements we can conceptualize progress towards a goal, along some path that begins at a source and ends at a goal. Like the word *in* the words *towards*, *along*, *begins* and *ends* are spatial relations that allow us to naturally infer information from the conceptual metaphor; for example, we can readily identify the places on the trajectory that have been visited and those that remain to be visited.

Importantly, these "Image schemas have a special cognitive function: They are both perceptual and conceptual in nature" [17, p. 31]. Thus, for example, in the following sketch of a Container schema we immediately see that four dots are in a container and one dot is outside of the container.



Similarly, with the Source-Path-Goal schema, we can see a source and a goal for a moving object. The dotted line indicates that part of the trajectory has yet to be realized and the arrow indicates the direction of the trajectory. If a moving object is at position x then we can conclude that all points between the Source and x have been visited and the points between x and the Goal have yet to be visited.

Lakoff and Núñez [15] claim that these cognitive structures are learned early in life in all cultures, and that people use them unconsciously in everyday reasoning and



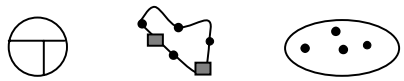
when reasoning about science and technology. It is for this reason that the example given in the introduction was so easily understood: It was a conceptual metaphor that blended the Container and Source-Path-Goal schemas.

2.2. Visual grammar

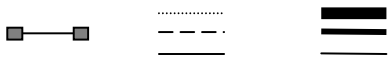
Ware [27, p. 214] uses the term *visual grammar* to characterize the regularities that are found across the majority of node-link diagrams, such as the UML modeling language, Petri nets, and so on. Based on established findings in perceptual psychology, he proposes 13 graphical codes, set in *italic* below, and corresponding meanings.

A *(1) closed contour* means an entity, object or node and the *(2) shape* and *(3) color* of such enclosed regions indicates a type of entity. The *(4) size of an enclosed region indicates* entity value. Examples:

By *(5) partitioning lines within an enclosed region* one can create within-entity structure, *(6) attached shapes* can indicate part-of relationships and *(7) shapes enclosed by a contour* indicate contained entities. Examples:



A *(8) linking line* means a relationship between entities exists, the *(9) linking-line quality* indicates the type of relationship, and the *(10) linking-line thickness* indicates the strength of the relationship. Examples:



Finally, *(11) spatially ordered shapes* can be used to indicate a sequence, *(12) tab connectors* can indicate a fit between components and *(13) proximity* can be used to indicate groups of components:

Together, these approaches seem to show a correspondence between "deep" linguistic and perceptual structures [14]. For example, the Container schema is represented by enclosing the shapes with a contour (7), and nested Container schemas can be represented by partitioning lines within an enclosed region (5). Proximity can be used to suggest containment without using a contour. Similarly, the Source-Path-Goal schema can be represented by a linking line (8). The other graphic codes can be used to enrich the Container and Source-Path-Goal schemas with additional information from the domain, thereby facilitating domain-specific reasoning. These approaches, in short, give us an analytic tool for recognizing and accounting for "naturalness" when people talk about and draw computational processes.

3. Research study

3.1. Aim and task

The original aim of the study was to investigate the technical knowledge that students have for how search engines work [9, 13]. As an instructor, I wanted to assess students' technical literacy in this area in order to improve classroom instruction.

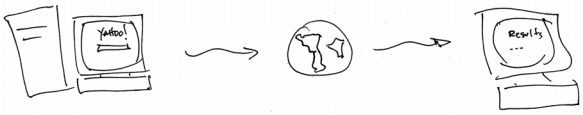
Students were thus prompted to sketch "how a search engine works" on 8 x 11 paper sheets during regularly scheduled classes. Participation was voluntary and anonymous. These sketches were used to promote classroom discussion and were also collected for more detailed analysis.

In the spring and autumn of 2003, a sample of 232 sketches from undergraduate and graduates students in Information Science was collected. The students reflect a great diversity in ages and educational backgrounds, with some students having relatively strong backgrounds in programming and system design and others having no background at all in programming.

To analyze the sketches, a normative model was created from textbook accounts of how search engines work [1, 19]. Using this normative model (see Appendix), five raters coded all 232 sketches for the presence or absence of the 14 concepts that make up the model. In summary, the mean number of concepts per sketch was about 4.5 (SD 3.0) with a low of 0 concepts (n = 25) and a high of 13 concepts (n = 2). The four most common concepts were *query* (72% of sketches), *results* (65%), *content* (55%), and *match* (52%). The full quantitative analysis is reported elsewhere [9, 13].

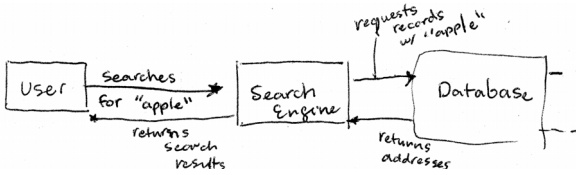
3.2. Qualitative analysis

The full sample of sketches reveals a tremendous diversity of approaches for explaining how search-engines work. Some of the sketches were largely representational, with little detail.



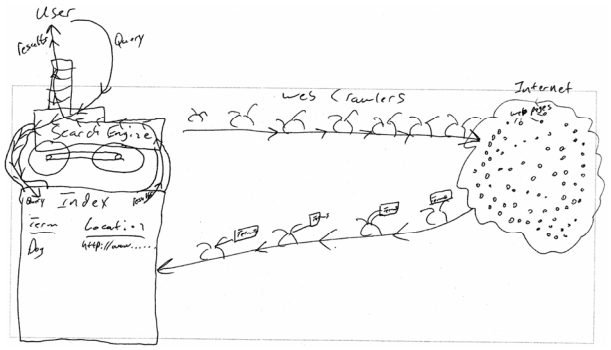
In this sketch, the query is depicted by a horizontal input box akin to Google's visual design (though the sketch uses the label Yahoo!). The query is sent into the world, somehow yielding Results. One might assume that the image of the world also depicts some large, diffuse matching process and the second display represents the first at a subsequent period of time. (Words excerpted from the sketches are underlined.)

A second class of sketches is largely systems-oriented, where box-and-line symbols are employed to depict key entities and processes. In this sketch the entities User, Search Engine, and Database are connected with directed communication pathways that are signaled by arrows and the words Searches for, requests, and returns.



In a third class of sketches, making up approximately 20% of the sample, search engines are explained by depicting computational process though more intricate visual structures, although none of these sketches used a formal notation such as UML.

Consider the following sketch.

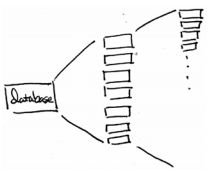


Beginning on the left, the sketch shows that a user submits a Query that flows into the Index contained within the Search Engine and evidently a matching process takes place through the Index. The Results flow back out the smokestack to the user. Associated with the Index are two columns, Term and Location, with the words Dog and http://www... appearing beneath these headings. The use of such specific examples is often used to explicate the meaning of a sketch. In this case the whitespace beneath Dog is important for signaling that the index contains more entries.

In the middle part of the sketch web crawlers are depicted leaving the search engine empty-handed and going into the Internet, returning with little boxes, which are labeled terms. On the right, the Internet is represented as a somewhat undifferentiated cloud containing points with two labeled webpages. Interestingly, but not essential for comprehension, the Search Engine is depicted with wheels, a drive train, and smokestack.

This sketch is noteworthy for employing multiple metaphors while maintaining compositional coherence. Inspecting it shows four major layers of information: 1) Conceptual metaphors for containing and connecting entities; 2) Words used to depict and clarify systemic entities (e.g., Term or Dog); 3) Labels for clarifying the meaning of the entities represented; and 4) Literal, and evidently non-essential metaphors (e.g., smoke stack, spiders), for conveying the problem domain. The last two "descriptive layers" are clearly important for comprehending the sketch but are arguably non-essential.

Many of the sketches that depict computational process are far less coherent. Consider this sketch:



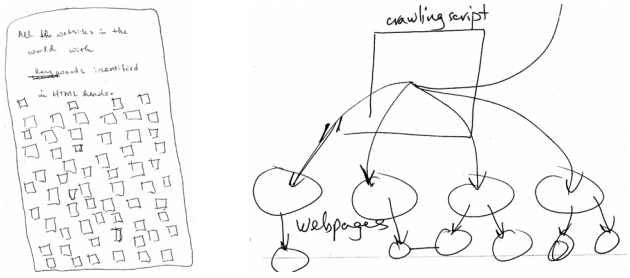
Data is sub-divided into smaller parts until the data can be easily search through [sic]. Much like a sorting algorithm

Here, we can recognize a tree structure where a database consists of rectangles, which, in turn, consist of rectangles. The four vertical dots, '....', and the use of a diagonal line appear to be used to indicate more of the same. This figure is imprecise and it raises more questions than it answers.

Even so, it enables metaphoric reasoning. For example, if those rectangles are smaller parts [of data], then we can ask "How small are they?", "How did they get broken up?", "Is additional information associated with the parts?" "How are the parts ordered vertically?" Thus, even an ambiguous sketch purporting to explain how a search engine works can initiate a questioning process by mapping from the source domain (the sketch's elements) to the target domain (search engines). Next, we isolate and discuss the mappings between graphic markings and concepts in the problem domain that can be discerned from the sketches.

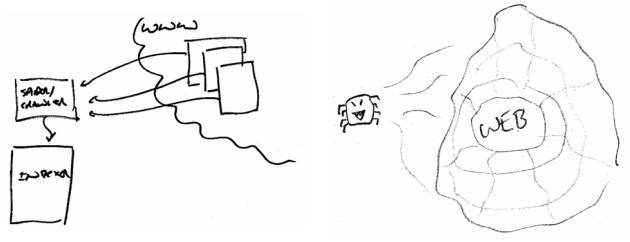
3.3. Example conceptual metaphors

3.3.1. Documents are points. The sketches often depict documents as single points or various geometric shapes; often, many instances are shown. Documents are often placed in a container (left) but can also be related by linking them together (right).

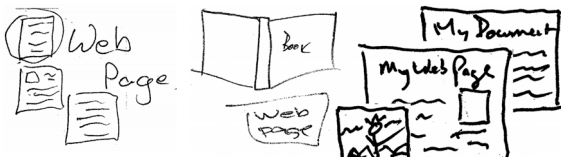


Captions and labels are often used to help interpretation. The caption on the left reads "All the websites in the world with keywords identified in HTML header." On the left, note that the label crawling script is used to clearly distinguish one visual element (a box) from the webpages (linked circles), and by virtue of the linked webpages we can infer the crawling is important here.

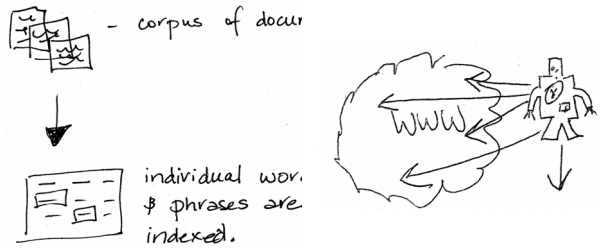
3.3.2. Containers. Organic shapes are often used to represent the internet. On the other hand, rectangles or other geometric shapes are typically used to contain computed or well-specified data.



3.2.3. Documents are text fragments. Sometimes documents are depicted in greater detail. Often, short wavy lines are used to represent text fragments that are placed within a container.

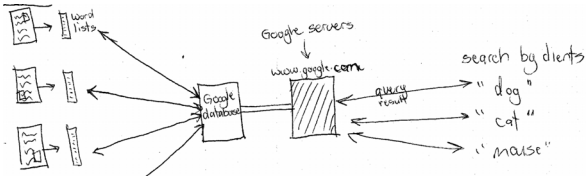


3.2.4. Transformations. Directed lines are used to depict a transformation of data from one form into another.

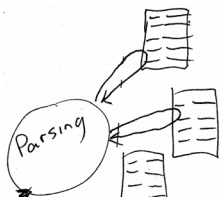


Interestingly, the meanings of the lines can depend on their sources and destinations. On the right, the set of four directed lines seems to indicate *the robot will visit the web* and the vertical line seems to indicate *the robot will send data onward*.

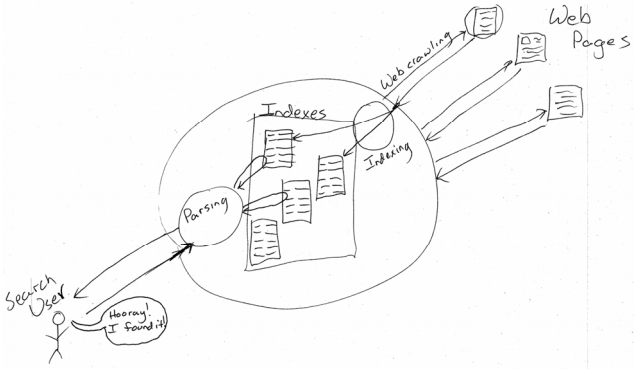
In the next example, documents, depicted with wavy lines, are transformed into word lists, depicted with shorter lines. The word lists appear to be merged at the Google database, as signaled by the converging lines.



Sometimes lines loop across objects, indicating that a process must *go out and come back* as with this sketch.

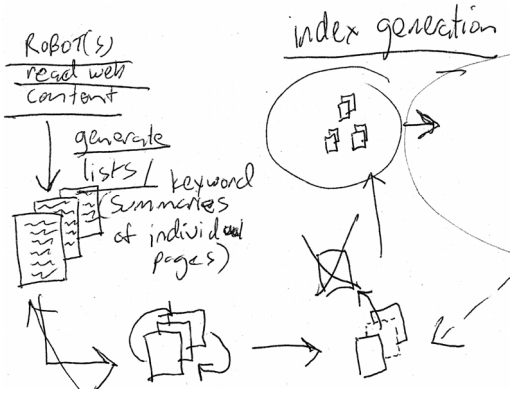


3.2.5. Grouping and abstraction. Sketches often show hierarchical structures for grouping and abstraction. In the following example, notice how the large circle encloses two smaller circles, one labeled Parsing and one labeled Indexing. As well, this diagram does the same with a rectangle, labeled Indexes, which encloses four smaller rectangles.



3.2.6. Iteration and selection. Ellipses ('…') are sometimes used to depict repeated operations but more often several examples are shown, leaving it to the reader to infer that additional instances might be involved. Rules for selecting items or conditions for stopping, if given at all, seem to be usually stated in word annotations.

The next sketch shows flow, looping, and selection. Directed links and the careful use of whitespace are used to represent flow. A group of three documents wrapped by a curved line seems to indicate some kind of looping process (bottom, towards left). Finally, consider the document represented by a dashed outline (bottom right). It is located between to other documents and it is the source of a line to a crossed out document. This assembly, in sum, suggests that some kind of decision process is eliminating documents.



4. Discussion

Approximately 20% of the 232 sketches depict the algorithmic operation of a search engine. As illustrated by the above excerpts, these sketches reveal a diverse range of conceptual and visual approaches. In fact, perhaps the most notable feature of the sketches is just how different they are. This can be seen especially in the images of robots, webs, documents, and so on but also in the overall composition of the sketches, how annotations are used to

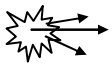
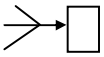
denote or clarify elements of the sketches, and the degree of completeness and correctness of the sketches.

Nevertheless, close inspection of the sketches seems to suggest an underlying regularity where search engines are described by showing how units of text are successively transformed into smaller fragments. This transformation process can be summed up by the conceptual metaphor: A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS. Under this metaphor, people explain the operation of a search engine by describing: 1) How web pages are broken up into words that, in turn, are stored in some fashion, and 2) How queries are broken up into words and matched against the previously stored word data, yielding identifiers to the original pages. To do this, the sketches, at varying degrees of completeness, show a sequence of transformative phases.

Seeking to capture the regularity across the sketches, Table 1 proposes a set of mappings from source to target domains. These mappings, from a sketch containing graphic and textual markings to the Search Engine domain, seek to establish a coherent structure for the metaphor, A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS.

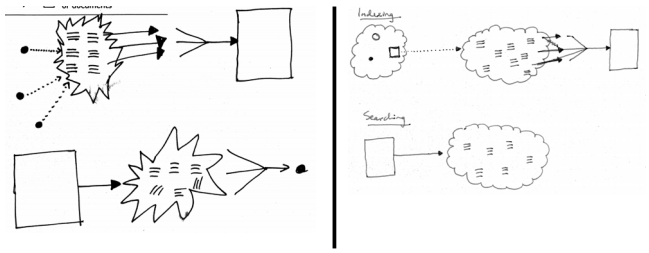
Next, some brief observations are made about this conceptual model.

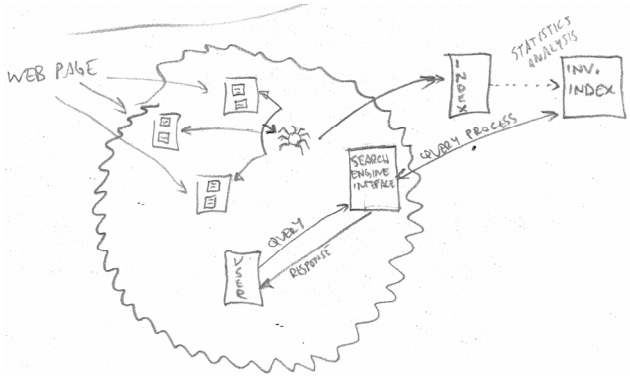
Table 1. Conceptual metaphor.

Source Domain Sketching		*Target Domain* Search Engine
Point or shape		Document
Short, wavy line		Text fragment
Cloud		Large, undifferentiated container
Whole-part assembly		Linked documents
Rectangle, circle, triangle, etc.		Computed or well-specified container
Directed line		Transform text fragment
Lines diverging from container		Process the contained items
Line attached to container		Process the contained items
Converging lines		Accumulate text fragments or documents

4.1. Properties of the notation. Quite obviously, this data flow notation is informal and imprecise—it is not executable. Yet, it can be used to conceptualize the design of a system by capturing the distinctions among entities and the rules for transforming one kind of entity into another. It can be used for fine- or course-grain analysis and by nesting containers the notation allows various levels of abstraction to be represented. While it allows people to depict text units, flows, and transformations, the rules for selecting or qualifying items must be described in annotations—a common practice in the sample of sketches analyzed and also seen in other studies of "natural" programming [8, 21].

4.2. Using the conceptual metaphor. When students are given this metaphor and asked to draw a sketch of how a search engine works, it clearly influences their representational work. Consider these three sketches, which were not part of the original sample.





Perhaps not surprisingly, the sketches are quite consistent in "surface" style and are also consistent with the underlying regularity of the original set of sketches. (These three sketches were drawn by students after a three week instructional module in Information Retrieval.)

While the metaphor may promote consistency in representation, it is not obvious that it actually helps students to reason about the operation of search engines or, more generally, algorithmic processes. The sketches have yet not been used as intermediate representations in design or programming tasks. To learn about the efficacy of the conceptual metaphor, this is a next logical step.

One approach is to replace the ground, SEARCH ENGINE, with a more narrow ground in the metaphor A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS. Then,

prompt learners to sketch solutions. The "natural" elements of students' problems could be further investigated with established techniques [21].

Finally, the conceptual metaphor can be used as a design tool for outlining a plan for implementing a computational process in a formal data flow language [22]. When it is used in this fashion, a key question is whether and how the association between the informal and formal representations is maintained. While this has been recognized as an important issue [20, 23], the benefits and costs of maintaining correspondences between a sketch and a computational process or formal model are not well understood.

4.3. Visual language design. How best to use metaphors—particularly explanatory or literal metaphors—in the design of visual languages is unclear [3]. Nevertheless, it seems that metaphors can be crucial for overcoming barriers in end-user programming [15]. This work shows that students do employ a wide range of literal metaphors in their explanations (spiders, robots, bookshelves, magnifying glasses, etc.) when given an open-ended task. While using metaphors in explanations is quite different from using metaphors in programming tasks, their use in these sketches does suggest that they are important for learners.

Nevertheless, the regularity in the sketches is found beneath these literal metaphors—it lies with the use of Container and Source-Path-Goal schemas and the graphic markings that convey them. Could it be that explanatory metaphors are effective to the extent that they represent these more fundamental cognitive structures in an overall coherent model? If mathematical structure is an analog for programming, then the work by Lakoff and Núñez [17] supports this possibility and gives a theoretical basis for considering "naturalness".

5. Conclusion

This exploratory study examined a sample of sketches drawn by undergraduate and graduate students in information science. Despite that these students do not come from a common community of practice with strong traditions in sketching, inspection of the sketches revealed an underlying regularity. The conceptual metaphor, A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS, was proposed to capture this regularity. This metaphor maps graphic markings that are easily sketched into concepts in the problem domain.

The major contribution of this exploratory study is that the combination of visual grammar and conceptual metaphor may provide a theoretical basis for considering the "naturalness" of a notation. Certainly, the underlying regularity seen in the sample of sketches can, at least to some meaningful degree, be understood through the combination of these two modes of analysis. From this, I propose three specific questions:

1) How might the conceptual metaphor, A SEARCH ENGINE IS A SERIES OF TEXT TRANSFORMATIONS, be used to teach people about search engines?

2) How could this informal notation be augmented with computational properties so that it could be 'run'?

3) How can conceptual metaphor, in general, be used to develop mappings between problem domains and programming notations?

More generally, this work suggests that there is much to learn about how sketching can be used to conceptualize the design of information systems. Studying how communities of designers adopt, develop, and appropriate notations and styles of sketching to represent computational process may provide key insights for supporting design and programming processes through notations. In other fields, such as architecture, sketching is an essential competency—it should be so in information system design too.

6. Acknowledgements

I wish to a thank Efthimis N. Efthimiadis for his collaboration in conceptualizing this work, and Peyina Lin, Kristene Unsworth, and Hui P. Yang for assistance with analyzing the sketches. Karen Fullerton provided valuable comments on an early draft of this paper.

7. Appendix – Normative model of search

In this normative model, based on textbook accounts of search engines [1, 19], a search-engine is divided into three phases, indexing, searching, and matching, with central concepts within each.

A. INDEXING: processing documents so they can be retrieved later.
   1. *Content*: The search engine processes web pages, web sites, and documents
   2. *Spidering/Crawling*: The search engine fetches web pages
   3. *Parsing*: Words from web pages are extracted and analyzed
   4. *Inverted-index-creation*: An index that maps words to web pages is created
   5. *Link-analysis*: The search engine analyzes the linking structure among web pages
   6. *Storage*: Web pages and indexes are stored at the search engine

B. SEARCHING: Users formulate a query and inspect results
   7. *User*: A person interacts with the search engine
   8. *User-need*: A 'need' triggers a user to perform a search
   9. *Query*: An interface is used to submit a query to a search engine

10. *Results*: The output from a search-engine is a list of web pages

C. MATCHING: Queries are matched against web pages

11. *Query processing*: Keywords and operators are extracted from the query
12. *Matching*: Words from the query are matched against words in web pages
13. *Accessing-inverted-file*: Keywords are used to access the inverted file
14. *Ranking*: A ranked ordering of web pages is created

8. References

[1] Belew, R.K., *Finding out about--A cognitive perspective on search engine technology and the www*. Cambridge University Press, 2000.

[2] Bellamy, R. K. E, "What Does Pseudo-Code Do? A Psychological Analysis of the Use of Pseudo-Code by Experienced Programmers", *Human-Computer Interaction, 9 (2), 1994, pp. 225-246.*

[3] Blackwell, A. F. and T. R. G. Green "Does metaphor increase visual language usability", *Proceedings 1999 IEEE Symposium on Visual Languages*, 1999, pp. 246-253.

[4] Blackwell, A. F., K. N. Whitley, J. Good and M. Petre, "Cognitive factors in programming with diagrams", *Artificial Intelligence Review*, 15, 2001, pp. 95-114.

[5] Chen, Q., J. Grundy, and J. Hosking, "An E-whiteboard application to support early design-stage sketching of UML diagrams", *Proceedings of the 2003 IEEE Conference on Human-Centric Computing*, Auckland, New Zealand, IEEE, 2003, 219-226.

[6] Cook, S. D. N. and J. S. Brown, "Bridging epistemologies: The generative dance between organizational knowledge and organizational knowing", *Organization Science*, 10 (4), 1999, pp. 381-400.

[7] Do, E. Y. and M. D. Gross, "Thinking with diagrams in architectural design", *Artificial Intelligence Review*, 15, 2001, pp. 135-149.

[8] Douglas, S., C. Hundhausen, and D. McKeown, "Toward empirically-based software visualization languages", *Proceedings of the 1995 IEEE Symposium on Visual Programming Languages*, 1995, pp. 342 – 349.

[9] Efthimiadis, E. N. and D. G. Hendry, "Search engines and how students think they work", *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 595-596.

[10] Garrett, J. J., "A visual vocabulary for describing information architecture and interaction design", Retrieved 03.01.06 from http://www.jjg.net/ia/visvocab/

[11] Gross, M. D., "The electronic napkin: Working with diagrams", *Design Studies*, 17(1), 2001, pp. 53-69.

[12] Henderson, K., "Flexible sketches and inflexible data bases: Visual communication, conscription devices, and boundary objects in design engineering", *Science, Technology, & Human Values, Science, Technology, & Human Values, 16*(2), 1991, pp. 448-473.

[13] Hendry, D. G. and E. N. Efthimiadis (in press). *Conceptual models for search engines*.

[14] Irani, P. and C. Ware, "The effect of a perceptual syntax on the learnability of novel concepts", *Proceedings of the Eighth International Conference on Information Visualization*, IEEE, London, UK, 2004, pp. 308-314

[15] Ko, A. J., B. A. Myers and H. H. Aung, "Six learning barriers in end-user programming systems", *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing*, IEEE, 2004.

[16] Lakoff, G. and Johnson, M. *Metaphors We Live By*. The University of Chicago Press, Chicago, 1980.

[17] Lakoff, G. and Núñez, R. E. *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books, New York, 2000.

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

[19] Liddy, E., "How a Search Engine Works", *Searcher*, 9(5). Retrieved 03.01.06 from http://www.infotoday.com/searcher /may01/liddy.htm

[20] Newman, M. W, J. Lin, J. I. Hong, J. A. Landay, "DENIM: An informal web site design tool inspired by observations of practice", *Human-Computer Interaction*, 18, 2003, pp. 259-324.

[21] Pane, J. F., C. A. Ratanamahatana and B. A. Myers, "Studying the language and structure in non-programmers' solutions to programming problems", *International Journal of. Human-Computer Studies*, 54, 2001, pp. 237-264.

[22] Pautasso, C. and G. Alonso, "Visual composition of web services", *Proceedings of the 2003 IEEE Conference on Human-Centric Computing*, Auckland, New Zealand, IEEE, 2003, pp. 92 - 99.

[23] Piller, B. and M. Apperley, "Computer-aided sketching to capture preliminary design", *Third Australasian User Interfaces Conference, Australian Computer Society, Inc.*, 2002, pp. 9-12.

[24] Petre, M. and R. Winder, "Issues governing the suitability of programming languages for programming tasks", *People and Computers IV*, Cambridge University Press, 1988, pp. 199-214.

[25] Schön, D. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983.

[26] Suchman, L. A., "Representing practice in cognitive science", *Human Studies*, 11, 1988, pp. 305-325.

[27] Ware, C., *Information Visualization: Perception for Design*. Morgan Kaufmann, New York, 2nd Edition, 2004.