Design Buddy: Providing Feedback for Sketched Multi-Modal Causal Explanations

Jon Wetzel and Ken Forbus

Qualitative Reasoning Group, Northwestern University
2133 Sheridan Road, Evanston, IL, 60201, USA
jw@northwestern.edu, forbus@northwestern.edu

Abstract

An important problem for engineering students is learning how to communicate. Based on collaborations with engineering design instructors, we are creating a system, *Design Buddy*, that is intended to help students learn how to explain their designs. The input is a sketched comic strip with graphical annotations that indicate forces and motions, plus a structured language system that enables students to enter causal models. The system provides feedback by comparing their explanation to its own reasoning about how the design might work, using qualitative reasoning to detect contradictions and gaps in the causal explanation. This paper focuses on a structured language system to provide information that is not easily communicated via sketching, and the new technique of *Sequential Explanation Analysis* to provide feedback based on the consistency of a student's explanation, both internally and with physical laws. Results from a pull-out study with engineering design students indicating that the system can indeed help them improve their explanations are also described.

Introduction

Communication is an important skill for engineers. They typically work in teams, and often work with clients, when creating and refining designs. Consequently, at Northwestern University students learning design are taught communication skills at the same time. Sketching is used ubiquitously in the early stages of engineering design. Consequently, instructors place a strong premium on the ability to communicate ideas via sketching. These instructors tell us that, unfortunately, this proves to be particularly difficult for most students. This motivated the creation of *Design Buddy,* a sketch-based system that is being built to help students learn to explain designs through sketches. [Wetzel & Forbus 2009] described a critiquing algorithm which used only sketched input, and a corpus analysis to argue for its generality. The new contributions here are (1) the extension of the critique algorithm to include causal explanations, entered by the student using a structured language interface in conjunction with sketching and (2) the results of a pull-out study with engineering students, which suggests that students can indeed improve their explanations by using the system.

We start by briefly reviewing CogSketch [Forbus *et al.,* 2008], which Design Buddy is built upon. Then we illustrate the system's operation with a simple example. Next we describe the critique algorithm for generating feedback. The results of the pull-out study are next, followed by related & future work.

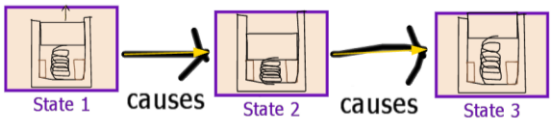

Figure 1: Sketch of a simple push-button with spring return.

CogSketch: The essentials

CogSketch is an open-domain sketch understanding system.[1] Most sketch understanding systems (cf. [Hammond & Davis, 2005]) focus on recognition, automatically identifying what the user draws as a member of a relatively small vocabulary of concepts. Such systems can work well in tightly constrained domains, like schematic diagrams or military symbology, but the range of concepts that arise in engineering design is huge. Moreover, unlike schematics, most concepts do not have agreed-upon visual symbols, and in reasoning about shapes and motion, the specific geometry of the sketch matters. Consequently, such systems are not good fits for this problem. By contrast, CogSketch provides several interfaces for specifying *conceptual labels* for categorizing what a glyph depicts. The labels are drawn from an OpenCyc-derived knowledge base, which includes extensions for qualitative and analogical reasoning, containing over 58,000 concepts. While even this range of concepts is not sufficient to completely cover engineering

[1] CogSketch is publicly available for download at http://www.qrg.northwestern.edu/software/cogsketch/index.html

design, it is already several orders of magnitude larger than what recognition-based systems can handle.

Figure 1 shows a simple behavior for illustration. Every CogSketch sketch consists of a set of subsketches. Figure 1 illustrates the *metalayer*, where every subsketch is treated as a glyph. Arrows drawn between subsketches express binary relationships between them. In Design Buddy, every subsketch is a qualitative state of the design's intended behavior, and the arrows between them are restricted to indicate state transitions. Once the student draws the first state, the rest of the states are typically created by cloning and modifying the first state, which greatly reduces drawing time. Note that, unlike traditional comic strips which impose a linear flow of time, these *comic graphs* [Forbus *et al* 2003] can include ambiguity (via multiple transitions out of a state), alternate paths (via multiple transitions into a state), and oscillation (via cycles), which broadens its expressive range.

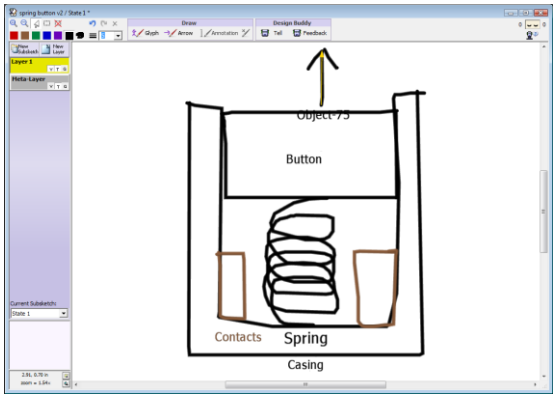

Figure 2: The first state of the example behavior. The arrow is a force annotation, indicating the intended direction of an applied force.

Figure 2 is a close-up of the first state of the behavior in Figure 1. Each sketch is made of *glyphs*, a collection of polylines that the user draws to depict an entity. Segmentation is manual, i.e., a button is pressed to start a glyph and pressed again to end it. This simple mechanism avoids using timeouts or pen-up heuristics that our users find limiting, especially when drawing complex shapes. The ink of a glyph is visually analyzed to construct qualitative spatial representations. Experiments with CogSketch indicate that it provides a reasonable model of aspects of human visual processing [Lovett *et al* 2009], providing a useful substrate for reasoning about mechanics. For example, it computes RCC8 relationships [Cohn 1996] among others to detect mechanical contact. CogSketch can also decompose a shape into its component edges and find the qualitative direction of the surface normal for a given contact surface pair. This spatial information is necessary for understanding the mechanics of the design.

Basic glyphs represent entities, and can be labeled with concepts from the knowledge base. For example, in Figure 2, the button is labeled as instance of the concept RigidObject and the spring is labeled as an instance of Spring-Device. Relationship glyphs, which are drawn as

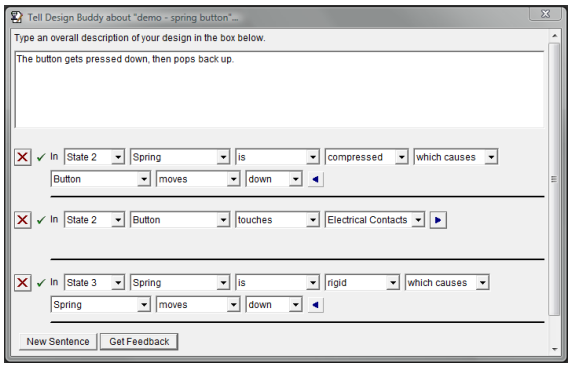

Figure 3: Structured language interface for complementing information in the sketch

arrows, are labeled using binary relationships, like causality in Figure 1. Annotation glyphs are used to add additional properties to the entity depicted by a glyph, e.g., the numerical value of a parameter or the direction of motion or forces. In Figure 2, an external force acting on the button is indicated by the arrow at the top.

Design Buddy Example

Consider the simple push-button with spring return illustrated in Figure 1. The student starts constructing the explanation of their design by drawing the comic strip. Once they start drawing, they also use a structured language interface to add information about the design. As explained below, this interface can provide information that is hard to sketch, such as indicating causal relationships between specific aspects of states, as opposed to whole states as in Figure 1. Figure 3 illustrates the structured language interface that Design Buddy provides for entering such information. While the surface form is textual, the underlying representation produced is predicate calculus, to support reasoning.

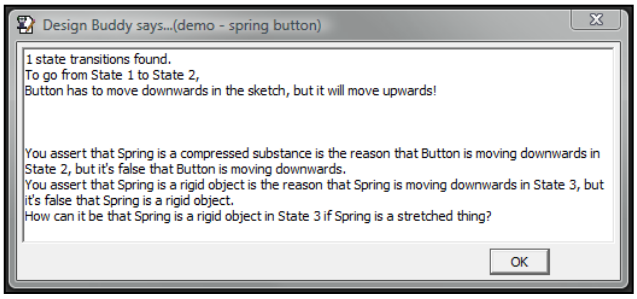

Figure 4: Design Buddy's feedback on the explanation. The top part is generated by STV while the bottom three sentences are generated by SEA. (See "Generating Feedback")

At any point, the user can ask for feedback. Design Buddy then analyzes the combination of the sketch and structured language input to look for inconsistencies and/or gaps in the explanation. These analyses result in questions and critiques for the user. In this case it finds three problems, as Figure 4 illustrates. The first is that the force annotation indicates that the button is being forced upward

in the initial state, while the sketch has it moving downward. The second is that the first sentence in Figure 3 has the button moving down in response to the compressed spring in state two when in fact it should pop back up. The third is that the spring is described as being rigid in the third state, which is inconsistent with it being a spring. After the student fixes these problems, by rotating the force arrow in the first state and changing the spring to be described as stretched in the third state, Design Buddy is satisfied with the explanation.

Next we describe the interface for structured language explanations, followed by our algorithm for generating feedback.

Structured Language Input

While sketching is very flexible, some things are best expressed in natural language. Robust natural language understanding in domains this broad remains beyond the state of the art. Consequently, we use forms for constructing restricted sentences, using templates that can be rendered into understandable natural language, but whose internal form is predicate calculus.[2] The basic sentence structure we use is:

In <*context*> <*subject*> <*verb*> <*object*>.

Here <*context*> is the state (i.e. subsketch) being discussed. Once the user has selected a context for a sentence, the <*subject*> field is populated with entities represented by glyphs in that subsketch. The <*verb*> field supports stating how <*subject*> moves or rotates, whether or not it possesses some property (e.g., is rigid), or that <*subject*> touches <*object*>. The possibilities for <*object*> vary according to the verb selected, as shown in Table 1. One example is the second sentence in Figure 3: "In state 2 Button touches electrical contacts."

Table 1: Template verbs and their possible objects as used in the structured language interface

Verb	Possible Object Field Values
moves	left, right, up, down, quadrants 1-4
rotates	clockwise, counterclockwise
does not move	None
does not rotate	None
is	rigid, fixed, springy, compressed, stretched
touches	all other objects in the subsketch

Sentences can either be simple or compound. To create a compound sentence, the user clicks the arrow button to the right of the object, revealing additional fields. The template for a compound sentence is two simple sentences joined with a causal relationship:

In <*context*> <*subject1*> <*verb1*> <*object1*>
<*causal*> <*subject2*> <*verb2*> <*object2*>.

<*causal*> is either A causes B (e.g. "which causes") or A prevents B (e.g., "which prevents"). Thus the student can make causal claims about specific aspects of the behavior and/or properties of components in their design. Returning again to Figure 3, the first sentence is: "In state 2, Spring is compressed which causes button moves down."[3]

Whenever a field is edited, the program checks to see if the template is completely filled out. If so, a green check mark appears next to it in the Tell window, letting the user know they have completed the sentence. The fact that the student asserted the sentence is then stored in the shared underlying knowledge representation.

Generating Feedback

Once the student completes the sketch and structured language input, they can request feedback. The goal of Design Buddy is to verify that the explanation given (i.e., the structured language and sketched input) is consistent internally and physically. If there are contradictions or unfounded claims, then Design Buddy provides feedback that highlights them, so that the student can debug their explanation.

To understand the possible behaviors of the system, Design Buddy uses *qualitative mechanics* (QM) [Nielsen, 1988; Kim 1993], which describes the physical interactions between objects via contact relationships and forces in two dimensions. Qualitative models are necessary in this task for two reasons. First, few numerical parameters are available at this stage in the design process. Second, it is important to provide feedback about causal theories in a human-like way, to help enculturate students in engineering practice. Our qualitative mechanics reasoning currently handles the behavior of arbitrarily-shaped rigid bodies, springs, and gears.

Design Buddy uses two algorithms to produce feedback. The first is State Transition Verification [Wetzel & Forbus, 2009], which focuses on the sketch alone. We summarize it briefly below for completeness. The second is *Sequential Explanation Analysis*, which combines the sketch and structured language explanation.

State Transition Verification (STV) takes as input a comic graph. It starts by examining each state transition to produce a list of compatibility constraints for the transition to occur, assuming the motion that the student described is correct. An example constraint in figure 1 is that based on the relative positions of the objects between state 1 and state 2, the button must move upwards. It then looks at the first state in each pair, using QM to predict what motions can actually occur. If the motion predicted by QM does not match the compatibility constraints, there is a contradiction within the explanation. For each type of contradiction there is a canned explanation template that is used in generating textual output. In addition to providing some direct critiques, QM infers useful facts such as surface contacts and normals, plus the net force and next

[2] This interface is inspired by [Frank & Szekely, 1998].

[3] Currently we are more concerned with expressiveness than perfect grammar.

motion of objects in each subsketch, all of which are used in the following step.

Sequential Explanation Analysis checks the structured language input statements against the sketch to see if they are mutually consistent. Sentences are processed in the order provided, since they can lead to additional assumptions about the design. The analysis marks each fact as either: (1) *proven*, i.e., inferred based on what is known from the sketch and the previous sentences processed, (2) *contradictory*, i.e., inconsistent with the beliefs about the design based on the analysis so far, or (3) *unproven*, i.e., neither a consequence of the analysis so far nor inconsistent with it. Unproven facts are assumed to be true while processing subsequent sentences, because they represent new information about the design.

Sentences in the structured language explanation are verified based on their content. Statements about contact are tested against the visual representation computed for the state in question. Statements about motion or rotation are tested against the qualitative mechanics analysis of the sketched comic graph. Statements about physical state (i.e., "is" statements) are tested against the conceptual labels provided for glyphs in the comic graph. Statements about causality are tested by assuming the causal antecedent and seeing if (a) the causal effect is believed and (b) that the causal antecedent is one of the assumptions underlying belief in the causal effect, as found by analyzing the dependencies in the underlying truth maintenance system.

When generating feedback for the explanation, the states of facts in the structured language explanation are used to generate part of the natural language response. (The rest is from state transition verification.) Contradictory statements lead to a contrast template being used, e.g., "You said that the stopper moves to the right, but I think it will move to the left." Unproven statements that should have been derivable lead to a gap template being used, e.g., "You said that the stopper moves right, but it's not clear why that will happen." If all statements that should be provable are indeed proven, then Design Buddy simply says "Everything you told me made sense."

Exploratory Study

In Fall 2009 we performed an exploratory study to gather formative feedback with undergraduate students taking an engineering design and communications (EDC) class. Here we focus on the following questions: (1) Can students use the feedback provided by Design Buddy to produce consistent explanations? (2) How much feedback is provided by SVT versus SEA?

Procedure

The study involved seven participants, paid hourly, all engineering undergraduate students taking EDC. One-on-one interviews were conducted in single sessions lasting between 45 minutes and 80 minutes depending on their availability and interest in continuing.

The participants used a tablet PC with CogSketch for the experiment. Two kinds of data were collected: (1) Video and audio were captured by screen-capture software (for 6 of the 7 students) and (2) the CogSketch sketch files produced by the student over the course of the experiment. Sketch files retain their edit history, and CogSketch can replay the drawing of the sketch and export detailed timing data for analysis.

The experiment protocol consisted of four parts. First, the students received a brief tutorial on pen-based computing and CogSketch. Second, they were asked to explain their current design class project to the interviewer, while using CogSketch to draw it. (The purpose of this step was to gather data about future breadth requirements and to see what language they would use if unconstrained.) The third step was a tutorial on Design Buddy, focusing on using the interface to create explanations. The fourth step tested the explanation system directly: Students were asked to explain to Design Buddy how a retractable ball-point pen works. The students were provided with a real pen to handle and think about. The pen was opaque and students were not allowed to disassemble it.

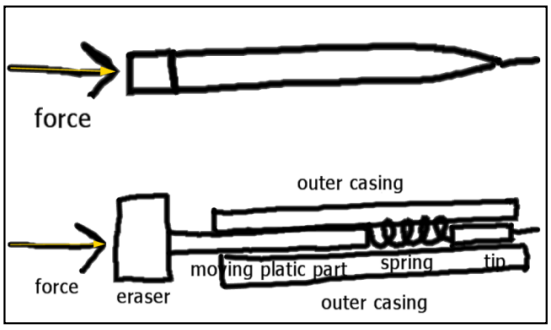

Figure 5: Early sketch (above) contrasted with later version (below). Student sketches changed as they interacted with Design Buddy.

Preliminary Findings

On average students spent under 15 minutes combined on the two tutorial sketches, which indicates that not much training is required to use the interface. All of the students were able to use Design Buddy to explain how the ink pen might work. The average time to complete an explanation was 38 minutes, with the shortest being 28 minutes and the longest being 60 minutes.

Each student drew a multi-state sketch, used the Tell window, and went through several feedback iterations with Design Buddy. On average, students went through 5.5 cycles of feedback, with the fewest being 3 and the most being 8. Figure 5 shows two sketches of the pen made by one student over the course of their session. The early sketch is highly abstract. Design Buddy reported that it didn't seem like any of the parts moved. This is because the student had initially drawn the pen as a single object. The student realized that they needed to add more detail to their sketch, so they redrew the pen to show the inner

workings. Most of the students drew sketches detailing the inner workings of the pen.

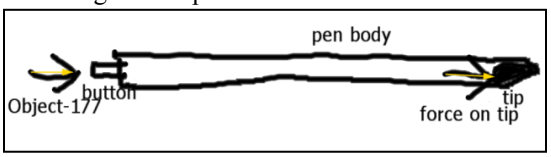

Figure 6: A student abstracts out the middle of this ink pen.

One student got around this process; rather than depicting how the pen worked internally, they depicted what was happening to the pen as viewed from the exterior (Figure 6). This more devious approach still kept the student busy for several iterations as they tried using different relations and annotations to express the motions.

Often students assumed that Design Buddy's visual processing was as robust as a human partner. For example, they would sometimes leave sizeable gaps between parts that they meant to be in contact. Other times they would forget that they needed to conceptually label a glyph. Occasionally they ran across an actual bug in the program. Each of these cases required the interviewer to explain limitations of the program. While students found the feedback itself clear, tracing back to the specific aspects of their explanation that were responsible for a reported problem was often difficult. We allowed the students to try diagnosing the problems on their own until they got stuck, at which point we stepped in. When it wasn't obvious to the experimenter either, we showed them how to use CogSketch's built-in reasoning browser to figure it out definitively. This browser is not student-friendly: It exposes the underlying predicate calculus form of statements and includes many facts necessary for implementing the reasoning but irrelevant to what the student needs to know. We plan on adding a *structured explanation system* [Forbus *et al* 1999] to provide a filtered, student-friendly drill-down system to address this problem.

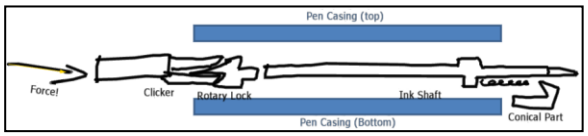

Figure 7: Student 194's explanation was very complex and precise, and they were unable to explain it using the template sentences.

How much utility does Sequential Explanation Analysis add over State Transition Verification? To explore this question, we examined how often advice was provided by each. Overall, students received 51% of their feedback from SEA. Student 194 (Figure 7) received no SEA feedback at all because their explanation was too complex for the template sentences. The student wrote their explanation in the Notes field for the sketch instead, which was not accessible to Design Buddy.

Having the students explain their design projects to the experimenter, as opposed to Design Buddy, provided information about their default communication strategies. When explaining to the experimenter, students used language more heavily than their sketches. When explaining to Design Buddy, they put more information into the sketch by adding conceptual labels, additional states, and drawing forces. Students primarily used structured language for describing motion. Students for the most part had no suggestions for additions to the structured language, though some did suggest alternate phrasings for existing phrases such as "compresses" instead of "is compressed". Another student suggested adding the ability to talk about multiple states at once. Given that our goal is to help students learn to use sketches to communicate, these results are encouraging.

Related Work

Argumentation and Explanation

In the Belvedere system [Suthers *et al* 2001], students used a visual language to construct arguments which were then critiqued based on their structure. However, the feedback focused on the structure of the arguments and ignored the actual content. Design Buddy, by contrast, incorporates a model of some kinds of reasoning an expert does about explanations for engineering designs.

Recognition-based approaches

Using sketch and speech as input, Bischel *et al.* [2009] were able to train classifiers which distinguished between strokes that were gestures versus strokes that were part of the depiction of a device. The gestures included both pointing to parts being referred to linguistically and for indicating spatial features, such as direction of motion. We observed both kinds of gestures when students were communicating with the experimenter, which suggests that adding support for gestures could increase the naturalness of the interface. However, spoken language input is problematic for many student environments, due to noise, the inconvenience of a headset, and the need to avoid disturbing others.

Recognition-oriented sketch recognition can be used for education, in circumstances where teaching people conventions for drawing something are important. For example, [Taele & Hammond, 2010] use sketching to teach Mandarin Phonetic Symbols. As noted earlier, in engineering design the shapes of parts in designs for mechanical domains are determined by physics not by convention, as they are for electronics and UML diagrams.

Discussion & Future Work

Helping engineering students practice giving explanations via sketching will, both we and their instructors believe, help them improve their ability to communicate. The structured language interface provides a way for students to communicate information that is not always easily sketched, such as causal models. The new technique of Sequential Explanation Analysis provides feedback about

the internal consistency of their explanation by comparing the structured language explanation with the qualitative mechanical analysis of their sketch. The pull-out study, while only involving a small number of students, provides strong evidence that the feedback Design Buddy already provides can help students improve their explanations.

While the progress described in this paper takes us closer to our goal, there are still several extensions we plan to make before attempting in-class experiments:

Richer qualitative reasoning: Currently we require students to draw every qualitative state explicitly in their behavior. For simple designs this is reasonable, but as designs get more complex, experienced designers leave out "obvious" intermediate states. Design Buddy will need more qualitative reasoning to fill in those missing states, and to evaluate when they make assumptions that are inconsistent with a student's explanation. Moreover, some design assignments use non-rigid materials, such as strings, and fluids. We plan on extending our current qualitative mechanics reasoner to incorporate qualitative process theory [Forbus, 1984] to handle this, with Kim's bounded stuff ontology [Kim 1993] to handle fluids.

Richer teleological vocabulary: The structured language interface needs to be extended to describe the purpose of the design, and explain how the behavior (which is the current focus of explanation) leads to this purpose being achieved.

More types of feedback: Currently Design Buddy's feedback focuses only on consistency in the explanation. Equally important is coverage: Does the explanation account for all of the important choices in the design? Simplicity and clarity are also important, as is detecting circularities. We plan to extend Design Buddy to critique explanations along these dimensions as well, as we scale up to more complex designs.

Additional studies planned: We plan to continue pull-out studies for formative evaluation, with the goal of moving to in-class experiments. We are also continuing to observe instructors providing feedback about design sketches, both in classroom group work and in one-on-one interactions, to improve Design Buddy's feedback.

Acknowledgements

This research was supported by the Spatial Intelligence and Learning Center, NSF SLC Grant SBE-0541957. We thank Bruce Ankenman, John Anderson, and Stacy Benjamin for their feedback and help, and for allowing us to work with their EDC students.

References

[Bischel *et al.*, 2009] Bischel, D., Stahovich, T., Davis, R., Adler, A., & Peterson, E. (2009). Combining speech and sketch to interpret unconstrained descriptions of mechanical devices. *International Joint Conference on Artificial Intelligence*. Pasadena, CA.

[Cohn 1996] Cohn A. Calculi for qualitative spatial reasoning. In *Artificial Intelligence and Symbolic Mathematical Computation*, LNCS 1138, eds: J Calmet, J A Campbell, J Pfalzgraph, Springer Verlag, (1996) 124-143.

[Forbus, 1984] Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, 24, 85-168.

[Forbus *et al* 1999] Forbus, K., Whalley, P., Everett, J., Ureel, L., Brokowski, M., Baher, J., Kuehne, S. (1999). CyclePad: an articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*. 114 (1-2): 297-347.

[Forbus *et al* 2003] Forbus K., Usher, J. and Chapman, V. (2003). Qualitative spatial reasoning about sketch maps. *Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence*, Acapulco, Mexico.

[Forbus *et al.,* 2008] Forbus, K., Usher, J., Lovett, A., Lockwood, K., & Wetzel, J. (2008). CogSketch: Open-domain sketch understanding for cognitive science research and for education. *Proceedings of the Fifth Eurographics Workshop on Sketch-Based Interfaces and Modeling*. Annecy, France.

[Frank & Szekely, 1998] Frank, M. and Szekely, P. Adaptive Forms: An interaction paradigm for entering structured data. In *Proceedings of IUI-1998*.

[Hammond & Davis, 2005] Hammond, T. & Davis, R. 2005 LADDER, a sketching language for user interface developers. *Computers & Graphics* 29 (518-532).

[Kim 1993] Kim, H. (1993). Qualitative reasoning about fluids and mechanics. Ph.D. dissertation and ILS Technical Report, Northwestern University. Evanston, IL.

[Lovett *et al* 2009] 1. Lovett, A., Tomai, E., Forbus, K. & Usher, J. (2009) Solving Geometric Analogy Problems through Two-Stage Analogical Mapping. *Cognitive Science.*

[Nielsen 1988] Nielsen, P.E. (1988). A qualitative approach to rigid body mechanics. (Tech. Rep. No. UIUCDCS-R-88-1469; UILU-ENG-88-1775). Urbana, Illinois: University of Illinois at Urbana-Champaign, Department of Computer Science.

[Suthers *et.al.* 2001] Suthers, D., Connely, J., Lesgold, A., Paolucci, M., Toth, E.E., Weiner, A., (2001). Representational and Advisory Guidance for Students Learning Scientific Inquiry. In: Forbus, K., and Feltovich, P. *Smart machines in education: The coming revolution in educational technology*. Menlo Park, CA: AAAI/MIT Press, pp7-35.

[Taele & Hammond, 2010] Taele, P. and Hammond, T. 2010. LAMPS: A Sketch Recognition-Based Teaching Tool for Mandarin Phonetic Symbols I. *Visual Languages and Computation.*

[Wetzel & Forbus, 2009] Wetzel, J. and Forbus, K. (2009). Automated Critique of Sketched Mechanisms. Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference. Pasadena, CA.