Explaining Serendipitous Recognition in Design

Linda M. Wills
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
linda@cc.gatech.edu

Janet L. Kolodner
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
jlk@cc.gatech.edu

Abstract

Creative designers often see solutions to pending design problems in the everyday objects surrounding them. This can often lead to innovation and insight, sometimes revealing new functions and purposes for common design pieces in the process. We are interested in modeling serendipitous recognition of solutions to pending problems in the context of creative mechanical design. This paper characterizes this ability, analyzing observations we have made of it, and placing it in the context of other forms of recognition. We propose a computational model to capture and explore serendipitous recognition which is based on ideas from reconstructive dynamic memory and situation assessment in case-based reasoning.

Introduction

When creative designers have been deeply engaged in a problem, they are often able to recognize solutions to it in their environment, even if they are not actively working on the problem at the time. The solutions recognized may be objects, behaviors, processes, techniques - anything that they observe or examine. This can often lead to key insights, sometimes revealing new functions or purposes for common design pieces. This paper takes a step toward understanding this ability by focusing on the serendipitous recognition of *objects* as solutions to pending (possibly suspended) problems during creative mechanical design.

Serendipity often plays a significant role in creativity. Its accidental nature may seem to put it out of reach for creating computer-based design systems that can take advantage of serendipity, or for supporting people in serendipitous discovery of solutions. However, we believe that being in the right place at the right time is not the difficult part; we can put a designer (human or computer-based) into a rich environment of stimuli where "accidents" will happen. The creativity comes in the preparation that allows recognition of a solution when it is present (Seifert, et al., 1994). This requires becoming immersed in the problem, redescribing it and viewing it from multiple perspectives, considering, comparing, and critiquing several options, so that when a relevant solution is spotted, the way it fits into the problem can be immediately discerned.

The following are typical examples of serendipitous recognition. They occurred during a mechanical engineering (ME) design project we observed in which a team of four students were to design and build a device to quickly and safely transport as many eggs as possible from one location to another. (The first author participated as a member of the team in order to become immersed in the design issues and observe as much of the design process as possible in a natural setting.)

Our first example ("bending springs") occurred while our designers were considering using a spring launching device and went to a home improvement store to look into materials. While comparing the strengths of several springs by compressing them, they noticed that the springs tended to bend. One designer wrapped a hand around the spring to hold it straight as it was compressed and said the springs would each need to be enclosed in a tube to keep them from bending. Another added that the tube would need to be collapsible (to compress with the spring). The designers could not think of an existing collapsible tube and did not want to build one due to time pressure. They gave up on the springs and started thinking about egg protection. During their search for protection material, they walked through the bathroom section of the store, where they saw a display of toilet paper holders. They immediately recognized them as collapsible tubes which could be used to support the springs.

The second example ("weighted rubberbands") occurred later in the design, after the designers had decided to use a cylindrical egg carrier with a spring launch device. They were working on a homework assignment which involved formalizing the tradeoffs between egg capacity, weight, and launch force as a multi-goal optimization problem to help them make the best choice for these variables. They were having trouble and were easily distracted from the problem. One distraction came from a designer who described a trick involving a moving cylinder (coffee can) she had seen the night before on a children's science TV show (Beakman's World). The episode showed how to make a coffee can that rolled back to you when you rolled it away. Batteries were taped as weights to rubberbands, strung through the center of the can. The weights caused the rubberbands to become wound up as the can rolled. As the rubberbands unwound, they caused the can to roll back to the starting location. The designers discussed whether this could be modified for use in their design (e.g., wind the rubberbands up and let their unwinding launch the device). However, they rejected the design for adding too much weight, since the design task had strict weight restrictions. Then they went back to their homework. Suddenly, one designer suggested using the *eggs* as weights on the rubberbands. This alleviated the weight problem because the weight of the eggs did not count into the restricted weight constraint.

There are three intriguing characteristics of this type of recognition. First, designers are able to recognize solutions to problems that have already been suspended. We call this *serendipitous recognition.*

Second, they are often able to recognize objects as solutions, even though this requires the object to play a role or provide a function different from its usual role or function. In our bending springs example, the toilet paper holder is not used to hold a paper roll, but to keep a spring from bending upon compression. In the weighted rubberbands example, the eggs not only play their usual role of being cargo/passengers in the egg carrier, but also are used in a nonstandard way to provide weight to make the device move. This type of recognition requires overcoming functional fixedness (Maier, 1931, 1970; Duncker, 1945).

Third, the solutions recognized are not always standard solutions to the problem. In the bending springs example, the problem was new to the designers and did not have standard solutions. In the weighted rubberbands example, the designers recognize a solution that is different from the existing standard solutions to the problem. In general, if a problem has standard solutions (or ones that are apparently applicable), fixation on these standard solutions must be overcome.

These three characteristics make this type of recognition more difficult to model than types previously studied. Since the solutions are nonstandard, it is not addressed by object recognition work (e.g., (Grimson, 1990)) which typically searches a scene for particular models of existing standard solutions to problems. The type of recognition we are interested in requires models of objects to be constructed on the fly. Jordan and Shrager (1991) model how people select objects for a nonstandard use, based on which object's physical properties relate best to the desired function. However, they do not address how the salient properties are derived or determined.

The opportunistic nature of serendipitous recognition makes research on predictive encoding (Patalano, Seifert, & Hammond, 1993) and opportunism (Hammond, 1989; Mueller, 1990) relevant to our research. However, they address opportunism in planning situations in which there are standard solutions to the problems at hand. The opportunities detected are those to fulfill suspended goals by recognizing that a standard plan for them can now be resumed. Our recognition involves recognizing objects that are not the standard solutions to the pending problems.

Some important issues serendipitous recognition raises are: To what extent does problem and solution need to be described and elaborated for the recognition to take place? How is this description created? How is the opportunity to solve a suspended problem noticed? What allows the relevant problem context to be recalled when its solution is seen?

This paper presents our analysis of the processes involved in serendipitous recognition of nonstandard solutions. Our hypothesis is that recognition arises from interactions between two processes: *problem evolution* and *assimilation* of proposed ideas into memory. We propose a computational model of serendipitous recognition based on this hypothesis. Our model draws on and extends ideas from reconstructive dynamic memory (Schank, 1982; Kolodner, 1983) and case-based reasoning (Kolodner, 1993), particularly situation assessment and evaluation processes.

Evolution and Assimilation

A key activity of designers is to understand, refine, elaborate, and re-define the problem. They view the problem from multiple perspectives and redescribe it in more familiar terms. This process, which we call *problem evolution*, reveals constraints, features, and properties to look for in proposed solutions. As design alternatives are proposed and explored, they are *assimilated* into memory; they are compared and organized, based on the criteria and features the designer has become attuned to through problem evolution. Some serendipitous recognition may arise from assimilation activities.

Problem evolution is driven primarily by the *evaluation* of proposed ideas, which, in addition to revealing flaws in the specification (such as contradictions and ambiguities), generates new criteria, constraints, and preferences that go beyond those given in the original statement of the problem. Assimilation itself can play a significant role in evaluation by drawing attention to features of proposed ideas that are unusual or particularly good or bad compared to other proposed ideas. This in turn can trigger a complete problem reformulation. So, while problem evolution can "set up" the reasoner to recognize solutions when they are stumbled upon, the recognition itself can sometimes actually trigger a problem redescription. This occurred in the weighted rubberbands example, which we analyze in depth in the next section.

These interacting processes fit well into a model of dynamic memory and case-based reasoning. A key idea underlying *dynamic memory* is that remembering, understanding, and learning are all inextricably intertwined. The ability to determine where something fits in with what we already know (understanding) is a key part of being able to assimilate objects in our environment into our problem solving. This may involve a useful reinterpretation of something already in memory and can result in a new way of indexing it in memory.

The process of elaborating and redescribing the problem specification corresponds closely to the process of *situation assessment* in case-based reasoning: redescribing a problem situation in the vocabulary of problems solved in the past (i.e., the indexing vocabulary of the reasoner's memory). These processes facilitate retrieval in compensating for the fact that we may not be able to anticipate how we might want to use some piece of knowledge when we enter it in memory. Situation assessment aligns the vocabularies of the current situation with that of previous problems we have encountered. Also, by providing several different ways of describing a problem and what would count as a solution, it allows entities to be reinterpreted in the context of the problem and serendipitously recognized as relevant to solving it.

Research into situation assessment and problem reformulation (e.g., in CASEY (Koton, 1988), CYRUS (Kolodner, 1983), MINSTREL (Turner, 1994), BRAINSTORMER (Jones, 1992), and STRATA (Lowry, 1987)), show different ways this can be done. Also, Sycara and Navinchandra (1989) have identified several index transformation techniques relevant to case-based design. We are building on and extending these ideas, exploring in particular how they can be synergistically integrated with evaluation, retrieval, adaptation, and assimilation processes.

Analysis of Examples

Bending Springs

For the designers to recognize the toilet paper holder as a solution to the problem of bending springs, they needed to create a

description of what solutions to this problem would look like. This description evolved as they thought about the problem, proposed solutions to it, and critiqued these solutions. Here is a closer look at how this description evolved.

As the designers were comparing the strengths of various springs, one designer compressed the springs between a thumb and forefinger and noticed that the spring bent, imposed lateral forces at the endpoints, and a variable longitudinal force. This was judged to be a problem. We hypothesize that this judgment was made based on a violation of the designers' expectations about how the spring would behave and reasoning about the consequences of the actual behavior in the context of their proposed design.

One designer wrapped a hand around the spring to hold it straight as it was compressed and said the springs would each need to be enclosed in a tube to keep them from bending. Wrapping a hand around something to make its shape conform to what you want is a standard technique. The subsequent tube proposal can be the result of a memory retrieval based on structural shape similarity.

Another designer added that the tube would need to be collapsible (to compress with the spring). This adaptation may have been suggested as a result of noticing that the hand wrapped around the spring hindered the compression of the spring because it was too longitudinally rigid. This could be fixed by making the tube longitudinally flexible (collapsible).

Weighting Rubberbands

In the bending springs example, the designers derived a concrete description of what they needed, which primed them to recognize it when they saw it. In our second example (the problem of weighting the rubberbands inside the cylindrical egg carrier), the description of what was needed did not fully evolve before the recognition (of eggs as weights) occurred. Rather, the recognition itself helped to redescribe the problem.

The original design, in which batteries were used as weights, was rejected because the batteries would add weight to the device. Their problem description - "find something to act as a weight without adding weight" - was overconstrained. So the problem was abandoned.

They went back to the optimization problem they were given for homework, which involved thinking about the tradeoff between launch force and egg capacity (the more eggs, the more force required because the eggs would increase the weight). Considering the eggs as providing weight prompted one designer to suggest the clever idea of using the eggs as weights on the rubberbands.

Our hypothesis is that the designer saw the relevance of the eggs to the weighting rubberbands problem due to its weight property, which they were focusing on in the homework. The eggs were different than the previously proposed solution (batteries) with respect to the weight property, since egg weight does not count into the total weight limit of the device. (The problem statement explicitly restricted the "weight of the device (not including the eggs).") This difference was noticed and seen to be a key advantage. It generated a refinement of the problem description. Instead of "provide weight, without adding weight," it became "provide weight, without adding weight that counts toward the weight limit." It is only by bringing the eggs into focus and re-interpreting them from the point of view of their weight that this other problem description was created. If the designers had redescribed the problem in this way to begin with, they might have immediately recalled eggs as a solution to this problem. However, the redescription seems to have been the result of the recognition rather than a prerequisite of it.

Modeling Implications

A number of interesting issues arise in considering how to model the problem evolution and assimilation occurring in these two examples.

1. Experimentation plays an important role in problem evolution. Its results are used in evaluating proposed designs and in suggesting solutions and adaptations (e.g., wrapping hand led to tube suggestion; compression hindered by wrapped hand led to the "collapsible" adaptation). Simulating or actually performing this type of experimentation (e.g., with a robot) is, of course, outside the scope of our modeling efforts. But we can provide the results of experimentation as input to our computational model.

2. As proposed solutions are generated and explored (e.g., by collecting experimental data about them), an evaluation process notices their problems (e.g., constraint or expectation violations) or good features. New evaluative issues emerge that go beyond the stated constraints on the problem. Navinchandra (1991) calls this *criteria emergence*. In addition, constraints in general (Bhatta, Goel, & Prabhakar, 1994) and relative priorities among them, also gradually emerge. This emergence is a major part of problem evolution.

3. The designers in our examples generated descriptions that allowed immediate recognition of satisficing solutions to the pending problems. They were concrete enough to be easily recognizable (for minimum inference at recognition-time) but abstract enough to be satisfied by a variety of different objects. They referred to both *structural* (cylindrical shape, length, radius, or weight) and *behavioral* (length varies) properties of objects. Matching these properties against objects under consideration sometimes requires drawing on knowledge about the object (e.g., what configurations or shapes it can take, whether it can stand on end, or whether its length can vary). This in turn requires that the object being viewed has been recognized as its usual identity (e.g., toilet paper roll holder) so that the associated structural, behavioral, and functional knowledge of the object can be matched against the evolved description of what is needed. In our computational model, the input from the environment is augmented by the equivalent of results of standard object recognition techniques, so that an object under view has both its current imagistic features and standard knowledge about its assumed structure, behavior, and function.

Computational Model

Based on our analysis of these examples and others from our exploratory study of the ME design project, we are constructing a computational model of serendipitous recognition. We are implementing this model in a system called IMPROVISER (Invention Modeled by Problem Redescription, Observation, and evaluation, Interacting SERendipitously).

IMPROVISER's proposed architecture has a problem evolution component which is modeled using situation assessment

procedures co-routined with evaluation techniques. The output of this component (i.e., the evolving specification) feeds into memory retrieval and update processes.[1] Retrieval interfaces with a library of design cases which models, in part, long-term memory. The specification is used as a probe to recall relevant design cases (for evaluation, elaboration, etc.).

Memory update is the complement of retrieval. It accumulates design alternatives proposed (i.e., those retrieved, elaborated, or viewed directly in the external environment) into a pool of design alternatives under consideration. It organizes and compares the alternatives with respect to each other, along the dimensions relevant to the problem specification. This is used to model the assimilation process. Recognition of a solution results when an alternative is stored that is a close match to the desired solution.

The data structure holding the set of design alternatives forms an extension of the long-term memory. We call this extension the *problem context*. It contains the set of explored design alternatives and the relevant set of descriptors from the specification. As the problem specification evolves, the focus changes on the relevant descriptors to be used for organizing alternatives in the memory (e.g., shape, construction cost, personal safety). For complex problems, with many subproblems, there are several subproblem contexts, which might overlap, depending on interactions between subproblems. When an alternative is entered into memory, it is interpreted with respect to the descriptors in the subproblem contexts to find the best place(s) to store the alternative.

The various subproblem contexts can be seen as dynamically constructed models of desired solutions, built during problem evolution. Recognition of instances of these models occurs as alternatives are entered into the most appropriate contexts through standard memory update techniques.

A key part of this assimilation process is noticing "interesting" similarities or differences between alternatives being added to some context. For instance, in the weighted rubberbands example, eggs were noticed to be different than previous proposals to the moving cylinder problem with respect to the weight property – one of the descriptors in the problem contexts for both the homework assignment and the moving cylinder problem. Knowing that the weight of eggs is exempt from the weight restriction makes the noticed difference interesting; it is directly related to the conflicting weight constraint, suggesting that this constraint should be re-evaluated with the eggs acting as weights. The success of this evaluation subsequently causes the weight constraint to be refined.

A set of monitoring procedures are associated with each process and watch for opportunities for further processing to occur. The opportunities noticed are placed on an agenda, maintained and accessed by strategic control heuristics.

A Proposed Scenario

This section gives a scenario of how IMPROVISER will model our bending springs example, once fully implemented. IMPROVISER starts with a partial specification which includes specifications for each subproblem in the current partitioning of the problem (launching, moving, stopping, and protecting the eggs). The launch subproblem specification contains a partial specification for a spring launch mechanism. (In the following, "??" denotes incompletenesses due to pending decisions; "..." denotes parts of the specification not shown.)

```
<Spec:
Subproblem: Launch
  Parts: Spring, Base
  Attached(Spring, Base, <position>)
  Spring:
    k: ??  ;; force constant
    x: ??  ;; spring displacement
  Launch-Force: (- (* k x))  ;; Hooke's law
  ...
Subproblem: Protect-Eggs ...
Subproblem: Transport ...
Subproblem: Stop ...
...>
```

This specifies that the launch mechanism should consist of two parts, attached to each other in a particular configuration (given in <position>). There is a pending decision as to the choice of spring strength (k) and how much it should be compressed (x) to achieve a certain launch force. (There are several other constraints involving the launch force which are not shown, such as constraints relating striking distance and launch force or relating the type and amount of egg protection material with the launch force it must cushion.)

IMPROVISER chooses to work on the pending decision concerning spring strength.[2] It asks an oracle to perform a trial-and-error experiment for it to help choose a spring from a set of springs. The oracle compares several springs and feeds back experimental data to IMPROVISER. The data is augmented with causal information about how the data resulted from the properties of the partial design. The oracle reports that the spring bends, which causes it to exert a weak, variable force in the direction of its axis and additional forces of variable magnitude and direction.

In general, there may be many results reported by the oracle. An important issue we are dealing with is how IMPROVISER directs its focus of attention to particular pieces of experimental data. In this case, the focus was on the launch mechanism and its force, so it is natural that IMPROVISER would attend to facts related to forces from the spring.

IMPROVISER notices that these results are not what is expected. It derives their consequences based on causal connections between constraints in the specification. The force along the spring's axis is weaker than the ideal launch force ($F = -kx$), which will make the device move slower and not as far as desired. The additional forces in various directions may cause an inconsistent, unpredictable motion.

IMPROVISER's evaluation monitors detect the negative consequences and update the specification to prohibit their causes, specifying that the spring in the launch mechanism must stay straight. In general, discovering the constraints to add to the specification which will require or prohibit some observed feature of a device involves reasoning based on a causal model of the device (Bhatta, Goel, & Prabhakar, 1994).

[1] This section sketches only the main data flow relationships between these processes. IMPROVISER has a flexible, opportunistic blackboard-style architecture which is guided by explicit strategic control mechanisms.

[2] How this decision is chosen, and, in general, how IMPROVISER's sequence of steps is controlled is an interesting modeling issue. However, it is not the subject of this paper, but see (Kolodner & Wills, 1993; Wills & Kolodner, 1994).

A standard method of forcing a small object to maintain a desired shape is by holding it in that shape with your hand. IMPROVISER asks the oracle to do this and the oracle reports that this causes the spring to stay straight.

IMPROVISER evaluates the results, judges the spring staying straight as positive, and updates the specification with the characteristics of the wrapped hand that are responsible. This specification is used to retrieve a standard design object (RIGID-TUBE) to provide those characteristics.

```
<Spec:
Subproblem: Launch
  Parts: Spring, Base, RIGID-TUBE
  Attached(Spring, Base, <position>)
  ENCLOSED(SPRING, RIGID-TUBE)
  Rigid-Tube:
    SOLIDITY: HOLLOW
    RADIUS: (+ RADIUS(SPRING) DELTA)
    LENGTH: (- LENGTH(SPRING) SM-DELTA)
    SHAPE: CYLINDRICAL
    LENGTH-VARIABILITY: CONSTANT
    RADIUS-VARIABILITY: CONSTANT
...>
```

Further experimental data from the oracle reveals that the rigid tube hinders compression of the spring, causing the spring displacement limit to be much smaller than expected.

```
ORACLE:
Length-Variability: Constant
  Causing
   ;; spring displacement is limited by tube length
   Actual-x(Spring) <= Rest-Length(Spring)
                       - Length(Rigid-Tube)
     Causing
      ;; ... much less than ideal displacement
      Actual-x(Spring) << Max-x(Spring)
```

IMPROVISER detects the undesirable limit on spring displacement and derives the consequences that the launch force will be weaker than is ideal ($F = -kx$), which will affect device speed and striking distance.

IMPROVISER reasons about the causes of the hindrance and updates the specification to require the tube to be collapsible (i.e., allow its length to vary). In other words, IMPROVISER refines the rigidity criterion to what is really needed - the radius to remain constant (lateral-rigidity) and the length to vary (longitudinal flexibility). This requires getting more detailed causal knowledge (not shown) from the oracle about what causes the spring to stay straight, to make sure it won't hurt to vary the length.

```
<Spec:
Subproblem: Launch
  Parts: Spring, Base, Rigid-Tube
  Rigid-Tube:
    ...
    LENGTH-VARIABILITY: VARIES
    Radius-Variability: Constant
...>
```

Using this specification as a probe to memory, IMPROVISER tries to recall a device that satisfies this specification. The retrieval fails.

Since no viable options are found, IMPROVISER suspends work on the launch subproblem and switches to a different subproblem: how to protect the eggs.

```
<Spec:
Subproblem: Protect-Eggs
  Parts: Cushioning-Material
  Cushioning-Material:
    Pressure-Resistance: Soft
    ...>
```

While looking for objects that satisfied this description, a toilet paper holder is observed through an oracle. The observation is a mix of image features and knowledge about the holder, once it has been identified through object recognition.

```
EXTERNAL OBSERVATION: TPH
  Structural properties:
    Parts: Cylinders C1 and C2, Spring S
    Associated-Part: Wall-Fixture
    Fits-Inside(C1, C2)
    C1: Solidity: Hollow
        Length: 1/2(Rest-Length(S))...
    C2: Solidity: Hollow
        Length: 1/2(Rest-Length(S))...
    Composition of Cylinders (C1C2)
      Solidity: Hollow
      Length: Length(S) + Delta3
      Shape: Cylindrical ...
    S: Rest-Length > Width(Wall-Fixture)...
    Enclosed(S, C1C2)
  Behavioral properties:
    States: Steady, Squeeze, Rest.
    Steady:  ;; btwn sides of wall fixture
      Length(S) < Rest-Length(S)
      x(S) = Length(S) - Rest-Length(S)
      Force-Btwn-Endpts: (- (* k(S) x(S)))
      Length(C1C2) = Width(Wall-Fixture)...
    Squeeze:  ;; being compressed
      Length(C1C2) < Width(Wall-Fixture)
      Length(C1C2) >= Max(Length(C1),
                          Length(C2)) ...
    Rest:  ;; outside wall fixture
      Length(C1C2) > Width(Wall-Fixture)
      Length(S) = Rest-Length(S)...
  Functional properties:
    Use: Hold paper roll
```

The most relevant problem context is retrieved, indexed by descriptors that are relevant to solving its pending problem (e.g., size and cylindrical shape of tube, spring enclosed in tube). The context contains the descriptors given in the specification for the problem, the options that have been proposed, and the degree to which each matches the descriptors. In this case, the problem context associated with the launch subproblem is retrieved and the paper holder is assimilated into it, based on the structural, imagistic properties of the paper holder and the knowledge associated with it. During this assimilation, IMPROVISER must check whether the tube length can vary by referring to the behavioral knowledge associated with the paper holder to see if the length of the composition of cylinders (C1C2) changes over the states.

A monitor of the assimilation process notices that the paper holder fits the subproblem specification better than any previous option, particularly with respect to variable length.

The results of this process are 1) a problem specification that has been elaborated (with the constraint that the spring must be enclosed in a tube) and refined (with finer-grain constraints on the rigidity properties of the tube), and 2) new knowledge learned about the functions of a common object (i.e., a toilet

paper holder has a new, additional function of keeping its internal spring straight).

Status and Open Issues

Our system currently contains implemented procedures for assimilation of alternatives into a single problem context, evaluation based on specification constraints, and standard memory indexing and retrieval, as well as data structures representing the case library, problem contexts, the evolving problem specification, and the opportunity agenda. Simple monitors surrounding the assimilation process have been implemented, but more are needed for this and the other processes. We currently do not have a general model for what makes some difference or similarity that is noticed "interesting." We are intrigued by the fact that objects can be noticed as being interesting and relevant to a pending problem before their relevance to the problem is fully understood.

As we extend IMPROVISER to handle multiple problem contexts, we need to deal with issues about how to maintain them. For example, how do they decay? What influences which ones are active (e.g., recency, interaction between the problems)? How do they change as related problems are worked on? How does knowledge of functional properties of an object inhibit the retrieval of a relevant problem context in which the object can be used in a new way. (This is important in modeling functional fixedness.)

More work is also needed to identify and define situation assessment procedures, elaboration techniques, and strategic control heuristics. We are also starting to understand how criteria, constraints, preferences, etc., emerge during evaluation, but more effort is needed in modeling this emergence.

Our intention in building IMPROVISER is not to automate design, but to test our hypotheses about the cognition of creative design. As we increase our understanding of creative processes, we will be better able to answer the question how best to assist human designers. This may include 1) aiding the formalization, reformulation, and refinement of specifications (Reubenstein & Waters, 1991; Johnson, Benner, & Harris, 1993), 2) bringing up evaluative issues (Domeshek & Kolodner, 1993), 3) retrieving pending problem contexts to help recognize the applicability of solutions, or 4) proposing new control strategies.

Acknowledgements

We appreciate the insights we have gained from Terry Chandler, Eric Domeshek, Alex Kirlik, Nancy Nersessian, Ashwin Ram, Mimi Recker, and our anonymous reviewers. We would like to thank Otto Baskin, Jon Howard, and Malisa Samtinoranont, for their invaluable cooperation. This research was funded in part by NSF Grant No. IRI-8921256 and in part by ONR Grant No. N00014-92-J-1234.

References

Bhatta, S., Goel, A., & Prabhakar, S. (1994). Innovation in Analogical Design: A Model-Based Approach. In *3rd Int. Conf. on AI in Design*. Lausanne, Switzerland.

Domeshek, E.A., & Kolodner, J.L. (1993). Using the points of large cases, *AI EDAM*, 7(2), pp. 87-96.

Duncker, K. (1945). On problem solving. *Psychological Monographs*, 58(270).

Grimson, W.E.L. (1990). *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press.

Hammond, K. (1989). Opportunistic memory. In *IJCAI-89* (pp. 504-510). Detroit, MI.

Johnson, W.L., Benner, K.M., & Harris, D.R. (1993). Developing Formal Specifications From Informal Requirements. *IEEE Expert*, 8(4), pp. 82-90.

Jones, E.K. (1992). The Flexible Use of Abstract Knowledge in Planning. (Tech. Rep. 28). Doctoral dissertation. Northwestern University, Institute for the Learning Sciences.

Jordan, D.S. & Shrager, J. (1991). The role of physical properties in understanding the functionality of objects. In *13th Cognitive Science Conference* (pp. 179-184). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kolodner, J.L. (1983). Reconstructive Memory: A Computer Model. *Cognitive Science*, 7(4), 281-328.

Kolodner, J.L. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan-Kaufman Publishers, Inc.

Kolodner, J.L. & Wills, L.M. (1993) Paying Attention to the Right Thing: Issues of Focus in Case-Based Creative Design. *AAAI Case-Based Reasoning Workshop* (pp. 19-25).

Koton, P. (1988). Reasoning about evidence in causal explanation. In *AAAI-88*. MIT Press.

Lowry, M. (1987). The Abstraction/Implementation Model of Problem Reformulation. In *IJCAI-87* (pp. 1004-1010). Milan, Italy.

Maier, N.R.F. (1931). Reasoning in humans II: The Solution of a problem and its appearance in consciousness. *Journal of Comparative Psychology*, 12, 181-94.

Maier, N.R.F. (1970). *Problem Solving and Creativity: In Individuals and Groups*. Belmont, CA: Brooks/Cole Publishing Company, Study 14, pp. 162-175.

Mueller, E.T. (1990). *Daydreaming in humans and machines*. Norwood, NJ: Ablex Publishing Corporation.

Navinchandra, D. (1991). *Exploration and Innovation in Design: Towards a Computational Model*. Springer-Verlag.

Patalano, A., Seifert, C., Hammond, K. (1993). Predictive encoding: Planning for opportunities. In *15th Cognitive Science Conference*, (pp. 800-805). Lawrence Erlbaum

Reubenstein, H.B. & Waters, R.C. (1991). The Requirements Apprentice: Automated Assistance for Requirements Aquisition. *IEEE Transactions on Software Engineering*, 17(3), pp. 226-240.

Schank, R. (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. Cambridge University Press.

Seifert, C., Meyer, D., Davidson, N., Patalano, A., & Yaniv, I. (1994). Demystification of Cognitive Insight: Opportunistic Assimilation and the Prepared-Mind Perspective. In R.J. Sternberg and J.E. Davidson (eds.), *The Nature of Insight*. Cambridge, MA: MIT Press. Forthcoming.

Sycara, K. & Navinchandra, D. (1989). Representing and Indexing Design Cases. In *Proceedings of the Second International Conference on Industrial and Engineering Applications of AI and Expert Systems* (pp. 735-741).

Turner, S.R. (1994). *MINSTREL*, Hillsdale, NJ: Lawrence-Erlbaum Associates, Inc. Forthcoming.

Wills, L.M. & Kolodner, J.L. (1994) Towards More Creative Case-Based Design Systems. In *AAAI-94*. Seattle, WA.