A Process Model of Experience-Based Design

Katia P. Sycara and D. Navinchandra

The Robotics Institute, Carnegie Mellon University

ABSTRACT

Human designers use previous designs extensively in the process of producing a new design. When they come up with a partial or complete design, designers perform a mental simulation to verify the design. We present a model for engineering design that integrates case-based reasoning and qualitative simulation. The model involves: (1) setting up the functional requirements, (2) accessing memory to retrieve cases relevant to the requirements, (3) synthesizing pieces of cases into designs, (4) verifying and testing the design, and finally, (5) debugging. This process is applied recursively till the design is complete and bug free. The model integrates different levels of representation and reasoning mechanisms in order to effectively support the design tasks.

INTRODUCTION

Design is the act of devising an artifact which satisfies a useful need, in other words, performs some function. Design is a complex task that challenges human creativity. This is particularly true in our domain of interest: engineering design. Underlying the design task is a core set of principles, rules, laws and techniques which the designer uses for problem solving. His expertise lies in his ability to use these techniques to produce a feasible design. The designer's expertise is a consequence of his experience and training, much of which is based on previous exposure to similar design problems (Pahl & Beitz 84).

Research investigating the role of experience in problem solving domains that involve understanding the behavior of physical devices has primarily focused on diagnosis (e.g., (Lancaster 88)). Engineering design is a domain that involves not only understanding of device behavior so as to recognize and explain faults but also conceiving and synthesizing device components. Previous AI research in engineering design (Mostow & Barley 87) has advocated the hierarchical decomposition of a design and re-use of plan steps that realize the functional specifications of the components. This technique is promising for domains such as software or circuit design because in these domains designs can be characterized as collections of weakly interacting functional modules, each of which implements one of the functional requirements. Good mechanical designs on the other hand are highly integrated, tightly coupled collections of interacting components. Moreover, an artifact (or artifact component) can be used to satisfy more than one function. These observations imply that a problem solver needs to have access to previous cases (or case pieces) as well as previous plans in terms of operators, preconditions and effects. In this paper, we present a model of engineering design that integrates Case-Based reasoning and qualitative reasoning to come up with new designs.

CASE-BASED REASONING AND ENGINEERING DESIGN

Case-Based Reasoning (CBR) is the problem solving paradigm where previous experiences are used to guide problem solving (Kolodner et al. 85, Sycara 87). Cases similar to the current problem are retrieved from memory, the best case is selected from those retrieved and compared to the current problem. The precedent case is adapted to fit the current situation, based on the identified differences between the precedent and the current case. Successful cases are stored so they can be retrieved and re-used in the future. Failed cases are also stored so that they will warn the problem solver of potential difficulties and help recover from failures. If a current case has features similar to a past failure, then the problem solver is warned not to attempt the failed solution. After the problem is solved, the case memory is updated with the new experience. In this way, learning is integrated with problem solving.

For design, case retrieval is done based not just on surface features but also on (a) the qualitative behavior of the device depicted in the case, (b) the causal relations in the explanation of the device's functions, and (c) device topology. To support retrieval, cases need to be represented at several levels ranging from a topological description of the device objects to a linguistic specification of function. At the intervening levels causal explanations of the device behavior and feature relations have to be incorporated. These levels capture the "mechanism", "causality", and "purpose" perspectives used by people to understand physical systems (White 88).

The mechanical design domain has several characteristics that make the problem very complicated and impose a set of requirements on a reasoner.

- During the design process, a designer transforms an abstract functional description for a device into a physical description that satisfies the functional requirements. In this sense, design is a transformation from the functional domain to the physical domain. In order to effect this transformation, a designer needs to reason at different levels of abstraction ranging from the physical to the functional.
- Good mechanical designs are often highly integrated, tightly coupled collections of interacting components with no obvious decomposition of the overall function into subfunctions. Previous cases represent good solutions to these interactions and can be profitably used.
- The initial functional description of the artifact is usually underspecified so that a designer needs to identify information "gaps" during the design process and generate new problem solving subgoals to resolve them.
- A complete design is synthesized from solutions to subproblems that capture desired subfunctions of the artifact. In engineering design, decisions relating to how certain functions are achieved might be taken at a linguistic or qualitative level. Considerable complication arises from the fact that although a design might be verified to be correct at these levels, simulation at the physical level might fail. The problem solver must synthesize snippets at one level of abstraction while making sure the parts will work together in physically correct ways.
- A design needs to be verified to ensure it meets its functional specifications. In engineering design, verifying that the component parts meet their specifications does not guarantee that the design as a whole will meet its specifications. Thus, both partial and complete designs must be verified.

Our process model addresses all the above requirements.

CASE REPRESENTATION

In dealing with physical systems a reasoner needs to retrieve cases based not only on the *physical attributes* of a device but also on its *functional behavior*. The case based problem solver should be able to work at several levels of abstraction ranging from the physical to the functional level. For example, while trying to produce a design to perform a particular function, a functional description may be used to retrieve cases. However, using the case to physically synthesize the design involves extracting appropriate physical features from the case. This requires that the representation be able to capture the relationship between physical form and qualitative function. We now present in some detail the types of representation used in our work.

1. ***Linguistic Description.*** Linguistic representations are best suited for direct indexing based on a matching linguistic cue. Case attributes provide such indices. Features that capture the physical description of the device (object features) need to be included. For example, a simple household water

tap can be indexed in terms of its function, to control water flow; its components, pipe, nozzle, handle, valve and seal; the material out of which it is made, brass; the type of device it is, mechanical; the places where it is intended to be used, kitchen, bathroom, water tank.

2. ***Functional Block-Diagramming.*** Devices can be viewed as black-boxes which take inputs and produce desired outputs. In the physical domain, three types of inputs and outputs have been identified: signals, energy and materials (Pahl & Beitz 84). A characterization of the relationships between the input and the outputs is the device behavior. In our example, the tap takes a material input (water) and outputs the water in response to the signal (open/close). The tap takes the input signal *Sn*, and the input water flow rate of *Qin* and produces the output flow rate of *Qout*. The temperatures of the input and outputs are *Tin* and *Tout*. The tap has the following qualitative relationships: (1) The inflow of water (*Qin*) monotonically increases with the signal *theta*. (2) The inflow is equal to the outflow (*Qin* = *Qout*), (3) Temperature does not change. (4) When *theta* is zero, there is no flow through the tap, and (5) When *theta* is 2π, then the flow is maximum. The device description at this level of detail does not capture the underlying behavior. This is done with the aid of causal explanations represented as acyclic graphs.

3. ***Causal Graphs.*** A causal graph is composed of links and nodes, where the nodes represent device objects and their attributes, while the links represent causal relations among the attributes. For design cases we use an augmented causal graph representation in which causal relations have qualitative equations (Forbus 84, Kuipers 86) associated with them.

4. ***Qualitative states and Configuration Spaces.*** The causal relationships that describe the behavior of an artifact refer to specific artifact components and relate status conditions such as position and size of the components. The next step is to associate the objects and their status directly to the object geometry. In our work, this is done through Qualitative State descriptions. Qualitative states provide a vocabulary for describing the device behavior. Transitions between the states in the vocabulary are expressed in the causal explanation. For example, the tap can take three states that are qualitatively significant: closed, partially open and fully open. These states provide limit cases for qualitative simulation of the causal network.

PROBLEM SOLVING STEPS

This section presents the steps of Case Based Problem Solving for Design. The process is recursively applied as new subgoals are generated during problem solving. Using the multi-level representation presented in section 3, the problem solver can use cases to address new design problems while switching from one representation to another as needed. This section provides details about the problem solving steps of our approach. We illustrate our approach with the aid of an example. The example is about the design of a Hot&Cold water Faucet which allows control of the mix of hot and cold water independently of the rate of flow of the mixture. The problem solving steps are as follows:

1. Development of a Linguistic Description. Surface features of the problem are determined. In design, surface matching has a lot of validity since similar form often embodies similar function. Several studies (Ratterman 87, Faries 88) have found that surface features have a major influence on the possibility of a case-based reminding. Surface matching is tried first. If a failure to get any useful information out of the matched precedents occurs, then structure matching is attempted.

A description of hot-water faucet is: a device which mixes hot and cold water allowing independent control of the temperature and rate of flow of the mixed water. The index attributes are hot, cold, water and flow-control. Contexts of usage are water, bath, kitchen, bar.

2. Describing the required Function. At the simplest level, the desired artifact can be viewed as a black-box which takes certain inputs and produces desired outputs. The function of the black-box is

described by qualitative relations explaining how the inputs and outputs are related. This function is provided by the user. It is the system's job to help realize an artifact which will convert the inputs into the desired outputs.

A functional black-box diagram of the faucet is shown in Figure 1. The qualitative functions of the faucet are: (1) A signal *Sm* controls the mix temperature *Tm* monotonically: (*Tm M+ Sm*). (2) The rate of flow of the mix (*Qm*) is controlled by the signal *Sf* monotonically: (*Qm M+ Sf*). Certain constraints have to be satisfied too: (1) When there is no flow, the temperature of the mix is zero, in other words, (*Tm M+ Sm*) does not hold. (2) The temperature of the mix is determined by: $Tm.Qm = Th.Qh + Tc.Qc$, where *Qm*, *Qh* and *Qc* are the flow rates of the mix, hot and cold streams and the *T*'s are the corresponding temperatures. (3) Mass is conserved through the system: $Qh + Qc = Qm$.

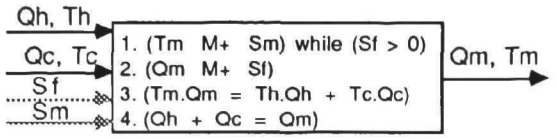

Figure 1: A Qualitative Description of a Faucet

3. Problem Analysis to Identify new Relations. Using the given functional description may not yield relevant cases. It is useful to analyze the given problem in order to generate new relations between inputs and outputs. New relations can be generated by propagating relations using simple combination and propagation rules.

For example, in the faucet problem we are given that the temperature of the mix increases with the mix signal *Sm*. From the equation for calculating the temperature of the mix *Tm* one can qualitatively infer that when there is some flow in the system ($Qm > 0$), then the rate of flow of hot water *Qh* monotonically increases with *Sm* while the rate of flow of cold water decreases. In other words, rate of flow of hot water is inversely proportional to the rate of flow of cold water (*Qh M− Qc*) and the rate of flow of cold water decreases with *Sm*: (*Qc M+ −Sm*) . Other relations which are derived from the given qualitative relations are that the flow of hot water monotonically increases with the signal *Sf* , the same is true for the flow of cold water.

4. Retrieval of Cases. A set of design cases (or case parts) bearing similarity to a given collection of features are accessed and retrieved. Similarity is determined using not only the existing features of the input specification, but also perturbations arising from index generation and reformulation strategies (Sycara & Navinchandra 89) as well as the derived qualitative relations from the above step.

Using the derived relations (*Qc M+ −Sm*) and (*Qh M+ Sm*) the common tap (described in the previous section) is retrieved since it is a water regulator which will change flow in response to a signal. Given that there is only one *Sm*, there has to be some way of splitting the signal into two signals of opposite sense. As it is not known how this function will be realized, it is treated as a new black-box (black-box1) in Figure 2. The figure shows another black-box (black-box2) which takes two flows and merges them. At this point in the problem solving, the system does not have, as yet, any way of controlling the total flow with *Sf*.

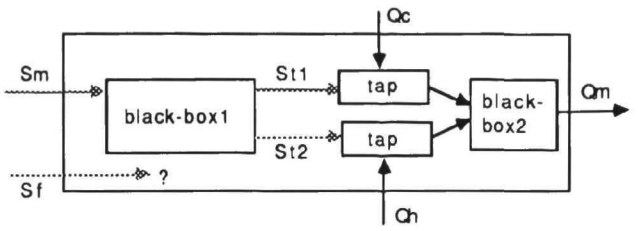

Figure 2: Functional Level Synthesis of the Faucet

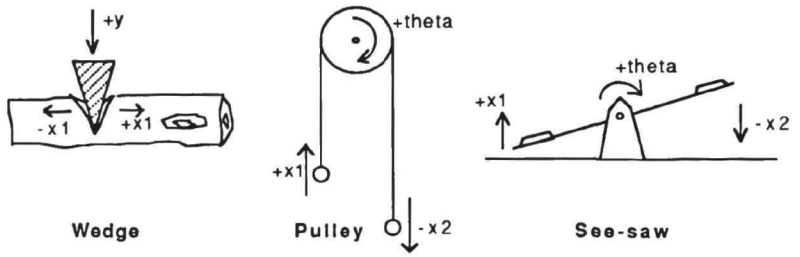

Figure 3: Cases showing one parameter increasing while another decreases

The next step is to find a way of realizing the split of the signal *Sm*. The required function may be used as an index into memory. In this situation, the index would be: "given a signal *Sm* the flow rate of one stream goes up while that of another goes down to maintain a constant total flow rate". For the given case memory, this index is overly specific and thus it fails to retrieve cases. The index is generalized to its corresponding qualitative statement: "given some signal one quantity goes up while another comes down proportionately". Operationally the index is given as:
(*Quantity*1 *M+ Signal*) $\wedge$ (*Quantity*2 *M− Signal*). This generalization retrieves the cases shown in Figure 3 where one parameter $x1$ increases while another $x2$ decreases given a signal (either of *theta* or y).

5. Extraction of relevant "snippets" from cases. A designer may retrieve and adapt not only whole cases but also pieces of cases that might embody appropriate principles useful for the design task. To access a snippet directly, indices based on features appropriate for the snippet are used; to extract a relevant snippet from a retrieved whole case, the subgoals of the problem solver are that are posted at this point are used (Navinchandra 88).

Let us consider the see-saw case retrieved in the last step. Extraction of the appropriate snippet from the see-saw case requires examining the underlying causal structure of the case. The qualitative relations for the see-saw are:

When $d(theta) > 0$: ($x1$ *M+ theta*) and ($x2$ *M− theta*),
When $d(theta) < 0$: ($x1$ *M− theta*) and ($x2$ *M+ theta*)
where: $d(theta) = d(theta)/dt$ (qualitative differential eqn)
$-theta^* \leq theta \leq +theta^*$ (limits are symmetric)
$0 \leq x1 \leq x^{max}$ and $0 \leq x2 \leq x^{max}$

The angular motion of the see-saw is limited by the ground. Let the limits be $\pm$ *theta**. Correspondingly, the maximum and minimum values of $x1$ and $x2$ are: zero and x^{max}. The algebraic relation among these parameters is: $x^{max} = LSin(\ theta^*\ )$, where L is the length of the see-saw board. Another useful relation is that x^{max} is double the height (h) of the see-saw's fulcrum above the ground: ($x^{max} = 2h$). These algebraic relations are useful in constructing the configuration spaces and for reasoning about parametric adaptation of design snippets. Reasoning about quantitative equations

together with qualitative relations is an important part of design process.

6. Snippet Synthesis. Partial designs have to be combined to produce a complete design. This is a difficult problem since undesirable interactions among snippets may occur. Snippet synthesis is done by ensuring that snippet preconditions are satisfied. Each snippet is treated as a small black-box with known inputs and behavior. Snippets are synthesized by attaching their inputs and outputs appropriately. When input types are incompatible, a new subgoal to find a way to convert one output into a compatible input is spawned. Even though each individual snippet has been tested to satisfy required subgoals, the synthesis process may uncover undesirable interactions at "snippet interfaces". These interactions have to be recognized and fixed. Even after the interactions have been fixed, an additional problem may arise. The synthesized solution might not satisfy the original set of specifications, although each solution component satisfies a subspecification. This characteristic is especially true in design. Thus, verification is required after each synthesis step.

Returning to the faucet example, a see-saw has been retrieved (Figure 3) for proportionate up and down motion. The next step is to find cases which will allow translatory motion to control water flow. Using the tap case's current context the cases shown in Figure 4 are retrieved.

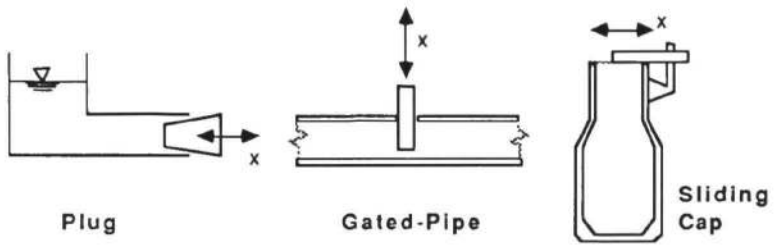

Figure 4: Using translatory motion to control orifice size

Using the causal explanations in the cases, snippets contributing directly to the required functions are retrieved and synthesized. At the qualitative level, the see-saw and the gated-pipe can be synthesized as shown in Figure 2. The see-saw function converts an input mix signal (*Sf*) into two translatory signals *St*1 and *St*2. These two signals are fed into basic "tap"s, one for cold water and the other for hot water. A synthesis of the existing snippets (at the physical level) is done by attaching the signal of one snippet to that of another as dictated by the Functional Level synthesis. An example of a rule that was used is: If the signals are motions, Then their degrees of freedom should match. For example, translation should match translation. The synthesis is shown in Figure 5. The design allows a signal to control the proportion of orifice sizes for the cold and hot water streams.

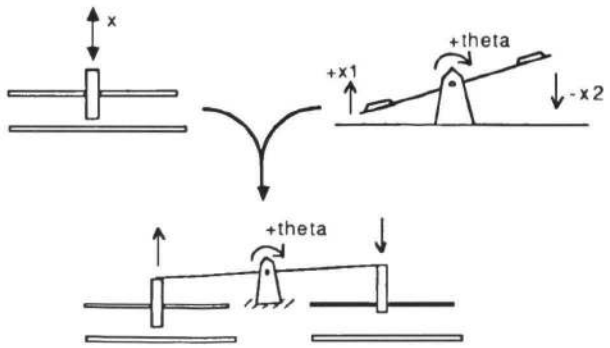

Figure 5: Physical Level Synthesis of the Faucet

*7. **Verification.*** During verification adverse interactions could lead to non-conformance of the design to the desired specifications. This is verified through qualitative simulation. If the simulation is correct, and if all the constraints are satisfied, then the design is successful. If not, debugging (next step) is attempted. A qualitative simulation of the tap discovers some "gaps" in the performance. The partial design generated thus far shows how to control the mix of the flows given the signal *Sm* but not how to control the total *Qm* without changing the ratio of hot and cold water. The design is not complete with respect to the signal (*Sf*) which controls the total flow through the tap (*Qm*).

*8. **Debugging.*** In debugging designs we have found that relevant cases can be retrieved by using the reasons underlying the bug as cues into memory (Navinchandra 87). Roughly speaking, the idea is to reduce a bug into "sub-bugs" which may relate either directly or analogically to cases in memory. The reasons underlying a given bug are determined by developing a causal explanation for the existence of the bug. Debugging involves a process of asking relevant questions and modifying them based on a causal explanation of the bug. These questions serve as cues into memory (Schank 86). When a bug is found, a corresponding question is posed to the Case Knowledge Base (CKB): "Has this, or some similar, bug been seen before? Is there a known way of repairing it?" If a relevant case is not found, the causal reasons for the bug are used to transform the question. For example, if for a given bug X, related cases are not found, then the debugger goes on to ask "What are the causes of X?", "If it is not known how to eliminate X, can its causes be eliminated?". This questioning process is recursively applied until a relevant case is found.

The process in the faucet design example proceeds as follows: The "gap" found in the verification step needs to be "filled". A way needs to be found to control *Qm* with *Sm* without changing the temperature of the mix. One of the causes of *Qm* is total orifice size (this relation is deduced from the causal graph). Consequently, the following question is generated: "How does one now control total orifice size without changing the ratio of hot and cold water flows?" In terms of constraints, it follows that the ratio: ($St1 / St2 = K^1$) has to be satisfied, while the signal *Sf* increases the total flow-rate ($St1 + St2$) monotonically. Correspondingly, in the see-saw case, one needs to achieve ($x1 / x2 = constant$) while ($x1 + x2$ *M+* *Sf*). From the see-saw equations it is known that $x1 + x2 = x^{max}$. It follows that we need a way of achieving (x^{max} *M+* *Sf*). This can be done by working from the see-saw equation: ($x^{max} = 2h$) and finding a way of getting (h *M+* *Sf*). This last goal is easily satisfied. As the height (h) is an independent parameter (no other constraints on it), it can be linked directly to the signal: (h = *Sf*).

Finally, by making correspondences between $x1$, $x2$ and $St1$, $St2$, and by recursively applying the process of retrieval, snippet extraction and synthesis, the final conceptual design of the tap can be developed (Figure 6).

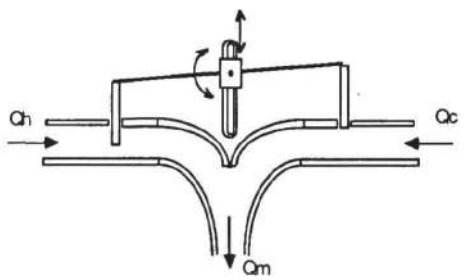

Figure 6: Conceptual Design of a Faucet

[1] A constant

CONCLUDING REMARKS

Problem solving in the domain of Engineering Design imposes a set of requirements on a problem solver: (a) the representation needs to capture and integrate several levels of abstraction from the linguistic to the physical, incorporating linguistic specifications, laws of physics, constraints and tolerances, (b) the problem solver should be able to reason both symbolically and analytically at different problem solving stages and integrate the process and results of its reasoning, (c) verification techniques should be incorporated in the problem solving. To deal with these requirements, we have presented a methodology for design that integrates use of past design cases with qualitative reasoning. Cases are represented at various levels of abstraction, and indices corresponding to these levels allow access to design cases at any of the abstraction levels. Past design cases similar to the current design are used to: focus on the relevant parts of the problem, form the basis for analogical reasoning, avoid past mistakes, and provide guidance in debugging. Qualitative reasoning determines appropriate indices for case retrieval, provides causal explanations of the behavior of an artifact, and forms the basis for verification to check whether the design meets its specifications.

REFERENCES

(Faries 88) Faries, J. M. and Reiser, B.J., "Access and Use of Previous Solutions in a Problem Solving Situation," *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, The Cognitive Science Society, Montreal, Canada, 1988, pp. 433-439.

(Forbus 84) Forbus, K., "Qualitative Process Theory," *Artificial Intelligence*, Vol. 24, 1984.

(Kolodner et al. 85) Kolodner, J.L., Simpson, R.L., and Sycara-Cyranski, K., "A Process Model of Case-Based Reasoning in Problem Solving," *Proceedings of IJCAI-85*, Los Angeles, CA, 1985, pp. 284-290.

(Kuipers 86) Kuipers, B.J., "Qualitative Simulation," *Aritifial Intelligence*, Vol. 29, 1986, pp. 289-338.

(Lancaster 88) Lancaster, J. and Kolodner, J.L, "Varieties of Learning from Problem Solving Experience," *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, The Cognitive Science Society, Montreal, Canada, 1988, pp. 447-453.

(Mostow & Barley 87) Mostow, J., M. Barley, "Automated Reuse of Design Plans," *Proceedings of the International Conference on Engineering Design* , February 1987.

(Navinchandra 87) Navinchandra, D., *Exploring for Innovative Designs by Relaxing Criteria and reasoning from Precedent-Based Knowledge,* PhD dissertation, M.I.T., 1987.

(Navinchandra 88) Navinchandra, D., "Case-Based Reasoning in CYCLOPS, a Design Problem Solver," in *Proceedings of the DARPA Workshop on Case-based Reasoning,* Kolodner, J., ed., Morgan Kaufman, 1988, pp. 286-301.

(Pahl & Beitz 84) Pahl, G., W. Beitz, *Engineering Design,* The Design Council, Springer-Verlag, 1984.

(Ratterman 87) Ratterman, M.J., and Gentner, D., "Analogy and Similarity: Determinants of accessibility and inferential soundness," *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, The Cognitive Science Society, Seattle, Wa., 1987, pp. 23-35.

(Schank 86) Schank, R.C., *Explanation Patterns: Understanding Mechanically and Creatively,* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

(Sycara 87) Sycara, K., *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods,* PhD dissertation, School of Information and Computer Science Georgia Institute of Technology, 1987.

(Sycara & Navinchandra 89) Sycara, K., D. Navinchandra, "Integrating Case-Based Reasoning and Qualitative Reasoning in Design," in *AI in Design,* J. Gero, ed., Computational Mechanics, U.K., 1989.