Proceedings of DETC '01
ASME 2001 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
September 9-12, 2001 Pittsburgh, Pennsylvania

DETC 2001/DTM-21702

REDESIGNIT - A CONSTRAINT-BASED TOOL FOR MANAGING DESIGN CHANGES

Gabriel Aguirre Ollinger
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
gaguirre@andrew.cmu.edu

Thomas F. Stahovich
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
stahov@andrew.cmu.edu

ABSTRACT

RedesignIT is a computer program that uses model-based reasoning to generate and evaluate proposals of redesign plans for engineered devices. These proposals describe how the design parameters could be changed to achieve a specified performance goal. Equally important, the program proposes complementary modifications that may be necessary to counteract the undesirable side effects of the primary changes. RedesignIT is intended for use during the first stages of a redesign project, when engineers need to make a quick, yet accurate assessment of the overall effects of a particular design change. The program uses qualitative device models, which allow it to compute redesign plans efficiently. With its ability to predict the collateral, and probably undesirable, effects of a design change, the program is well suited to aid product designers in deciding on the feasibility of introducing design changes to a product. RedesignIT employs methods of artificial intelligence, especially qualitative reasoning, causal reasoning, and heuristic search.

INTRODUCTION

Customers have ever increasing expectations about the quality and performance of the products that they buy. In order to keep pace with this demand for value, companies in practically every business field have to come up with frequent upgrades of their products. One of the main difficulties in redesigning a product is that a proposed design change can have many undesirable side effects besides the intended ones. In complex engineered devices such as automobiles, these side effects are often hard to predict, especially when they cross the boundaries between the device's subsystems.

Clearly, a manufacturer could save considerable amounts of resources and effort if it were possible to make a quick, yet accurate assessment about the overall effects of a design change, before making a commitment to implementing the change. As a response to this need we are currently developing RedesignIT, a model-based computational tool that generates and evaluates proposals of redesign plans for engineered devices.

RedesignIT is designed for use at the beginning of a redesign task. Its purpose is to aid the designer in answering the following questions:

- What is the complete set of possible changes —within the scope of the model—that can be made in order to achieve the design goal?
- For each of these possibilities, what is the complete list of both certain and probable side effects?
- How can each of the side effects be remedied?
- Which design change has (1) the greatest influence on the design goal and (2) the least cost in terms of implementation and undesirable side effects?

RedesignIT receives as input a model of the device under study. Because the purpose of the project is to help the designer understand how a change to one part of a device generates changes elsewhere in the device, the model we use consists mainly of the relevant physical quantities, and the causal relationships between them. In general, these relationships can be expressed in qualitative, quantitative or semi-quantitative terms. We have found a semi-quantitative representation, based on orders of magnitude, to be especially effective. Such a representation allows comparisons between the effects that different design changes have on the design constraints, without incurring the expense of building a mathematically exact model. The program's task is to use this model to generate several combinations of changes to the quantities, by which the redesign goals, as intended by the designer, can be accomplished. Redesign goals are expressed as a desired

change in the magnitude of one or more quantities. We refer to these as the target quantities, and to the desired change in their magnitude as target changes. Examples of a redesign goal can be an increase in the output torque of a motor, or a reduction in the cycle time of a machine's operation.

The program's output consists of several possible redesign plans by which the target changes can be achieved. A redesign plan is expressed as a combination of changes to some of the exogenous quantities of the system, i.e. quantities whose values can be directly set by the designer. One of the important characteristics of our program is that it is designed to generate redesign plans in which undesirable effects will be kept as small as possible. Along with every redesign plan, the program reports the degree to which the redesign plan can be expected to achieve the redesign goal, as well as the nature and severity of the side effects. The program ranks the proposed redesign plans according to several criteria of effectiveness. RedesignIT has been conceived mainly as a tool for performing a form of "what-if" analysis. It satisfies its purpose by enabling the designers to make a well-informed decision before starting to elaborate a detailed design.

A CASE EXAMPLE

We shall use a concrete redesign problem to illustrate the program's operation, starting with the construction of a directed dependency graph. The problem consists of modifying a four-stroke, turbocharged diesel engine in order to achieve a larger output torque. At the same time, certain constraints like the durability of the engine and the level of emissions must be preserved. A simplified diagram of a four-stroke diesel engine is shown in Figure 1. The diesel engine takes in air, compresses it and then injects fuel into it. The heat of the compressed air ignites the fuel spontaneously and causes the gases in the combustion chamber to expand, driving the piston down.

After identifying the physical quantities that are relevant to the operation of the diesel engine, we make a comprehensive list of causal relationships between those quantities, and use the information to build a directed dependency graph. The graph we made for the program to analyze the engine contained over 80 variables. Figure 2 shows only a very simplified version of the graph; we will refer to this figure in several places to illustrate how the program works. The details of the graph will be explained below, but at this point we can use it to show the kind of redesign plans that are generated by the program.

The redesign task is to modify the engine to achieve a larger output torque (T). The program will search for ways to achieve this goal by identifying quantities in the graph that (1) have a causal influence on T, and (2) whose values can be directly set by the designer. The quantities that have the latter property are called exogenous quantities; they are shown in gray boxes in Figure 2. Upon examining these quantities, the program will find that the length of the piston stroke (Δx) has a causal influence on T (because a longer stroke allows the engine

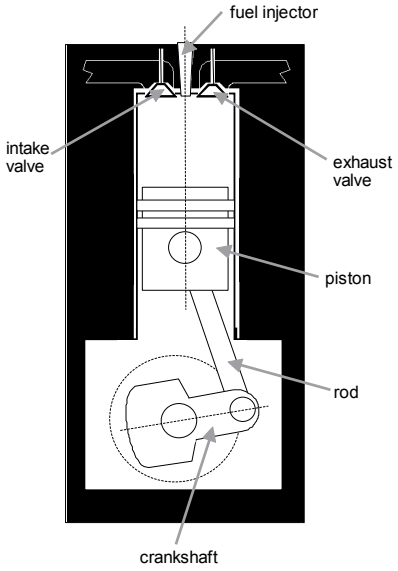


Figure 1. Diagram of a diesel engine.

to admit a greater mass of air during the intake cycle, thereby generating a greater impulse on the piston during the expansion cycle). Given the type of the causal influence (M+, or positive monotonic), the program will conclude that an increase in T can be achieved by increasing the piston stroke. However, increasing the piston stroke will produce changes to several other quantities. These changes, or side effects, include an increase in the friction on the piston, and therefore a reduction in the durability of the engine[1]. Additionally, the emissions of particles can be expected to increase as well. In most cases, effects like these are undesirable because they tend to violate constraints set by the designer or by the physics of the engine. Therefore, the redesign plan does not consist solely of increasing Δx; the program will search for additional exogenous quantities that can counteract the side effects.

The program will gradually complete the redesign plan by proposing additional changes, such as increasing the power of the oil pump in order to improve the lubrication of the piston, and improving the mixing effect of the fuel jet by increasing the injection period. The search for additional quantities ideally terminates when there is a redesign plan in which all side effects are counteracted. However, in some cases the engine's design may not allow a complete elimination of side effects. In such cases the search will terminate when there are no remaining exogenous quantities that can remedy the side effects, or when changing any remaining exogenous quantities exceeds the expected benefit of doing so.

[1] This conclusion is arrived at through the following reasoning: a large piston stroke means that a greater mass of fuel/air mix will be available for combustion. Burning a greater mass of fuel/air mix means a higher output of heat and, consequently, a rise in the cylinder's temperature. A hotter cylinder will increase the distortion of the piston and, as a result, the friction between the piston and the cylinder wall will also increase. Greater friction will increase the wear of both components, and hence reduce the engine's durability.

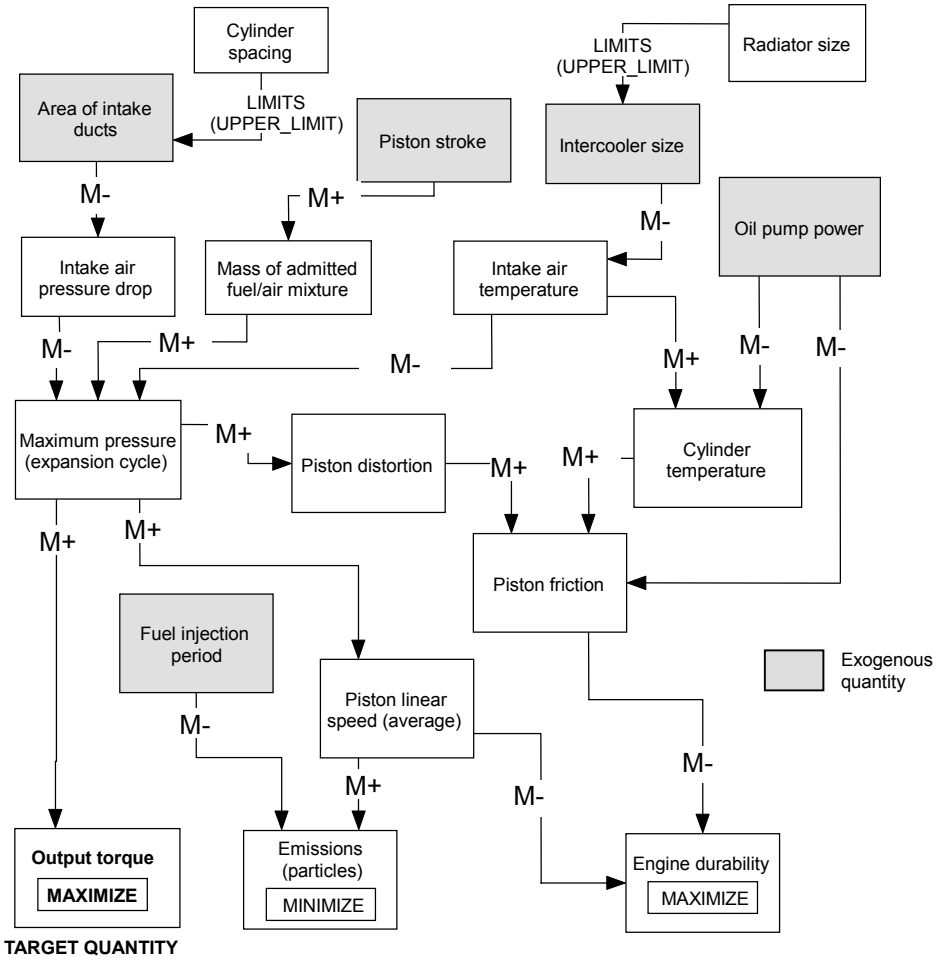


Figure 2. Representation of a diesel engine using a directed dependency graph.

REPRESENTATION OF AN ENGINEERED DEVICE

This section describes the elements that form the representation of a device. These elements are mainly quantities, constraints applied to them and causal influences between quantities.

We use a qualitative measure to describe changes to quantities and magnitudes of causal influences. No attempt is made to give precise numerical values of these magnitudes; instead, we represent approximate notions of magnitude such as a "typical" value, a "large" value and a "small" value. We have chosen certain orders of magnitude (i.e. powers of 10) as landmarks to distinguish between these values. Therefore, the possible magnitudes that can be used in our device representation are:

- ORDER_LOW (small value): magnitudes in the order of 10^{-1} or less.
- ORDER_ZERO (typical value): magnitudes in the order of 10^{0}.
- ORDER_HIGH (large value): magnitudes in the order of 10^{1} or higher.

The use of orders of magnitude enables the program to add and multiply changes as they propagate through the dependency graph, and to make comparisons of facts about quantities, without the cost of implementing a fully detailed, numerical model of the device.

Quantities

The term quantity in our representation refers to both the physical properties of the components of a device and descriptions of the device's operation. Examples of the former are the calorific power of the fuel and the tensile strength of the pistons' material. Examples of the latter are the linear speed of the piston and the durability of the engine.

Because we are interested in modifications to a design, we reason about the magnitude of change rather than the absolute value of a quantity. There are two ways in which the magnitude can be changed. If a quantity is exogenous, its magnitude can be directly changed by the designer. If the quantity is not exogenous, changes to it are a result of causal influences from other quantities that have changed. Ultimately, a change to a non-exogenous quantity is the result of changes to exogenous quantities. A change in the magnitude of a quantity is expressed by the direction, which can be either an increase or a decrease, and an absolute difference that is specified by a qualitative order of magnitude.

Constraints on quantities

A constraint is an expression of a design requirement placed upon a particular quantity. The program uses four basic types of constraints:

- FIXED: a single value is acceptable for the quantity. Any deviation from that value constitutes a violation of the constraint, as in the case of dimensions of standard, off-the-shelf components.
- MAXIMIZE: it is desirable to make the value as high as possible, as in the case of the durability of the engine.
- MINIMIZE: it is desirable to make the value as small as possible, as in the case of the friction.
- RANGE: there is an acceptable range of values for the quantity. An example of this type of constraint would be the temperature of the engine.

Because a constraint of any of the types described above is, in a sense, a property of a specific quantity, we refer to these types as intrinsic constraints.

The concept of constraint on a quantity is especially important, because the effects of a redesign plan are expressed in terms of how constrained quantities are affected. Furthermore, a target change is specified as a MAXIMIZE or MINIMIZE constraint on a target quantity. Such constraints, by definition, are not satisfied by the device's current design, so it becomes necessary to make changes to the device. A redesign plan is considered effective if it makes the target quantity comply with its constraint[2]. In a similar way, the unexpected side effects of a redesign plan are expressed in terms of constraints. A side effect is desirable to the extent in which it produces or increases compliance with constraints on quantities other than the target quantities. A side effect is undesirable if it produces constraint violations.

In modeling constraints, we have included a measure of the importance of a constrained quantity. In the current implementation of the program, we use integer values to signify different levels of importance, with the highest value being assigned to target quantities. These importance factors are used in the calculation of the benefits and costs resulting from a change to an exogenous quantity. (Section "Calculation of cost/benefit of an exogenous quantity change" below describes this issue in more depth.)

Causal influences

Causal influences describe how a change to one quantity affects the other quantities. We currently consider four common types of causal influences: M+, M-, LIMITS (AS_UPPER_LIMIT), LIMITS (AS_LOWER_LIMIT).

[2] In the case of a MAXIMIZE constraint, the target quantity becomes compliant if it is made to increase. A further increase will result in a greater degree of compliance. Similarly, a MINIMIZE constraint is complied with by decreasing the value of the quantity.

The first two, M+ and M-, are similar to the monotonic relationships commonly used in qualitative simulation (cf. Kuipers [3]), except that we also include the notion of the direction of causality. Consider some of the factors influencing the output torque (T) of the engine. A physically precise model for the behavior of T would be given by the following equation:

$$T = \frac{1}{4\pi}(p_b \cdot \Delta x \cdot A_t)$$

where p_b = cylinder mean pressure
Δx = piston stroke
A_t = piston area

This is a mathematical statement that represents part of the physics of the engine. However, it contains no information about causality. For the purposes of causal reasoning, we want a model to capture causal relationships such as "an increase in Δx produces an increase in T", and "a decrease in Δx produces a decrease in T". Equally important is the fact that these relationships are directional; a change in Δx produces a change in T, but a change in T does not produce a change in Δx. These relationships are expressed concisely by the statement T = M+(Δx). In a similar fashion, we find that a decrease in the size of the intercooler produces an increase in the temperature of the intake air. This relationship is expressed as Intake_air_temperature = M- (Intercooler_size).

In the case of M+ and M- relationships, an order of magnitude is specified for the causal influence. An example of a fully specified causal influence is shown in Figure 3; the program will use the magnitude of the change in the "cause" quantity, plus the definition of the M- relationship, to calculate the magnitude and direction of the change in the "effect" quantity.

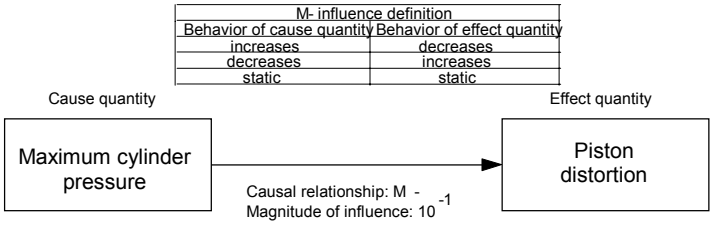


Figure 3. Specification of a causal influence.

The LIMITS relationships describe situations in which the magnitude of one quantity imposes either an upper or lower limit on the value of a different quantity. In some cases, the quantity that produces the LIMITS constraint can be a physical property of a component. For example, the greatest amount of fuel that can be injected into the combustion chamber is limited by the mass of the air in the chamber (given that a certain ratio of fuel to air must be maintained). This relationship is expressed concisely as Air_mass_at_igntion LIMITS_(UPPER_LIMIT) Fuel_mass_at_ignition. In other cases, the LIMITS constraint may be a result of geometry. For example, the spacing between cylinders (see Figure 1) places an upper limit on the area of the air intake ducts. Usually, if a change to a quantity violates a

LIMITS constraint, the result will be damage to the device's operation, damage to a specific component, or both.

The specification of a LIMITS constraint requires the user to give the initial qualitative magnitude (ORDER_LOW, ORDER_ZERO or ORDER_HIGH) of the difference d between the limited quantity Q, and the limiting quantity L. A proposed design change can violate a LIMITS constraint if it causes the absolute value of d to become zero, i.e. if it causes Q to cross the boundary imposed by L; such a violation is considered an undesirable side effect. However, identifying violations of constraints of this type is more involved than in the case of intrinsic constraints (like MAXIMIZE or MINIMIZE). When the program attempts to infer the effects of an exogenous quantity change on a LIMITS constraint, the possible outcomes are compliance with the constraint, constraint violation, or risk of constraint violation. (The next section explains the process for determining the effect of an exogenous quantity change on a LIMITS constraint.)

Directed dependency graph

The complete list of causal relationships in the device model is used to construct the directed dependency graph (see Figure 2). In this graph, the M+ and M- relationships are especially relevant, because they are the relationships that actually propagate magnitude changes from one quantity to another. Therefore, they are the relationships responsible for generating the target changes in the device.

PROPAGATION OF CHANGES

This section describes how changes propagate through the directed dependency graph. If we examine the quantity "piston stroke" (Δx) in Figure 2, we can observe several chains of causally connected quantities that originate with Δx. For example, if Δx changes, the maximum pressure during the expansion cycle will change also (there is an M+ relationship). The latter will in turn produce a change in the output torque.

Chains of this kind are known as causal paths. The existence of a causal path implies that, if a change is applied to the quantity at the start of the path, every quantity downstream in the path will be changed as well. This phenomenon is known as the propagation of a change. Because any quantity in a given device model can have causal relationships with more than one successor, change propagation usually takes the form of a tree propagation. Figure 4 shows the complete propagation tree that starts from quantity piston stroke.

Change propagation is the key concept in developing redesign plans. If a change is applied to the correct quantity, that change will propagate through the device, and ultimately modify the target quantity in the desired way. However, because the causal relationships form a tree, there are almost always side effects. If, for example, the target quantity in our redesign problem is an increase in output torque (T), one can conclude that the target change can be achieved by increasing the piston stroke (Δx). Apparently, our redesign target can be achieved by modifying a single property of the engine. However, it can also be seen that quantities such as the durability of the engine and the particle emissions will change as well, and these side effects may prove to be undesirable. Therefore, one central problem that our program addresses is how to counteract the side effects that arise from those changes that are intended to achieve a redesign goal.

The change propagation mechanism is straightforward. Given a causal pair, the program takes (1) the direction of change in the cause quantity and (2) the type of causal influence involved, and matches them against the propagation rules shown in Table 1 to find the change in the affected quantity. Note that in the case of LIMITS relationships, what changes is not the value of the affected quantity itself, but rather the value of one of its limits.

Causal influence	Change in cause quantity	Change in effect quantity
M+	Value increases	Value increases
	Value decreases	Value decreases
	Value static	None
M-	Value increases	Value decreases
	Value decreases	Value increases
	Value static	None
LIMITS_(UPPER_LIMIT)	Value increases	Upper limit increases
	Value decreases	Upper limit decreases
	Value static	None
LIMITS_(LOWER_LIMIT)	Value increases	Lower limit increases
	Value decreases	Lower limit decreases
	Value static	None

Table 1. Rules of change propagation.

The order of magnitude of the change is found by multiplying the qualitative order of magnitude of the change in the initiating quantity with the qualitative order of magnitude of its causal influence. Because qualitative orders of magnitude are expressed as ranges of values (except for ORDER_ZERO), the result of this multiplication can be a range as well. For example, if the change in the initiating quantity is ORDER_HIGH ($\geq 10^1$) and the causal influence is M-, ORDER_HIGH ($\geq 10^1$), the magnitude of the change in the effect quantity will be, in absolute terms, greater than or equal to 10^2, and therefore will be considered ORDER_HIGH.

Evaluating constraints

As we have discussed previously, one of the program's functions is to determine the effect that the propagation of a change will have on a quantity's intrinsic constraints. A change experienced by a constrained quantity can either be neutral to its constraint, or have any of these effects: increased constraint compliance, risk of violation or definite violation. The program determines these effects by using the rules in Table 2.

To determine if a LIMITS (non-intrinsic) constraint is violated, the program compares the qualitative changes in the quantity (Q) and the limit (L) to the initial qualitative distance (d) between the quantity and the limit. If the changes to Q and L

Fixed		Maximize		Minimize		Range	
Change	Effect	Change	Effect	Change	Effect	Change	Effect
Decreases	Violation	Decreases	Violation	Decreases	Compliance	Decreases	Neutral
Increases	Violation	Increases	Compliance	Increases	Violation	Increases	Neutral
Static	Compliance	Static	Neutral	Static	Neutral	Static	Compliance

Table 2. Effects of change propagation on constraints.

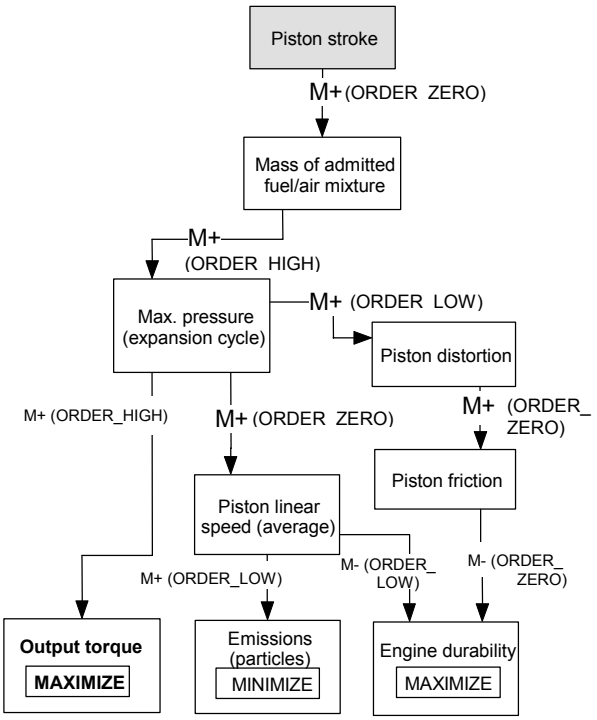


Figure 4. Propagation tree for quantity "piston stroke"

tend to increase the distance between them, there is no constraint violation. If the changes decrease the distance an amount qualitatively equivalent to d, there is a risk of violation. If the distance decreases more than d, then there is a constraint violation.

ALGORITHM FOR CONSTRUCTING REDESIGN PLANS

Relationship between exogenous quantities and constrained quantities

An exogenous quantity is defined as a quantity that is subject to no M+ or M- causal influences, but can be subject to LIMITS influences. The program identifies exogenous quantities by simply traversing the list of quantities in the model, and checking every quantity to see if the above definition applies to it. In our case example (see Figure 2), the following are exogenous quantities: piston stroke, area of the intake ducts, intercooler size, oil pump power, and fuel injection period.

To carry out the redesign task, the program focuses on how the exogenous quantities affect the constrained quantities, including the target ones. Exogenous quantities that have a causal influence on the target quantity are what we call effective exogenous quantities. Examples of effective exogenous quantities in our redesign task (i.e. quantities that have a causal influence on the output torque) are the piston stroke, the area of the intake ducts, and the size of the intercooler.

Constructing a table of entailed causal influences

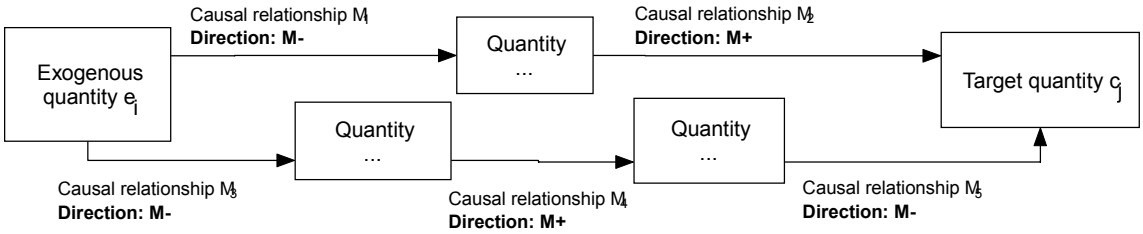
If introducing a change to exogenous quantity e_i can change a constrained quantity c_j, then e_i is said to have an entailed causal influence on c_j. This type of causal relationship can be represented in the same way as a direct casual influence between two quantities (see Figure 3), that is, as an M+ or M- influence of e_i on c_j. For every exogenous quantity, the program builds a list of constrained quantities upon which it has an entailed causal influence, along with the magnitude and direction of the influence. Given an exogenous quantity e_i, the test for finding the constrained quantities influenced by it consists of applying a test change (we choose an increase of ORDER_HIGH) to e_i, and propagating the change throughout the causal influence graph. Any constrained quantity c_j that experiences a change as a result of this propagation can be said to be subject to a causal influence from quantity e_i. This test also serves to find the direction of such causal influence: if constrained quantity c_j experiences an increase, the entailed causal influence is M+, whereas if the change is a decrease, the influence is M-. From the engine's model it can be determined, for example, that an increase in the piston stroke (Δx) will contribute an increase in the target quantity, output torque (T).

The magnitude of the entailed causal influence of exogenous quantity e_i on a constrained quantity c_j is found by first identifying all the causal paths that lead from e_i to c_j. When more than one path exists, we speak of a confluence of causal influences. The program then finds the causal influence magnitude associated with each path; it does so by multiplying the influence magnitudes of all causal links on the path. The total magnitude of the causal influence of quantity e_i on quantity c_j is the sum of the magnitudes computed for each individual path. (See Figure 5.)

Calculating the change on a constrained quantity

The perturbation that a single exogenous quantity e_i causes to a constrained quantity c_j is equal to the change of e_i multiplied by the entailed causal influence of e_i on c_j. In the context of a redesign plan, there may be several exogenous quantities that are changed, and thus the total change of c_j is the sum of the perturbations produced by each of the exogenous quantities. If the redesign plan is formed by N_e exogenous quantities, the total change Δc_j is:

$$\Delta c_j = \sum_{i=1}^{N_e} M_{ij} \cdot \Delta e_i \qquad (1)$$



Net causal influence of e_i on c_j: $OM(M_{ij}) = -OM(M_1)OM(M_2) - OM(M_3)OM(M_4)OM(M_5)$

Figure 5. A confluence of causal influences.

where M_{ij} = magnitude of the entailed causal influence of exogenous quantity e_i on constrained quantity c_j.

Calculation of cost and benefit of redesign plan

When constructing redesign plans, the program must be able to determine which exogenous quantity is the best one to change next. Furthermore, it must be able to determine the best direction and magnitude of change for that quantity. If each exogenous quantity had an entailed influence on only one constrained quantity, finding the correct direction of change would be trivial. But, in most cases, an exogenous quantity has entailed influences on several constrained quantities, so a change to the exogenous quantity will simultaneously produce compliance with some constraints and violation of others.

To evaluate the usefulness of extending a redesign plan by changing a particular exogenous quantity, the program uses metrics for the overall costs and benefits of the plan. We assume that implementing a change to exogenous quantity, e_i, has an (intrinsic) implementation cost, P_i. Furthermore, we assume that every constrained quantity, c_j, has an importance I_j, describing how important it is that the constraint be satisfied.

In general the overall benefit of a redesign plan is calculated as:

$$B = \sum_{j=1}^{N_s} \Delta c_j \cdot I_j \qquad (2)$$

where B = total benefit of the plan,
Δc_j = is the magnitude of the net change to constrained quantity c_j due to all exogenous quantity changes in the plan,
N_s = number of constrained quantities that are satisfied by the plan,
I_j = importance of constrained quantity c_j.

Similarly, the overall cost of a redesign plan is given by:

$$C = \sum_{i=1}^{N_e} P_i + \sum_{k=1}^{N_v} \Delta c_k \cdot I_k \qquad (3)$$

where C = total cost of the plan,
P_i = cost of implementing the change to exogenous quantity e_i,
N_e = number of exogenous quantity changes in the plan,
Δc_k = the magnitude of the net change to constrained quantity c_k due to all exogenous quantity changes in the plan,
N_v = number of constraints violated by the plan,
I_k = importance of constrained quantity c_k.

The overall quality of a redesign plan is the difference between its benefit and cost:

$$V = B - C \qquad (4)$$

We call this value, V, the "change value" of the redesign plan. The best redesign plan is the one with the largest change value. Conversely, if the change value of a redesign plan is negative, the plan is not useful.

When evaluating the costs and benefits of a plan, simple numerical values are used. The orders of magnitudes of the changes are replaced by the appropriate numerical values. For example ORDER_LOW becomes 10^{-1}. The values of the importance factors, I_j, are assigned by the designer. Determining appropriate numerical values can take some tuning, but in general we use values in the range from 1 to 10, with 1 representing an unimportant constraint and 10 representing a target quantity.

Search procedure

A complete redesign plan is defined as a combination of changes to some or all of the exogenous quantities, by which the redesign targets are achieved and any undesirable side effects are nullified or minimized. The program generates possible redesign plans by using best-first search to generate different combinations of changes to the exogenous quantities. When performing the best-first search, the partial redesign plans are sorted in order of decreasing change value (benefit – cost).

1. Generate valid exogenous quantity changes. As a preprocessing step prior to performing the search, we prune out some of the possible changes to exogenous quantities to reduce the amount of search that is performed. To explain the pruning process, we first need to make a few observations about exogenous quantities:

a) We denote as "effective" an exogenous quantity that has a causal influence (direct or entailed) on one or more target quantities c_j within the model. Exogenous quantities that have no such influences —but have causal influences on other constrained quantities— are referred to as "violation solvers", because their main usefulness is for correcting the constraint violations generated by the effective exogenous quantities.

b) For each exogenous quantity, the designer specifies the maximum allowable order of magnitude of change. For example, the engine design does not allow an increase in the area of the intake ducts by a factor of 10 or greater; therefore, the maximum possible order of magnitude for a change to them is ORDER_ZERO.

The first test in the pruning procedure is to determine the best direction of change for each exogenous quantity. The program first tries an increase of the highest order of magnitude allowable for that quantity, and then a decrease of the same magnitude. If the exogenous quantity is effective, the program chooses the change direction that (1) achieves the most compliances with the target quantities, and (2) has a positive change value V. If neither direction yields a positive V, the exogenous quantity is rejected as a candidate for redesign plans. If the exogenous quantity is a violation solver, the program chooses the direction that generates the greatest value of V, again, rejecting the quantity if neither direction produces a positive V.

Having determined the correct direction of change at the highest order of magnitude, the program now tests changes in the same direction at lower orders of magnitude. For these changes to be considered valid, they must yield a positive change value V.[3]

Quantity e_i		Proposed change		
Category	**Name**	**direction**	**magnitude**	**ranking #**
Effective	e_1 = Piston stroke	increase	ORDER_LOW	1
Effective	e_2 = Intake ducts area size	increase	ORDER_ZERO	2
		increase	ORDER_LOW	3
Effective	e_3 = Intercooler size	increase	ORDER_ZERO	4
		increase	ORDER_LOW	5
Violation solver	e_4 = Oil pump power	increase	ORDER_ZERO	6
		increase	ORDER_LOW	7
Violation solver	e_5 = Fuel injection flow rate	decrease	ORDER_LOW	8
		decrease	ORDER_ZERO	9

Table 4. List of valid exogenous quantity changes.

Only those quantity changes that survive the pruning are considered valid for use when searching for redesign plans. Once the valid changes have been identified, they are sorted by change value V, so that the highest change values are at the top of the list. This list constitutes the basis for the search queue. Note that the list naturally divides the exogenous quantities into effective quantities and violation solvers. For the case of the diesel engine model, the ordered list is shown in Table 4.

2. Performing best-first search.

Redesign plans are generated using best first search. The search queue consists of candidate (partial) redesign plans. Initially, the queue contains single change redesign plans corresponding to each of the valid, effective, exogenous quantity changes identified in the preprocessing step (e.g., Table 4). The queue is sorted in decreasing order of the change value.

To perform a step of search, the first candidate redesign plan is removed from the queue and checked to determine if it is a complete redesign plan. A redesign plan is considered to be complete, although not necessarily optimal, if any of the following conditions are satisfied:

- All targets have been achieved and all constraint violations have been repaired.
- The maximum size for a redesign plan has been reached.
- There are no remaining exogenous quantity changes to add to the plan.

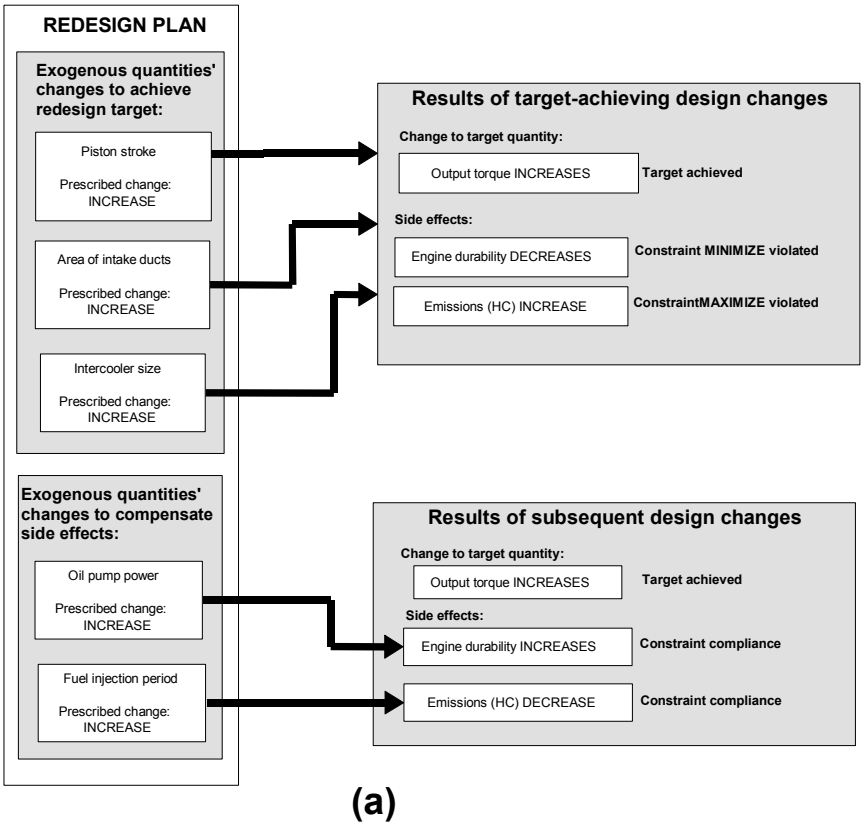
If the plan is a complete solution, the program presents it to the user. The user can request that the program finds a better solution, in which case the search continues until a better solution is found (or the queue is empty). The user can continue to ask for better solutions in this fashion. Once the user is satisfied with the current best solution, the search terminates.

If the candidate plan removed from the queue is not a complete plan, and does exceed a user specified maximum plan length, it is expanded into multiple new candidate plans, each having one additional change. The successors to a candidate plan are determined by the last exogenous quantity change Δe_n added to that plan. A valid successor is obtained by selecting any one exogenous quantity change that appears lower than Δe_n in the list of valid exogenous quantity changes (Table 4). For example, let us suppose that the current redesign plan is formed by the following changes:

Piston stroke	increase	ORDER_LOW
Intercooler size	increase	ORDER_ZERO

The last exogenous quantity change appended to this plan was Intercooler_size (INCREASE, ORDER_ZERO). From Table 4, there are four possible successors to that change:

[3] Sometimes a change of a lower order of magnitude does not produce enough benefit to overcome the implementation cost.



Quantity	Proposed change	
	direction	magnitude
Piston stroke	increase	ORDER_LOW
Intake ducts area size	increase	ORDER_ZERO
Intercooler size	increase	ORDER_LOW
Fuel injection period	increase	ORDER_LOW
Oil pump power	increase	ORDER_ZERO

(b)

Figure 6. Redesign plan for the diesel engine. (a) General form. (b) Orders of magnitude of changes.

Oil pump power	increase	ORDER_ZERO
Oil pump power	increase	ORDER_LOW
Fuel injection flow rate	decrease	ORDER_LOW
Fuel injection flow rate	decrease	ORDER_ZERO

Therefore, the program will generate four "child" redesign plans.

In general, when expanding a candidate redesign plan, the program generates all possible successor plans. For a successor to be valid, it must satisfy all of the following criteria:

- The new redesign plan must have a greater change value than the parent redesign plan.
- The new redesign plan must not undo any target change that has been achieved by the parent plan.
- The new redesign plan must not violates a constraint on a quantity that was already complied with, unless there is a possible way of repairing the constraint violation. To verify this, the program checks if there are any exogenous remaining quantities that have the capability of counteracting the constraint violation.

To complete the search step, all valid successors plans are added to the queue (the parent plan is removed) and the queue is sorted. The process then continues on in this fashion.

Search results: redesign plans for the diesel engine

Figure 6 shows a sample redesign plan form the diesel engine example. (We used the full engine model with 82 quantities.) In this plan, all of the effective exogenous quantities (piston stroke, intake ducts, area size and intercooler size) have been used, thereby allowing a great degree of compliance with the redesign target, which was to increase the engine's output torque. However, there were constraint violations that had to be remedied by complementary changes to other exogenous quantities. For example, the reduction in the engine's durability was corrected by increasing the power rating of the oil pump. Additionally, the program recommended an increase in the fuel injection period as a means of increasing the combustion efficiency, and consequently reducing particle emissions.

The solution in Figure 6 is actually the best solution that was found for a specified maximum plan length of 5 exogenous quantity changes. An interesting property of this solution is that, in the case of the intercooler size and the injection period, the program chose changes of a low order of magnitude, even

though we specified that higher orders of magnitude would be acceptable. The reason for this was that higher magnitude changes would have caused constraint violations that could not be fixed within the given engine model. For example, a very large increase in the size of the intercooler would have violated a constraint against increasing the weight of the engine, and there would have been no way to repair this violation.

Through this example, and others, we have found that best-first search does help to find a valid solution quickly. However, it also tends to identify lots of similar solutions. For example, the first two solutions may be immediate successors of the same particularly good parent plan. Thus, in using best-first search there is a tradeoff between speed and variety. Of course, however, if the search in continued long enough, all possible plans will eventually be generated.

RELATED WORK

The form of causal reasoning we use is similar to the chains of behavioral states used by Sembugamoorthy and Chandrasekaran [1]. Their model uses a representation language for describing a device in terms of structure, behavior and function; behaviors are captured as a series of states in the device's components. However, in our representation we have abstracted away structural information. We model behaviors solely in terms of quantities and their causal relationships. A basic technique for identifying exogenous quantities that can achieve a target change is outlined by Keller [2]. This technique, based on causal ordering, generates causal paths between the exogenous quantities and the target; the causal relationships between pairs of quantities are derived from the qualitative differential equations used to model the device's behavior. However, the kind of redesign plans described in that work do not consider the propagation of changes throughout the complete model of a device, and therefore no actions are proposed to counteract side effects of the changes.

RedesignIT uses concepts of qualitative simulation to describe the behavior of a quantity (cf. Kuipers, [3]). However, we have found it necessary to provide a measure of the magnitude of causal influences, by using qualitative orders of magnitude. Additionally, our definition of monotonic relationships (M+ and M-) incorporates the notion of causal direction.

A substantial amount of work has been done to address the issue of modeling engineered devices for redesign purposes. Goel and Chandrasekaran [4] use function-structure models for the redesign task, but the focus of their work is on corrective redesign, aimed at repairing faults occurring during the device's operation, rather than on correcting the side effects of design actions. The goal of the system they describe is mapping functional faults to structural faults. At the core of their model is a structural representation, based on assembly to component decomposition. This representation contains causal relationships in the form of pointers to domain knowledge. Once a structural fault is identified, repair proposals are drawn from an indexed memory of design strategies. Therefore, the system relies on the availability of such knowledge to find redesign solutions.

Otto and Wood [5] propose a redesign model based on product disassembly and component subtraction to identify functions of components. The goal is to identify ways to modify or replace components to improve device performance. A key part of their model is a functional diagram in which functional nodes are linked by flows of energy, signals, or materials. Function is mapped to structure via a morphological matrix. Otto and Wood establish redesign intent in the form of a set of metrics, with target values assigned to them. The possible strategies for redesign are parametric redesign or adaptive redesign. Our own redesign strategy is similar to parametric redesign but, while we perform qualitative simulation of parameter changes to generate qualitative redesign plans, the strategy of Otto and Wood is focused on numerical simulation and optimization. Different approaches are also used to manage conflicts in the device behavior: RedesignIT focuses on compensating undesirable changes on constrained quantities, while their system performs changes in morphology (i.e. changes in structure, and structure to function mapping).

The KRITIK2 system described by Goel and Stroulia [6] performs diagnosis tasks of three types: (1) device does not perform desired function, (2) design results in undesirable behavior, and (3) specific structural element misbehaves. Our own work is in the line of their task type (2). However, their emphasis is on correcting such behavior while preserving function. By contrast, our goal is improved functionality while maintaining compliance with design constraints. They use structure-behavior-function models in which the causal model is a dependency graph composed of structural parts and transitions between states, while we use quantities and their causal influences. Their model contains pointers to behavioral states, which can be substance schema, component schema, or field schema; in similarity with our notions about exogenous quantities, their model uses the concepts of range and intensity of a causal influence. Diagnosis for a type 2 task is performed by specifying the undesired behavior, plus a desirable behavioral state; the goal is to produce a specification of structural elements that, if modified, can produce the desired behavior. RedesignIT focuses on independent parameters (exogenous quantities), and how to modify them to achieve a target in a design parameter.

RedesignIT specifically addresses the issue of iteration in the design process. By finding and testing possible design changes, and generating detailed information about the effects of such changes, the program can become a tool for developing design strategies for families of devices. Thus it can play a complementary role to the system called LearnIT, described by Stahovich [7]. LearnIT is a system that learns design strategies

from observing a designer's actions when solving parametric design problems. While RedesignIT is used to generate redesign plans through simulation of changes, LearnIT can be used to infer an underlying strategy in those plans, which can in turn be applied to families of similar devices.

FUTURE WORK

The redesign plans that RedesignIT generates are abstract: they specify which quantities should be changed, and in which directions, in order to achieve the performance goals. However, they do not specify numerical values for the quantities. The program's primary task is to identify the parts of the design that will be affected by design changes. This helps focus the designer's effort on the relevant parts of the design. In our ongoing work, we plan to explore the use of more detailed device models as a means of refining the abstract redesign plans into more detailed plans.

In our continued work, we will also be developing techniques to allow the program to explain its decisions, specifically why it chooses a particular exogenous quantity for a redesign plan. These explanations will help the designer better assess the quality of a proposed redesign plan. For example, the program may choose a specific exogenous quantity because it can generate a high benefit, i.e. it has a large impact on the target quantity. However, this may produce a constraint violation that cannot be counteracted. The program will report that the permanent constraint violation is being accepted because of a high expected benefit.

Currently RedesignIT assumes the device is capable of providing all of the necessary functions. The program's task is to adjust the design parameters to achieve particular performance levels. We plan to explore techniques that allow our program to identify design modifications that will achieve new functionality. For this task, the work of Shimomura *et al*. [8] may be relevant. With their approach, desirable behaviors are achieved or enhanced by constraint propagation through a directed dependency graph of functions, rather than quantities. Their representation describes influences between functions such as "function enhanced by" and "function decomposed into."

CONCLUSION

In order to judge if a particular design change is feasible or desirable it is necessary for the designer to have knowledge about the side effects of the proposed change, because very often the side effects can outweigh the benefits expected from redesign. We have developed a program that generates proposals for achieving redesign goals, identifies side effects (potential or certain), and suggests additional changes to counteract those effects. The program helps the designer to understand the possible consequences of a redesign before resources have been committed to detailed design tasks, prototyping, and testing. This kind of tool will be particularly useful for making modifications to large scale engineered systems for which it is not possible for one designer to know all aspects of the design.

This work demonstrates the usefulness of causal influence models for planning redesign projects. Our program uses a device model describing the important physical quantities and the causal relationships between them. The advantage of this representation is that it directly focuses on the mechanism by which a design change propagates through a system. It also enables a program to detect possible side effects of a design change, and identify means of remedying those side effects.

ACKNOWLEDGEMENTS

This work has been supported by the National Science Foundation under Award Number 9813259.

REFERENCES

[1] Sembugamoorthy, V. and Chandrasekaran, B., 1986, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems", *Experience, Memory and Reasoning*, Kolodner, J., and Reisbeck, C., eds., Lawrence Erlbaum associates, Hillsdale, NJ.

[2] Keller, Richard M., et al., 1989, "Compiling Diagnosis Rules and Redesign Plans from a Structure/Behavior Device Model: the Details," Stanford University Knowledge Systems Laboratory, KSL 89-50.

[3] Kuipers, B. J., 1994, "Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge," MIT Press, Cambridge, MA.

[4] Goel, A., Chandrasekaran, B., 1989, "Functional Representation of Designs and Redesign Problem Solving," Proc. Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Detroit, Michigan, pp. 1388-1394, Los Altos, California, Morgan Kaufmann Publishers.

[5] Otto, K., Wood. K., 1999, "Product Evolution: a Reverse Engineering and Redesign Methodology," submitted to *Journal of Research in Engineering Design*.

[6] Goel, A., Stroulia, E., 1996, "Functional Device Models and Model-Based Diagnosis in Adaptive Design," *AIEDAM*, 10, pp. 355-370, Cambridge University Press.

[7] Stahovich, T., 1999, "LearnIT: a System that can Learn and Reuse Design Strategies," Proceedings of the ASME 1999 Design Engineering Technical Conference, Las Vegas, Nevada, DETC99/DTM-8779.

[8] Shimomura, Y., *et al*., 1995, "Representation of Design Object Based on the Functional Evolution Process Model," ASME Design Engineering Technical Conferences, DE-Vol. 83.

