NASA Contractor Report 198474

Integrating Post-Manufacturing Issues Into Design and Manufacturing Decisions

Charles F. Eubanks
Stanford University
Stanford, California

May 1996

Prepared for
Lewis Research Center
Under Grant NGT–51193

National Aeronautics and
Space Administration

Annual Technical Report
for
NASA GSRP Project NGT-51193

Integrating Post-manufacturing Issues into Design and Manufacturing Decisions

Charles F. Eubanks
September, 1994 - August, 1995

Introduction

This project focuses on research into some of the fundamental issues underlying the design for manufacturing, service and recycling that effect engineering decisions early in the conceptual design phase of mechanical systems. I am investigating a systems-based approach to material selection, manufacturing methods and assembly processes related to overall product requirements, performance and life-cycle costs. I am placing particular emphasis on concurrent engineering decision support for post-manufacturing issues such as serviceability, recyclability, and product retirement.

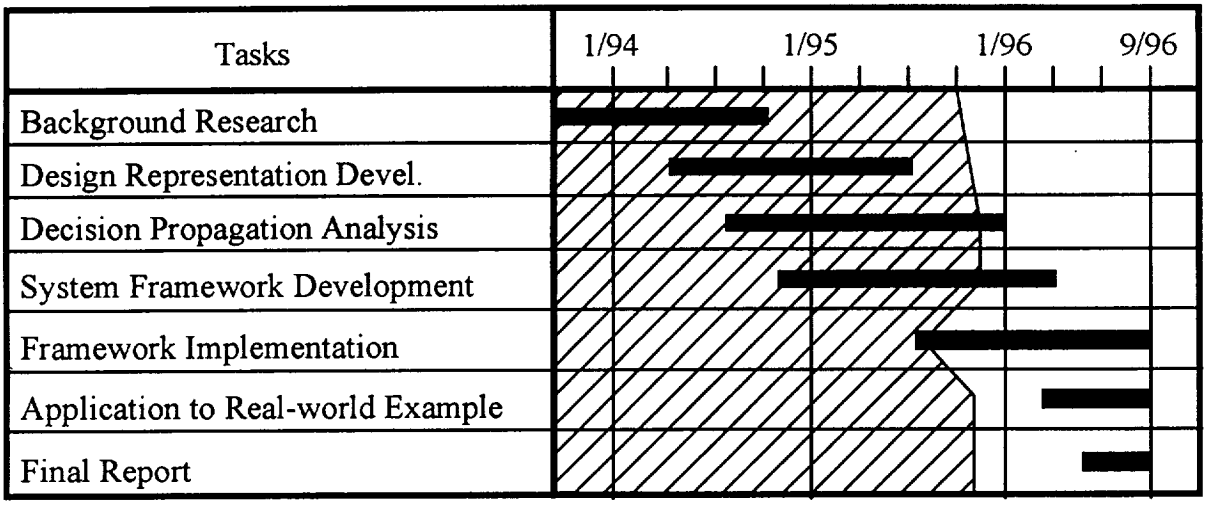
Over the past 15 years, we have seen a growing body research on several concurrent engineering issues. Probably the most mature of these is design for assembly (DFA) (Boothroyd & Dewhurst, 1972; Homem de Mello & Sanderson, 1991; Sturges & Kilani, 1992), with design for producibility (Priest, 1988; Arimoto, et al., 1993) and design for manufacturability (Poli, et al., 1992; Fathailall & Dixon, 1994) also receiving attention. We now see several institutions pursuing a combination of these disciplines as design for manufacturing and assembly (DFMA). These issues are very important, but fail to address costs incurred after the product leaves the factory.

Rising warranty costs have focused attention on the issue of design for serviceability (Makino, et al., 1989; Berzak, 1991; Eubanks & Ishii, 1993), particularly among automotive and major appliance manufacturers. It is clear that component manufacturing and assembly play a major role in overall quality and reliability, and that system configuration and assembly methods contribute to the ease of service. More recently, design for environmental compatibility (Navin-Chandra, 1991; Glantschnig, 1993; Marks, et al., 1993a) has become a factor that focuses attention on how the materials that make up a product and manufacturing processes used to create the product impact the earth's natural resources. All of these considerations point towards the need for an integrated approach to product design and manufacturing, and the need for decision support systems to aid engineers early in the design process.

Second Year Progress Summary

The project timeline shows that the primary goals for the second year was to complete the design representation phase, and to continue work on the decision propagation analysis, system framework development, and begin the framework implementation. The project is on schedule for the most part, although research into the framework implementation has been delayed.

Project and Research Timeline
NASA GSRP Project NGT-51193
Progress as of September 1, 1995



Research focus during the second year

The primary focus of our research over the past twelve months has been in the area of design decision propagation analysis. Key factors are 1) capturing the conceptual design decision process using methods and representations suited to that process, and 2) assuring, as much as possible, consistent and complete delineation of the parameters and operating conditions that will guide the layout and detail stages of the system design.

In the context of product design, the process begins by defining customer requirements, then performing a functional analysis to generate the design concepts and define the design problems to be solved. (Suh, 1990; Ullman, 1992) This process generally proceeds until lower level functions can be mapped to components. However, as Ullman (1993) points out, the design process should be seen as a more parallel development of form and function. Approaching the problem from the standpoint of looking at relationships between function and structure as the design develops, we realized that there is an opportunity to incorporate the ability to perform advanced Failure Modes and Effects Analysis (FMEA) as a means of testing the validity of our method. Preliminary work by Di Marco, et al. (1995) showed that such an analysis could be extracted from a fairly simple function-to-structure mapping, but

also showed some weaknesses of using a standard functional analysis model, and with the ways in which FMEA is carried out on consumer and industrial products.

Outlining the FMEA problem

Like serviceability, the major problem associated with traditional FMEA and diagnosability analysis methods is that they occur way too late in the design process, because it relies on the specification of the components that make up the device. Generally, the required component information is only available after completion of the prototype component and system design phase. Thus, as indicated earlier, any shortcomings in the design that might be identified by FMEA can be very expensive and difficult to correct or mitigate.

Our industry collaborators have also indicated that FMEA's on some consumer goods tend to be carried out on sub-systems, without necessarily addressing system wide effects. For example, a critical design criteria for an automatic ice maker is the alignment relative to dead level. Appropriate alignment assures that the water level is even throughout the ice maker freezing tray, so that ice cubes freeze evenly. Uneven freezing leads to hollow ice cubes on one extreme, and brittle (over frozen) on the other extreme. Problems develop when the ice maker is correctly aligned with respect to the freezer, but the freezer is not dead level with respect to the earth. A standard FMEA would miss this failure mode, because it does not account for issues related to the device's interface with the rest of the system. Other system-wide variables that effect ice maker operation include incoming water pressure and freezer temperature. Variations in alignment, water pressure, and freezer temperature contribute to nearly half of ice maker service calls.

Several automated FMEA systems have been developed for use in analyzing electrical systems, since faults and failures can be more easily characterized as numerical quantities. Ormsby, et al., (1991) developed a concept for automated FMEA employing qualitative reasoning in a model-based environment as a means of making the analysis extensible to other domains. Computer-based diagnosis systems have been a popular research subject for the past several years, as evidenced by Hamsher, et al. (1992) Abu-Hanna, et al., (1991) showed that functional design models can be used in model-based diagnosis systems. In the mechanical engineering domain, Umeda, et al., (1992) used functional representations for diagnosis and self-repair of a copy machine. Morjaria, et al., (1992) have developed diagnostic systems based on belief network technology, which employs causal networks and probability theory to reason from symptom to failure in large industrial systems. More pertinent to this research is work by Clark and Paasch (1994), who show that function to structure mapping can be used in the early stages of design to assess diagnosability; i.e., a measure of the ease of isolating the cause of a malfunction. They present a method for substituting functions and sub-functions for component performance measures in the early stages of design to make diagnosability assessments.

Behavior Model Development

Basic concepts and definitions

Structure is defined as the physical topology of a device or system, including the components that make up the system, and the relationships between the components.

The definition of function is fairly well established, and is usually stated as what must be done without specifying how it is to be achieved. Functional analysis is probably the most widely accepted practice for defining designs in the conceptual phase. Engineers begin by defining the overall function which the device is to perform, and decompose it into sub-functions that delineate the design problems to be solved. (Suh, 1990; Ullman, 1992) The usual practice of effective functional decomposition stresses suppressing the definition of the "hows" as long as possible so that we can 1) better understand the design problems to be solved, and 2) identify existing components that may fulfill the functional requirements. (Ullman, 1992)

Behavior is not quite as well defined, but normally follows the notion that it is "how (an) expected result is attained" (Keuneke, 1989), or the "detailed description of internal physical action based on physical principles and phenomena." (Welch & Dixon, 1994) However, using these definitions, the line between function and behavior blurs very quickly during the process of functional analysis. Finger and Rinderle (1989) recognized this, and used the term function "to indicate the subset of *behaviors* which are required for the device to perform satisfactorily." Struges (1992) brings this into focus when he describes his functional block diagram decomposition (Figure 1) with the statement:

> "The nodes to the left of a function node represent the reason *why* a function is included: a higher-level function. The nodes to the right are functions describing *how* the function is performed..."

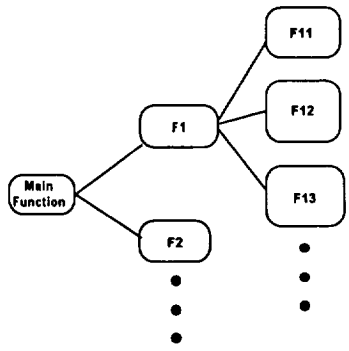


Figure 1: Functional Block Diagram Schematic

The distinction we have generally seen between models for function and behavior is the latter's use of pre- and post-conditions; i.e., what conditions must be true in order for the behavior/function to take place, and what conditions exist given that the behavior/function has taken place.

Behavior modeling has received a good deal of attention on the theoretical level, particularly in the AI community. In a function-behavior-structure modeling construct, behavior knowledge forms the link between the functions and structure of a device. As mentioned earlier, this information is vital to issues related to diagnosis and serviceability of systems. Many researchers in the AI community use the notion of causal chains or networks that are derivable either from the functional description of a device (Keuneke, 1991), from its structure, (Kuipers, 1984), or from the aspect of qualitative physics. (deKleer and Brown, 1984) More recently, researchers in the field have developed more rigorous definitions and methods for describing the behavior of devices from the aspect of causal process descriptions of devices (Iwasaki & Chandrasekaran, 1993) and causal ordering based on process models. (Iwasaki & Simon, 1994)

Restating Ullman's conjecture, the design process should be seen as a more parallel development of form and function. Given that this form-function link is created early in the design process, then we should be able to perform FMEA and diagnostic analyses early in the design process. As shown in Figure 2, FMEA begins with a failed or degraded component, and attempts to identify the end-effect, usually expressed as a malfunction or misbehavior. For example, if the ice maker thermostat exhibits a failure mode of "stuck open", ice will form in the tray, but the ice maker will not cycle to push the ice into the ice bucket, and refill with water. The observed effect is "no ice in bucket, ice in tray". Diagnosis entails the same notion, but occurs in the opposite direction, starting with an observed misbehavior, and attempting to identify the failed component. Using the previous example, we would observe "no ice in bucket, ice in tray" and work to the conclusion that the cause was "thermostat stuck open".

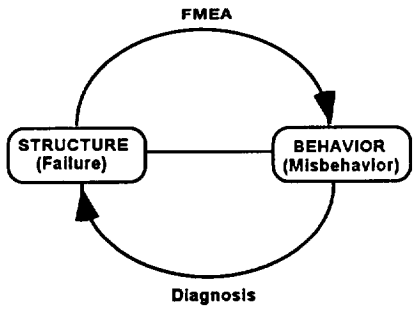


Figure 2: FMEA / Diagnosis Relationship

Thus, if we can develop a method capable of performing FMEA in the early stages of design, then we should be able to provide insight into issues surrounding reliability, diagnosis, and serviceability in these early stages as well. Key elements required to develop this capability include:

- a behavior model suitable for use in the early stages of design
- a structural model suitable for use in the early stages of design
- a framework linking these two models
- inferencing methods for evaluating effects of both behaviors and misbehaviors on system operation
- a user intuitive interface consistent with the conceptual design process

The clearest advantage of using behavior modeling over traditional functional modeling for this analysis is that it requires specification of initial system states and the transitions necessary to obtain the desired system states. The major departure from behavioral modeling methods mentioned in the previous section is that the full specification of the device state is not necessarily known early in the design phase. Therefore, we must adapt methods for dealing with sets of states, as opposed to individual states.

Initial concept development on behavior modeling for design

Gleaned from the literature, we submit the following definitions:

Variable: a triple (<object>, <attribute>, <value>)
where:
<object> can be any physical or conceptual entity
<attribute> is a distinctive quality or characteristic of the object
<value> is a quantification of the object attribute

State: a set of quantified state variables

Behavior: a transition from one state to another; i.e.,
initial state → behavior → final state

Behavior definition and state space partitioning

Design begins when we 1) recognize a need, and 2) decide to build a device to satisfy that need. At this point, we have no idea what the device will look like, how it will perform, etc. What we do know is some initial existing condition, or state, that we wish to alter to create some final desired state. For example, I recognize a need for ice cubes to be present in the ice bucket of my household freezer; i.e.,

Initial state: no ice cubes in ice bucket
Desired state: ice cubes in ice bucket

Conceptualizing a universe S of all the possible states that any device that we might design may exhibit, we have essentially partitioned the state space into two regions, each of which is a set of states. One set consists of all possible states, where a state is some unique combination of state variables, which have the common state variable (ice bucket, ice cube

level, empty), while the other set consists of all possible states which have the common state variable (ice bucket, ice cube level, not empty). Identifying these sets as S_1 and S_2, respectively, we have:

S_1 = {(ice bucket, ice cube level, empty), (obj_2,*, *), (obj_3, *, *),..., (obj_n, *, *)}
S_2 = {(ice bucket, ice cube level, not empty), (obj_2, *, *), (obj_3, *, *),..., (obj_n, *, *)}

where (obj_i, *, *), i = 2, ..., n, represent as yet unknown objects whose attributes may take on any value. Therefore, if we take the intersection of all of the possible states contained in the sets S_1 and S_2, we would have:

$$\bigcap_{S_1 \in S} S_1 = \{(\text{ice bucket, ice cube level, empty})\}$$

$$\bigcap_{S_2 \in S} S_2 = \{(\text{ice bucket, ice cube level, not empty})\}$$

We now define the desired behavior, b_1, for our device as "deposit ice cubes in bucket", which causes the state transition to take place. We can envision the resulting state space and transition as shown in Figure 3:

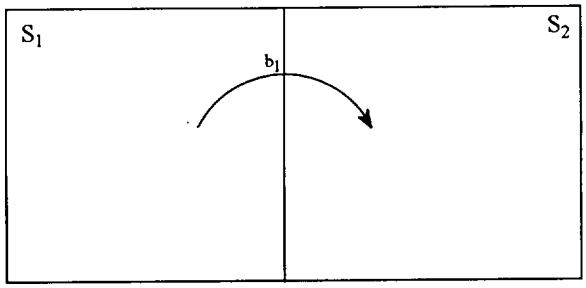


Figure 3: Partitioned State Space with Transition

We can also represent the transition as a flow diagram:

$$S_1 \xrightarrow{b_1} S_2$$

We may also know some general operating conditions that we either expect will exist, or which must exist due to the physical requirements of the process. In this case, the device must exist in an environment cold enough to freeze water, and maintain it in a frozen state. Once again, the state space can partitioned into sets of states which have state variables:

(environment, temperature, ≤ 32)
(environment, temperature, > 32)

We have now defined four sets of states:

$$\bigcap_{S_1 \in S} S_1 = \{(\text{ice bucket, ice cube level, empty}), (\text{environment, temperature, } \leq 32)\}$$

$$\bigcap_{S_2 \in S} S_2 = \{(\text{ice bucket, ice cube level, not empty}), (\text{environment, temperature, } \leq 32)\}$$

$$\bigcap_{S_3 \in S} S_3 = \{(\text{ice bucket, ice cube level, empty}), (\text{environment, temperature, } > 32)\}$$

$$\bigcap_{S_4 \in S} S_4 = \{(\text{ice bucket, ice cube level, not empty}), (\text{environment, temperature, } > 32)\}$$

Note that the set of states S_4 is, under normal circumstances, not possible. Laws of nature (physics, chemistry, etc.) further partition the state space, separating device states that are possible from those that are impossible; i.e., states that violate the 2nd Law of Thermodynamics, Newton's Laws, etc. The implication is that a transition from initial state in the set S_1 to any state in the set S_4 cannot occur, simply because the laws of nature prevent ice from existing (at least for extended periods) in an environment with temperature greater than 32. In addition, state S_3, is possible, but is undesirable, since there is no possible transition from this state to the desired condition of having ice cubes in the bucket. These areas can be visualized as shown in Figure 4:

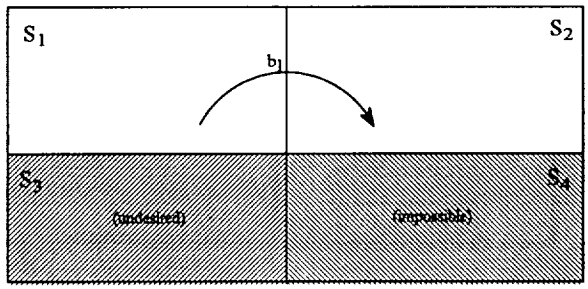


Figure 4: Partitioned State Space with Undesired and Impossible States Defined

Among the 4 definable sets of states, we consider 2 to be desirable, both of which have as a state variable the necessary operating condition (environment, temperature, ≤ 32). Thus, the set intersection of elements common to each set yields:

$$\bigcap_{S_1 \in S} S_1 \cap \bigcap_{S_2 \in S} S_2 = \{(\text{environment, temperature, } \leq 32)\}$$

In addition to desired, undesired, and impossible sets of states, we may partition into sets of states which are either unknown, or not applicable. Unknown sets are possible combinations of state variables which cannot be designated as either desired or undesired, and may represent behavior side-effects. Sets of states become not applicable when design decisions exclude them from the realm of possibility; e.g., deciding to use an electric motor for energy input as opposed to a hand crank. It is possible to represent these decisions as a hierarchical

breakdown of a desired behavior into desired sub-behaviors. Behavioral decomposition is covered in a later section.

Reasoning about failures

At this point, we can assume that some device exists whose desired behavior is to deposit ice cubes into the ice bucket. We have also implicitly assumed the existence of an operating environment with a temperature less than or equal to 32. Let's assume that we begin with the conditions described as S_1, and consider two possible failures: 1) the failure of our device to deposit ice cubes in the bucket, and 2) the failure of the environment to exist at a temperature less than or equal to 32. In the case of failure 1, no state transition takes place. We simply remain in state S_1 ad infinitum, or until failure 1 is corrected. In the case of failure 2, we make the transition into the undesired state, S_3, there to remain until failure 2 is corrected, which allows us to transition to state S_1 , and then on to S_2. It is reasonable to conclude that failures, either in the device itself or in the system supporting it, can manifest themselves as either 1) the failure to transition to the desired end state, or 2) the transition to an undesired state.

A behavior (b_1) is uniquely defined by both it's initial state, or pre-conditions (S_1), and it's final state, or post-conditions (S_2). (Iwasaki & Chandrasekaran, 1992) As such, we can define any transition into an undesired state uniquely, and make the claim that the transition from some desired state (S_1) into an undesired state (S_3) results in a unique and undesired behavior (b_1'). It is also possible to define another undesired behavior, b_1'', that transitions from S_2 to S_3, also a result of failure 2. We can assign both b_1' and b_1'' the label "freezer failure". (Figure 5) Failure 1, on the other hand, can be thought of as a non-behavior, $\neg b$, representing a case where there is simply no transition to another state. Since we would like to be able to identify failures, we could choose to assign $\neg b$ the label "ice maker failure".

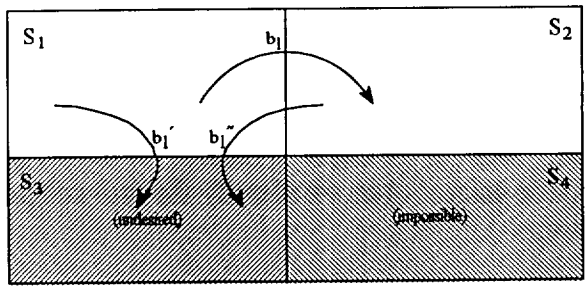


Figure 5: Representation of Undesired State Transitions

From the previous discussion, we can state the FMEA problem as:

"Given an undesired behavior or a non-behavior, what is the resulting device state?"

Using our example, we would pose the FMEA problem as:

"Given S_1 and "freezer failure", how would that effect the system?"

From the transition graph, we see that the answer is S_3.

In much the same way,. we can state the diagnosis problem as:

> "Given an undesired final state, what desired behaviors did not occur, or what undesired behaviors did occur?"

Using our example, we would pose the diagnosis problem as:

> "Given that the device is in S_3, what happened?"

We see that there are 2 undesired behaviors that could have occurred, both of which represent "freezer failure".

The preceding examples are simple and straightforward, because we are working a very high level of device abstraction, and thus at a very high of system aggregation;, i.e., entire sub-systems. As we decompose the operation of the device into more detailed descriptions of both behavior and structure, we need more sophisticated techniques for describing and reasoning through the FMEA and diagnosis problem, a primary thrust of this research. The methods developed will parallel work described in Goel & Chandrasekaran. (1989)

Behavior decomposition

The device design proceeds with decisions about how the device is to perform its desired behavior, with each decision partitioning the state space. Using the notion of functional decomposition, we can begin to perform behavioral decomposition to delineate the sub-behaviors required to accomplish the overall device behavior. Continuing with our example, behavior b_1 can be decomposed into 2 sub-behaviors:

> b_{11}: create ice cubes
> b_{12}: deposit ice cubes in bucket

which create state transitions:

$$S_{11} \xrightarrow{b_{11}} S_{12} \quad S_{21} \xrightarrow{b_{12}} S_{22}$$

Note that the label of b_{12}, "deposit ice cubes in bucket," is not unique, as it is the same label used for b_1. However, it is unique when we identify the pre- and post-conditions. For b_{11}, the initial and final states are:

S_{11} = {(ice bucket, ice cube level, empty),
(environment, temperature, ≤ 32),
(ice maker, ice present, false),
(obj_4, *, *), (obj_5, *, *),..., (obj_n, *, *)}
S_{12} = {(ice bucket, ice cube level, empty),
(environment, temperature, ≤ 32),
(ice maker, ice present, true),
(obj_4, *, *), (obj_5, *, *),..., (obj_n, *, *)}

For b_{12}, the initial and final states are:

S_{21} = {(ice bucket, ice cube level, empty),
(environment, temperature, ≤ 32),
(ice maker, ice present, true),
(obj_4, *, *), (obj_5, *, *),..., (obj_n, *, *)}
S_{22} = {(ice bucket, ice cube level, not empty),
(environment, temperature, ≤ 32),
(ice maker, ice present, *),
(obj_4, *, *), (obj_5, *, *),..., (obj_n, *, *)}

At this level of granularity, note that S_{12} and S_{21} are representations of the same set of states, which we shall designate simply as S_{12}. Because the post-conditions of b_{11} are therefore precisely the pre-conditions of b_{12}, we can say that b_{11} and b_{12} are causally linked to form the state transition graph:

$$S_{11} \xrightarrow{b_{11}} S_{12} \xrightarrow{b_{12}} S_{22}$$

Intuition tells us that any pre-conditions required for b_1 must also be required for b_{11}, and that any post-conditions resulting from b_1 must also result from b_{12}, if this decomposition is valid. In state space terms, any evaluated variables common to partition S_1 must also be common to partition S_{11}, and likewise for S_2 and S_{22}; i.e.,

$$\bigcap_{s_{11} \in S} S_{11} \cap \bigcap_{s_1 \in S} S_1 = \bigcap_{s_1 \in S} S_1$$

$$\bigcap_{s_{22} \in S} S_{22} \cap \bigcap_{s_2 \in S} S_2 = \bigcap_{s_2 \in S} S_2$$

In addition, the afore mentioned operating condition (environment, temperature, ≤ 32) must, by definition, be pervasive throughout the decomposition of desired behaviors:

$$\bigcap_{s_{11} \in S} S_{11} \cap \bigcap_{s_{12} \in S} S_{12} \cap \bigcap_{s_{22} \in S} S_{22} = \bigcap_{s_1 \in S} S_1 \cap \bigcap_{s_2 \in S} S_2$$

It would be prudent at this point to note that the decomposition of a behavior into sub-behaviors is not necessarily unique. The details of the decomposition hierarchy will be a direct reflection of the design team's implementation decisions. Therefore, any representation model must support multiple sets of decomposed child behaviors for each parent behavior.

I submit (without proof, for the time being) that S_2 and S_{22} represent the same set of states, since the "ice maker" variable is now assuming all possible values. Therefore, the state transition graph becomes:

$$S_{11} \xrightarrow{b_{11}} S_{12} \xrightarrow{b_{12}} S_2$$

Graphically, the state space appears as shown in Figure 6:

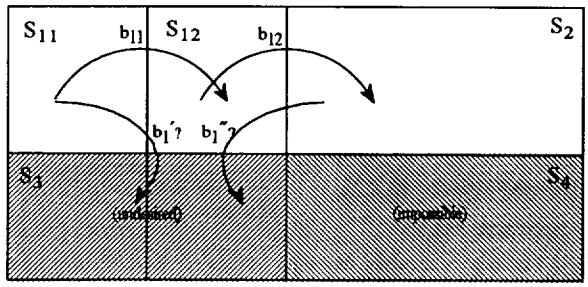


Figure 6: Representation of a Decomposed State Space

This situation now begs the question of whether the undesired behaviors are similarly decomposable and will they follow the same decomposition rules? It would appear so in our simple example, but may not be the case in general.

Implications for system development

What we see developing is a set of methods and rules which can be used to both guide and partially specify the way in which behavioral decomposition proceeds. For example, if the designer declares an operating condition variable value to be pervasive, and later in the decomposition attempt to assign a conflicting value to it, the system can easily perform a consistency check and flag this as a mistake. The same can be done between levels of decomposition, since conditions pervasive in the parent behavior must exist in all states in the child behaviors. Because we are grounding the specification and decomposition in logic and set theory, we should be able to perform automated, detailed analyses and simulations of device behavior, reason about conditions which depart from desired behaviors, and analyze the results of those departures.

The method we are proposing is rigorous in that it requires specification of pre- and post-conditions, yet flexible in that we require no specific syntax by which the design team must define the device operation. One obvious question to be answered is whether we can continue to develop a meaningful representation and a tractable reasoning system without resorting to the use of a specific syntax.

Concept development on device structure representation

Another major effort required by this research is the development of a robust representation of device structure. We have seen how it is possible to define failures and their effects in terms of undesired states and transitions to those states. However, the FMEA, and the diagnosis models generated from it, make more sense to engineers when place in the context of components and/or subassemblies. We need a comparable model for device structure, capable of capturing as much knowledge about the physical aspects of the device as possible, as early as possible in the design phase.

As shown in the previous discussion, behavioral decomposition will generally define the nature of the sub-systems that will eventually make up the device, and possibly some of the necessary support systems for the device to operate properly; e.g., the freezer as a support device for the ice maker. Thus, there is some notion of behavior to structure mapping that is simply implicit in the behavior modeling process. Ullman (1993) points out that in many cases some structural decisions are made in these early stages, indicating that the structural representation needs to be developed in parallel with the behavioral model. Of course, as the decomposition of device behaviors continues, behavior descriptions will approach a detailed enough level to 1) warrant the use of engineering equations to describe the state transition, or 2) be mapped directly to a know artifact which performs the desired behavior. (Ullman, 1992; Suh, 1990) At this point, we are into the more traditional aspects of mechanical engineering component design, which will begin to add design details that will be used in the development of 3-D models.

As reported previously, I have been working on an object-oriented approach to a structural representation syntax, applicable to the early stages of design, which aids in the definition of the basic physical objects, and relationships between these objects, which define a mechanical system. (Eubanks, 1994) I have established the basic elements of the system and their semantic relationships. (Figure 7) A good deal of work remains to refine this model, and determine the necessary object attributes.

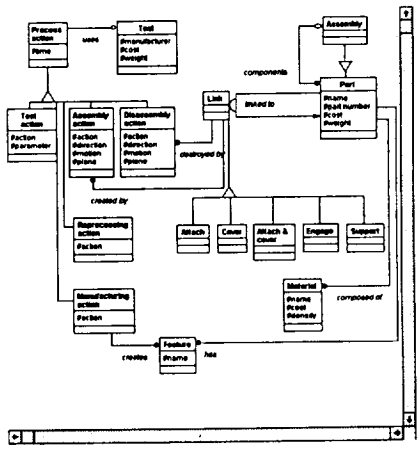


Figure 7: Representation of Existing Structural Representation

Where previous reports separated the notions of function and structure, we have now combined them under the consolidated concept of behavior. The object-oriented approach of the behavior model should dovetail with this structural model quite nicely. In addition, it may be possible to establish object links to CAD systems that support component databases. However, most commercially available packages do not support this option, or do so in a very limited fashion. (Busick & Chong, 1995)

At every stage of the design process, we want key elements and relationships established as shown in Figure 8.

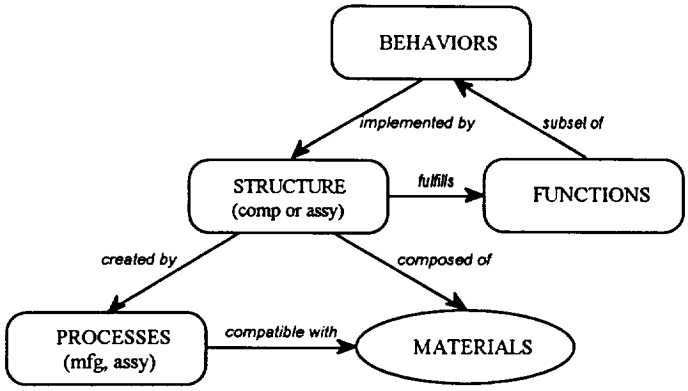


Figure 8: Key Elements and Relationships for Behavior/Structure Model

By creating the representations in parallel, and providing a direct link between the descriptions of the device operation and descriptions of the physical entities that implement those actions, I believe that we can generate pathways for inferencing strategies necessary to perform advanced FMEA analyses, which in turn lead to advanced diagnosis and serviceability analysis.

Inferencing within the structural model

Given that different teams might be working on different parts of a design, the parts still have to fit together somehow. In general, this means that features of parts contained in subassemblies will share a structural relationship. Using an automotive transmission as an example, suppose we have one team working on the case, and another working on the gear train, in particular the rear main shaft bearing. A simplified form of the instantiations for these entities using the proposed structural representation would appear as shown in Figure 9.

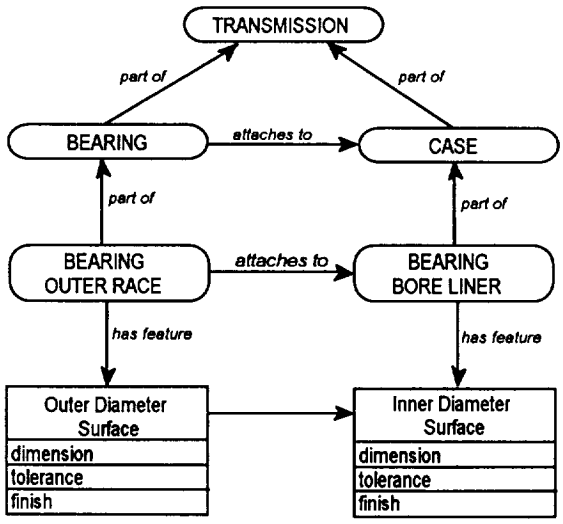


Figure 9: Interdependencies in a Concurrent Engineering Framework

Just like any good concurrent engineering model, this model records the interdependencies of the parts and features of one design team with those of the other team. In the event that the shaft design team considers changing some attribute of the outer diameter, they can obtain a list of the effected features on the bearing bore liner. The natural extension of this idea is a constraint management system.

Another inference that can be drawn is that the liner and race materials are in physical contact. Suppose that the case design team considers changing the material to magnesium (for what ever reason). They could request the affected attributes of the race and see that the race material is steel, and realize that there is the possibility of galvanic corrosion between these two materials. The natural extension here is a rule-based system of design practices to flag these types of potential problems.

In the above example, the link between the bearing OD and the liner ID is a natural consequence of the fact that the bearing and the liner are physically linked. Of course, the system would not know exactly what features, but it would be reasonable to establish a consistency checking rule that effectively says "if two parts are in direct contact, then they must have features in direct contact." Note that, while not implemented, we implied similar rules in the definitions of our LINKER relationships; e.g., if one object covers another, then that object must be attached to something else, otherwise it would be floating in space.

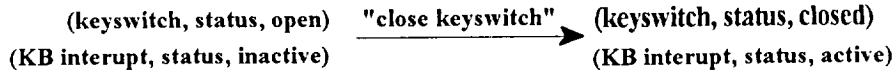
Model integration and user interface issues

Any tool must be presented to the user in a form that resembles, as closely as possible, the representations and thought processes that the user employs to solve the problem. In most cases, the interface must employ both verbal and graphical representations to be effective, with a great deal of emphasis placed on intuitive graphical methods.

Similar to my approach with the LINKER interface design, I will rely, as much as possible, on a mixture of graphical and verbal representations as the source of input. Since the basic concept of the behavioral model is a sequence of events, I envision an input screen along the lines of:

$$S_{11} \xrightarrow{b_{11}} S_{12} \xrightarrow{b_{12}} S_{22}$$

This would mean that state and behavior description would require another page or a dialog box for input or review. An even better approach, if it can be done without too much clutter on the screen, would be to make the symbols explicit. Using the example of a computer keyboard, the states and behaviors would appear as:



The behavior to structure mappings present another challenge in terms of a good and intuitive representation to the user. In his work on function-structure mapping, Di Marco (1995) used graphical formats for entering the function and structure trees separately, then used this information to form hierarchical lists in order to do the mapping. This approach avoided the problem of the considerable screen clutter of an icon-based system, but provided a less than intuitive feel for the mappings.

However, suppose we have non-state changing requirements, such as aesthetics or packaging. On a very basic level, addition of packaging or aesthetic requirements simply add more constraints to the structural implementation of the functional requirements of the device. The interesting notion here, applied to the idea of behavior modeling, is that packaging and aesthetics do not necessarily contribute to change of state operations per se; e.g., a button contact closing will result in a change of state, but the location, shape, marking, etc., does not. What the location, shape, marking, etc. may contribute to ergonomic aspects of the design, which will in fact exist as part of the design requirements, and probably should be considered part of the device function. Consequently, we would expect to see a high-level function of "easy to use", which would then be decomposed, using standard functional analysis techniques, to the level where we would, as usual, map to structural entity, and a particular feature of that structural entity; e.g., those little raised nubs on the "f" and "j" keys that let you know by tactile feedback where the home row is located.

I now decide to map the behavior "close keyswitch" to a physical entity called "key," a part of the keyboard, which provides the necessary behavior. It turns out that the marketing surveys indicate that the users want the keys marked so they can tell which ones are which. I'm thinking Braille, but my industrial designer says we should provide a label. In essence, then, the feature "label" implements the function "distinguish keys." Graphically we can envision the relationships as shown in Figure 10.

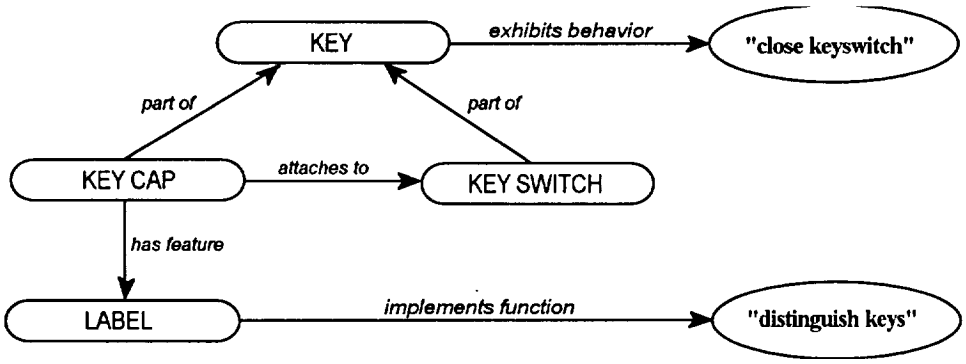


Figure 10: Keyboard Structure to Behavior Mapping Fragment

The representation shown in Figure 10 gets the point across, but it is still a fragment of a much larger picture; a picture that may have to be composed via other fragments, such as Figure 11.

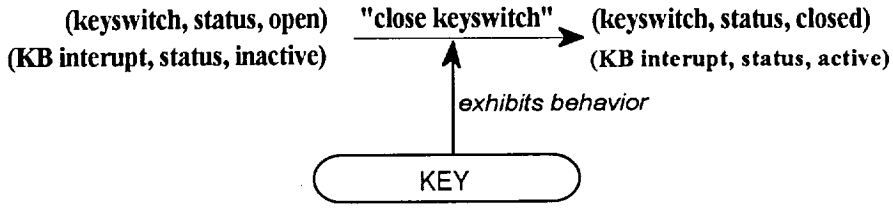


Figure 11: Structure to Behavior Mapping in a Keyboard Behavior Model Fragment

Even text based input and output must have a visual component. While the LINKER representation has a nice visual aspect, our collaborators at GE felt that it was also necessary to have a standard parts tree, or bill of materials form for the structural representation, an aspect that also appears in Di Marco's (1995) work. These hierarchical representations will appear in several places in the proposed work. For even moderately complex designs, putting entire part hierarchies on a single page, even if it's a pretty big one, will result in a fair amount of information overload for the user. This may require a similar use of model fragments where we limit the visual extent of the hierarchy to only one or two levels. For example, if my interest is in the keyboard key, I might have a representation such as Figure 12.

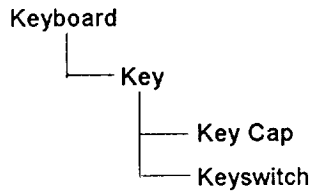


Figure 12: 3-Level Hierarchical Representation

Clicking on "Keyswitch" would bring produce Figure 13.

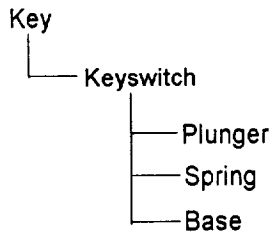


Figure 13: 3-Level Hierarchical Representation - Lower Indenture

My views on presentation of outputs is much the same. Our experience with the LASeR program showed that intuitive graphical outputs, usually in the form of charts, graphs, or forms, convey the most information and have the greatest impact. The bottom line, as I see it,

is to be as visual and as intuitive as possible, using graphics or text where appropriate to convey the maximum amount of information as efficiently as possible.

Another factor from the conceptual design development standpoint is linking to sketches or existing drawings. Computer-based drawing packages make electronic sketches a real possibility, while scanners can handle input of back-of-the-napkin sketch. Hypertext systems may be one way of linking conceptual design inputs, like functional block diagrams or behavior models, to initial design sketches, or even to part specifications, where the behavior links directly to a known component.

Error checking application and presentation

I see two basic type of error checking that would apply to this type of a system deployed in the early design stages: 1) consistency checking, and 2) dependency checking.

Consistency checking

Early in the development of the behavior model, the nature of the information that the system will process will be extremely qualitative; e.g., systems will be active or inactive, shafts will be rotating or not, etc. Given this form of information, the goal of the error checking will be to ensure that the specifications remain consistent throughout the behavioral decomposition that will describe the operation of the device. In the preceding keyswitch example, a pre-condition for the "close keyswitch" behavior is that the keyswitch status is open. Suppose that in some previous behavior description, I had specified that one of the post-conditions was that the keyswitch status was closed. As I move through the chain of behaviors, I would note that these two state variable values conflict, producing an inconsistent specification of device state. Optimally, the program would display the conflicting conditions, and ask for a clarification, such as:

BEHAVIOR MODEL CONSISTENCY CHECK: ERROR

Behavior: "click mouse button"
Post-con: (KB interupt, status, active)

Behavior: "close keyswitch"
Pre-con: (KB interupt, status, inactive)

BEHAVIOR "close keyswitch" REQUIRES (KB interupt, status, inactive)
PREVIOUS BEHAVIOR "click mouse button" RESULTED IN (KB interupt, status, active)

RECOMMEND:
MODIFY CONDITIONS OF EXISTING BEHAVIORS
INSERT BEHAVIOR WITH RESULT(KB interupt, status, inactive)

Supplying recommendations will require recognizing error patterns and having a lookup table or a rule-base of known fixes for those errors. For the example above, the recommendations are directly related to the inherent flow of the behavioral relationships. In the structural model, consistency checks can become quite extensive because of the number and types of relationships that exist between the various data objects. For example, if a user specifies "plastic" as the material, then "die casting" is an inconsistent specification as a manufacturing

process. This points to the need once again of a type of lookup table of materials and acceptable manufacturing processes.

Dependency checking

Dependency checking will take the form of either a less rigorous approach to constraint management, or a rule set of good design practices. This type of checking is particularly applicable when we have different teams working on different portions of the same design. The presentation to the user will probably take a form similar to the above, presenting the user with the two elements in conflict, why the conflict exists, and possible ways to resolve the situation.

Levels of representation within the system

At all levels, I expect to see a mapping between physical entities (PE's) themselves, and also between PE's and behavior or function specifications. There may also be a mapping between a function or behavior, and a feature of a physical entity, but I think that that mapping should be thought of as a modifier, since it is still the PE that provides the feature that fulfills the function or behavior. The fins on a motorcycle cylinder provide an example of a behavior to feature mapping, where the behavior might be specified as "remove excess heat from cylinder."

My perception is that the resulting system will exhibit the characteristic of one large system in that it will only have one basic data structure, and thus will lend itself to a centrally located file server. From the usability standpoint, though, different design teams should only have to see the elements of the system that they are working on, and those elements directly dependent on them. Using the transmission example, the members of the case design team are obviously concerned with some elements related to, or features of, the bearing, but will not be concerned about the speed ratio for reverse gear. Therefore, while all elements and features of the outer race specification would be part of their visible data partition, all elements of the design of the reverse gear would be transparent to them. There will be data partitioning, but at the same time there will be some crossover due to the coupling of design elements.

Levels of the design will be captured by the hierarchical nature of the design description. Going back to the keyboard example, the behavior fragment could be thought of as representing an intermediate level behavior. A higher level behavior regarding the entire keyboard as a unit might look like:

Only one of the state variables is specified, because at a higher level we are only concerned that pressing a key makes the keyboard interrupt status change from inactive to active. To decompose this behavior, we must decide what press key means: closing a switch, compressing a membrane, etc. I chose to use a switch with a keycap, and ended up with the behavior specification in the previous section that involved closing a switch. A key point is

that, no matter how we decide to decompose "press key," it must result in the KB interrupt being active - the results must be consistent.

We can also exploit the hierarchical and consistent nature of the decomposition to examine design alternatives as well. As long as the coupling at the higher levels is maintained, we can "plug" various representations of alternatives into the representations of the entire system and perform evaluations, and we can do this because we are maintaining one consistent data structure at every level of the design representation.

These same arguments hold for the structural representation also, since design decisions will be inherent in the components we choose to make up our subassemblies. The logical extension along this line is the ability to build design decision histories by taking "snapshots" of the system, and storing them for review and reevaluation.

From this central system, we can then integrate the flow of information to and from the various tools that work at lower levels of abstraction, like FEM and solid modeling packages, when the use of those packages is appropriate. The data structures for the structural representation include both part specification, and specifications for part features, the two elements that form the core of most structural modeling packages. For example, our model links the feature object with the manufacturing process that is used to create it. Now, suppose I am designing a steel plate that has a feature "3/8 inch hole" created by a "drilling operation" and that I specify this prior to building my geometric model. A simplified, partial data representation might appear as shown in Figure 14.

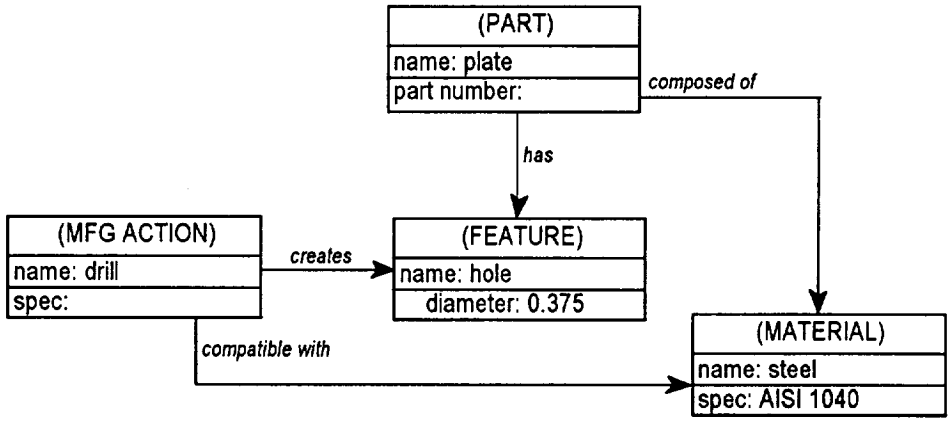


Figure 14: Manufacturing Relationships in the Structural Model

Conceptual extensions

Integration with other design tools

Let's assume that I have the ability to link this information to other design tools. (I realize this is a very rash assumption at this point.) I create the representation of my plate, and link it to the above part specification. I now pull up a parametric representation of a hole (assuming I have one), and link it to the above feature. My CAD package accesses the diameter information and sizes the hole for me. I can continue building models, linking to existing concepts generated earlier. When the design is mature enough, I can send the information to my FEM package, it can access the material specification, which could link to a database containing all the necessary parameters (modulus, density, etc.) for its calculations. The information could also flow the other way. For example, if I were able to identify parts by name as I create them in a solid modeling environment, then that information can be used to create the database entries, details of which could be filled in later. Following on the ideas established in the LINKER model, the user can also specify relationships between elements of the design. If the structural modeling package is capable of inferring these relationships, then this information could be fed to the database, and the assembly and disassembly plans can begin to take shape. I think it might be impractical to go the other way with this information, since it lacks any geometrical data.

The key needs are common part identifiers and standard interface data structures. Using the concept of a part, i.e., some component in a design, comes the closest to supplying a common data item to work from that is applicable to most design packages. The fact remains that most geometric modeling packages do not support this concept right now. Those that do (e.g., the latest IDEAS package) have a very limited parts library, and it is in a non-standard format that cannot access or be accessed from outside databases. (Busick and Chong, 1995)

We also need standard data structures in order to provide portability. Each package will need to have a hook to and from the central database that passes and receives the information in a way that each can understand. The object-oriented approach that we are employing in this development shows the most promise for providing this portability. Object-oriented methods have proven to be the next standard for software development, particularly in the realm of data structure development (e.g., OSU's CIS 680 data structures course), and distributed systems. (Booch, 1994)

Advantages and disadvantages of common design data bases

Advantages

The major advantage is the ability to provide a central access point for basic information on as many aspects of a device's design as possible. Since the proposed system is to be applied primarily to early design phases, I have not attempted to include data structures in the proposed specification for spatial information or for various types of numerical results from

tools such as FEM packages. I do believe that it forms the basis for accessing information necessary in the later stages of design also.

The idea of accessing common design bases also allows for access to information of existing parts. While obviously valuable for configuration (routine) design, there is a strong push in industry to standardize part use. This effort results in savings by decreasing the amount of design work necessary to field a system, decreasing accounting of parts and part specifications, decreasing the inventory and spares necessary for maintenance and service, and allowing for high volume purchase or manufacture of standard parts. An automated search to flag the existence of suitable parts early on could be a major benefit in this regard. This involves a matching process which could be very difficult without constraining the semantics of the behavior model, as indicated previously.

Along this same line, another aspect of having a common design base would be to present the designers with alternative component structures in the component search task that fulfill the design requirements; i.e., when one behavior or function maps to multiple components or subsystems. In the previous example, where the design description called for a behavior "maintain contact between part A and part B," the system could present a list of fasteners including mechanical or chemical type. It could also list something like Velcro as a possible candidate, which may be perfectly suitable for the task, but may not have otherwise occurred to the design team to use.

Disadvantages

One major disadvantage that I see is the increased computing and information overhead that all of this will entail. The amount of information inherent in any reasonably complex system will be enormous on its own. Tracking down dependencies and maintaining consistency in all of the data objects could be a formidable task. I can also foresee a great deal of effort expended just in maintaining the data base with current information on parts, materials and processes.

This raises another interesting point that I've seen engineering functions at GE, GM, and Ford dealing with, and that is "how do you maintain data integrity." Even in the context of a single (but fairly complex) design, it is not inconceivable that someone could enter erroneous data that would become part of a central data system. Given a situation, as I envision, where that information is propagated to all other interdependent entities, the effect could be disastrous. There is a need for error checking, but this becomes very difficult in knowledge-based systems, since there is nothing to fall back on, like a check calculation (e.g., solving the ideal gas law for R and checking against a known value).

Use of conceptual design libraries

The behavior model, proposed as a means of defining a device in the conceptual design phase, may provide a way of accessing design library entries. Library entries will be a composite of the functional, behavioral and structural description of a design object, thus the task of

mapping between various representational views becomes extremely important. The key element is being able to recognize that a structural entity, be it a part or subsystem, or a class of structural entities (i.e., fasteners, motors, valves, etc.) exists that fulfills the functional/behavioral specifications. Therefore, the structural entities will need to have a specification of exhibited behavior associated with them that fully describes the applicable pre- and post-conditions that will result from the component or system performing its behavior. This is then compared to the behavior required by the design to determine suitability, as shown in Figure 15.

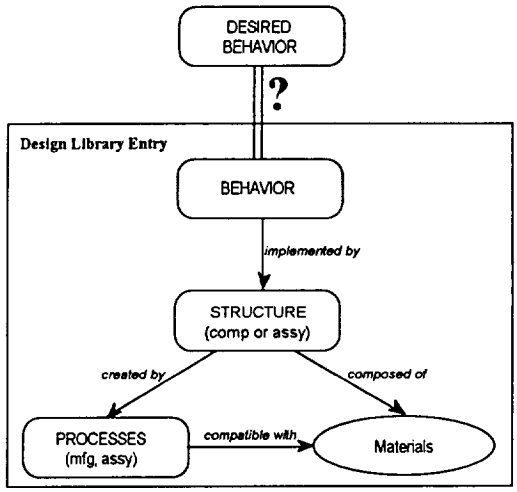


Figure 15: Behavior Matching in a Design Library

For example, a design description may call for a behavior "maintain contact between part A and part B." My design library would need to have a behavior description "maintain contact between two parts" that maps to a class of components known as fasteners. We could seek the match via "maintain contact between *" and automatically infer the use of some kind of fastener, be it a mechanical fastener, an adhesive, a clamp, etc. Further specifications, such as separation force of a boiler plate derived from internal pressure, would be used to narrow the possible candidate to bolts or rivets larger than a certain size, certain grades of epoxy, etc. This may require the use of semantic matching (which means the implementation of a semantic system) between the behavioral representation under development and the library entries.

Another source of information for analysis packages may exist in the pre- and post-condition specifications for the component behaviors. These conditions are state variables which will include items like operating temperature and power requirements at very high levels of abstraction, and items such as loads and torques at lower levels of abstraction. While I can say that this information should be available within the behavior model structure, I cannot yet say exactly how it will be made available. That lies within the scope of my proposed research.

Regulatory issues

Regulatory issues pose a very interesting set of constraints on design. My experience with the medical devices industry, at least from the conceptual design standpoint, showed that it basically boils down to the implementation of a rule-based system that serves to interpret the regulations imposed on the design. For example, only certain materials are approved for contact with the human body, and even fewer for placement inside the human body. The proposed system could be configured to perform various checks, via a rule-based system, to ensure that material specifications were appropriate, or that testing procedures (another object in our model) were complied with. The rule-base would have to be partitioned, since various levels of regulation exist depending on the application of the device under development. A key need is standard data structures so that the rule-base can communicate with that facet of the database to which it applies.

Applying framework concepts to post-manufacturing issues

Evaluating the effects of design decisions

We began the process by looking at a semantic syntax, as outlined in the previous annual report, and developing a preliminary object model representing the relationships of various design issues (materials, manufacturing, assembly, testing) to the elements under design. This model comprises the more "physical" aspects of the design artifacts, and have characteristics defined as attributes in the model. These objects also have methods associated with them that provide the means for evaluating the design issues. The methods will be a direct reflection of the type of evaluation available; i.e., explicit closed form solutions as equations, and qualitative methods as a knowledge- or rule-base).

From decision analysis, we know that decisions are made traversing down through a decision hierarchy, while the effects of decisions are propagated back up the hierarchy. Placing the discussion in the context of design, decisions are made as we move down through the parts hierarchy; i.e., from conceptual to detailed design. The effects of decisions, of course, propagate up through the hierarchy.

Service and maintenance issues

Model and inferencing techniques:

Cunningham and Cox (1972) point out that during the early phases of equipment or system design, the key requirement affecting service and maintenance decisions is overall system availability. Further, they place more emphasis on inherent availability as a function of designed-in maintainability. Inherent availability can be expressed as Equation 1.

$$A_i \frac{MTBF}{MTBF + MTTR} \quad (1)$$

where: MTBF = mean time between failure
MTTR = mean time to repair

MTBF is an input to the system based on reliability analysis. We can use the reciprocal of MTBF as the failure frequency input into the service cost equation. We can compute MTTR by summing the time required to perform each service step, as we do in our current serviceability analysis system. One question that remains is exactly what service actions (parts needing replacement or repair) are required when a particular part fails.

The answer can be provided by an FMEA capable of generating multi-failure scenarios. For example, an oil pump failure may result in a main crank bearing failure, requiring both items to be replaced. As stated in the behavior model development section, we can identify failure effects in two ways: 1) by the results of non-behaviors, or 2) by examining known failure paths. Suppose we have the model fragment shown in Figure 16.

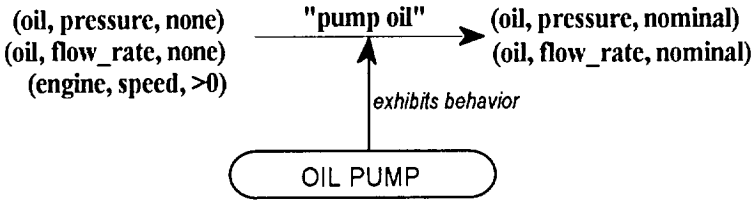


Figure 16: Oil Pump Behavior Model Fragment

If we can ask the question "What if the oil pump fails?", our answer can be inferred from the fact that the "pump oil" behavior has *not* occurred. Therefore, we know that the oil pressure and flow rate are none, since their states have not changed. In the context of the serviceability question, we defined, by direct inference (Di Marco, et al., 1995), the service operation:

(failure: oil pump) ⟶ **(replace: oil pump)**

In addition, we can also infer a diagnosis path by determining the behavior that did not take place, based upon the deviation of state variable from their expected values. In our example, we expected oil pressure and flow to be nominal, and instead they are none. The behavior that did not happen was "pump oil," since its preconditions match the existing state, and postconditions match the expected state. The oil pump is the element responsible for implementing this behavior, so that we can infer:

Expected conditions:	(oil, pressure, nominal)
	(oil, flow_rate, nominal)
Existing conditions:	(oil, pressure, none)
	(oil, flow_rate, none)
Non-behavior:	"pump oil"
Mapped element:	oil pump
Conclude	(failure: oil pump)

Experience tells us that no oil flow or pressure generally leads to crank bearing failure, so that we can augment the model with a failure path:

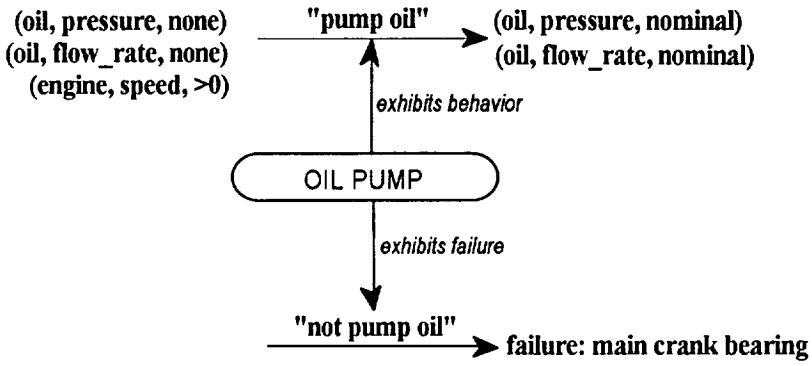


Figure 17: Augmented Oil Pump Behavior Model Fragment

The serviceability inferencing will now include:

(failure: oil pump) ⟶ (failure: main crank bearing) ⟶ (replace: main crank bearing)

Our previous work demonstrated the ability to infer service steps from our structural representation (Eubanks & Ishii, 1993), and was designed around the concept of single element service operations, such as "replace oil pump." In the context of multiple simultaneous failures, we must modify this concept to exclude duplicate labor steps that might result from generating the step sequences separately. In the above example, inferencing on both "replace oil pump" and "replace bearing" will generate steps such as "remove oil" for both repairs. These duplicate steps will need to be eliminated from the list of total service steps prior to calculating the service cost. There may also be items that may require replacement whenever the system is down for maintenance or repair, typically things like gaskets, seals, working fluids, and other consumables. We can handle these situations by flagging the part for replacement whenever it appears in a service step list.

Life-cycle service costs:

Given an arbitrary repair action j consisting of labor steps 1 through n, a simplified model of the life-cycle service cost for that repair action is shown in Equation 2.

$$LCSC_i = f_j \sum_{i=1}^{n} \left\{ \left[(t_L + p_L) \times c_{LR} \right] + \left[c_P + p_P \right] \right\}_i \quad (2)$$

where: t_L = labor time (hours)
p_L = labor time penalty (hours)
c_{LR} = labor rate ($/hour)
c_P = part or material cost ($)
p_P = part or material cost penalty ($)
f_j = frequency of repair action j
n = number of labor steps in repair action j

The above cost equation assumes that repair action j is only performed as needed; i.e., when something fails.

In mission critical systems, we generally try to anticipate such failures, and establish periodic maintenance, during which time several inspections and minor repairs are made in order to prevent these type of failures. Given a particular maintenance cycle k consisting of 1 through m maintenance actions are performed, a simplified model of the life-cycle maintenance cost is shown in Equation 3.

$$LCMC_k = f_k \sum_{j=1}^{m} \sum_{i=1}^{n} \left\{ \left[(t_L + p_L) \times c_{LR} \right] + \left[c_P + p_P \right] \right\}_{i,j} \quad (3)$$

where: m = number of maintenance actions in maintenance cycle k

Our previous works demonstrated the ability to infer service steps from our structural representation (Eubanks & Ishii, 1993), and was designed around the concept of single element service operations, such as "replace oil pump." In the context of either multi-task maintenance operations or multiple simultaneous failures, we must modify this concept to exclude duplicate labor steps that might result from generating the step sequences separately. For example, an oil pump failure may result in a bearing failure, requiring both items to be replaced. If we generate the labor steps separately, both lists may contain steps such as "remove engine" and "remove oil." These duplicate steps will need to eliminated from the list of total service steps prior to calculating the service cost.

There will also be items that may require replacement whenever the system is down for maintenance or repair, typically things like gaskets, seals, working fluids, and other consumables. We can handle these situations by flagging the part for replacement whenever it appears in a service step list.

Multi-task maintenance operations will be input by the user, along with a specified frequency or MTBM. Multiple simultaneous failures can be inferred by the automated FMEA algorithm, currently under development, based on MTBF data. Given complete statistical data (median and distribution) for MTBM and MTBF, we could run a discrete system simulation model to generate the maintenance and service events. In the early stages of design, we will probably

be limited to estimated times, for which we can calculate event frequency over the life-cycle. For example, if we specify the MTBM for a system to be 2000 hours, and the system has an estimated life of 25,000 hours, the we can estimate the maintenance frequency, f_m, using Equation 4.

$$f_m = \frac{Estimated\ Life}{MTBF} = \frac{25{,}000}{2000} = 12.5 \cong 12 \qquad (4)$$

A similar calculation will yield the service frequency, f_s, based on MTBF.

In a detailed maintenance and service cost analysis, we should consider including three additional major costs: the cost to remove and replace the sub-system from the system (i.e., cost to R/R a jet engine from the aircraft), the cost of system downtime, and, if necessary, cost of parts or subsystem transportation. The precise application of these costs depends on the service logistics decisions; i.e., is the system down for part replacement, or down for the entire length of the repair.

Recylability issues

In my previous report, I discussed the concepts of Design for Product Retirement, DFPR, and presented a system retirement cost equation. The focus of DFPR is strictly end of product life-cycle. Material life-cycle is a concept that attempts to encompass the idea that a material may be reprocessed and either reused in a similar application, or used in a lower performance application, over several product life-cycles before being discarded due to the degradation of material properties. (Ishii, et al., 1994)

Application to service parts

We need to expand the notion material life-cycle into the use phase of the product life-cycle. For example, we generally replace automotive engine oil as many as 40 times over the 10 year life-cycle of the vehicle. The material and labor costs associated with this requirement are a significant part of the life-cycle cost of using the product. As a consequence, we need to expand the analysis to include the environmental impact of service and periodic maintenance. The modified life-cycle service cost can be calculated using Equation 5.

$$LCSC_i = f_j \sum_{i=1}^{n} \left\{ \left[(t_L + p_L) \times c_{LR} \right] + \left[c_P + p_P \right] + \left[c_{dispose} + \left(c_{refurb} - v_{refurb} \right) + \left(c_{recycle} - v_{recycle} \right) \right] \right\}_i \qquad (5)$$

where: $c_{dispose}$ = cost of disposal
c_{refurb} = cost of part of assembly refurbishment
v_{refurb} = value of refurbished part of assembly
$c_{recycle}$ = cost of material reprocessing/recycling
$v_{recycle}$ = value of reprocessed/recycled material

In this equation, the cost of material recovery is borne by the labor cost terms (first set of brackets) of the service cost equation, which is a function of system configuration (i.e., component access) decisions. The level of service or maintenance (part, module, subassembly) is reflected in the part cost terms (second set of brackets). The impact of reuse, recycling, and disposal costs is reflected in the last set of brackets, and will be a function of material selection, and end of material-cycle disposition (as opposed to end of material-life disposition).

The material reprocessing conundrum

Product retirement plans require design for disassembly and material reprocessing. Both of these must be balanced against the costs to design and manufacture a product, and its performance in the field.

A great deal of work continues on design for disassembly. Japan's MITI Mechanical Engineering Laboratory continues to work on technologies such as "swarming robots" for massive and, for the most part, destructive disassembly of larger systems such as automobiles. Paul and Beitz continue to investigate fastener design for ease of disassembly in response to Germany's so-called "take back" laws. Unfortunately, we can design for disassembly all we want, but if all were left with is nice clean piles of non-reprocessable junk, then we gain very little.

For the recyclability analysis, the burning questions still involve reprocessing technologies. Current reprocessing technologies capable of generating engineering grade materials are generally applied to single material waste streams, or to non-chemically mixed waste streams. Non-chemically mixed refers to parts or assemblies where materials can be removed using either mechanical means (e.g., brass bushings in plastic, coatings on glass), or by selective chemical or thermal removal (e.g., sweat furnaces for printed wiring boards) Reprocessing technologies for many engineering composites and compounds (e.g., thermoset plastics, exotic metal alloys) are unknown.

But how do we compute costs which are unknown or uncertain? In general, the answer to that question is that, for the time being, we must simply rely on knowledge, and any inferences that can be made from that knowledge. Thus, our use, both past and present, of knowledge based systems for product retirement and recycling knowledge. For example, we have developed very qualitative methods using material compatibility models (Marks, et al., 1993), which remain valid, and will be used for reprocessing cost estimates in this model. This is not without precedence. We continue to see serviceability issues stated primarily in terms of knowledge, such as providing adequate clearance, clearly labeling service items, etc. The key element to be addressed is how to integrate design issue knowledge into the more qualitative elements of product design.

Framework integration

It makes sense to select materials and processes that provide for the minimum life-cycle cost for a given level of performance. Factoring in the cost or value of a material at the end of the product life-cycle may change the outcome of that decision point. For example, a part which costs $1.00 to produce and is worth $0.40 at product retirement is a better value than a part which costs $0.75 to produce and has no retirement value. We bring this aspect of material cost into the framework as shown in Figure 18.

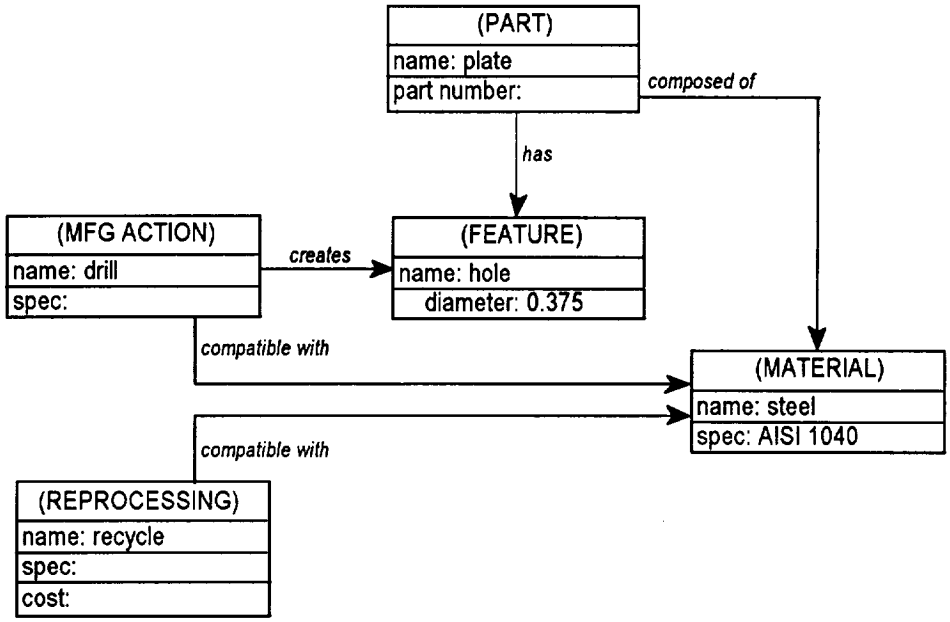


Figure 18: Integration of Reprocessing into Structural Model Framework

While the manufacturing action may be specified in terms of material removal rates, machine cost, etc., the reprocessing action will probably be a knowledge-based application that returns an estimate based on qualitative evaluations. In our past work, we have used a simple if-then rule structure to implement the knowledge component of our product retirement analysis program. (Marks, et al., 1993) While good for proof of concept, we should consider using a commercial package designed for knowledge/rule based applications.

Future work

Over the next 12 months, we will finalize the structure of the system framework, and implement a prototype program to demonstrate the concepts presented in this report. The implementation will feature modules for the device behavior and structure representations of the framework, and will be tied together via a common link to the physical entities that comprise the system being designed. The system will incorporate a semi-automated FMEA module based on a description of device behavior, and life-cycle cost evaluation modules for assembly, serviceability and recyclability. We anticipate that advances in technology, particularly in the area of recyclability, will bring new and better models and evaluation

methods. Thus, we will employ an object-oriented topology for the system to take advantage of the ability to add and modify the various programs used to evaluate the various life-cycle costs without disturbing the basic system framework. We intend to provide a full specification of the program structure, and implement a PC-based prototype system as a proof of concept.

References

Abu-Hanna, A., Benjamins, R., Jansweijer, W. (1991), "Device Understanding and Modeling for Diagnosis," IEEE Expert, Vol. 6, No. 2, pp. 26-32.

Arimoto, S., T. Ohashi, M. Ikeda and S. Mitakawa (1993), "Development of Machining-Producibility Evaluation Method (MEM)," *Annals of the CIRP*, vol. 42/1/1993, pp. 119-122.

Berzak, N. (1991), "Serviceability by Design," *Proc. of the 23rd Int'l SAMPE Technical Conference*, October 21-24, 1991, pp. 1060-1071.

Booch, G. (1994), Object-oriented Analysis and Design with Applications, 2nd ed., Benjamin/Cummings Publishing Co., Inc., Redwood City, CA.

Boothroyd, G. and P. Dewhurst (1983), *Design for Assembly: A Designer's Handbook*. Boothroyd Dewhurst, Inc., Wakerfield, RI, 1983.

Busick, D. and C. Chong (1995), Personal communication, August 22, 1995.

Clark, G. and R. Paasch (1994), "Diagnostic Modeling and Diagnosability Evaluation of Mechanical Systems," *Proc. of the ASME 6th International Conference on Design Theory and Methodology (DTM '94)*, Minneapolis, MN, pp. 179-189.

Cunningham, C. and W. Cox (1972), Applied Maintainability Engineering, John Wiley & Sons, New York.

deKleer, J. and J. Brown (1984), "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, 24(3), pp. 7-83.

Di Marco, P., C. Eubanks and K. Ishii (1995), "Service Modes and Effect Analysis: Integration of Failure Analysis and Serviceability Design," *Proc. of the 15th Annual International Computers in Engineering Conference (CIE '95)*, Boston, MA, pp. 833-840.

Eubanks, C. and K. Ishii (1993), "AI Methods for Life-cycle Serviceability Design of Mechanical Systems," *Artificial Intelligence in Engineering Journal*, 8(2):127-140.

Eubanks, C. (1994), "Annual Technical Report for NASA GSRP Project NGT-51193," September, 1994.

Fathailall, A. and J. Dixon (1994), "A Method for Early Manufacturability Evaluation of Proposed Tolerance Plans for Thin-Walled Parts," *Proc. of the ASME 6th Int'l Conf. on Design Theory and Methodology*, Minneapolis, MN, pp. 9-17.

Finger, S. and J. Rinderle (1989), "A Transformational Approach to Mechanical Design Using a Bond Graph Grammar," *Proc. of the ASME 1st Int'l Conf. on Design Theory and Methodology*, Montreal, pp. 107-116.

Glantschnig, W. (1993), "Green Design: A Review of Issues and Challenges," *Proc. of the 1993 IEEE Int'l Symposium on Electronics and the Environment*, Arlington, VA, pp. 74-77.

Goel, A. and B. Chandrasekaran (1989), "Functional Representation of Designs and Redesign Problem Solving," *Proc. of the 11th Int'l Joint Conf. on Artificial Intelligence*, Detroit, MI, pp. 1388-1394.

Hamscher, W., Console, L., de Kleer, J. (eds.) (1992), Readings in Model-based Diagnosis, Morgan Kaufmann Publishers, San Mateo, CA.

Homem de Mello, L. and A. Sanderson (1991), "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Trans. on Robotics & Automation*, vol. 7, no. 2, pp. 228-240, April, 1991.

Ishii, K., C. Eubanks, P. Di Marco (1994), "Design for Product Retirement and Material Life-cycle," *Materials and Design Journal*, 15(2): 225-233.

Iwasaki, Y. and B. Chandrasekaran (1992), "Design Verification Through Function- and Behavior-Oriented Representations," *Artificial Intelligence in Engineering Design '92* (J. Gero, ed.), Kluwer Academic Publishers, Boston, pp. 597-616.

Iwasaki, Y. and H. Simon (1994), "Causality and Model Extraction," *Artificial Intelligence*, (67)?Kluwer Academic Publishers, Boston, pp. 597-616.

Keuneke, A. (1991), "Device Representation: The Significance of Functional Knowledge," *IEEE Expert*, vol. 6, no. 2, April, 1991, pp. 22-25.

Kuipers, B. (1984), "Commonsense Reasoning About Causality: Deriving Behavior from Structure," *Artificial Intelligence*, 24(3), pp. 169-203.

Makino, A., P. Barkan, L. Reynolds and E. Pfaff (1989), "A Design for Serviceability Expert System," *Proc. of the ASME Winter Annual Meeting 1989*, San Francisco, CA, pp. 213-218.

Marks, M., C. Eubanks, M. Shriver and K. Ishii (1993), "Life-cycle Design for Recyclability," *Proc. of the JSME-ASME Joint Workshop on Design '93*, Tokyo, Japan, pp. 19-24.

Marks, M., C. Eubanks, K. Ishii (1993), "Life-cycle Clumping of Product Designs for Ownership and Retirement," *Proc. of the ASME 5th International Conference on Design Theory and Methodology*, Albuquerque, NM, pp. 83-90.

Morjaria, M., M. Simmons, J. Stillman (1992), "Development of Belief Networks Technology for Diagnostic Systems," General Electric Corporate Research and Development Technical Information Series, GE CR&D, Schnectady, NY, April, 1992, 35 pages.

Navin-Chandra, D. (1991), "Design for Environmentability," *Proc. of the ASME 3rd Int'l Conf. on Design Theory and Methodology*, Miami, FL, pp. 119-125.

Ormsby, A., J. Hunt, M. Lee (1991), Towards an Automated FMEA Assistant, *Applications of Artificial Intelligence in Engineering VI*, eds. Rzevski, G. and Adey, R., Computational Mechanics Publications, Southampton, Boston, pp. 739-752.

Poli, C., P. Dastidar and R. Graves (1992), "Design Knowledge Acquisition for DFM Methodologies," *Research in Engineering Design*, (1992) 4:131-145.

Priest, J. (1988), *Engineering Design for Producibility and Reliability*, Marcel Dekker, Inc., New York, 1988

Sturges, R. (1992), "A Computational Model for Conceptual Design Based on Function Logic," *Artificial Intelligence in Design '92*, Kluwer Academic Publishers, Boston, pp. 757-772.

Sturges, R. and M. Kilani (1992), "Towards an Integrated Design for an Assembly Evaluation and Reasoning System," *Computer-Aided Design*, vol. 24, no. 2, pp. 67-79, February, 1992.

Suh, N. (1990), *The Principles of Design*, Oxford University Press, New York, 1990.

Welch, R. and J. Dixon (1994), "Guiding Conceptual Design Through Behavioral Reasoning," *Research in Engineering Design*, (1994) 6: 169-188.

Ullman, D. (1992), The Mechanical Design Process, McGraw-Hill, Inc., New York.

Ullman, D. (1993), "A New View on Functional Modeling," *Proc. of the 9th International Conference on Engineering Design (ICED'93)*, The Hague, pp. 177-193.

Umeda, Y., T. Tomiyama, Y. Yoshikawa (1992), "A Design Methodology for a Self-Maintenance Machine Based on Functional Redundancy," *Proc. of the 4th Int'l. Conference on Design Theory and Methodology*, Scottsdale, AZ, pp. 317-324.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1996	Final Contractor Report

4. TITLE AND SUBTITLE
Integrating Post-Manufacturing Issues Into Design and Manufacturing Decisions

5. FUNDING NUMBERS
WU-505-63-5B
G-NGT-51193

6. AUTHOR(S)
Charles F. Eubanks

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Stanford University
Stanford, California 94305-4021

8. PERFORMING ORGANIZATION REPORT NUMBER
E-10206

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-198474

11. SUPPLEMENTARY NOTES
Project Manager, Christos C. Chamis, Structures Division, NASA Lewis Research Center, organization code 5200, (216) 433-3252.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 39
This publication is available from the NASA Center for AeroSpace Information, (301) 621-0390.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

An investigation is conducted on research into some of the fundamental issues underlying the design for manufacturing, service and recycling that affect engineering decisions early in the conceptual design phase of mechanical systems. The investigation focuses on a system-based approach to material selection, manufacturing methods and assembly processes related to overall product requirements, performance and life-cycle costs. Particular emphasis is placed on concurrent engineering decision support for post-manufacturing issues such as serviceability, recyclability, and product retirement.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Manufacturing; Methods recycling; System; Materials; Assemblies; Product requirements; Life-cycle costs	35
	16. PRICE CODE: A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

National Aeronautics and
Space Administration

Lewis Research Center
21000 Brookpark Rd.
Cleveland, OH 44135-3191