A Computational Model for Conceptual Design Based on Function Logic

R.H. Sturges

EDRC 24-96-92

A COMPUTATIONAL MODEL FOR CONCEPTUAL DESIGN BASED ON FUNCTION LOGIC

R. H. STURGES
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract. Function logic and function block diagrams have been successfully employed in preliminary and conceptual design processes for several decades. This paper describes a computational model of this process with extensions of the manual approach. It provides for a systematic identification and definition of form and function variables and identifies a three-level function/allocation/component information structure to represent the state of the design. We outline the inputs, outputs and operations on the form and function variables as a key step prior to the synthesis process. We also illustrate by example how to transfer functional designs across specialist domains.

1. Introduction

Concurrency of product and process design tasks has become recognized as extremely valuable in reducing and predicting life cycle cost. The short-term benefits of such concurrency are reflected in reduced time to market and reduced manufacturing cycle time (Andreason et al., 1988). The longer term benefits accrue from the leverage which well-reasoned design methodologies exert over the product life cycle (Sturges et al,, 1986). The decisions made at the earlier stages of a design have a disproportionately large share of impact than the latter. Thus, it is the preliminary or conceptual stages of the Form-Function synthesis process which we are interested in representing and facilitating. Conceptual design is distinguished from other phases of the design process as illustrated in Figure 1 (Westinghouse, 1984). The activities in the preliminary and conceptual phases differ from the latter detailed design work. In mechanical design, conceptual issues are largely functional, with less emphasis on form. Conventional detailed design, on the other hand, requires us to synthesize forms which will not compromise the given functions.

The conceptual phase concerns the problem of coming up with new ideas or new solutions to older problems (Pugh, 1981). Good conceptual design means innovation, and an innovative design comes about when one deliberately tries to create one (Perkins, 1981). For example (Bailey, 1978),

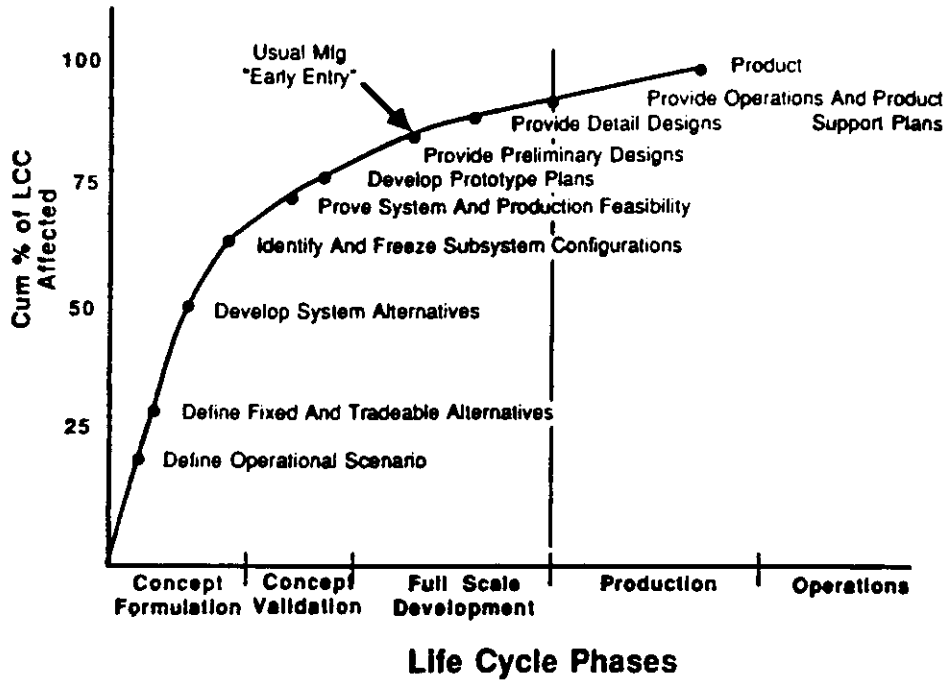


Fig 1. Conceptual Design in the Life Cycle Cost of a Product

> An engineer carefully studies power losses in a coal-fired plant and is able to increase efficiency by 0.1%. Another engineer studying the same data conceives of the idea of using direct energy conversion to use the waste heat and increases efficiency by 5-10%.

Although the essential result (efficiency) is the same, the functions used to produce it are very different. The former approach represents optimization of a given functional model. The latter represents modification of this functional model. The issue in conceptual design theory is to understand the processes which lead to innovation and to create tools which generate such step changes in function in an orderly and repetitious basis. Processes for prompting innovative design are currently typified by "Buhl's Seven Steps" which are shown in Table 1. Similar approaches are given by Bailey (1978). Processes for managing conceptual design and the process of innovation have long been described in the Value Analysis and Value Engineering literature (e.g., Bytheway, 1964; Ruggles, 1971; Miles, 1982). This method, when applied by a small group of engineers, has been shown to be consistently effective in achieving focused conceptual design goals, such as

achieving a given level of product performance, redesigning to reduce costs below a given threshold, etc. A study by the American Ordnance Association of a sampling of 2000 of its projects revealed improvements in cycle time, reliability, quality and maintainability in excess of 60% (Prendergast, 1982). That the process works is not debated. Understanding and translating the process into software tools remains a challenge and an opportunity.

TABLE 1
Buhl's seven steps (Perkins)

Recognition. Recognize that a problem exists and decide to do something about it.
Definition. Define problems in familiar terms and symbols; dissect into sub-problems; determine limitations and restrictions.
Preparation. Compile past experience in the form of data, ideas, opinions, assumptions, etc.
Analysis. Analyze preparatory material in view of defined problems, inter-relations, and evaluation of all information that could bear on the problem.
Synthesis. Develop a solution or solutions from developed information.
Evaluation. Evaluate possible solutions. Verify and check all facets of the solution. Reach a decision.
Presentation. Plan a strategy for convincing others and carry it out.

Tools for carrying out the essence of these conceptual design activities include detailed checklists, data retrieval and management systems, function logic, evaluation methods, and presentation techniques. The single common element in each of the examples above is the expression of a design at the conceptual stage in functional terms (the function block diagram or FBD), and the deliberate manipulation of this functional representation (the function logic process). In this paper we describe a model for conceptual design in which functional representations are extended to include other elements of the manual process and in which the manipulation of the functions is facilitated. We have employed the terminology of the Society of Value Engineers in dealing with functional representations of conceptual design (Bytheway, 1965). Specifically, *function* refers to largely domain independent characteristics or behaviors of elements or groups of elements of a design. The *intent* of a design is expressed by the totality of its functional elements and their structure. The *basic function* refers to the single intended output or use of the product, with *secondary functions* describing necessary but less critical constraints. *Side effects* refer to unintended behaviors which derive from implementation decisions.

2. A Computational Model for Conceptual Design

2.1. OVERALL ARCHITECTURE AND APPROACH

In this section we describe a representation of and a systematic approach to conceptual design based on function logic. The central concept of the approach is to capture design intent through a chain of functional description and reasoning and to highlight the dependencies among the sub-functions of a design. This ability is crucial to life cycle success since the *basis* for a specification changes with time, resources, market, technology, and the evolution of the form of the design itself. This basis and its evolution are rarely present in design representations. Layout and detail drawings tell us "what" but rarely "why." The reverse process of extracting function from form is problematic since critical information must be synthesized to replace that which was discarded along the way. Even a very detailed functional specification gives you only "the answer" while discarding the numerous questions which led to its substance. It is these questions which need to be reexamined when a design changes to meet a new need or requires analysis for improved performance. Also, the interfunctional dependencies of a design ("degree of coupling") are not obvious from its specifications and yet may be highly sensitive to them. Functional dependence is often realized only after a given technology (form of manufacture) or component (form of artifact) is chosen. In short, form-function synthesis processes need not only specifications; they also need a way to systematically produce and manage these early in the evolution of a design.

As mentioned in the introduction, our representation of conceptual design, based on Miles (1982), employs functional descriptions of products or processes according to a set of linguistic and hierarchic rules. Existing artifacts are analyzed by developing their function logic in a bottom-up fashion; new artifacts are synthesized generally in a top-down fashion (Bytheway, 1971). Our extensions to the FBD representation (Sturges et al., 1990a) comprise a three-tiered structure (Figure 2) consisting of: (i) function blocks (compact verb-noun descriptors of what the design *does* rather than what it *is*) with links to other blocks; (ii) allocations (constraints, performance requirements, specifications and resources (Kantowitz and Sorkin, 1987); and (iii) components (artifacts that satisfy the given function). Design intent is captured and managed during the conceptual design process by each tier of this structure as follows.

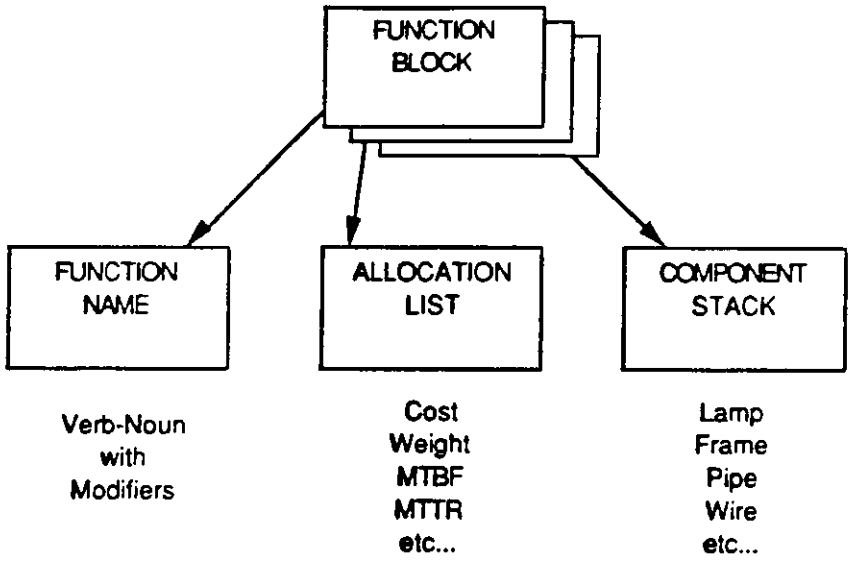


Fig. 2. Three-tier structure for a function logic diagram.

2.2. INPUTS

The *problem* in conceptual design is the articulation of the function in sufficient detail to suggest and monitor the development of form through existing knowledge and decision-making methods. Since the process in its manual implementations is reversible, the inputs may be derived from two sources. In forward design, the *inputs* are verbal, syntactic and numeric descriptors which evolve with understanding and negotiation. In reverse design, the inputs comprise the set of such descriptors attendant to each of the given components and subassemblies of an existing design. Since there is little evidence of design proceeding exclusively in the forward mode, descriptors from related but independent designs will always be present (Ullman et al., 1988). In this section we describe a representation of and a systematic approach

2.3. FORM AND FUNCTION VARIABLES

The function variables consist of verb/noun pairs and links between these. The general form of the FBD is shown in Figure 3. The function block (or node) contains the function name (what is done) expressed as a generic noun/verb pair to describe the function of the product or process. The verb must be active; the noun must be measurable. 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: lower-level functions. Links connect each high-level function with its lower-level function according to this how/why relationship. Links other than how/why have been identified between function blocks, viz.: causal, temporal, informational, alternative and revisional (Sturges and Kilani, 1990). A description of each link is given in Table 2. Strict hierarchy is not required since a more specific function may satisfy more than one less specific function in the diagram.

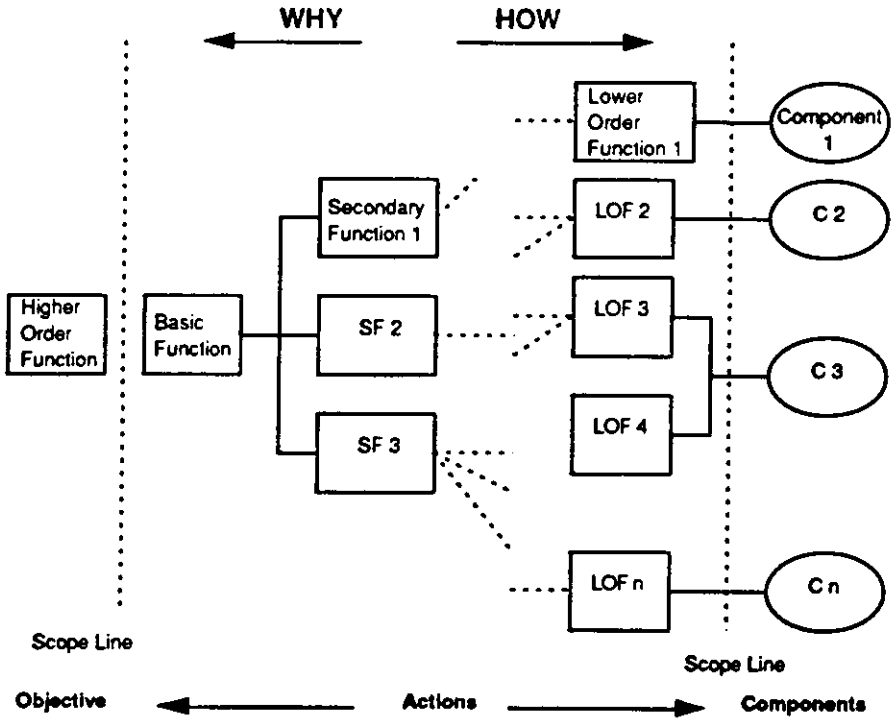


Fig. 3. The general form of a function logic diagram.

TABLE 2.
Links used in function block diagrams

And/Or: The "and" links are the conventional links indicated by Bytheway (1971), and are represented as solid lines connecting the blocks. The "or" links suggest a viable alternative solution for decomposing a function block. These links are represented as dotted lines.

Temporal: The "temporal" link connects functions that occur "at the same time," as suggested by Ruggles (1971). Usually, these function blocks are portrayed vertically and are connected with a dotted/dashed line. An arrow indicates the process flow, or which function must occur before the other. Temporal links are also used to specify a material flow, because the stages in the process occur in time or event sequence.

Causal: The "causal" link is a result of the Miles (1982) side-effect concept and is represented as a solid line with an arrow to indicate which function was created as a side effect. The allocations associated with the side-effect function should be checked with other blocks in the FBD to discover opportunities and/or conflicts. For example, the creation of heat in an engine could be useful in the design of the cabin heating system.

Information: The "information" links indicate which functions are involved in an exchange of some type of information. This is represented as a dashed line with an arrow to indicate the direction of the information flow.

Revision: Tracing the design changes and the thought processes which led to them due to the changes in the design constraints is a valuable, but time consuming effort. We suggest that a record be kept through the use of the "revision" link with a time/date/author stamp before being filed away for future reference. These links are represented as solid, cross-hatched lines. Application of such links can be found in Subramanian (1990).

Chaining: "Chaining" links appear when a function block "decomposes" into only one function rather than several. This suggests that the noun/verb pair was not general enough or that an alternative was considered but discarded as non-viable.

The form variables consist of the allocations attached to these verb/noun pairs. The allocation list supplies what needs to be known about the design

as it evolves: constraints, performance requirements, specifications and resources. When complete, the allocations specify the behavior of each function and set the constraints for the form of the design. The synthesis of form itself in the component choices and details is not carried out by the conceptual process but, rather, by the parallel process of proposing and testing alternatives. When component decisions are made "early," their characteristics are attached to the related function blocks. Since they provide a compact description of a set of fixed constraints, they are not considered form variables.

2.4. REPRESENTATION OF FORM AND FUNCTION VARIABLES

First, the *basic function* of the design is established by agreement of the design team. If the basic function cannot be accomplished by a single known component, it is decomposed into several functions which collectively perform the function. These secondary functions may then be translated into components or recursively decomposed. The function decomposition process continues until: (1) the decomposition process is out of scope for the project, (2) there exists a synthesis technique which will complete the decomposition or propose artifacts, or (3) each function can be mapped into a component or structure that will accomplish it directly. Notice that so-called "non-functional" designs are included in this process: purely aesthetic functions such as **attract customer** or **enhance image** are valid and decomposable into artifacts. Such results are for the most part preliminary since practical designs rarely feature a one-to-one correspondence between functions and components.

Second, the allocations for each function block are developed and passed to the neighboring blocks by inheritance rules. Frequently, the allocation list attached to the basic function is only imprecisely known at the beginning of the processes (Paz-Soldan and Rinderle, 1990). As the lower-level functions are satisfied by artifacts or supported functions (which see, below), new categories of information are specified, passed up to the basic function and inherited by certain lower-level functions of the structure. Thus, the allocation list supplies what needs to be known as the design evolves. Its values drive the process of type, number and dimension synthesis through methods outlined in, e.g., Finger and Dixon (1989.)

Third, candidate artifacts and/or supported functions (general purpose models of behavior, coded domain-specific knowledge, formulas and examples) are specified to satisfy the lowest-level functions, which are in turn embodied by the higher-level ones. At this level, the creative process is constrained to issues of domain selection and expression of form through quantitative techniques.

2.5. OPERATIONS PERFORMED ON THE REPRESENTATIONS

2.5.1. *Operations Performed on the Representations.* As mentioned above, the descriptor of a function is a verb/noun pair. The verb must be active, such as "move" or "support." Passive verbs such as "provide" or "allow" are not permitted, since no action is requested. Nouns must be measurable, but cannot specify an artifact. Thus, "load" or "heat" are possible, but "effect" or "bracket" are not. This noun convention avoids domain-specific reasoning and ideation. This apparently simple construction represents a design in a fundamentally abstract form. It requires a deep understanding of the problem at hand and promotes discussion, especially among a group of designers, since it is often difficult to think about a design in this way without practice. The resulting set of descriptors, connected by links, captures the full *intent* of the design in a reasoned hierarchy

Function links represent the relationships between the function descriptors. The primary link between functions in a hierarchical sense is the *how/why* relation as illustrated in Figure 4, the FBD of a mousetrap. If no logical connection can be found, the part is subject to elimination or the function block diagram revised. Conversely, the process of creating new function blocks through successive decomposition is both guided and encouraged by the discipline of the how/why link. The expanded types (Table 2) represent the design and its reasoning in greater functional detail than the basic how/why logic without adding domain-specific information. An example with the information link is given below.

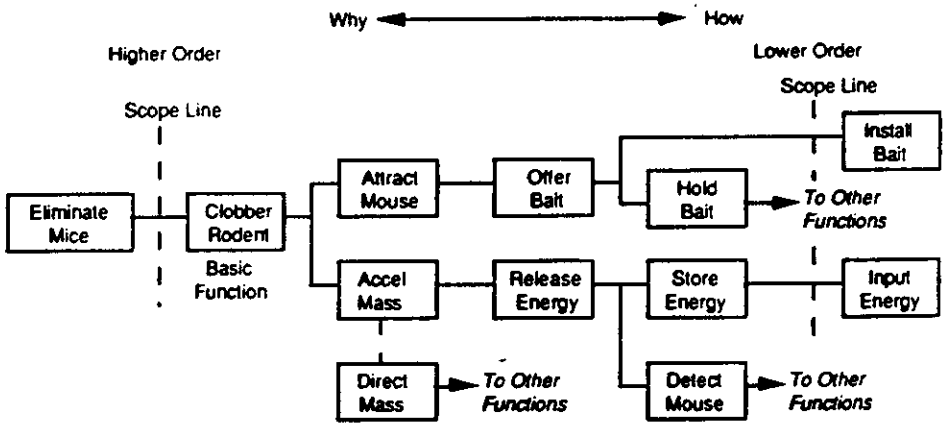


Fig. 4. Preliminary function block diagram of a mousetrap.
After Maurer (1984)

The FBD for any given design problem or product does not uniquely specify its physical form. Each verb/noun pair represents a conceptual decision, which is subject to the familiar "brainstorming" and creativity techniques, but is conceptually free from physical constraints. In this way function logic supports and encourages creativity.

2.5.1. *Operations of Form Variables.* During the development of the function logic for a product or process, and ideally before detail design begins, the designer must address the issue of who or what agent will perform each identified function. This process is known as *function allocation.* In addition to resources, the allocation list contains constraints, performance requirements and component specifications. An initial investigation into a more complex control problem, a motor/tachometer speed controller (Figure 5), indicated that there was no means in the function logic representation that could be used to represent the dynamic behavior of a system. This behavior, which includes the gain, natural frequency, damping, etc., is contained therefore in the allocation list.

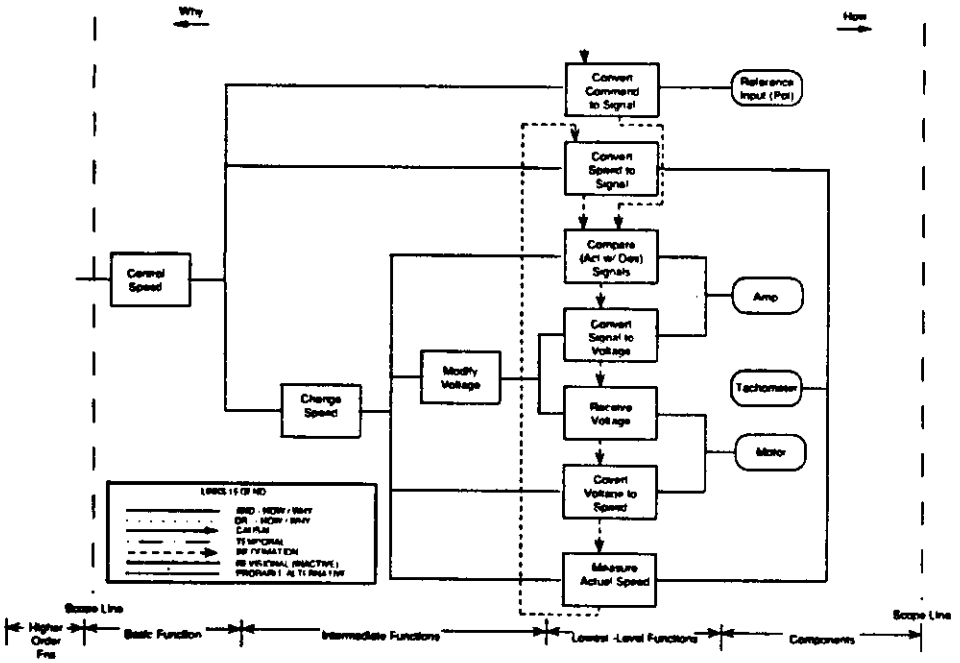


Fig. 5. A motor/tachometer speed controller FBD

In complex designs, the amount of information to be managed grows rapidly. However, the information that is required for any particular portion of the design will most likely be only a small fraction of the total amount

specified. A structure for the allocation list would allow smaller blocks of information to be discarded, or retained and propagated at that level. We have adopted a structure based on the verb classification of signal, energy or material as suggested by Pahl and Beitz (1988). This structure is given in Table 3. This structure also manages the allocation list according to: (1) arithmetic needs; (2) design domain; (3) disinheriting or filtering; or (4) distribution of allocation information.

TABLE 3.
Structure of an allocation list
(for position control example)

	Signal	Energy	Material
Cost	x	x	x
Weight	x	x	x
Size	x	x	x
Shape			x
Light		x	
Heat		x	
Humidity		x	
Human Factors	x		
Power		x	
Energy		x	
Voltage			x
Current			x
Speed			x
Load			x
Output:			
Distance			x
Type of Link			x
Maximum Allowable Error		x	
Change in Position	x		
Change in Direction	x		
Type of Controller			x
Design Domain	x		
State Space Equations	x		
Component Specifications:			
Gain of Encoders			x
Motor Characteristics			x
Side Effects			x

2.6. CONTROL

In addition to the above rules for FBD analysis and synthesis, one can apply specific steps to detect the existence of linkages in the function structure other than the basic how/why. At the present time, we have discovered a test for the information loops in the functional representation of a design which is independent of implementation domain, but closely related to abstract models of control theory (not to be confused with controlling the form-function process itself).

While the FBD is a general conceptual design representation, it may not always provide the designer with an intuitive "feel" for system structure or performance. The designer may be more comfortable with a more traditional form, a control loop for example. Since these are two different representations for the same system, it should be possible to create a procedure to transform one to the other. The study of domain crossing in design representation began with the reverse engineering of a model helicopter (Sturges et al., 1990a). This work revealed a linkage which connects functions that share or exchange information. The verb indicates the existence of the information link and its direction, that is, whether the block sends and/or receives the information. The form of this information flow is unspecified, but the associated noun defines the measurable quantity. Typically, the blocks which exhibit an information link are the most specific or elemental, that is, they lie to the far right on the FBD just to the left of their associated components. By creating a classical control block diagram from an FBD, the rules governing the conversion process between the control domain representation and function logic have been determined.

An FBD can also be developed from the control loop. Since there is more information about the design contained in the FBD, certain additional information needs to be extracted from the control loop designer. Thus, more rules are required for this case than for the FBD to control loop conversion process. These rules define how the control loop can be represented on the FBD, and how a control loop can be discovered on a developing FBD. Given a control loop, the governing rules identify which of the corresponding function blocks need to be connected. Simple examples from the control domain, such as a motor/tachometer speed controller of Figure 5, have been used to develop these rules.

For example, a conventional control block diagram was developed from the function block diagram of Figure 5. Figure 6 depicts the transformation process using the set of rules for manipulating the elements of the FBD presented in Table 4. Figure 7 shows the completed control loop. In this case, an FBD has been translated into another design domain: one of signal flows and operators. The FBD is the source of more detailed information

than the control block diagram (e.g., constraints and performance requirements). However, the translation back to a usable FBD from the control loop representation is possible with the inverse set of rules presented in Table 5.

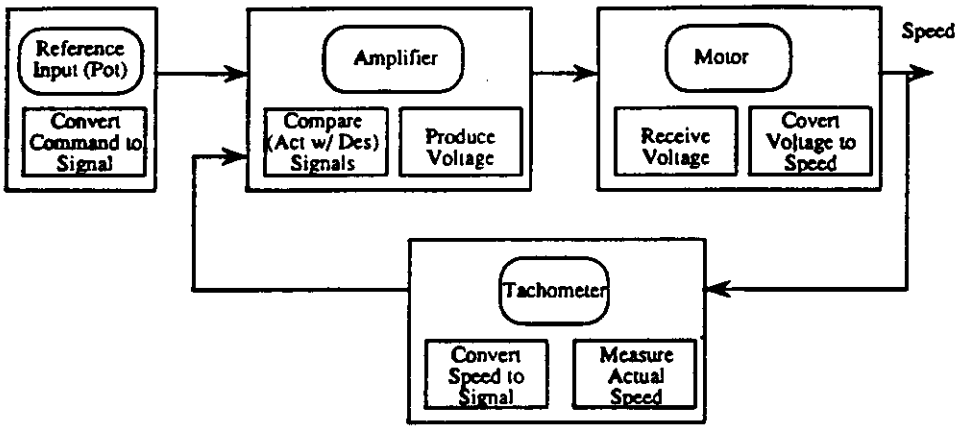


Fig. 6. The transformation of the FBD of Figure 5

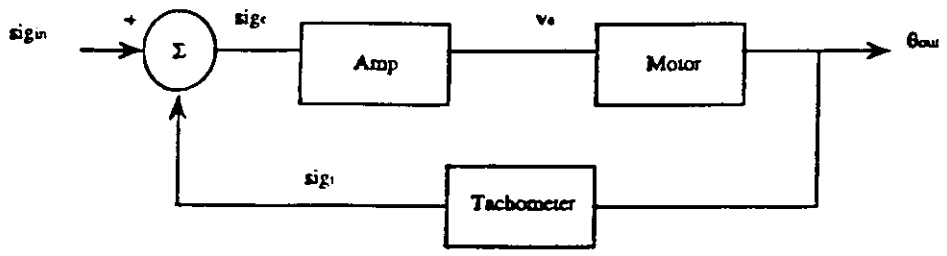
TABLE 4.
Conversion rules for FBDs to control block diagrams

- Check verbs for input/output properties. (These verbs will have one "input" and one "output" for information; arithmetic functions usually have two "input" ports.)
  - Link matching input and output nouns. If not, check logic or source of information. Avoid synonyms.
  - Indicate the direction of the flow (input to output and output to input) with an arrowhead on the information link.

- Once the information links have been completed, change to control domain:
  - Assign the basic function (leftmost occurrence of a measurable noun) to the output node of the loop.
  - Place the summing junction from the arithmetic function.
  - Attach input and feedback sources ("Receive" or "Determine" and a corresponding noun) to the arithmetic function.
  - Connect control elements to the loop output node by their information flow links. Connect feedback element functions from the desired output to the feedback source located at the summing junction.

TABLE 5.
Conversion rules for control block diagrams to FBDs

- Determine the function of each of the control block components by asking "why" they are necessary.
- Draw the control loop on a piece of paper with a Post-it™ to represent the artifact in each block, and placing the lowest-level function over it.
  - The function representing the input source, which might be out of scope, should be considered to exist before the summing junction (e.g., Receive Desired Signal).
  - The information links will adhere to general control loop rules, where each function block contains at least one input and one output node.
  - The output signal from the loop identifies the basic function, at least for the scope of the problem (e.g., Control Velocity). The noun of the output signal corresponds to the noun on the leftmost function.
- Begin the FBD by placing the artifacts to the extreme right and then placing the lowest-level functions to their left.
- The basic function would be placed to the extreme left (inside the scope line).
- Use how/why logic, to develop the intermediate functions, which connect the basic function with the lowest-level functions.
- Check verbs with one "input" and one "output" for information links (arithmetic functions which usually have two "inputs").
- Match the input and output nouns. Place an arrowhead on the information link following the direction of the flow (input to output and output to input).
- Optionally, place information-related functions on a vertical line, top down, in the FBD as follows: Input signal, Feedback signal, Summing junction, Control elements and Feedback elements.
- The corresponding components would be placed to the right of the function (inside scope lines) answering how the function is accomplished.
- Map direction and destination of the information links consistent with the control loop lowest-level functions (not the components).



Equivalent Control Block Diagram

Fig. 7. The completed control loop

2.7. OUTPUTS

The principal output is a valid FBD with its allocations satisfied. In some cases the allocations of the lowest level function block will identify a component with definite geometric and material properties. The details of the FBD may include revisions, alternatives, and process-dependent functions. The allocation lists will include behavioral and compliance information which would be used as inputs to optimization and synthesis processes.

3. Conclusions

The essential result of the function logic decomposition process mentioned above is a reasoning structure relating each component to the basic function of the design. The social context in which the FBD was developed depends on the engineers to somehow manage the quantitative data which eventually must be assigned to each component (Subramanian et al, 1989). Also, the how/why linking of functions is insufficient to represent other relationships. Finally, the representation of components as something other than functional and out of scope (i.e., performance and form) artificially divorces abstract functional descriptions from reusable artifacts.

These essential problem needs are addressed by a model of conceptual design for use in a computational environment. As introduced above, the function block is considered as a three-tier structure: (i) the function descriptors and the links which logically connect them, (ii) an allocation list associated with each block, and (iii) the components that jointly satisfy the requirements of the function and the allocation list. In this representation, the function descriptor retains the original noun/verb pair that describes in functional terms what is to be accomplished.

The links which provide the logical connection between the blocks specify the original existence reasoning, but are expanded in type to include information flow requirements, temporal relationships, causal connections, viable and non-viable alternatives and functional revisions. The allocation list attached to a function is a dynamic information structure containing the relevant design specifications, performance requirements, resources and component specifications.

The component level is the functional hook into detailed design analysis and synthesis methods through the supported function structure. Used in reverse, the component level captures functional information about an artifact in a context which includes design intent.

The design intent is captured by the FBD throughout the development of the product since a record is kept of the alternatives that are discarded and the revisions made. The logic behind each design can therefore be understood by both the expert and novice designer at any time. It remains to develop computational systems which can reason within this structure independently of the designer.

References

Andreason, M. M., Kahler, S. and Lund, T. (eds): 1988, *Design for Assembly*, New York, Second Edition.

Bailey, R. L. (ed.): 1978, *Disciplined Creativity for Engineers*, Ann Arbor Science Publishers, Ann Arbor, MI.

Bytheway, C. W.: 1965, Basic function determination techniques, *Proceedings of the Fifth National Meeting - Society of American Value Engineers*, **11**, April 21-23, 1965.

Bytheway, C. W.: 1971, The creative aspects of FAST diagramming, *Proceedings of the SAVE Conference*, 1971.

Finger, Susan and Dixon, John R.: 1989, A review of research in mechanical engineering design. Part I: Descriptive, prescriptive and computer-based models of design processes, *Research in Engineering Design*,1, 51-67.

Kantowitz, B. H. and Sorkin, R. D.: 1987, Allocation of functions, *in* G.I. Salvendy (ed.), *Handbook of Human Factors*, John Wiley & Sons, New York, pp. 355-369.

Miles, Lawrence D. (ed.): 1982, *Techniques of Value Analysis*, McGraw Hill Book Company, New York, Second Edition.

Pahl, G. and Beitz, W.: 1988, *Engineering Design: A Systematic Approach*. Springer-Verlag, New York, Edited by Ken Wallace.

Paz-Soldan, J. P. and Rinderle, J. R.: 1989, "The alternate use of abstraction and refinement in conceptual mechanical design," *ASME WAM*, San Francisco, CA, also EDRC 24-22-90, Carnegie Mellon University Engineering Design Research Center, September.

Perkins, D. N.: 1981, *The Mind's Best Work*, Harvard University Press, Cambridge, MA.

Pugh, S.: 1981, Concept selection - a method that works, *International Conference on Engineering Design*, ICED 1981, Rome, Italy.

Ruggles, Wayne F.: 1971, FAST - a management planning tool *SAVE Encyclopedia of Value*, **6**, 301.

Sturges, R. H., Dorman, J. G. and Brecker, J. N.: 1986, Design for producibility, Westinghouse Productivity and Quality Center.

Sturges, R. H., O'Shaughnessy, K. and Kilani, M. I.: 1990a, Representation of aircraft design data for supportability, operability, and producibility evaluations, EDRC Project Report Number: 14513, Carnegie Mellon University Engineering Design Research Center.

Sturges, R. H. and Kilani, M. I.: 1990b, A function logic and allocation design environment, *Proceedings for ESD Fourth Annual Expert Systems Conference and Exposition*, Detroit, MI, April 3-5, 1990.

Subramanian, E., Podnar, G. and Westerberg, A.: 1989, n-DIM: n-dimensional information modeling - a shared computational environment for design, Carnegie Mellon University Engineering Design Research Center, September 1989.

Ullman, D. G., Dietterich, T. G. and Stauffer, L. A.: 1988, A model of the mechanical design process based on empirical data," *AI EDAM* 2(1) 33-52.

Westinghouse Corporate Services Council: 1984, Report on life cycle costs, Westinghouse Productivity and Quality Center, Pittsburgh, PA.