A Systematic Approach to Conceptual Engineering Design

K. O'Shaughnessy, R.H. Sturges

EDRC 24-65-91

A Systematic Approach to Conceptual Engineering Design

Kathleen O'Shaughnessy
Graduate Research Assistant

Robert H. Sturges, Jr.
Assistant Professor

Pittsburgh, Pennsylvania
July, 1991

Abstract

The issue in conceptual design theory is to understand the processes which lead to innovation and to create tools which generate step changes in function in an orderly and repetitious manner. Detailed form-function synthesis methods which generally follow the conceptual phase of the design process require input specifications. An extension of function logic is proposed as a means to systematically promote innovation, manage conceptual design information, and produce such specifications. A three-tiered model for conceptual design is presented comprising function structure, allocation, and components. Applications of the model are given to include design intent, information flow, functional variants, and system behavior. Detailed examples are given with references to current methods.

A Systematic Approach to Conceptual Engineering Design

1.0 Introduction

Research in understanding Form-Function synthesis is enjoying an expansion as greater emphasis is placed on design theory by Industry and University [Westerberg, et al.,1989]. A consistently key element of this research lies in the representation of products and processes so that they may be efficiently analyzed and synthesized for optimum performance. 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]. It is the preliminary or conceptual stages of the Form-Function synthesis process which we are interested in representing and facilitating.

In this paper we will describe a representation of and a systematic approach to conceptual design. This approach aims at aiding synthesis and evaluation at the early stages of design evolution and enhancing communication between team members, either in the same or different design domains. As a conceptual process, it "begins"* with questions and "ends" with detailed specifications usable by current systematic synthesis techniques. (We should recognize that there actually is neither a beginning

nor an end *to* the conceptual process, except as limited by the actual number of **hours and** resources we choose to **apply to a** design effort.)

The greater contribution of the approach lies, however, in its ability to capture design intent through a chain of 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 design itself. This basis and its evolution are rarely captured in design representations. 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 are not obvious from its specifications and yet may be highly sensitive to them. Functional dependence is often realized only after a given technology or component is chosen. In short, form-function synthesis processes need not only specifications, they also need a way to systematically produce and manage them early in the evolution of a design.

We begin by defining the scope of conceptual design and investigating some of the recent work in the area of design representation and the needs of conceptual design in particular. We then discuss the function logic process as currently practiced [Miles, 1982; Bytheway, 1971]. We then propose the expansion of the function block diagram (FBD) representation as a three tiered structure, consisting of: (i) function blocks (what the design *does* rather than what it *is)* with links; (ii) allocations

(constraints, performance requirements, specifications and resources); and (iii) components (artifacts that satisfy the given function). We discuss applications of each tier of this structure showing how design intent is captured and managed during the conceptual design process. Detailed examples are also given which illustrate the use of the new concepts of allocation arithmetic and supported functions which derive from the representation.

2.0 Conceptual Design

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. Detailed design requires us to employ generally well-established tools to solve problems which have been well-specified in form or structure beforehand. 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],

> 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 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 conceptual design are currently typified by "Buhl's Seven Steps" which are shown in Table 1. Similar approaches are given by [Bailey, 1978; Miles, 1982; and Sturges, et al., 1986]. Tools for carrying out the essence of these steps include detailed checklists, data retrieval, and management systems, function logic, evaluation methods, and presentation techniques. The steps of any particular method also require a set of inputs and produce a set of outputs which we can list and compare.

3.0 Related Work

[Pahl and Beitz, 1988] have divided the design process into the following four distinct phases: (1) clarification of the task and development of the design specifications, (2) conceptual design, (3) embodiment design, and (4) detail design. Figure 2 shows each of these four phases as well as the steps that are required for each phase. Clarification involves the collection of requirements (the demands and wishes) and the constraints, which results in a specification list. We will see that function logic [Bytheway, 1965] is a method for accomplishing the first two phases. The *task* is equivalent to the *basic function.* The *specification* list corresponds directly to the *allocation* list in our extension of function logic.

Figure *3* shows the individual steps involved in conceptual design, as presented by Pahl and Beitz. The abstraction has already been completed in the development of the basic function. "Establish Function Structures" (from the overall function to the subfunctions) decomposes a function according to the process flow of material, energy and signals. "Search for Solution Principles" and "Combining Solution Principles" can be accomplished by a generic supported function. "Selection of Suitable Combinations" is completed by the designer, where possible alternatives are provided by a supported function. "Selection of Concept Variants" and "Evaluation of the Concept Variants" is achieved with the aid of a specific *supported function,* which we will treat in detail in section 6.1. Most of the Pahl and Beitz systematic approach concentrates on the embodiment phase of the design process, as does the concept selection process of [Pugh, 1981].

The concept variants of Pahl and Beitz corresponds to the functional variants presented by [Hundal, 1991]. Hundal advocates the use of a functional description and functional decomposition for the conceptual design phase. Similar to Pahl and Beitz, Hundal proposes that the decomposition be based on process flow, rather than on a functional development This approach is conceptually similar to the FBD process. The functions are listed and then decomposed into subfunctions. Solutions are sought only after the functional decomposition has been completed. The functional variants are generated by: (1) relocating the functions within the diagram or moving the system boundary, (2) subdividing the function, or (3) combining the functions. These resulting functional

variants can be manipulated to explore different alternatives without requiring a detailed analysis of the proposed solution.

Neither Pahl and Beitz nor Hundal present a definitive or comprehensive method for accomplishing the decomposition at a high level of abstraction. In contrast, a strictly functional hierarchy is used for an FBD, based on how a function is implemented rather than when. We will see the energy flows described by Pahl and Beitz and Hundal are contained in the allocation list. Similarly, the signal flow will be indicated by an information link; the material flow by a temporal link.

For example, Figure 4 shows a completed function structure of a specimen testing machine as presented by Pahl and Beitz. This diagram indicates that their function structure is based on a process flow of material, with signal and energy flows based on that process. In either a new design or the redesign of a product, the order of operation for the machine has been predefined, thereby eliminating some potential alternatives. In contrast, Figure 5 shows the FBD which directly corresponds to the function structure of this machine. The specimen, or material flow, in Figure 4 is first held and then it is loaded (or tested). This corresponds to the Load Specimen (onto machine), Hold Specimen and Perform Test on the FBD. This sequence of events must occur in a particular time order. The FBD shows that these events are functionally independent, but are connected by time, indicated by the temporal links. The signal flows (for both force and deformation) correspond to the information links on the FBD.

[Finger and Rinderle, 1989] have developed a method for transforming a "specification graph" into components and possible configurations. Figure 6 shows a specification graph, and Figure 7 shows the corresponding FBD for the transmission system. The FBD does not directly show the prime mover, this is considered to be out of scope. However, the input power is received from the Prime Mover, and is indicated by the information link. The Load is also considered to be out of scope. A general solution for the FBD is given by the behavior of the artifacts (e.g., bearings have resistance, and the gears have inertia), whose values have not determined at this stage. This general representation corresponds to the abstract primitives. These primitives include the compliance of the shaft (1/k), the inertia (Ji and J2) and number of teeth (ni and 112) of the gears, as well as the resistance of the bearings (b). These parameters are found in the allocation list corresponding to the lowest level function blocks. The abstract primitives are then compiled into a specification graph. The transformation process of the specification graph also relies on access to a "known component base" and related equations, which is part of a specific supported function by our extended convention.

The axiomatic approach presented by [Suh, 1988] describes design as a three step process: "(1) problem definition which results in the definition of functional requirements and constraints; (2) the creative process of conceptualizing and devising a solution; and (3) the analytical process of determining whether the proposed solution is a rational solution consistent with the problem definition." This description is similar to the one presented by Pahl and Beitz. Suh has chosen to include the embodiment and detailed design phases of Pahl and Beitz into the third phase. The

"**functional requirements**" of Suh equate to the basic function of the FBD; the "constraints" **and** "design parameters" are contained in the allocation list The function logic process is a means of formalizing the second step of the axiomatic process. The analytical step is aided by supported functions with the information contained in the allocation list

Suh has developed software design tools in which axioms, corrollaries and theorems were programmed in PROLOG. However, only the interaction between the user and the program has been described at this time. When the program is developed into a working model, it would be used during the analysis phase of design to facilitate the decision making process. This program could also be used in function logic as a general guideline for the development of domain specific supported functions.

4.0 Function Logic for Conceptual Design

4.1 Current Practice

A formal approach to conceptual functional design is provided by [Bytheway, 1965] in the form of *a function logic diagram.* Originally developed to stimulate design creativity and foster communication, this systematic approach to the conceptual design process relies on early identification of design goals in functional terms. The resulting description is called the *basic function* of the design. 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 mat will accomplish it directly. Selecting and integrating the components at this stage will lead to the accomplishment of the design goal. Such results are for the most part preliminary since practical designs rarely feature a one-to-one correspondence between functions and components.

The general form of the function block diagram (FBD) is shown in Figure 8. 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 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 simply connect each high level functions with its lower level functions according to this how/why relationship. As mentioned above, the functional decomposition sequence continues until we reach a set of functions that can be accomplished by available components or synthesis techniques. The vertical "scope lines" [Ruggles, 1971] limit the scope of the problem under study. Function blocks are shown as rectangles; components are represented in rounded corner rectangles and appear at the right hand scope line. These artifacts do not need to be described further, even though they may represent complex items, such as an engine.

4.2 Extension*

The essential result of the function logic decomposition process is a reasoning structure relating each component to **the** basic function of the design. Hie social context in which the FBD was developed depends on the engineers to somehow manage the quantitative data which eventually must assign to each component. 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 artificially divorces abstract functional descriptions from reusable artifacts.

We have developed several extensions to function logic which address these problems and increase its utility for conceptual design. The function block is considered as a three-tier structure (Figure 9): (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.

4.3 An Aid to Conceptual Design

The function logic method recognizes the social environment in which design takes place and encourages communication among groups of engineers with differing specialization. Functional definition of a problem specifically avoids domain-based design prejudices. All of Buhl's Seven Steps from definition through evaluation are embodied in the extended function logic approach, as the examples below will show. Presentation may also, of course, be formally designed through the same reasoning structure.

On a more detailed level, function logic aids conceptual design specifically with respect to information flow. In any design, there is information about the designed product and about the design process itself. The information about the product includes the constraints, performance requirements, resources and component specifications. This is contained in the allocation list. The information about the process is exchanged between functions. This is shown on the FBD by the information links. During the development of an FBD, this information flow requirement is suggested by the nouns and verbs of the function descriptor and its allocations. The representation for such information flow is independent of the type of information being exchanged.

The design intent is captured by the FBD throughout the development of the product; 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. Moreover, the FBD can also be used as a means of communication between designers in different domains. We give an example of domain conversion in section 5.1.

5.0 Application

The application of extended function logic will be presented with respect to each layer of the generalized function block in Figure 10: structure, allocations and components.

5.1 Layer #7: Function Structures

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

Boeing uses these as second order actions

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 10, 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 include the information link, the temporal link, the causal link, the alternative link with chaining, and revision link. A description of each link is given in Table 2. The expanded link set represents the design and its reasoning in greater functional detail than die basic how/why logic without adding domain-specific information.

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.

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 [Stnrges, 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 die 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 11, have been used to develop these rules.

For example, a conventional control block diagram was developed from the function block diagram of Figure 11. Figure 12 depicts the transformation process using the set of rules for manipulating the elements

of the FBD presented in Table 3. Figure 13 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 4.

5.2 Layer #2: Function Allocation

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, an inverted pendulum on a moving cart, 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 in the allocation list.

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 5. 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.

AllocationArithmetic

The process of allocating values among functions is generally carried out by hand, applying sets of rules or guidelines built from experience with a particular design domain. A function logic model of conceptual design which manipulates allocations of product or process performance and the role of its operators needs to consider at least two computational problems: (1) determining and evaluating allocation choices or tradeoffs, and (2) tracking and distributing these allocations among the function blocks. The former problem is far from well understood, and the reader is referred to [Kantowitz and Sorkin, 1987 and Meister, 1985] for excellent technical approaches and reviews of current practice. The latter problem is also not well defined, and the following four cases are examined in more detail by [Sturges 1990]: (1) diverging function allocations, (2) artifact allocations, (3) converging function allocations, and (4) alternative function allocations. Figure 14 shows example structures of allocation arithmetic.

Design Domain

At the conceptual design phase, not all necessary information is provided initially, or more information is required at a later stage to determine which of several alternative solutions will provide the best result. To assist this information gathering process, it is necessary to detect what information or values have been omitted. The breakdown of the allocation list according to design domain is useful to this end. It will become easier to recognize if any necessary information is missing from the original specifications at the basic function level. This will ultimately aid in the search for components to satisfy these requirements and constraints.

Disinheriting (Filtering) of Allocation Information

While all constraint and performance requirements must be satisfied, not all of them are required at each functional level. Arranging all the information from the basic function through to each lower level is time consuming and impractical. The structure of the allocation list facilitates the filtering of irrelevant design information. For example, a Support Load function would consider "color" to be irrelevant; therefore, the "aesthetic" allocations would not be considered at the lower level functions of Support Load.

DistributionofAllocationInformation

All of the information contained in the allocation list at the basic function level must be distributed to each of the relevant, connected, lower level functions. The relevant constraint information and design requirements must be propagated to each level so that a component can be selected which will satisfy these requirements. The distribution process also propagates the component specifications and associated side effects back up through the higher level functions once they have been selected. When this information is introduced in the allocation list at the basic function level, it is then be propagated back down through the other intermediate and lower level functions if relevant. This process is repeated until all the components have been selected.

For example, in the design of a vehicle, the heat generated by the motor is a side effect in the engine section, but may be useful in the design of the heating system. Figure 15a depicts the distribution of allocations by design domain and by filtering. To satisfy the Modify Voltage function, an amplifier is chosen, which adds heat to the allocation list. Figure 15b shows the propagation of this side effect back to the basic function, and Figure 15c shows the redistribution of this new allocation list to the other branches.

53 Layer #3: Components

As mentioned in section 3 above, embodiment, or detailed design procedures which are independent of domain are still largely unknown in mechanical design. Domain specific component synthesis methods, by contrast, are plentiful and growing to meet specific product needs [e.g., Birmingham, et al., 1989]. The present and evolving form-function synthesis methods noted above do specify a form of "input specification" which is precisely satisfied by the conceptual function logic method of section 5.1. The management of this same data has been structured by the allocations of section 5.2. Thus, while the conceptual design technique presented here may not be expected to produce component designs, it systematically provides the information for detailed design methods which do.

6.0 Examples

6.1 Supported Functions

Combining Function Structure and Function Data data defines a class of components by intended use. We term this combined set a *supported Junction,* and use it to select a specific component satisfying a functional need and the criteria found in the allocation list.

The process of value analysis and value engineering recognizes and indeed capitalizes on the fact that much design effort is spent applying

previous **known** solutions to current problems and constraints. A designer will try **to avoid** designing a bolt or a spring unless the constraints are sufficiently **narrow to** preclude the use of a standard item. Manufacturer's catalogs are designed to facilitate rapid evaluation and selection of their offerings (while carefully differentiating themselves from the competition). Handbook design data similarly contains generic models for often occurring primitive functional artifacts. These types of resources are useful for developing supported functions.

A supported function can be either general or specific. A general supported function is one that can be used as a template for more specific supported functions. This is useful for early evaluation and later detailed design. A general supported function is also a generic model for well known components, processes or subfunctions. Access to as wide an assortment of functional alternatives as possible helps make the systematic conceptual design process work. Models of a design which are built functionally express requirements and constraints in the same language as such supported functions.

A specific supported function is comprised of databases of known components and processes with their associated design equations (e.g., obtained from design handbooks or manufacturers' catalogues). Handbook data and artifact specifications are structured in functional form to facilitate access to functional alternatives. Such a database contains low level functions that help satisfy higher level supported functions. These low level functions are arranged by the designer in a hierarchy until the component level is reached.

Figure 16 shows an example of how a supported function operates. The function block that specifies the supported function Store Energy can be satisfied in many ways. The best alternative based on a specific set of allocations is readily chosen from the ones presented. An example of domain specific supported functions can be found in [Sturges and Kilani, 1990b]. A typical supported function template is shown in Figure 17. The inputs are the function name and the associated allocation list The outputs include values to some of the allocations (e.g., a compliance with some specification), component alternatives and a possible physical configuration.

6.2 Position Control System

The systematic conceptual design of a position control system will be used to illustrate the function logic process, and is shown in Figure 16. The example position control system is part of a computer output device. The basic function is identified as Output Character. The associated allocation list is also shown beneath the Output Character function block. The Get Character function is displayed to show the temporal and information dependencies of the function blocks, even though the higher and lower order functions of this block are considered to be out of scope. A step-by-step analysis of the FBD, its associated allocation list and a relevant supported function follows:

Layer #i: Function Structure: Blocks and Links

The decomposition of the FBD with its links corresponds to the development of the overall function into subfunctions, as defined by in Pahl and Beitz. A functional decomposition has the advantage of not eliminating any potential alternatives by predetermining the form of the solution until much later in the process. The development in this example is primarily based on *how* a function is accomplished, rather than a process flow. The time dependence of a process flow is indicated on an FBD by the temporal links. At first glance, this example may appear to follow a "flow chart" process, but that may be due to the fisheye view that has been adopted for displaying the diagrams.

The Output Character branch is started only after the Get Character sequence has been completed, which is indicated by the temporal link. Since the information about the character to be output is received from Get Character, there is also an information link connecting these blocks. In order to output a character, the designer must first choose where the character will be placed. The alternatives include saving the character in a file (Save Character), printing the character on paper (Print Character) or displaying the character on a screen (Display Character). These alternatives are represented on the FBD by an "or" link.

In order to print a character, the character must be positioned correctly at the printer and then placed onto the paper (Position Character and Place Character, respectively). Since both of these

functions are required for Print Character, they are connected by an "and" link. Position Character is accomplished by determining the position of the desired character (Compute Desired Position) and controlling the position (Control Position) of the output device. Therefore, they are connected to Position Character by an "and" link. In order to determine desired position, the change in the position and the change in direction from the current position must be determined. These functions are connected to Compute Desired Position by an "and" link. The Control Position function calls a supported function called "Control Position," which automatically adds the sections of the FBD shown in Figures 18c and 18d.

To control the position, the desired position signal must be received (Receive Desired Signal), the actual position must be converted to an actual signal (Convert Actual Position to Actual Signal), the actual and desired signals are then compared (Compare Actual and Desired Signals) and the position should be changed accordingly (Change Position). These functions are connected to Control Position by an "and" link. To change the position, a gain must be obtained (Create Gain) and the desired position must be produced (Generate Position), which are also connected by an "and" link.

To create the gain, the error signal must be increased in magnitude. These two blocks are connected by a "probable alternative" link. If the designer chooses to use a proportional (P) controller, there is only one block connected to Create Gain, and one of these should be eliminated. However, the designer could choose to use either a proportional/integral

(PI) or a proportional/integral/differential (PID) controller. Then, the probable alternative link will be replaced by an "and" link since there are several functions that are required (e.g., amplify error, integrate error, differentiate error, and add error components for a PID controller). Generate Position is accomplished by converting the amplified error signal to the actual position and by then measuring the actual position. Convert Error to Actual Position and Measure Actual Position are connected to Generate Position by an "and" link.

The blocks of the control loop within the "Control Position" supported function are connected by information links, as shown in Figure 18c. The Receive Desired Signal function receives its information from Compute Desired Position (both distance and direction). The Convert Actual Position to Actual Signal function receives its information from a measuring device. The Compare Desired and Actual Signals function compares the desired and actual signals, and then produces an error signal reflecting the difference. The Generate Actual Position function changes the amplified error signal received from Create Gain into the actual position. The error signal is amplified by the controller, which is then fed into the Convert Error to Actual Position function.

Figure 18d shows the internal functioning of the control element block, Convert Error to Actual Position. The decomposition of this function block occurs when the designer selects the domain that will be used. This decomposition corresponds to the functional representation of the state space equations, which are given in Table 6. Another information link entering some of these function blocks represents a disturbance input.

A disturbance input is usually not modeled very readily, but its effect can be easily measured. Moreover, a disturbance may be due to the selected components or due to some environmental (internal and/or external) condition. For example, the "generate" functions may have an error due to a conversion (e.g., e to i'). Some energy may be lost as heat in friction or mechanical losses in a motor.

Layer #2: Function Allocation

The allocation list contains the information that facilitates the search for, and the selection of, the best solution. The structure of the allocation list allows only the relevant specifications to be connected to a function. Therefore, it is easier to verify that all constraints have been met, or to eliminate alternatives that do not satisfy the requirements as well as another alternative. A structure of the allocation list is described in Section 5.2. This structure can be sorted in four ways: (1) based on arithmetic needs, (2) based on design domain, (3) to disinherit values or (4) for distribution. The additions to the allocation list during the development of Figure 18 is given below.

Arithmetic Needs

The output of Determine Distance and Determine Direction are the change in position and direction, respectively, required to achieve the desired position. The Compare Desired and Actual Signals block compares the desired and actual, and then produces an error signal of the difference.

Design Domain

The Save Character function prompts the designer about the format of the file. The Display Character function requires information about the format of the screen (e.g., if there is to be a new window). The Position Character function needs to have the maximum allowable error (the difference between the actual and desired positions). The Place Character function requires information about the format of the desired output.

The Control Position function block prompts the designer for information about the natural frequency and damping, and automatically calls a "Control Position" supported function. The Create Gain function prompts for information about the type of controller to be used (P, PID, etc.). The Convert Error to Actual Position function block initially prompts the designer to choose a design domain.

Disinherit values

The following lists show what parts of the allocation list were filtered from the function blocks. The structure of the allocation that can determine which values can be disinherited has not yet been determined. Therefore, this section must be completed by the designer without the aid of a computer program.

Some of the ergonomic factors that are considered in a design include the height for operation, the shape for accessibility, the clearness of layout, lighting and modes of operation. These human factors are filtered out because the following functions do not interact directly with the user Save Character, Position Character, Place Character, Received Desired Signal, Convert Actual Position to Actual Signal, Compare Actual and Desired Signals, Amplify Error, and Measure Actual Position.

The following functions will not have an external load attached for operation (such as a motor driving a load, or a voltage source suppling a load). This load information has been eliminated from: Save Character, Display Character, Compute Desired Position, Create Gain, Receive Desired Signal, Convert Actual Position to Actual Signal, Compare Desired and Actual Signals, and Measure Actual Position.

The following functions will not output the results directly to the user: Compute Desired Position, Receive Desired Signal, Convert Actual Position to Actual Signal, Compare Desired and Actual Signals, Measure Actual Position, and Convert Error to Actual Position children (except Generate and Integrate 8').

The following functions do not require the information about the maximum allowable error: Compute Desired Position, Generate Actual Position, Receive Desired Signal, Convert Actual Position to Actual Signal, Compare Desired and Actual Signals, and Measure Actuai Position.

No control information is required by the following functions: Receive Desired Signal, Convert Actual Position to Actual Signal, Compare Desired and Actual Signals, and Measure Actual Position.

Distribution

The information about each element in the state space equation that is generated at each level is subsequently added to its associated allocation list, also shown in Figure 18d. The Receive Desired Signal function receives its information from Compute Desired Position (both distance and direction), which is reflected in the associated allocation list. The Convert Actual Position to Actual Signal function receives its information from a measuring device.

The Receive Desired Signal function receives its information from Compute Desired Position (both distance and direction). The error signal is amplified by the controller, which is then fed into the Convert Error to Actual Position function. Once the components have been selected, the associated specifications can be distributed. This information includes the gain of the encoders, the motor characteristics (e.g., K, Ra, La, stall torque, no load), and the side effects (e.g., heat).

Figure 19 shows the form of the allocation list at the basic function level for two cases. The first case shows the allocation list at the beginning of the design process. The second case shows the allocation list after the

general form of the solution has been determined The characteristics and side effects are reinherited from the lowest level functions.

Layer#5: Components

Once the FBD has reached the lowest level and its allocation list has been formulated, the general form of the conceptual design can be obtained. This can occur without conducting an in depth analysis of the proposed solution, and is, therefore, the last stage of conceptual design. The general form must satisfy the function as well as the constraints and requirements in the allocation list, and can be obtained through the utilization of a supported function. This general form can be represented through equations relating the elements of the equations.

The example in Figure 18c corresponds to the printwheel control system of [Kuo, 1982]. Each of the lowest level functions is mapped to a general representation for the corresponding artifact. Receive Desired Signal, Convert Actual Position to Actual Signal, and Measure Actual Signal can be implemented by position encoders (for the desired and actual positions). Compare Actual and Desired Signals and Amplify Error correspond to a power amplifier and microprocessor controller. Convert Error to Actual Position corresponds to the dc motor and printwheel.

Each of the these artifacts can be represented by the behavior that they exhibit. However, the exact values for these characteristics do not have to be determined at this stage. For this example, the position encoders have gains (K). The dc motor can be represented by the inductance and

resistance of the armature (La and Ra), the motor torque constant (K0, inertia (J), the viscous friction coefficient (B), etc. This general representation of the behavior corresponds to the abstract primitives in the work of Finger and Rinderle. The abstract primitives can then be developed into a specification graph, which is shown in [Finger and Rinderle, 1989]. The specification graph will determine which combination of physical components can best satisfy the design criteria.

The general representation of the behavior of the components is also consistent with the development of the concept variants as given by Pahl and Beitz or the functional variants of Hundal. Moreover, once the general form has been determined, there are many methods available to determine what artifact will satisfy these parameters. The input to some of the these functions are the lowest level function blocks (e.g., the embodiment phase of Pahl and Beitz). The input to others may be the general form of the solution (e.g., the abstract primitives of Finger and Rinderle).

Layer #2 Revisited: Recursive Allocation Synthesis

Once a general solution to one area of a design has been determined, it may have an effect on some other area of the design. It is therefore desirable to propagate the component specifications back up to the basic function. At the basic function level, these new specifications would then be redistributed to the lower level functions in a recursive process. For example, one component that has been selected for the Convert Error to Actual Position function is a printwheel. If this selection had been a laser, then the direction would not need to be determined for Compute Desired

Position, and could be eliminated. In addition, Compute Desired Position and Determine Position would then be connected by a probable alternative link, and could be combined.

Most often, a designer will mentally check the characteristics of a particular component against the design specifications on a continuous basis. It is unclear at this stage in this research if the computer should interact on the same basis. Further investigation is required to determine if the reinserting of the allocations should occur when the new specifications are added, or if it should occur only when requested by the designer. It would probably be advantageous to allow the designer to choose whether to have the redistribution process continuous or not.

7.0 Conclusions

Function logic as extended here is a method for systematically performing the conceptual design process. As such, it facilitates the designer, both expert and novice, to develop many possible solutions. The form of the solution is not chosen until the FBD has been completely developed. The FBD is also a means for communication between designers in different domains. It was shown that a conversion process can be developed between function logic and the control domain. An FBD incorporates information about both the design process and the designed product. The reasoning structure is retained and a layout of the process is inherent in the links and blocks of the FBD.

Functional definition of a problem specifically avoids domain-based design prejudices. All of Buhl's Seven Steps from definition through evaluation are embodied in the extended function logic approach. According to Pahl and Beitz as well as Hundal, conceptual design begins with a functional decomposition of the overall function into subfunctions. Function logic provides a reliable method for decomposing the functions, and also for connecting the subfunctions to the higher level functions and/or to other subfunctions (e.g., information or temporal). Combined with allocation data, this general representation of the behavior corresponds to the abstract primitives in the work of Finger and Rinderle. The abstract primitives can then be developed into a specification graph.

When the overall function has been decomposed sufficiently, the designer will then begin to search for general solutions to the lowest level functions that satisfy all of the design specifications. The allocation list associated with the lowest level function contains all the specifications and parameters that will affect that particular function. The automation of the search process for potential solutions is attractive because unnecessary data has been eliminated by a filtering process, which ensures that relevant specifications will be satisfied efficiently.

The issue in conceptual design theory is to understand the processes which lead to innovation and to create tools which generate step changes in function in an orderly and repetitious basis. The extension of function logic offers a systematic way to encourage design solutions without domain-prejudice and to generate and maintain a "thought trail" (how&why), rather than a "data trail" (what), of the progress of a design.

8.0 Future Work

Future directions for the analysis of function logic are numerous. Current research has focused on product design. Further investigation is required to further function logic to efficiently handle process design. Further investigation into the temporal link is suggested to apply it to material flow in a process design. It should also be determined if the arrow on a temporal link indicates the existence of information (signal) flow or a causal connection. In addition, the allocation list must reflect this temporal link between functions.

The role of function logic will also be extended into other domains. The conversion to and from the control domain has been developed. Future research will include the areas of civil engineering, process management (both process and product) and software design (both process and product).

Many of the steps in manipulating and creating an FBD are attractive for automation, but are presently manual. A taxonomy of the allocation classes should be developed which will allow for automatic filtering. The current method is completed by the design, correlating the noun/verb pair with experience as to what is relevant and what can be discarded. Figures 18c and 18d displayed the decomposition of one function block into many other function blocks once the domain has been chosen. Investigation is needed to determine the rules governing this decomposition and substitution.

The structure of the supported function is virtually the same as any FBD. Thus, a completed conceptual design can be treated as yet another supported function, but with a well-developed design history and intent. Exploiting these new design structures in a "design with features" environment is an exciting prospect for the future.

(7916 Words)

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, interrelations, 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.

Table 2. Unks 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 suggests 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 by 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.

Table 3. 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 which 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 4. 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).

Table 5. Structure off 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

Table 6. State Space Equations for Position Controller Example Source: [Kuo, 1982].

$$\begin{bmatrix} i_a' \\ \omega_m' \\ \theta_m' \end{bmatrix} = \begin{bmatrix} -Ra/La & -Kb/La & -KKs/La \\ Kj/J & -B/J & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} i_a \\ \omega_m \\ \theta_m \end{bmatrix} + \begin{bmatrix} K/La \\ 0 \\ 0 \end{bmatrix} \theta_r$$

Where:

K_s	Gain of the encoder
K	Gain of power amp
Ra	Resistance of motor armature
La	Inductance of motor armature
K	Torque constant of the motor
Kb	Back emf constant of the motor
J	Inertia of the printwheel and the motor
B	Viscous friction of the motor and load shafts

The error voltage is the input to the motor, and the output is a shaft rotation, 6. The load that is attached to the motor is the printwheeL The printwheel turns at the same speed and in the same direction as the output shaft of the motor. The above equations are correlated to the function blocks of Figure 16d as follows:

1. Generate i' Components corresponds to the above equation for i_a\ which includes components of i_a, (0m» and 6_m. To obtain i_a, these components must be added and then integrated (Add and Integrate ia)*

2. Generate co_m' Components corresponds to the above equation for co_m\ which includes components of i_a and (Dm* To obtain cdm, these two components must be added and then integrated, which corresponds to Add and Integrate ω_m'.

3. Generate 6m* Components corresponds to the above equation for 6_m', which equals com. To obtain 6_m, integrate com (Generate and Integrate 0m')*

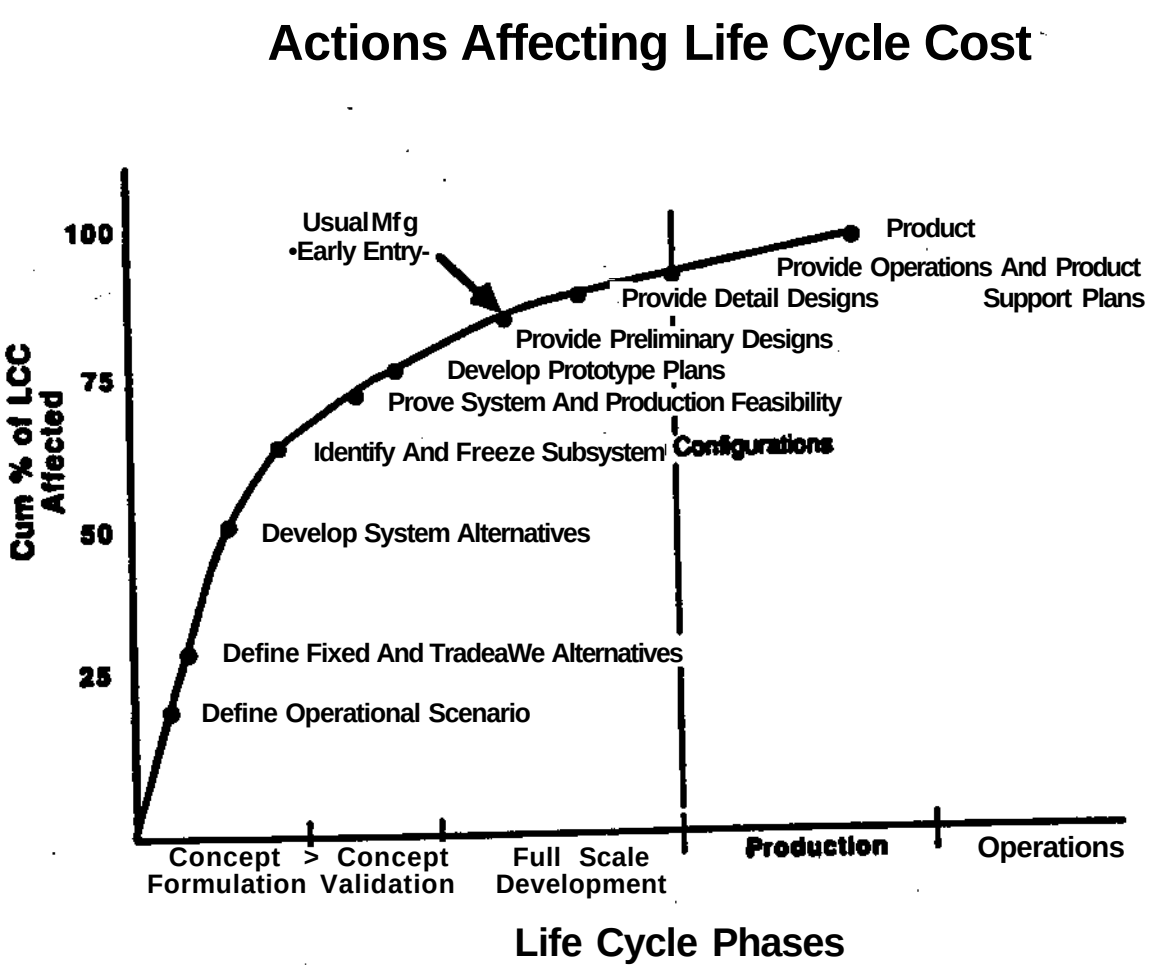


Figure l. Conceptual Design in the Ufe Cycle Cost of aProduct
Source: [Westinghouse, 1984]

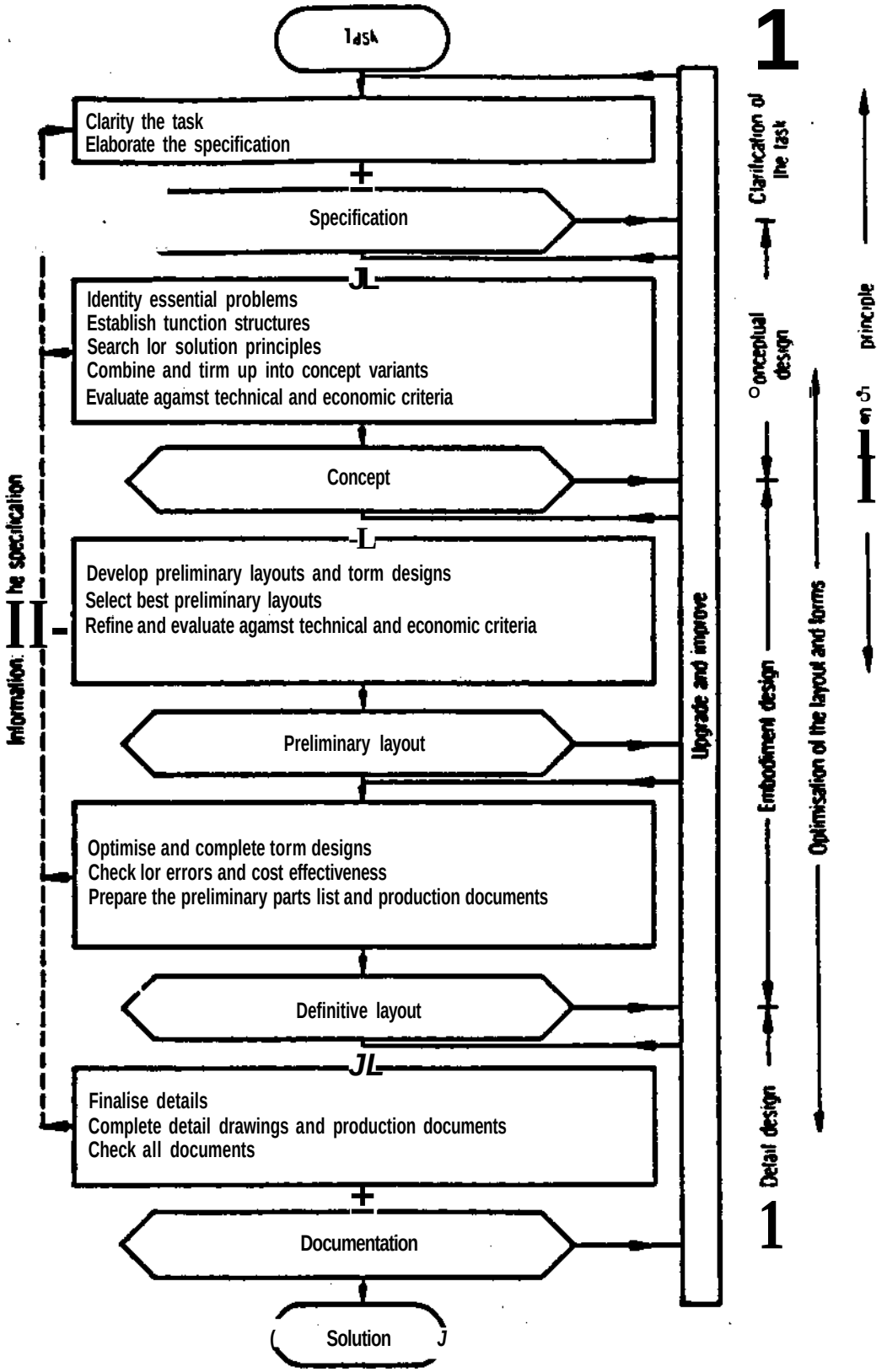


Figure 2. Steps of the design process.
(After Pahl and Beitz,1988)

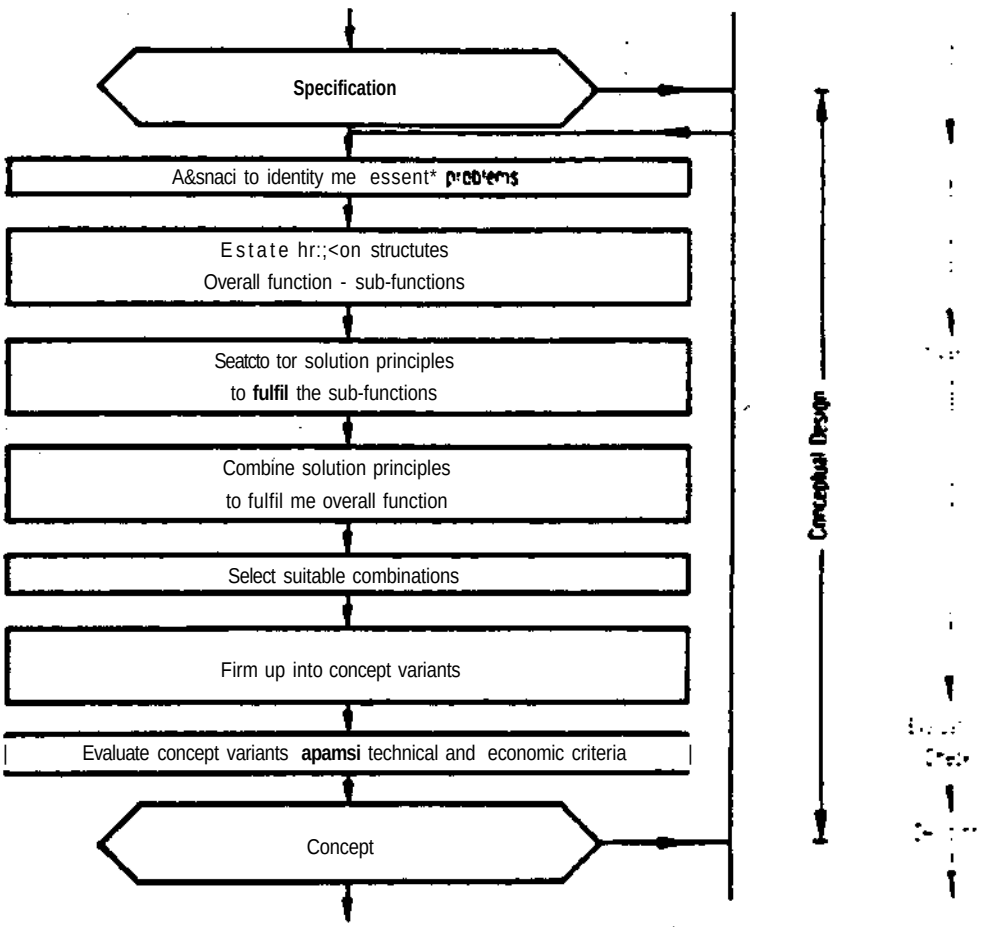


Figure 3. Steps of conceptual design.
(After Pahl and Beitz, 1988)

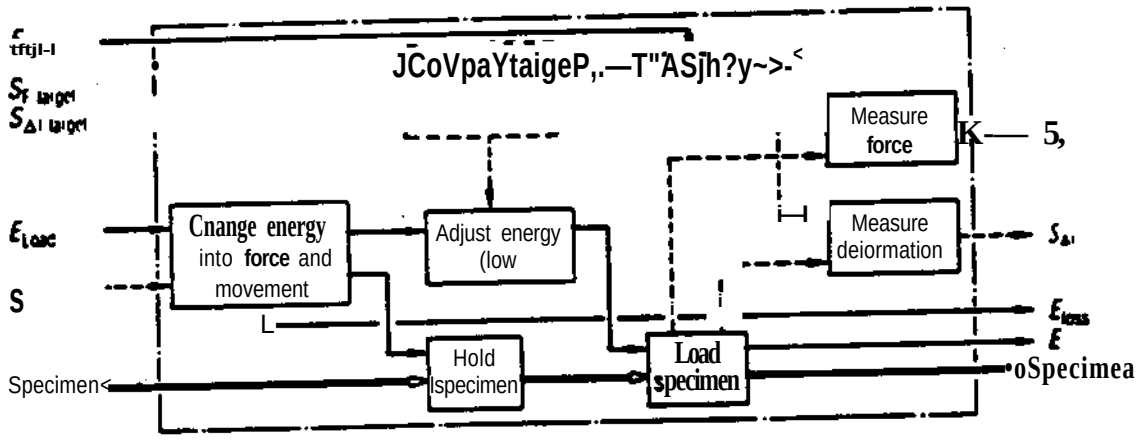
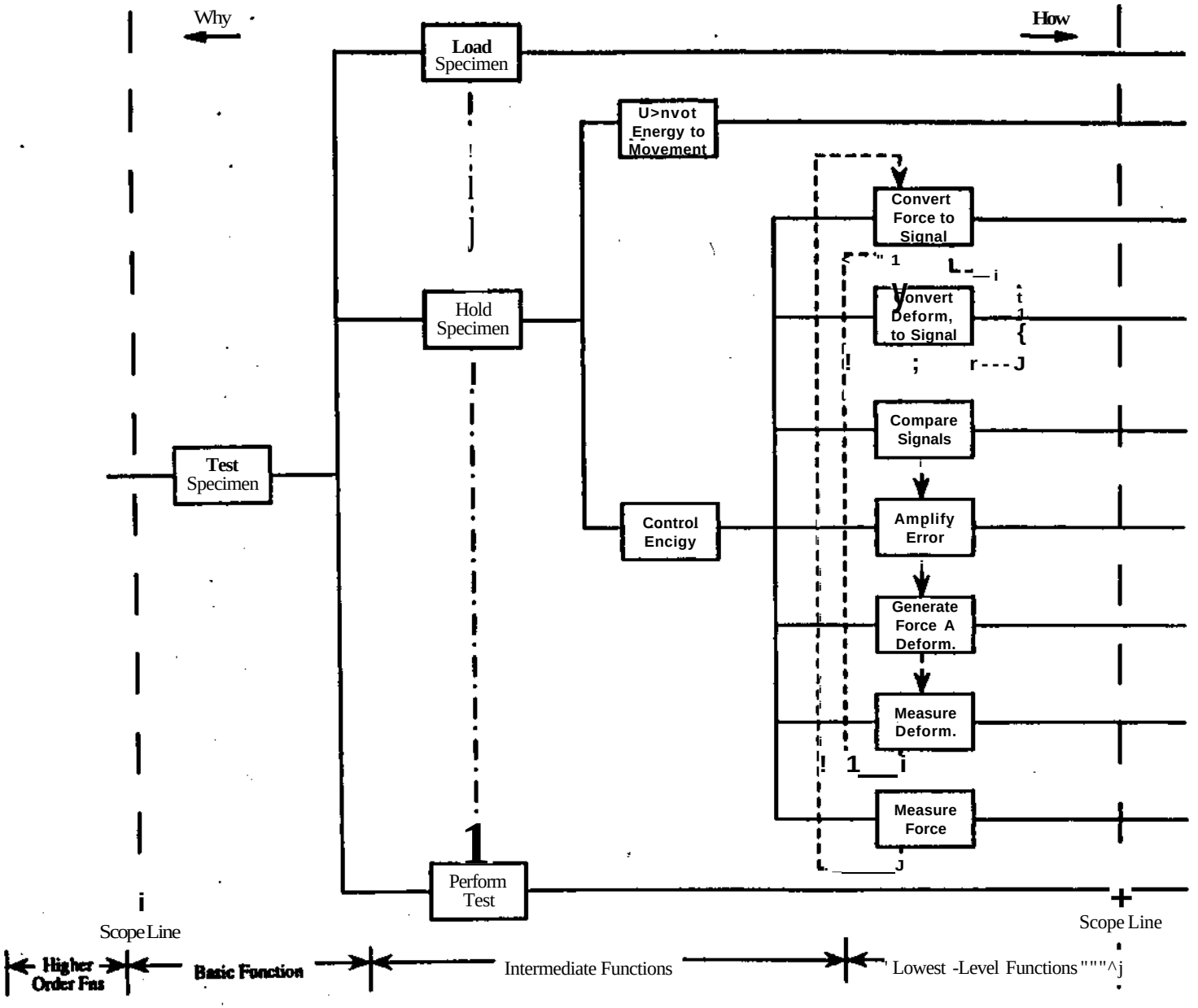


Figure 4. Completed function structure of the overall function of a specimen testing machine. (After Pahl and Beitz, 1988)



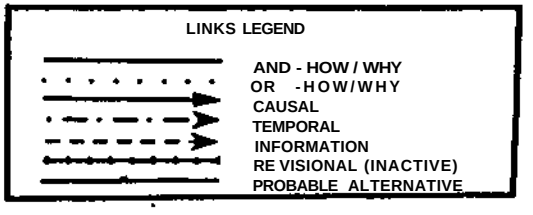


Figure 5. FBD for the specimen testing machine of Figure 4.

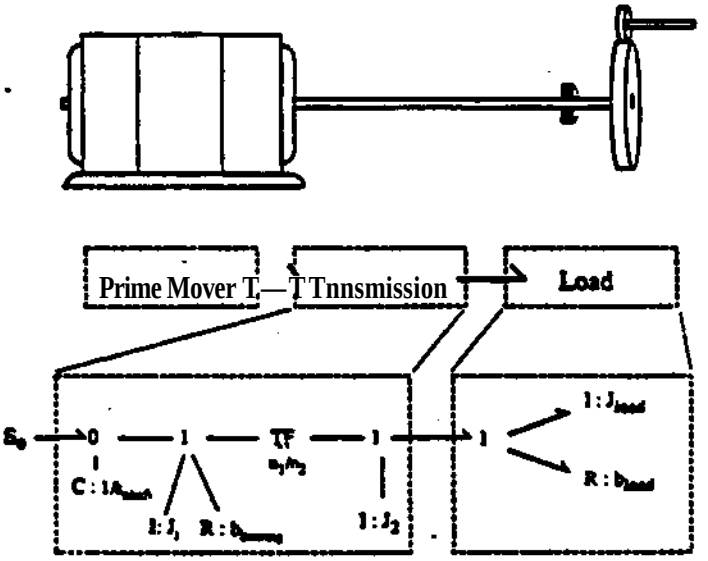


Figure 6. Specification graph for a transmission system. (After Finger and Rinderle, 1989)

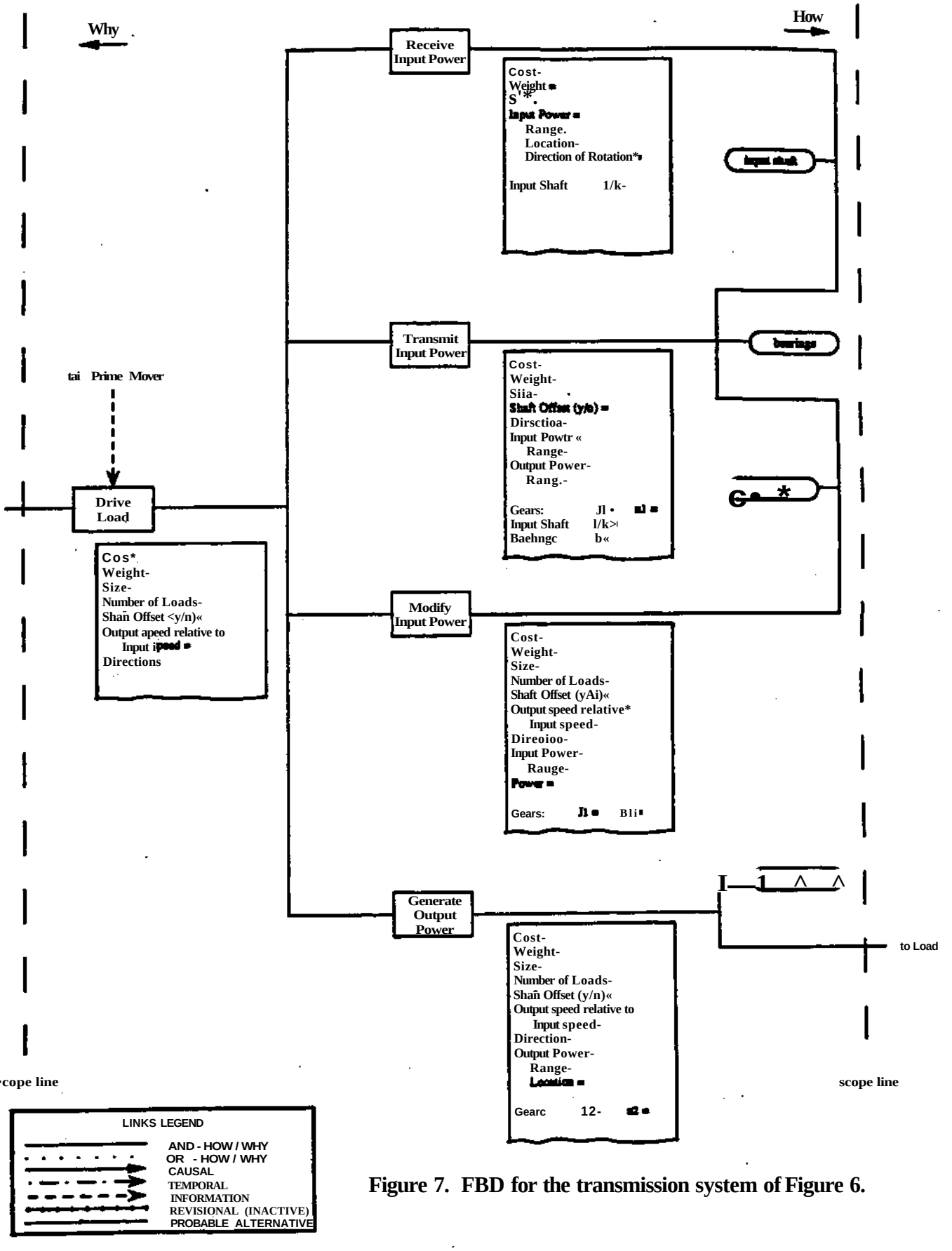


Figure 7. FBD for the transmission system of Figure 6.

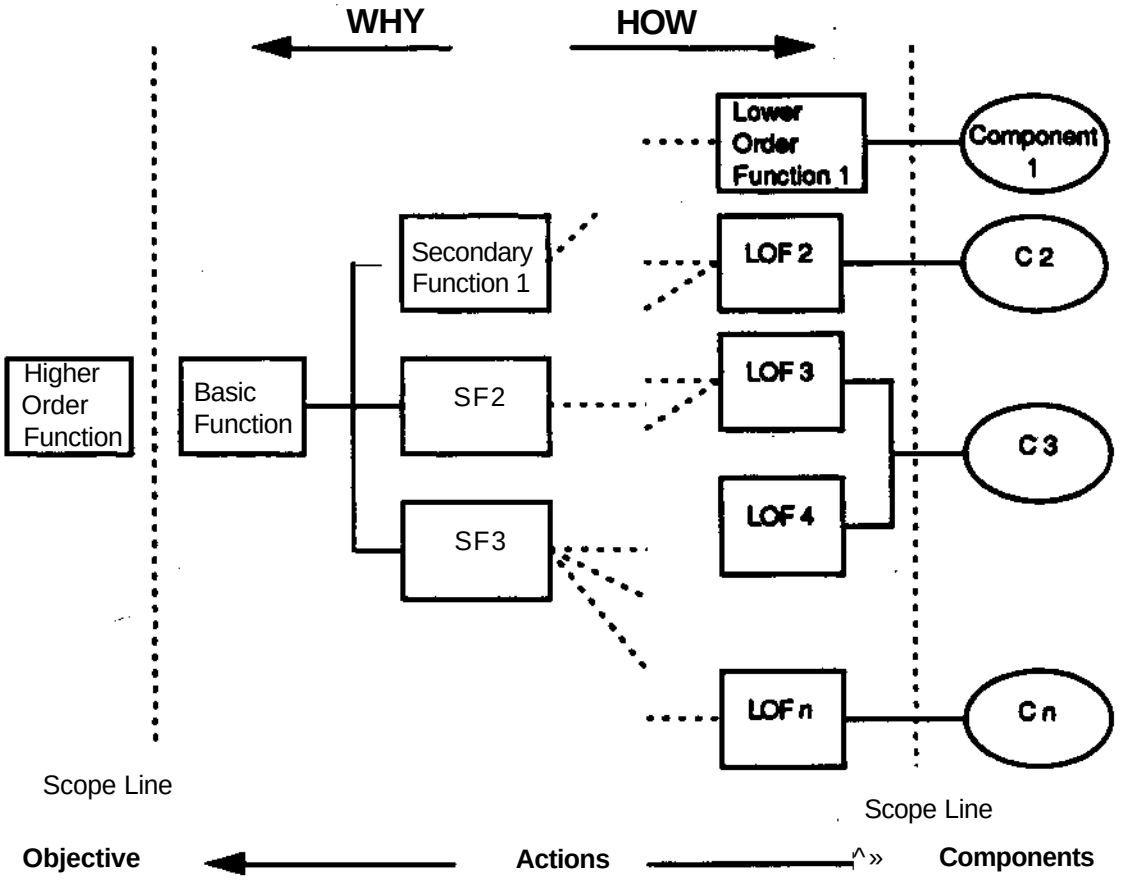


Figure 8. The general form of a function logic diagram.

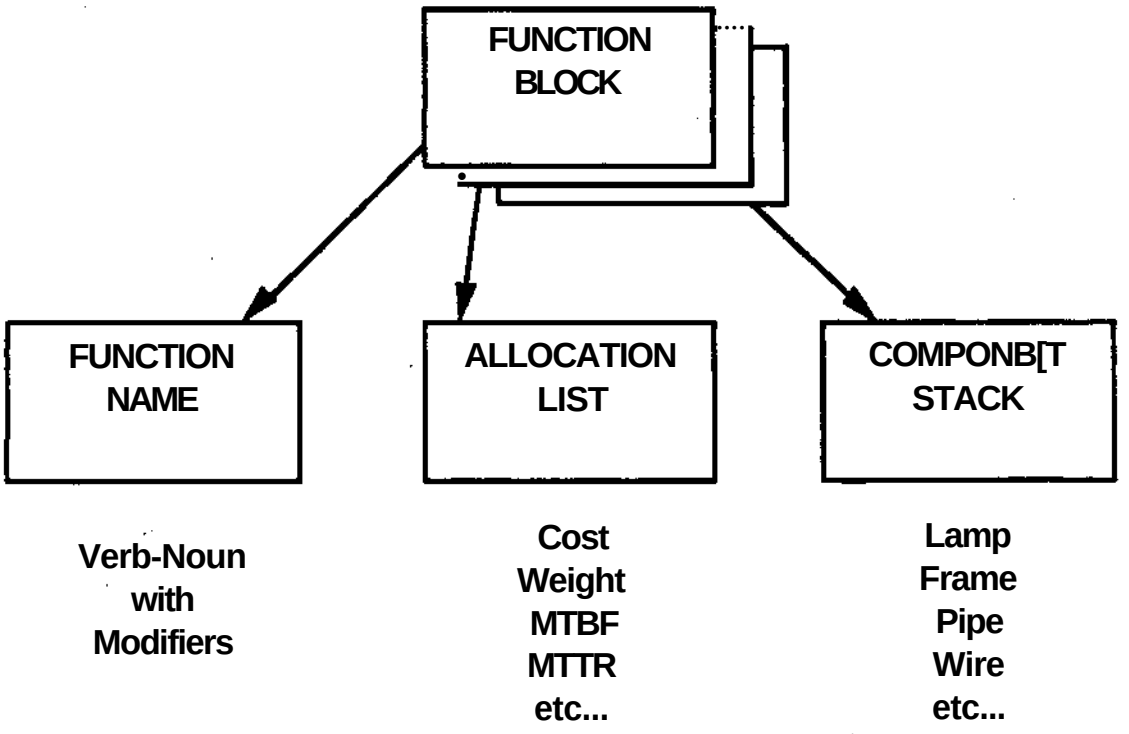


Figure 9. Three tier structure for a function logic diagram.

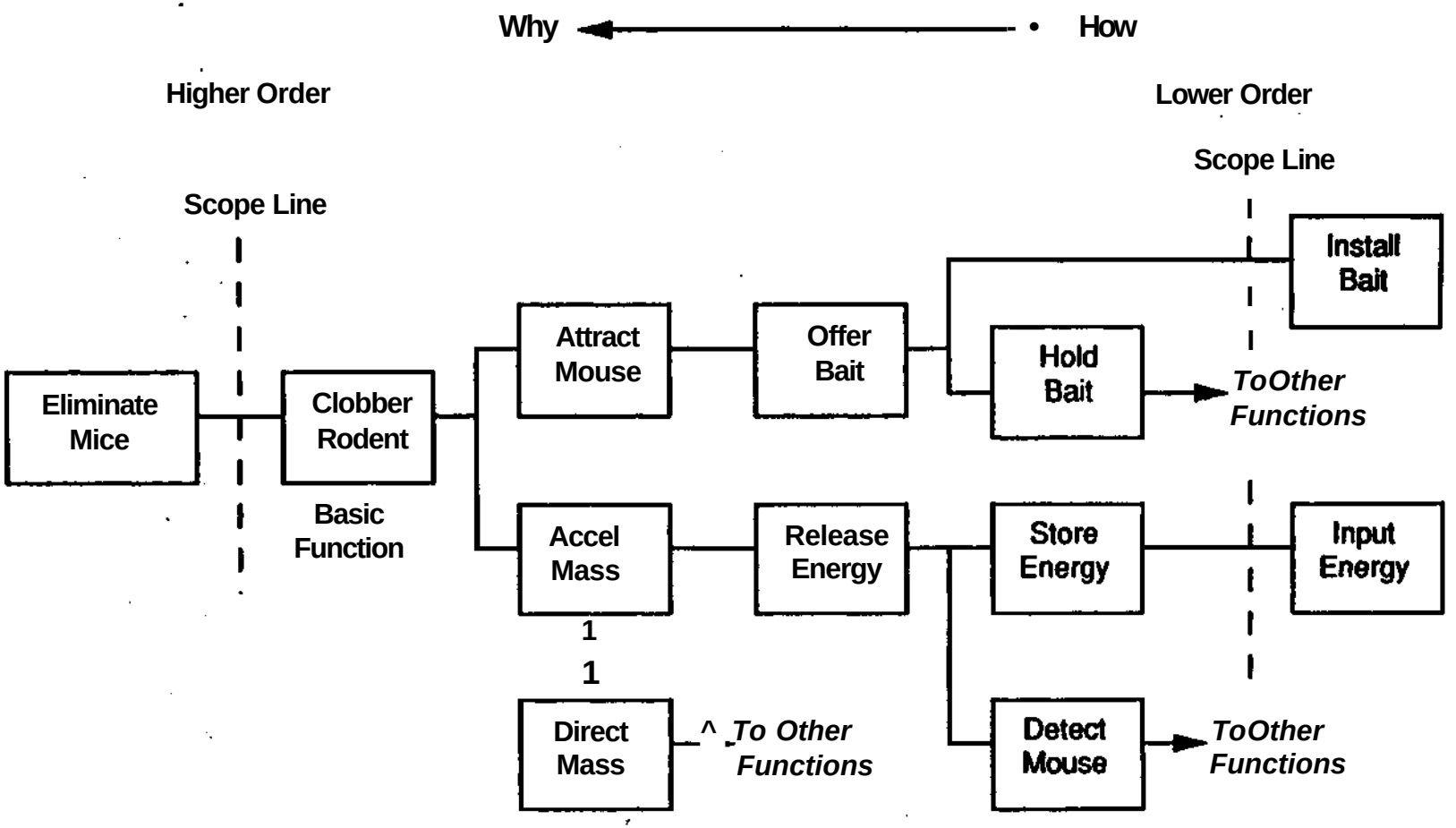


Figure 10. Preliminary function block diagram of a mousetrap. (After Maurer, 1986)

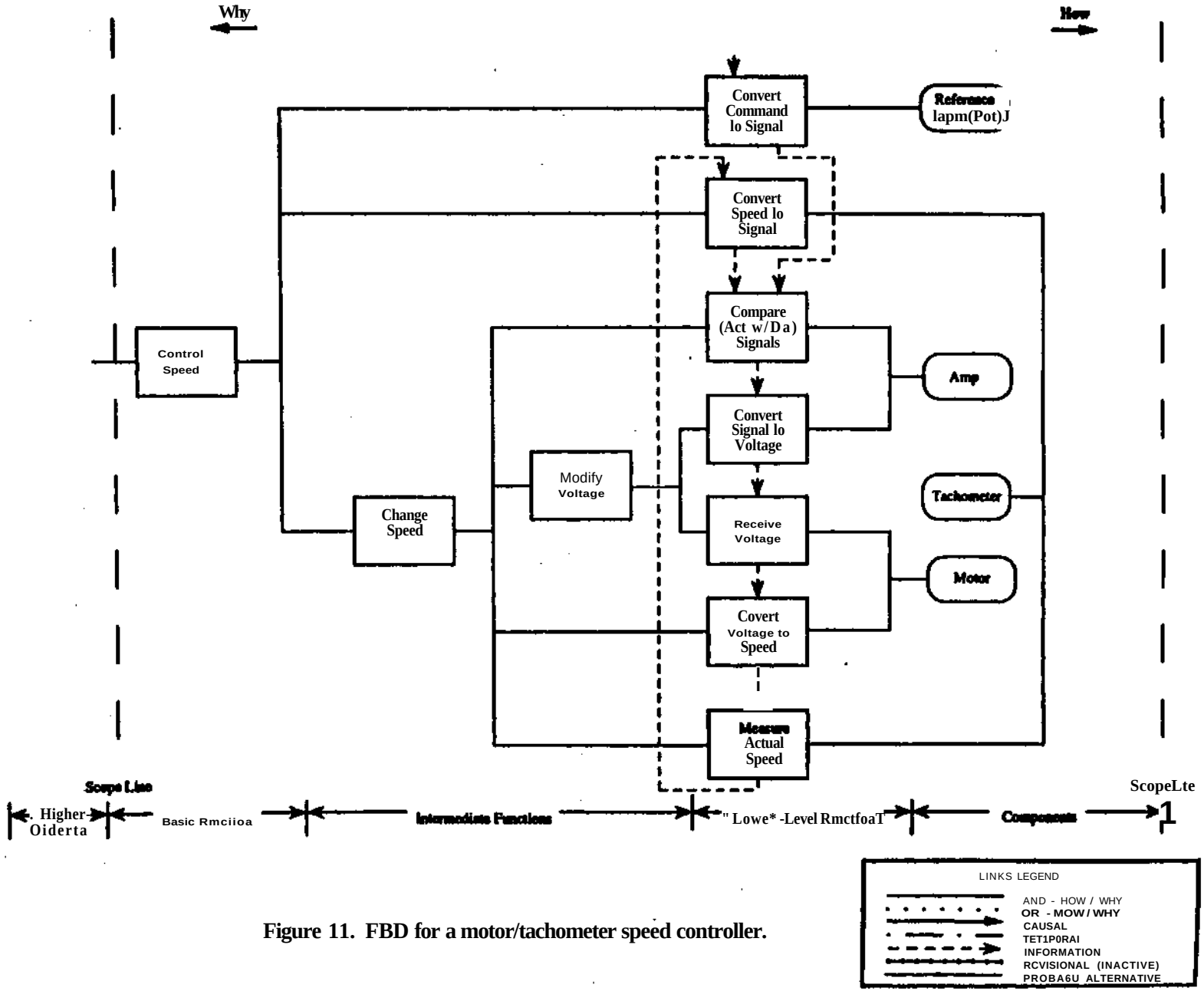


Figure 11. FBD for a motor/tachometer speed controller.

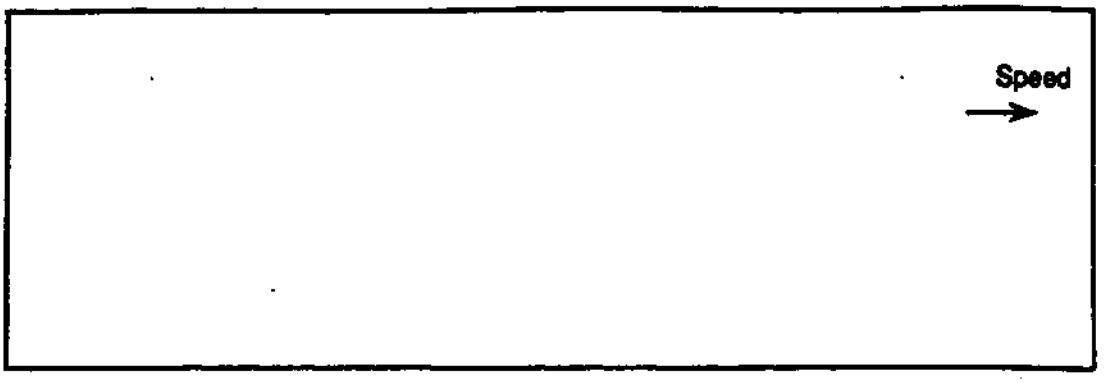
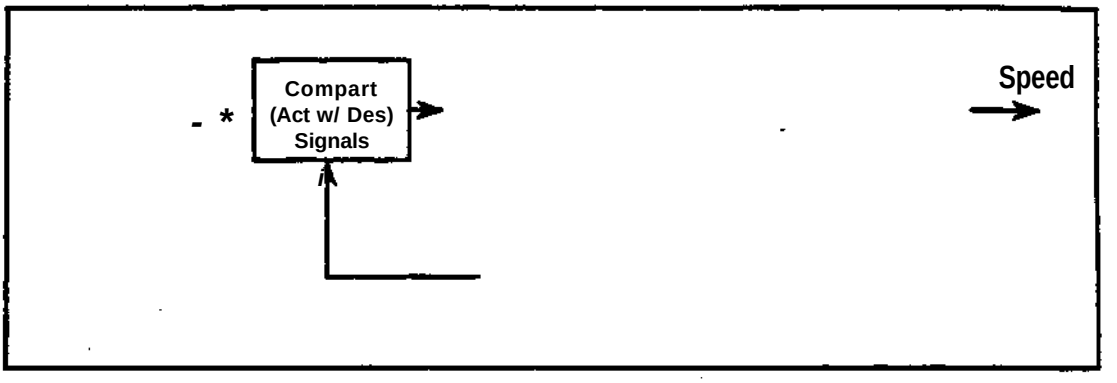
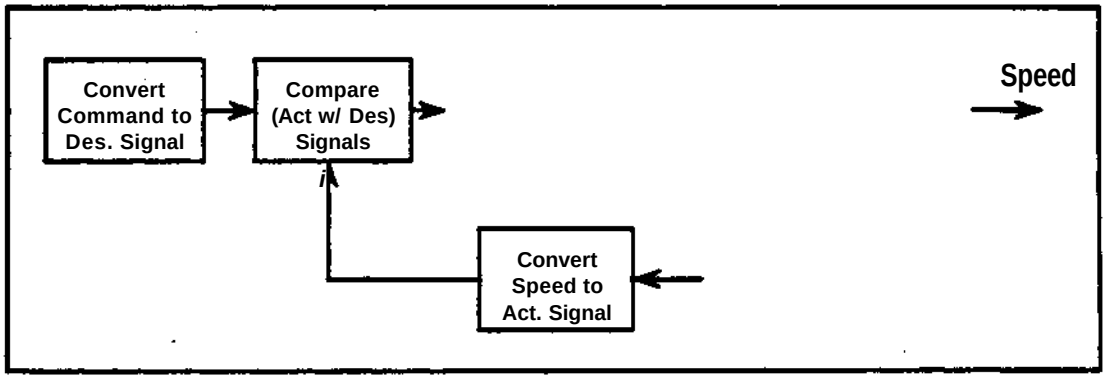


Figure 12a. First step in converting the FBD of Figure 11 into a control loop.



Hgure 12b. Second step in converting the FBD of Hgure 11 into a control loop.



Hgure 12c. Third step in converting the FBD of Figure 11 into a control loop.

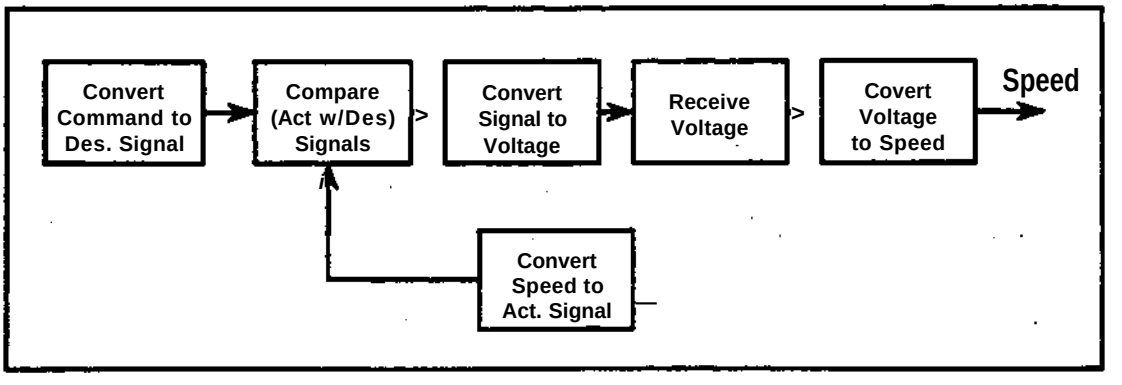


Figure 12d. Fourth step in converting the FBD of Hgure 11 into a control loop.

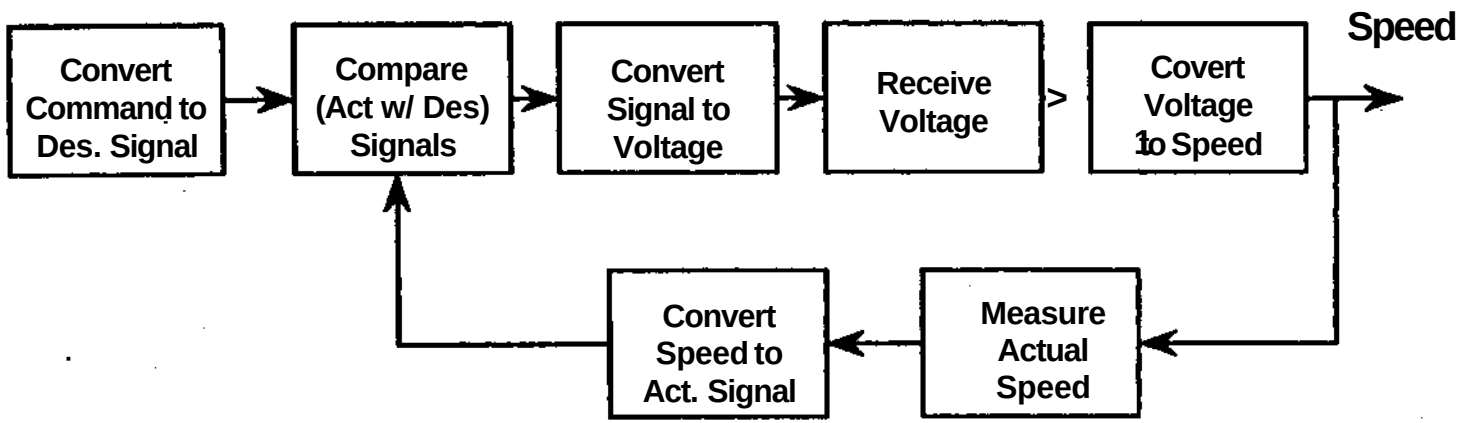
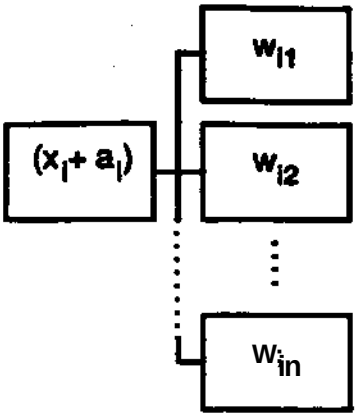
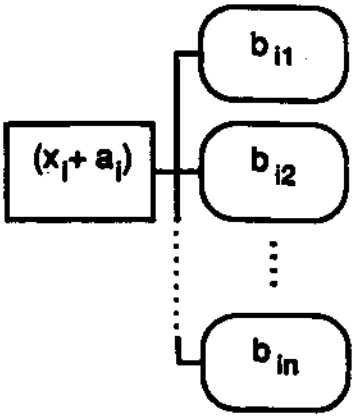


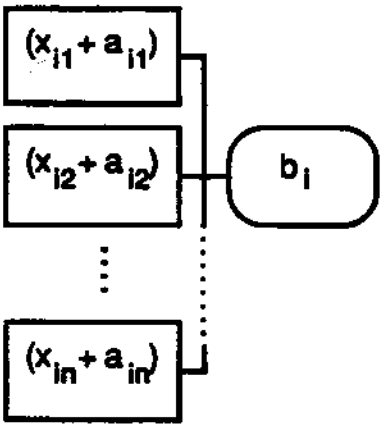
Figure 13. Completed control loop for the FBD of Figure 11.



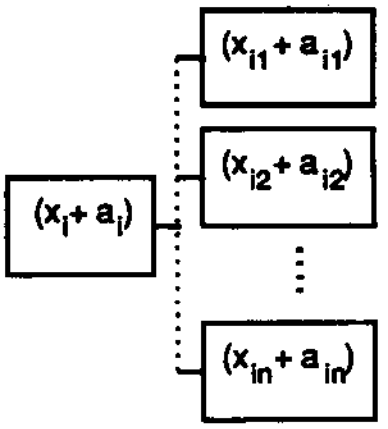
Case I:
Diverging Function Allocations



Case II:
Artifact Allocations



CaselR
Converging Function Allocations



Case IV:
Alternative Function Allocations

Figure 14. Allocation Arithmetic Examples

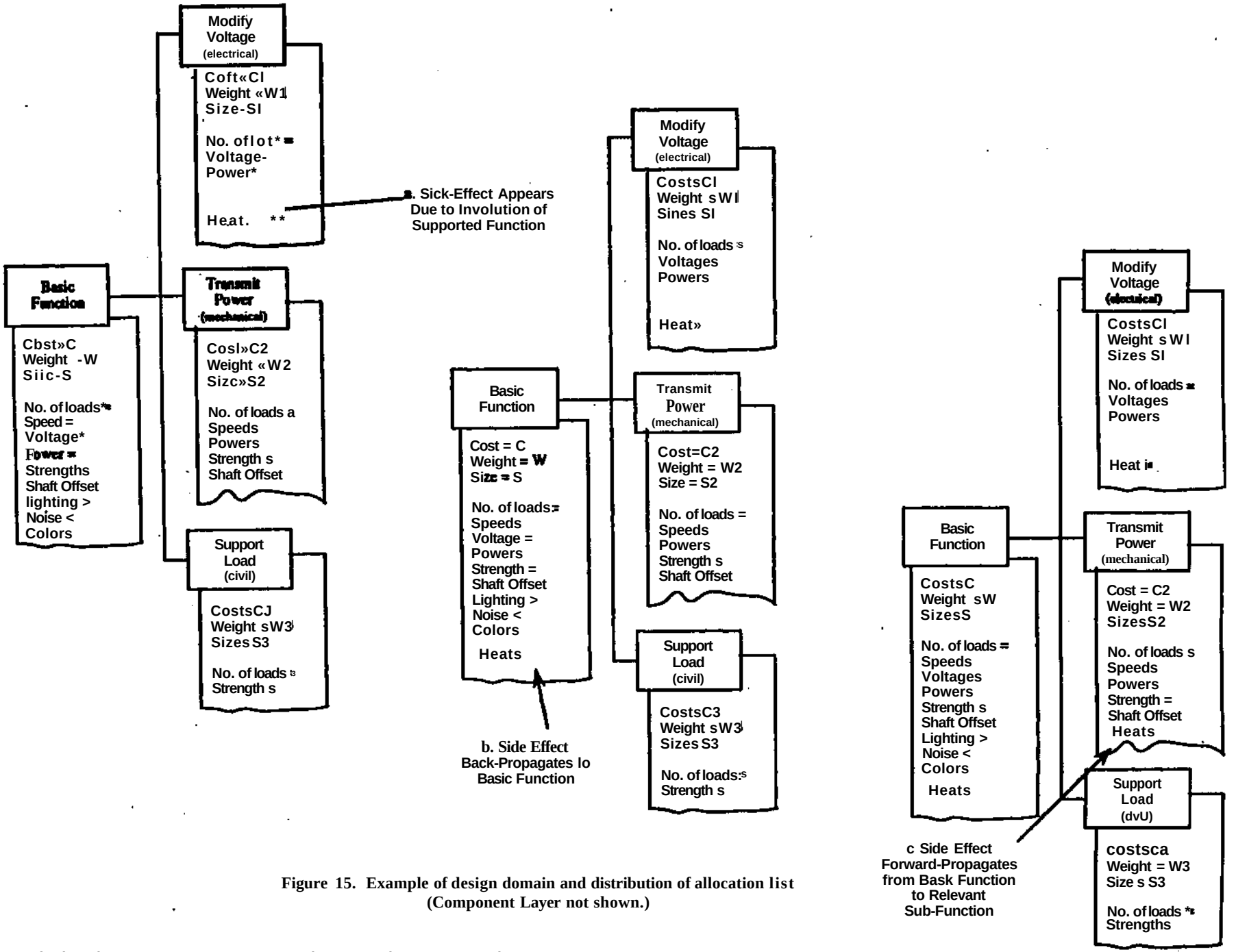


Figure 15. Example of design domain and distribution of allocation list (Component Layer not shown.)

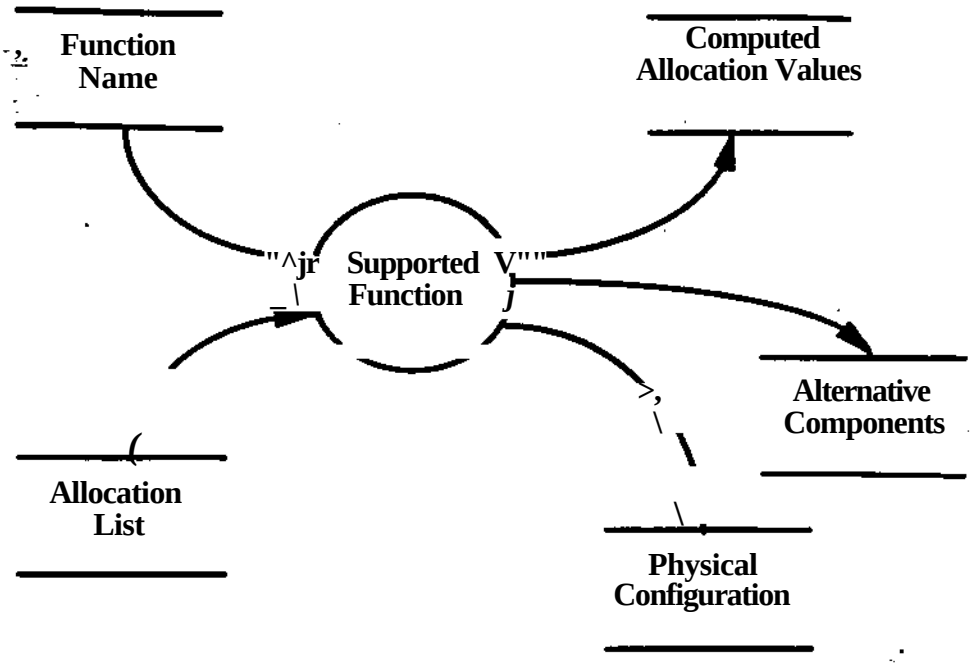


Figure 16. Example of a supported function template.

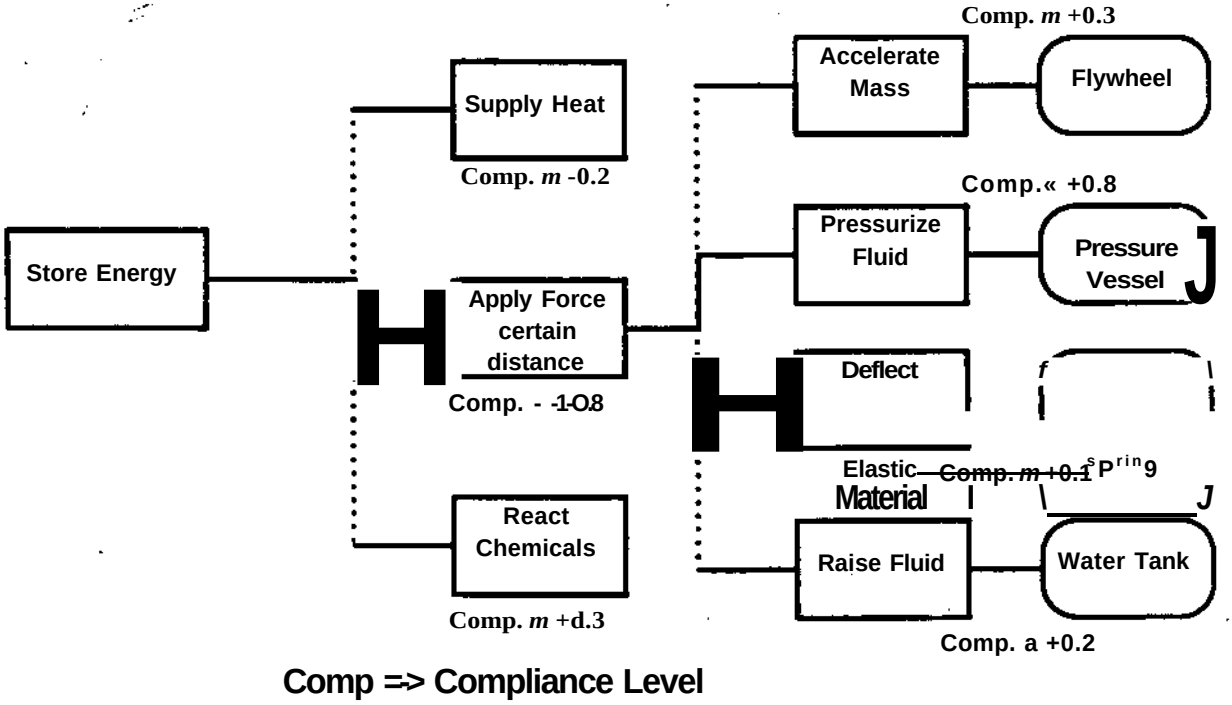


Figure 17. Example of a supported function.

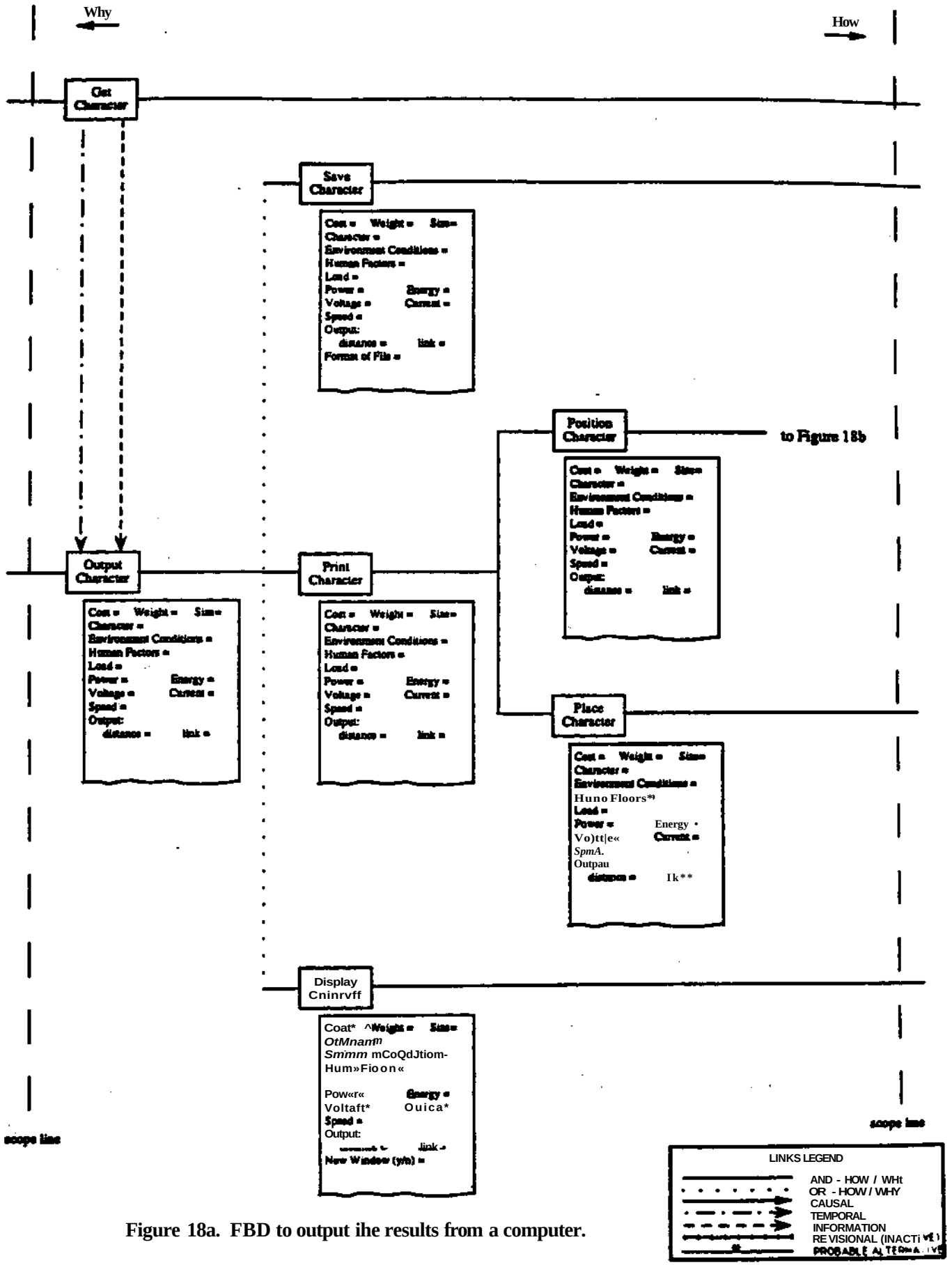


Figure 18a. FBD to output ihe results from a computer.

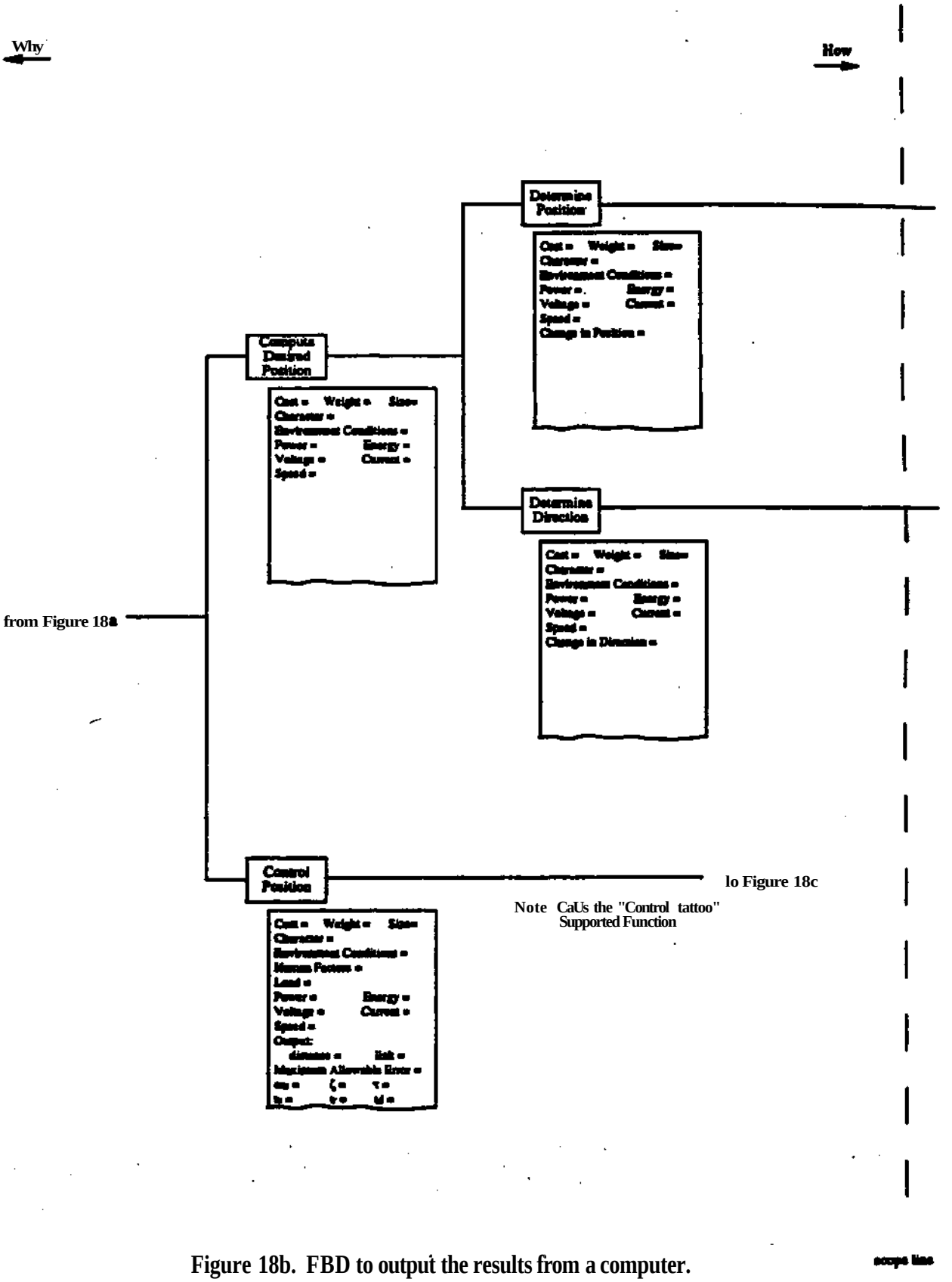


Figure 18b. FBD to output the results from a computer.

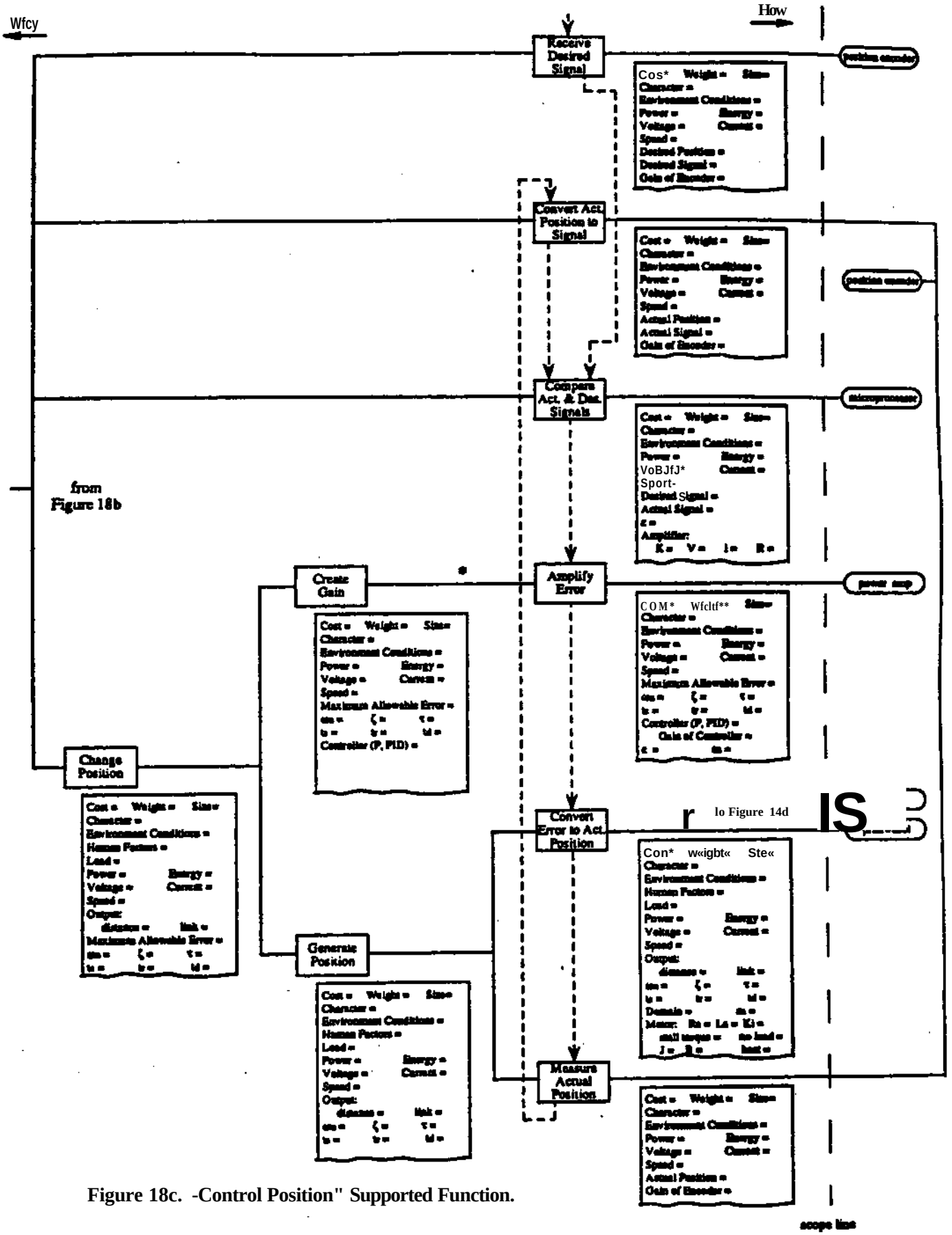


Figure 18c. -Control Position" Supported Function.

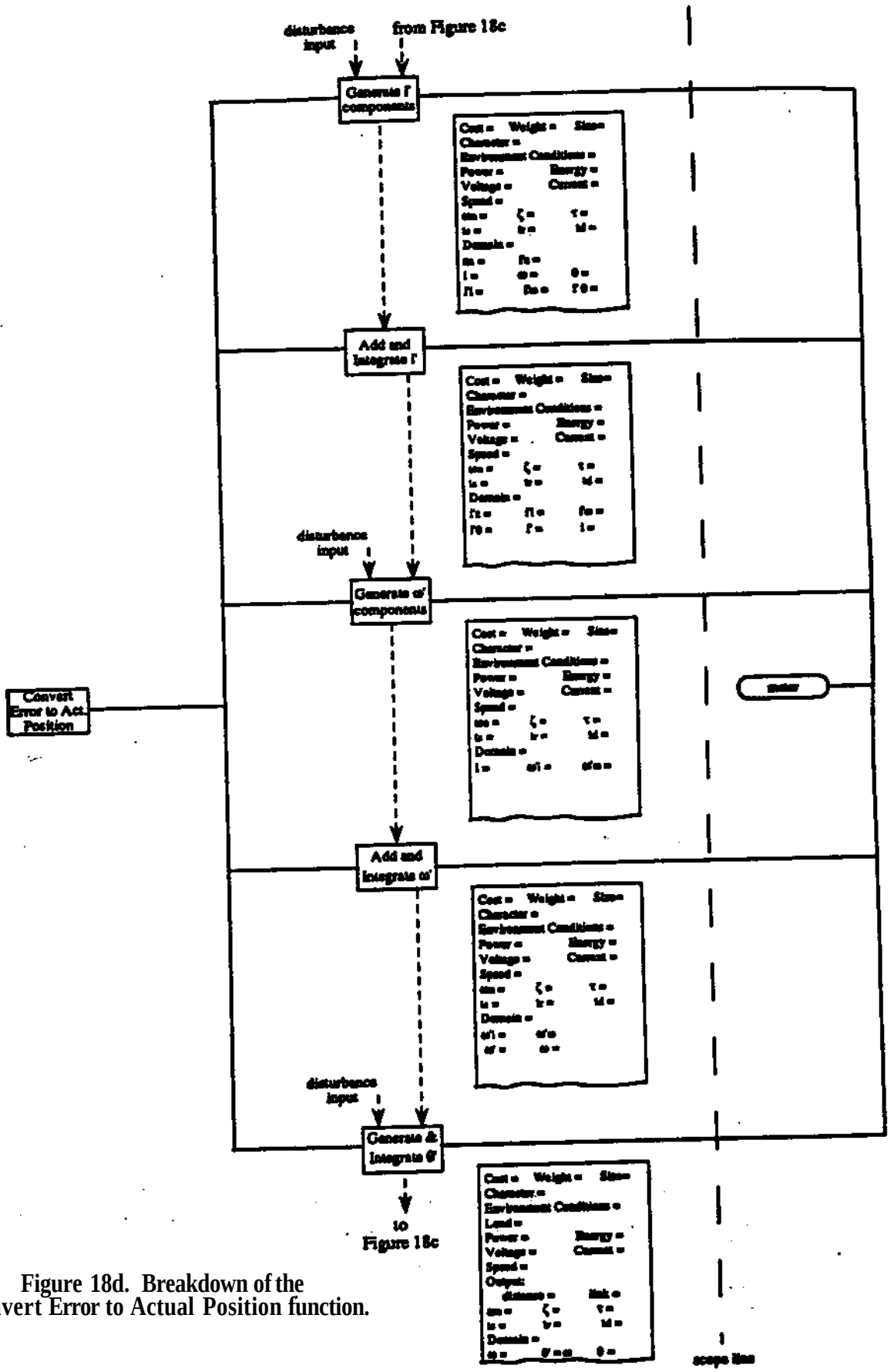
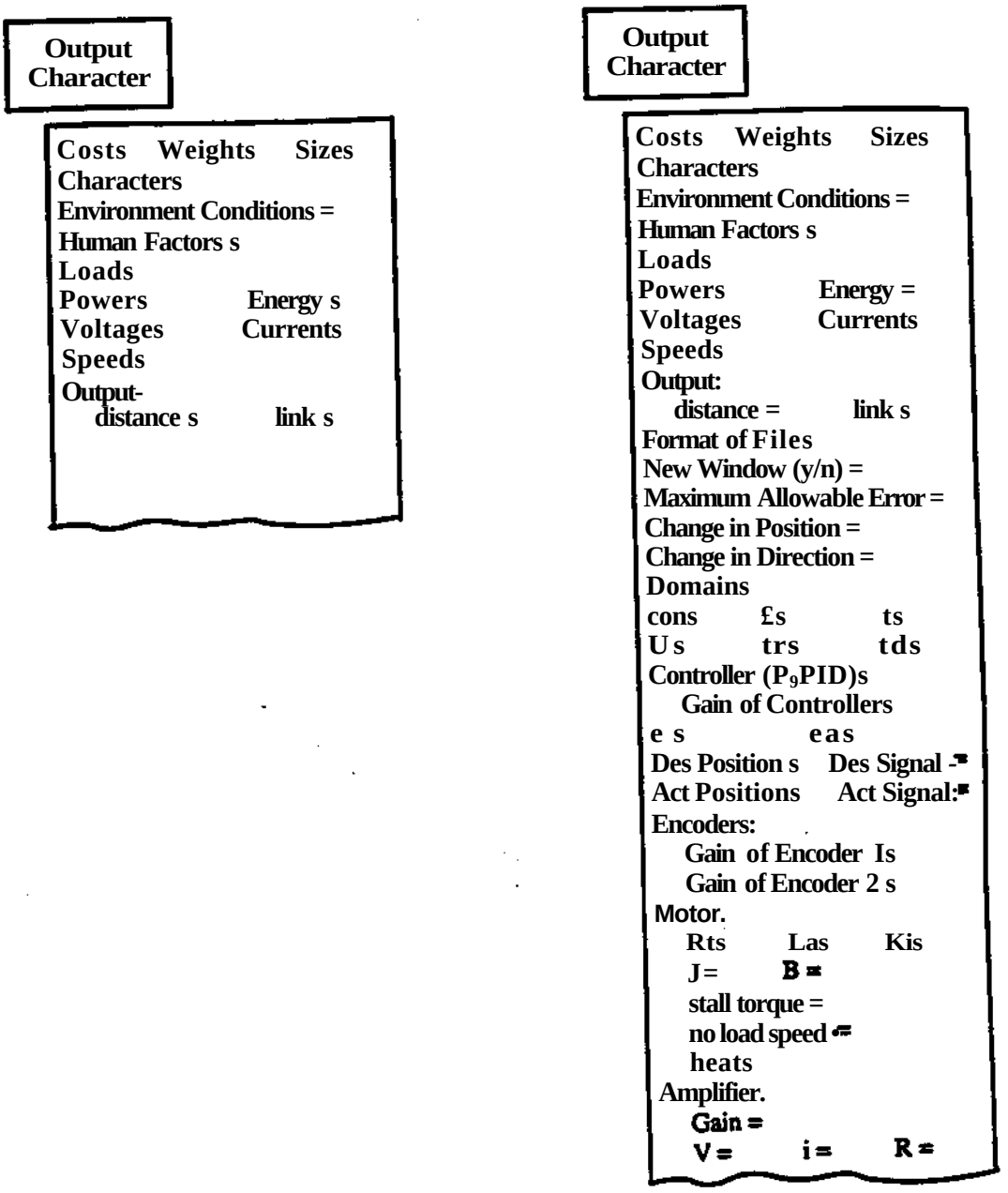


Figure 18d. Breakdown of the Convert Error to Actual Position function.



Case I:
List that is given at the outset of the design

Case II:
List after component specifications have been propagated back up from the lowest level functions

Figure 19. Allocation list from the basic function of Figure 18.

References

Andreason, KBC, Kahter, S. and Lund, T. (1988), *Design for Assembly,* 2nd ed.. New York.

Bailey, RX» (1978), Disciplined Creativity for Engineers, Ann Arbor Science Publishers, Ann Arbor, ML

Beggs, R. M., Cine, C, Etzl, J., Fischer, C, and McCoy, t. (1989), "Automated Design Development Support System (ADDSS)," Boeing Vertol Company, PO Box 33126, Philadelphia, PA 19142.

Birmingham, W. P., Gupta, Anurag P. and Siewiorek, D. P. (1989), "The MICON System for Computer Design", Carnegie Mellon University Engineering Design Research Center 18-10-89.

Bytheway, C W. (1965), "Basic Function Determination Techniques'*. Proceedings of the Fifth National Meeting - Society of American Value Engineers, VoL 11, April 21-23, 1965*

Bytheway, C W. (1971), 'The Creative Aspects of FAST Diagramming", Proceedings of the SAVE Conference, 1971.

Finger, S. and Dixon, J. R. (1989), "A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive and Computer-Based Models of Design Processes[99], *Research in Engineering Design,Vol.* 1, pp. 31-67.

Finger, S. and Rinderle, J. (1989), "A Transformational Approach to Mechanical Design Using a Bond Graph Grammar", *First ASME Design Theory and Methodology Conference,* Montreal, Quebec, Scpthember 1989.

Finger, S. and Rinderle, J. R. (1990), 'Transforming Behavioral and Physical Representations of Mechanical Designs", Carnegie Mellon University Engineering Design Research Center Report 24-35-90.

Furnas, G. W. (1986), "Generalized Fisheye Views", *Human Factors in Computing Systems,* ACM Proceedings CHI 1986, Boston, April 13-17,1986.

Hoover, S. P. and Rinderie, J. R. (1990), "A Synthesis Strategy for Mechanical Devices". *Research in Engineering Design,* 1:87-103.

HundaL M.S. (1991), "Use of Functional Variants in Product Development", ASME Design Theory and Methodology Conference, Miami, FL, September 1991.

Kantowitz, B.R, and Sorlrin, R.D., (1987), Allocation of Functions. In G.I. Salvendy (Ed.), *Handbook of Human Factors* (pp 355-369), New York: John Wiley & Sons.

Kaufman, JJ. (1982), "Function Analysis System Technique (FAST) for Management Application" and "Function Analysis System Technique (FAST) for Management Application, Part II", *Value World,* July/September 1982 and October/December 1982.

Kuo, Benjamin C. (1982), *Automatic Control Systems,* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Mainer, J. A.*(1986), "What is Value Analysis/Value Engineering," Society of American Value Engineers, Northbrook, 1L 60062

Meister, **D. (1985),** ***Behavior*** *Analysis and Measurement* ***Methods,*** New Yoik: John Wiley **& Sons.**

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

PahL G. and Beitz, W. (1988). *Engineering Design: A Systematic Approach,* Springer-**Veriag, NY, Edited by Ken Wallace.**

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

Prendergast, J.F., and Westinghouse Corporate Value Analysis Staff, (1982), "Value Analysis Handbook," Westinghouse Productivity and Quality Center, Pittsburgh, PA 15230-0160.

Pugh, S. (1981), "Concept Selection - A Method that Works", ***International Conference on Engineering Design,*** **ICED 1981, Rome, Italy, March 9-13,1981.**

Rinderle, J. R. (1986), "Implications of Function-Form-Fabrication Relations on Design Decomposition Strategies", Computers in Engineering, 1986, Gupta, G., ed., American Society of Mechanical Engineers, New York, 1986, pp. 193-198.

Ruggles, W. F. (1971), "A Management Planning Tool", ***Proceedings of the SAVE Conference.***

Suh, N. P< (1988), ***The Principles of Design,*** **Oxford University Press, NY.**

Sturges, R.H., Dorman, J.G., and Brecker, J.N. (1986), '"Design for Producibility," Westinghouse Productivity and Quality Center, 1986.

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 01-30-90.

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. (1990), "n-DIM: n-Dimensional Information Modeling - A Shared Computational Environment for Design", Carnegie Mellon University Engineering Design Research Center, September 1989.

Westerberg, A.; Grossmann, I.; Talukdar, S.; Prinz; F.; Fenvas, S.; and Maher, M. L. (1989), "Applications of Artificial Intelligence in Design Research at Carnegie Mellon University's EDRC, Carnegie Mellon University Engineering Design Research Center 05-30-89.

Westinghouse Corporate Services Council (1984). "Report on Life Cycle Costs," Westinghouse Productivity and Quality Center, Pgh PA 15230-0160.