Carnegie Mellon University
Research Showcase

Department of Mechanical Engineering | Carnegie Institute of Technology

1-1-1992

A computer-aided function logic sketchpad

Robert H. Sturges
Carnegie Mellon University

Mohammad I. Kilani

Carnegie Mellon University.Engineering Design Research Center.

Follow this and additional works at: http://repository.cmu.edu/meche

Recommended Citation
Sturges, Robert H.; Kilani, Mohammad I.; and Carnegie Mellon University.Engineering Design Research Center., "A computer-aided function logic sketchpad" (1992). *Department of Mechanical Engineering.* Paper 57.
http://repository.cmu.edu/meche/57

This Technical Report is brought to you for free and open access by the Carnegie Institute of Technology at Research Showcase. It has been accepted for inclusion in Department of Mechanical Engineering by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

A Computer-Aided Function Logic Sketchpad

Robert H. Sturges, Mohammad I. Kilani

EDRC 24-97-92

A Computer-Aided Function Logic Sketchpad

Robert H. Sturges, Assistant Professor
Mohammad I. Kilani, Research Assistant

Carnegie Mellon University
Department of Mechanical Engineering
Pittsburgh, PA 15213-3980

Abstract

Function logic has been an effective approach to improved preliminary and conceptual design for several decades. This paper describes a computer-based function block diagram development tool to aid a designer in the practice of preliminary design and analysis. Its functions include assistance in the identification and the definition of main and secondary functions, identifying links between interrelated functions, and automatic generation of functional block diagrams for the developing system. Limitations of the function logic method and the tool are discussed.

(Word Count: 4184)

1. Introduction

In a typical design problem, a set of specifications are to be met by selection of components that perform designer-determined functions in a configuration which satisfies certain constraints. Preliminary design involves a high level search for that set of function/component/configuration combinations that shows the best promise of accomplishing the design objectives.

To achieve an efficient design that makes effective use of the currently available technology, it is sometimes necessary to widen the design search space beyond previous practice to uncover a wider range of alternative function/component/configuration combinations that can accomplish the design objectives. It is also necessary to evaluate the merits of each of these alternatives on cost, quality and performance scales. A considerable amount of experience is required for the generation and evaluation of alternatives. This experience is rarely possessed by one designer. A group of designers and field experts usually work together as a team to generate and evaluate alternatives.

Recent studies of the design process [Sub 90] revealed some problems in the the current approach to design team interaction mechanisms. Participants have a non-uniform view of the design process, a very high volume of data needs to be managed, and design guidelines and other important design data are often difficult to

retrieve. In addition, the current design state is not easily characterizable [Ull 88].

One approach to providing the above facilities and avoiding some of the above difficulties employs a common language for design that suits all design specialists and can be general enough to encompass a wide variety of design alternatives. This language expresses function rather than artifacts and forms the central element of the Value Engineering (VE) process [Mi 82]. When we describe the design functionally we are thinking of what items *do or perform* rather than what they *are*. Functional representations provide a common descriptive base for all specialists and assists in the discovery of redundant activities at the early stages of the design. Further, this language motivates the generation of alternatives that would accomplish the same thing with better quality or reduced costs. For example, a study by the American Ordnance Association of a sampling of 2000 of its VE projects revealed improvements in cycle time, reliability, quality and maintainability in excess of 60% [Pre 82].

We are currently investigating the potential of a three-level function/allocation/component model of a design to serve as an extension of the elementary function descriptors of VE. We have employed the terminology of the Society of American Value Engineers in dealing with representations of design practice [Stu 90]. Specifically, *function* refers to largely domain-independent characteristics or behaviors of elements or groups of elements of a

design. *Allocation* is the process or the result of assigning specifications and resources, and may be domain-specific. The *intent* of a design is expressed by the totality of its functional and aesthetic elements, their structure and their effects.

We describe in this paper a function diagramming tool, based upon an extension of the manual approach of [By 65], that will assist in systematic identification and definition of design functions. In addition, this representation promotes connection of these functions to their allocations and to actual components. We are investigating the types of links that exist between functions on the function block diagram (FBD) and the effect of these on hardware implementations of the interrelated functions.

2. Function Logic and Function Block Diagrams

A systematic approach of function logic relies on early identification of design goals and describing them in functional terms. The resulting description is called the *basic function* of the design. If the basic function cannot be accomplished by a single component, it is decomposed into several functions which collectively perform the function. These secondary functions may then be translated into components or recursively decomposed further. The function decomposition process continues until we can map each function to a component (and *vice versa*, if we are "reverse engineering" a product or process for analysis.) When at this stage,

the integration of the components will lead to the accomplishment of the design goal. This integration step is usually not specified by the typical FAST (Function Analysis System Technique) diagram [By 71], but rather only outlined with the details left to the design specialist. The result of the function decomposition process is a reasoning structure relating each component to the basic function of the design. This structure can be presented by a diagram that takes the form of a directed graph. The nodes of the graph represent the design functions, basic, intermediate and low order. Its branches represent the reasoning relationship between its functions.

A representative function in the FBD is shown in figure 1. The function node contains the function name (what is done) along with information or allocations related to that function. The nodes to the left of a function node represent the reason why a function is done. This reason, goal or objective is termed a higher order function. The nodes to the right are functions describing the methods by which the function is performed. To describe function diagram methodology, we now give an example of a function block diagram for a readily recognizable product.

2.1 Overhead Transparency Projector Example:

A function block diagram for an overhead slide projector is shown in figure 2 below. The basic function of the transparency projector is identified to be "Enlarge and Project Image". This achieved by "Direct Light", "Focus Light" and "Illuminate

Transparency" secondary functions. Each of these secondary functions degenerates to a lower level function as shown.

In this example, we notice link types other than the basic Why/How links. The "Create Light" function has an unwanted side effect "Produce Heat". Dealing with this side effect requires a set of functions starting with the "Dissipate Heat" function. In addition, two necessary functions "Protect User" and "Enhance Appearance" could not be arrived at simply by answering "How" questions. These functions were linked by a secondary link and led to the function "Encase Product".

In this diagram, we also note that some important functions like "Maintain Task" and "Assure Safety" are linked through a relationship other than How/Why. Both of these functions, when decomposed, led to some important components in the circuit breaker. In addition, the functions "Allow Reset" and "Change Position" are related with a "side effect" link since we need to reset conditions if the position of the switch is changed.

2.2 Observations on Functional Design

In the preceding section we presented an approach for the design process based on functional description of the design objective. This description is followed by successive decomposition of the basic function into secondary functions. The decomposition

proceeds until reaching a level which is directly realizable by available components or more abstract design specifications.

The described method possesses the advantage of providing a general description and decomposition of the design problem in a way which does not constrain the design to specific implementation. Also, the method can accommodate any pre-assigned hardware specifications imposed on the design and can present these constraints to the design team appropriately attached to their corresponding functional descriptors. A conceptual design environment incorporating a database of functions connected to libraries of technologies that implement these functions can provide the designer with a powerful tool for generating alternative solutions to a specific design problem. We envision a design situation in which a functional description of the design problem allows the designer to examine a wide range of technologies applicable to his design. By appropriate integration to design evaluators, the designer can rapidly choose the technology that best suits his design requirements.

By providing a common language for the design, functional description allows a uniform view of the design problem among designers without losing generality or restricting implementation. A function like "Generate Power" can be understood by all specialists while still allowing a variety of implementations ranging from a turbine-generator to a dry cell. The three-level function/allocation/ component approach to design representation allows one to monitor the evolution of the design as it progresses.

A dynamically growing function block diagram may serve as a basis for a design evolution record. Here, we keep not only the final hardware selection but also all the components or technologies investigated. The reason for selecting a particular technology and the specification of all those technologies investigated may also be kept. The resulting diagram provides a convenient description as well as a justification record for the current design status.

Functional thinking throughout the design process provides the potential for the above advantages. Creating a function block diagram and describing the design functionally, however is still a difficult task. During our experiments in creating function block diagrams, several difficulties were observed:

Limitations of How/Why Logic:

It is not always possible to connect all the components in the design to the design objective using How/Why relationships. As noticed in the previous examples, certain components related to the "Assure Safety" and "Maintain Task" functions could not be arrived at by asking the "How" question starting from the design objective. These functions were necessary functions and had to be connected using another type of link. The questions that should be asked to arrive at these components were of the "What if" kind. The exact nature of these questions is currently investigated.

Linguistic Limitations:

Another difficulty in producing the functional description may come from the user's linguistic inability to name the function that describes what the components are to do. One way to overcome this is to provide a dictionary of verbs and associated nouns that usually combine to produce an acceptable function definition. The preferred approach employs a particular subset of classified verbs and nouns [Ja 91], namely, nouns which are either qualitative, quantitative, or conceptual, but not concrete. The verb set includes transformation and control, but excludes passive generation verbs such as "allow", "provide", etc.

3. Rules for Building Function Block Diagrams

The following general rules, based on [By 65] and extended as noted, have been adopted in building function block diagrams:

I. Rules for Naming Functions:

1. Functions are described using one verb and one noun combinations. Examples are "Support Weight", "Move Mass", and "Transmit Torque."

2. Active rather than passive verbs should be used. Active verbs like "transmit torque" allow alternative selection while passive verbs like "provide shaft" limits the selection of alternatives.

3. Nouns should be measurable when possible but should not name specific parts. "Force", "Mass", and "Energy" are some examples.

4. Words which predetermined the methods of accomplishing a function should be avoided. The function "weld rivets" would be better stated as "attach fasteners."

5. Functions should be defined in broad generic terms. The broader the definition, the wider the range of possible functional alternatives.

II. Rules for Connecting Functions:

1. All the functions describing what components do should be connected to the the basic function of the device. A component that is unconnected to the basic function is by definition unnecessary.

2. Lower level functions connect to higher level functions using "How/Why" relationship. If it is not possible to connect functions using "How/Why" reasoning, a side effect relationship need to be investigated.

3. Some functions produce undesirable side effects. An example of this is a "Covert Energy" function which produces heat. Eliminating these side effects by special functions like "Dissipate Heat" leads to a set of components that exist only to remove or exploit this side effect.

4. Function block diagrams may not be hierarchical. There are situations where two or more high level functions are accomplished by one low level function or, more commonly, one artifact.

5. The general form of an FBD is a directed graph. The basic function of the design is at the top of the graph while the components are at the leaves.

III. Types of Links and Their Context:

The following types of links were found to exist in studies [Stu 90] of several complex function block diagrams [Beg 89]:

1. How/Why links:

These are the most commonly encountered links in the diagram. A How link (an "includes" relationship) connecting a high level function to a number of lower level functions indicates that the high level function is accomplished by these low level function. Conversely, the high level function is the reason why each low level function exists (an "included by" relationship.)

2. Causal links:

Depending on the functional attributes under consideration, a causal link ("affects / affected by") may be established between functions. This link may have a sign to indicate which function is causing and which is affected by the causality relationship. Consider

the power attribute of the "Generate Energy" function that is necessary to accelerate a vehicle. Any function that contributes to the weight of the vehicle may be connected to the "Generate Energy" function by a causality link. The "Generate Energy" function is affected by this causality relationship.

We are currently investigating other linking types which express the requirements of simultaneity, information transfer, and alternative configurations.

4. The Implementation of a Function Block Diagram Sketchpad

In this section we describe an implementation of an FBD Sketchpad which provides a set of facilities to help the designer in function-related activities at the preliminary design stage. The sketchpad is implemented in Macintosh HyperCard and makes extensive use of the graphic and highly interactive interface provided in this environment. Currently, three forms of functional design assistance is provided by the FBD Sketchpad. These include assistance in the identification and the definition of main and secondary functions, identifying links between interrelated functions, and automatic generation of functional block diagrams for the designed system.

We are aware that designers do not limit themselves to functional thinking even at the early stages of the design [Ull 88].

The reason for this is usually more than a simple inability or inattention on the part of the designer to think functionally. Design constraints are often imposed such that certain functions be implemented by specified kinds of hardware components. These requirements prevent further subdivision of some functions into its subfunctions in the preferred functional way. To account for this dichotomy, we are currently considering methods for allowing three-level function/allocation/component design. The current implementation allows the designer to attach components and allocations (text) to functions at any level. The designer may describe the function in terms of its allocations or components even before giving a name to these functions. The implementation, however, does not perform any kind of checking on the consistency of component assignment between functions and sub-functions, nor does it propagate component assignment backwards along the function tree.

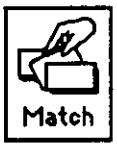
4.1 System Overview

In this section we provide an overview of the main activities that can be performed by the implemented module. We will also discuss the interface form provided for the designer in the module. Three modes of operation are presented, these are function definition and navigation mode, block diagram viewing mode, and analysis mode. We will describe each of these below.

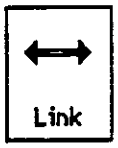
Function Definition and Navigation:

Figure 3 shows the function definition and navigation template. The right section includes function information fields and buttons for related function links. The function information fields include data related to the function such as function name, keywords, components, etc. The data in the field can be edited or modified with a mouse click in the field desired. The buttons in the right side lead to related functions. The buttons show the kind of relationship such as "include" or "affect." On the left side of the Definition and Navigation template lies a control panel that includes the navigation and definition buttons. These buttons are:

For creating and defining new functions

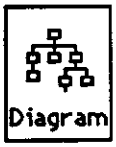


For identifying interrelated functions based on keywords.



For establishing links between interrelated functions.

For performing analysis on data entry.



For automatic generation of function block diagrams.

We will discuss the use of each of these buttons in detail below.

Analysis Template:

The Analysis activity is not yet implemented in this exploratory module. However, a sample analysis template is provided to trigger suggestions on the kind of analysis that would prove useful for the designer. The analysis template, shown in figure 4, has two buttons and an analysis field. The button labeled [Test] displays all the functions defined in the function definition and navigation mode along with the corresponding values of numerical data attached to those functions. The [Return] button returns the user to the function definition and navigation mode.

Function Block Diagram Generation Template:

Figure 5 shows the function block diagram generation template. In this template, function block diagrams for the designed system may be generated automatically. The current implementation generates function trees starting with the functions lately visited, i.e., the function where FBD display is requested.

As shown in figure 3, three buttons are available in the FBD generation mode, [Generate], [* delete], and [Return]. The [Generate] button is used to

generate and display the FBD for the designed system. The button deletes the FBD from the display. The returns the user to the function definition and navigation mode. The use of these buttons for their prescribed operations will be described in detailed instructions in the following sections.

4.2 Using the FBD Sketchpad

In this section we describe the process of using the FBD sketchpad. We assume the user to be familiar with the Macintosh environment.

Starting the Module:

Double click the file on the desktop to get to the function definition and navigation mode. A function definition and navigation template appears. A demonstration default system will be automatically defined. One can modify or add to this system as described below.

Defining New Functions:

Drag the mouse to the button on the left of the template. A simple list for elementary function verbs will appear below the cursor. One can select any of these verbs by dragging the cursor to that verb. When the cursor is placed over a verb, the verb gets

highlighted and a list of nouns associated with that verb appears. Dragging along the desired associated noun causes the noun to become highlighted. By clicking on the selected noun a new function gets created with the selected verb-noun combination as its name. To define a function whose name is not included in the elementary functions list, simply click on the button.

Finding Interrelated Functions Based on Key Word Entry:

The button on the left of the function template may be used to find interrelated functions based on key word entry. Select the key word desired from the the key words list by double clicking on it. Click the button while the keyword is highlighted. If a matching keyword is found in any of the defined functions, that function will be displayed with the key word framed. One can find other functions sharing the same keyword by continually clicking on the button.

Link Establishment Between Interrelated Functions:

Once a link between two functions is identified, the button may be used to establish this link. Two kinds of link establishment are currently supported; "include-included by" and "affect-affected" by links. Links may be established between an existing function and a function to be newly defined (usually, this is an include link). In addition, a link may be established between two existing functions based on matching keywords (usually, this is an affect link).

To establish a link between an existing function and a new function to be included by the existing function, click on the button. A dialog box with the default name "include" appears. Click OK if you want an include link or wish to change the name. (Note that only include or affect links are identified by the system at this time. If another link name is given, an unidentified link name error message appears and the linking process is aborted). Once the link name is identified, a dialog box appears for the function to be linked. Click the button to define a new function and a link will be established between the existing function and the newly defined function with the new name specified.

Another use of the button is to establish a link between two existing functions based on a matching keyword. To do this, one needs first to select a keyword from the keywords list attached to one of the functions on which the link is to be established. With the keyword highlighted, click the button. A dialog box for defining the link name appears. Enter the link name and click OK. Click "match" on the next dialog box and a link gets established between the starting function and the other function with the same selected keyword in its keywords list.

Once a link is established between two functions, a match button with the function name gets created. The button appears on

the function definition portion of the template. Clicking on this button sends the user to the function linked by the link just established. This function represents the "other side" of the link. On the template of this linked function, a matching button pointing to the original function also appears. Clicking on this button sends the user back to the original function where the link originated.

Analysis of the defined functions:

As mentioned previously, the analysis function is still in its preliminary implementation stage. By clicking on the button, an analysis template appears. Clicking the Test button on this template displays the functions currently defined along within the corresponding numeric data associated with these functions.

Function Block Diagram Generation:

The button in the bottom of the control button area can be used to generate block diagrams for the designed system starting with the most recent function visited. Click on this button to get to the function block diagram generation mode. Click the button to view a function tree starting with the selected function. The

button may be used to delete the diagram, and the button returns the user back to the function definition and navigation mode.

5. Future Work:

Several development opportunities exist for the current system. We are investigating ways for displaying three-level function/allocation/component diagrams and ways to handle diagrams that extend beyond the display area of the monitor screen, for example, a hierarchical "fisheye" view [Fur 86]. In addition, we are considering methods for providing automatic assistance in the function allocation process. It is now clear that the function allocation process is highly subjective and depends on the judgment of more that one person. In addition this process is repetitive because designers may not comply with design goals or because management constraints would not assign enough allocations as required by the design. We are working to reduce the repetitiveness of this process by giving the designer an insight into the man/machine allocations required for his system and are investigating design rules which may be used for this purpose.

References:

[Beg 89] Beggs, R. M., Ciric, C., Etzl, J., Fischer, C., and McCoy, T. (1989), "Automated Design Development Support System (ADDSS)," Boeing Vertol Company, PO Box 33126, Philadelphia, PA 19142.

[By 65] Bytheway, C.W., 1965. "Basic Function Determination Techniques". *Proceedings . . . Fifth National Meeting - Society of American Value Engineers*, Vol. 11, April 21-23, 1965.

[By 71] Bytheway, C. W., 1971. "FAST Diagrams for Creative Function Analysis," *Journal of Value Engineering*, Vol. 71-3, pp 6-10.

[Fur 86] Furnas, George W. (1986), "Generalized Fisheye Views", *Human Factors in Computing Systems*, ACM Proceedings CHI 1986, Boston, April 13-17, 1986.

[Ja 91] Jakobsen, K., Sigurjónsson, J., and Jakobsen, Ø., 1991. "Formalized Specifications of Functional Requirements," *Design Studies*, Vol 12, No. 1.

[Mi 82] Miles, L. D., (1982), *Techniques of Value Analysis*. New York: McGraw Hill Book Company, Second Edition.

[Pre 82] Prendergast, J.F., and Westinghouse Corporate Value Analysis Staff, (1982), *Value Analysis Handbook*, Westinghouse Productivity and Quality Center, Pittsburgh, PA 15230-0160.

[Stu 90] Sturges, R. H., O'Shaughnessy, K., and Kilani, M. I. (1990), "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.

[Sub 90] Subramahnian, E., Podnar, G., and Westerberg, A., 1990. "N-Dimensional Information Modeling - A Shared Computational Environment for Design", Carnegie Mellon University Engineering Design Research Center, September.

[Ull 88] 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.

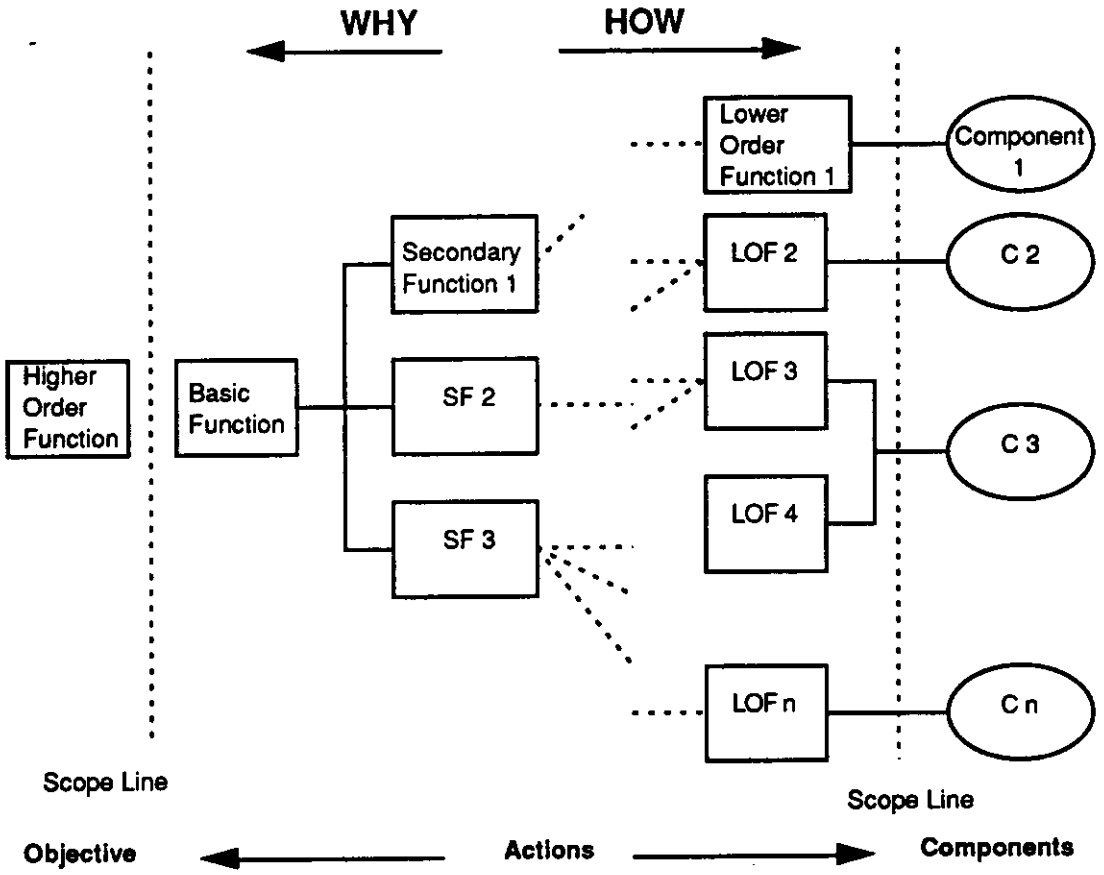


Figure 1. A Generic Function Block Diagram

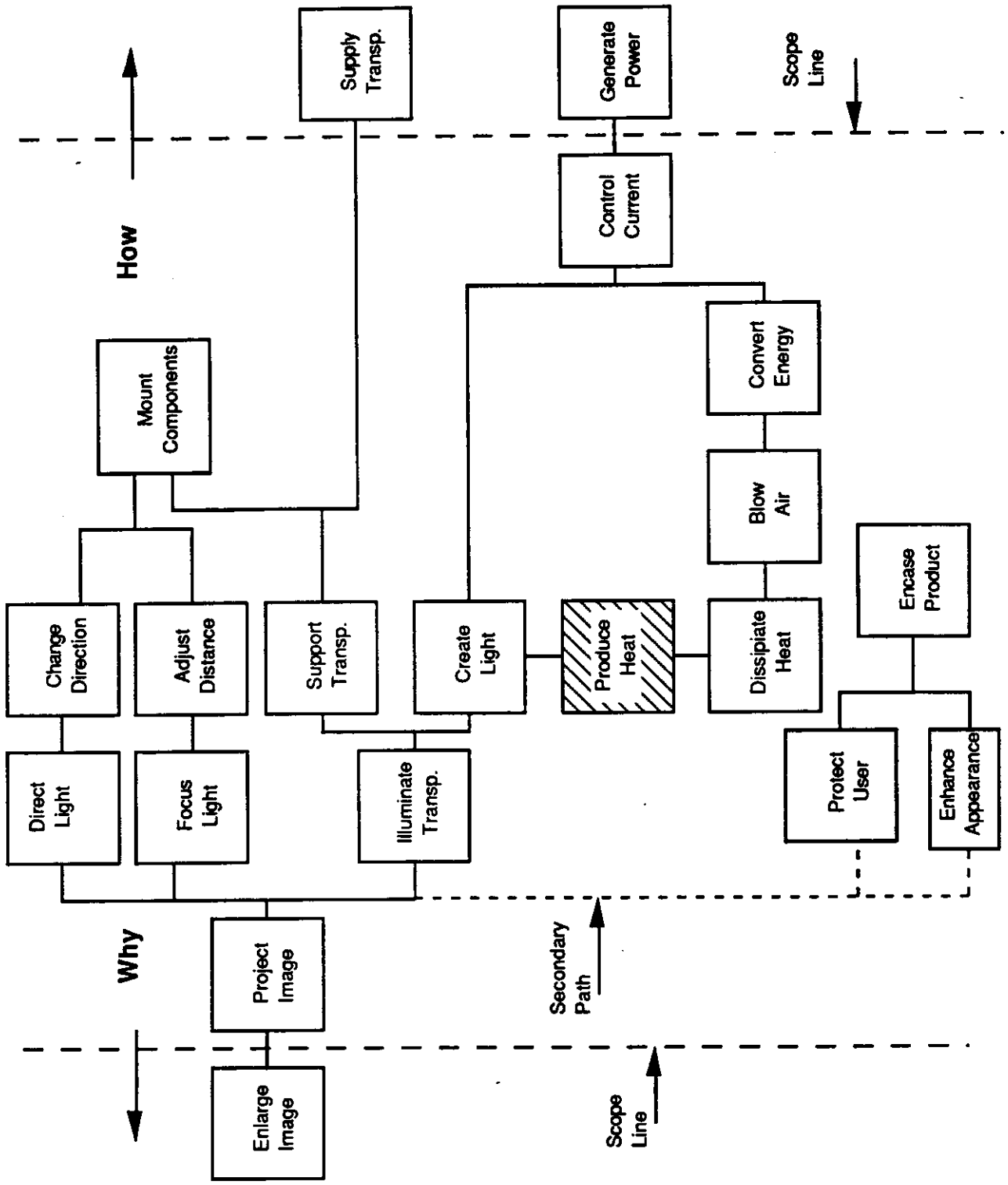


Figure 2. Overhead Transparency Projector FBD Example

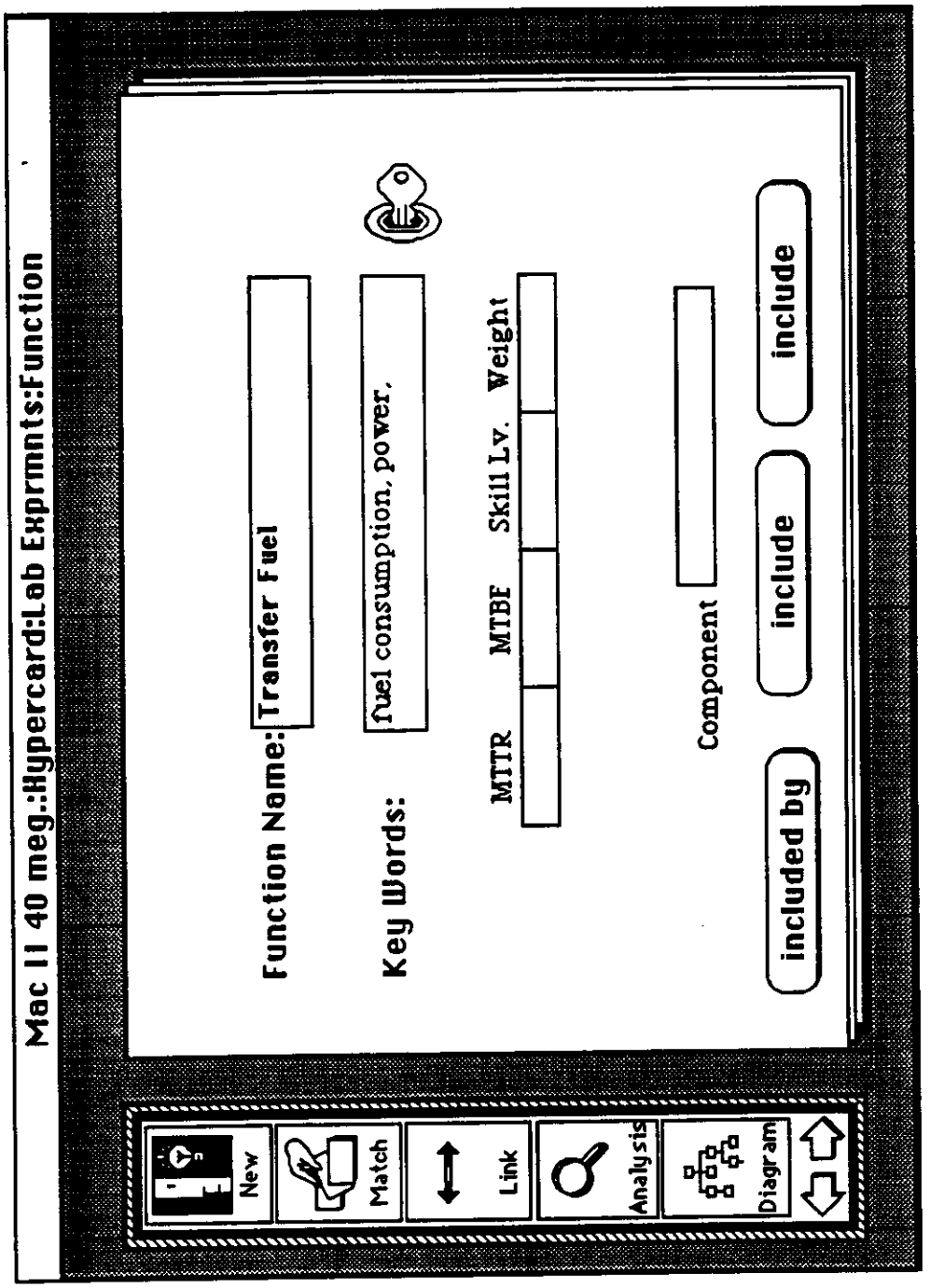


Figure 3. Function definition and Navigation template.

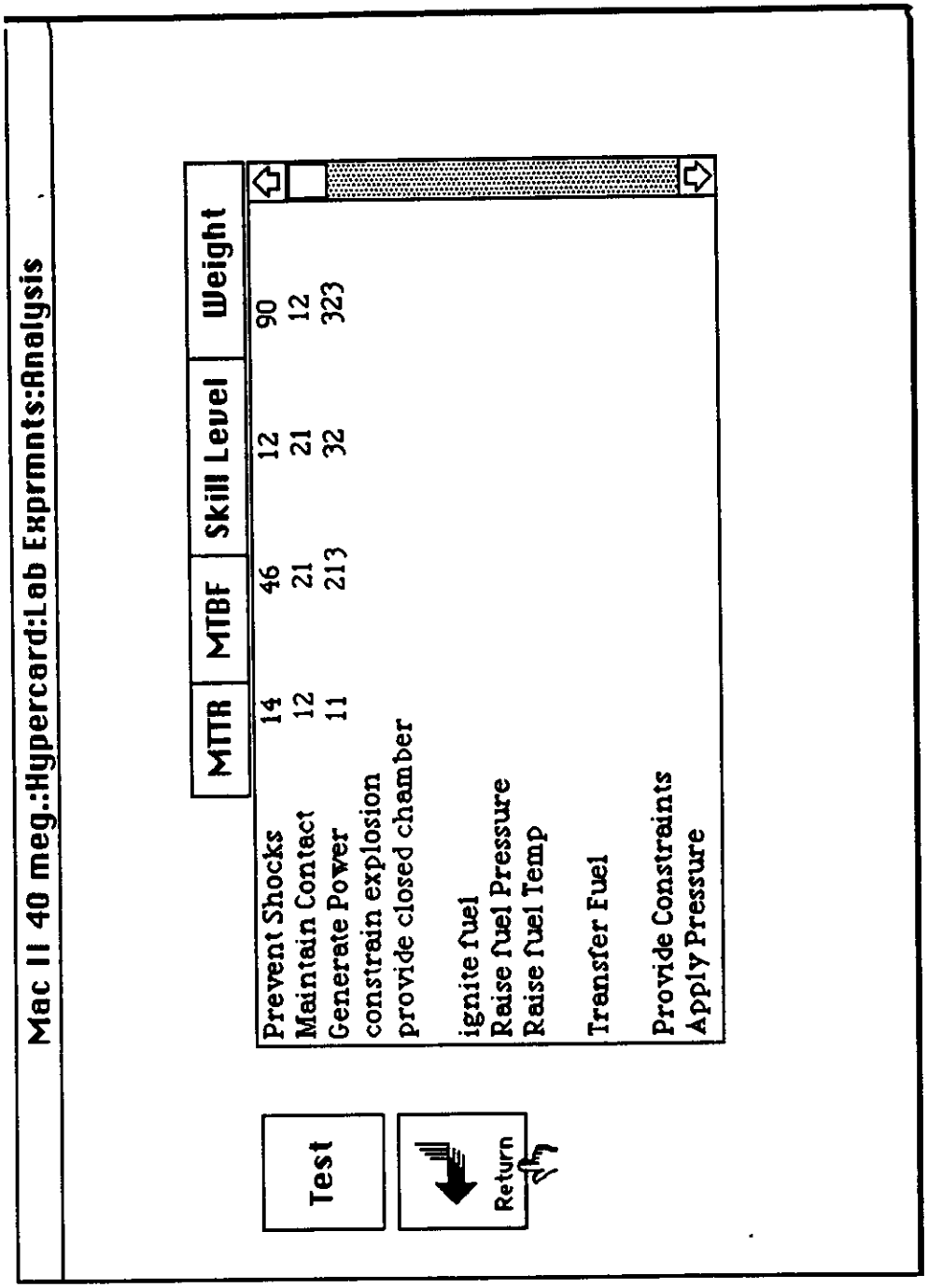


Figure 4. Analysis Template

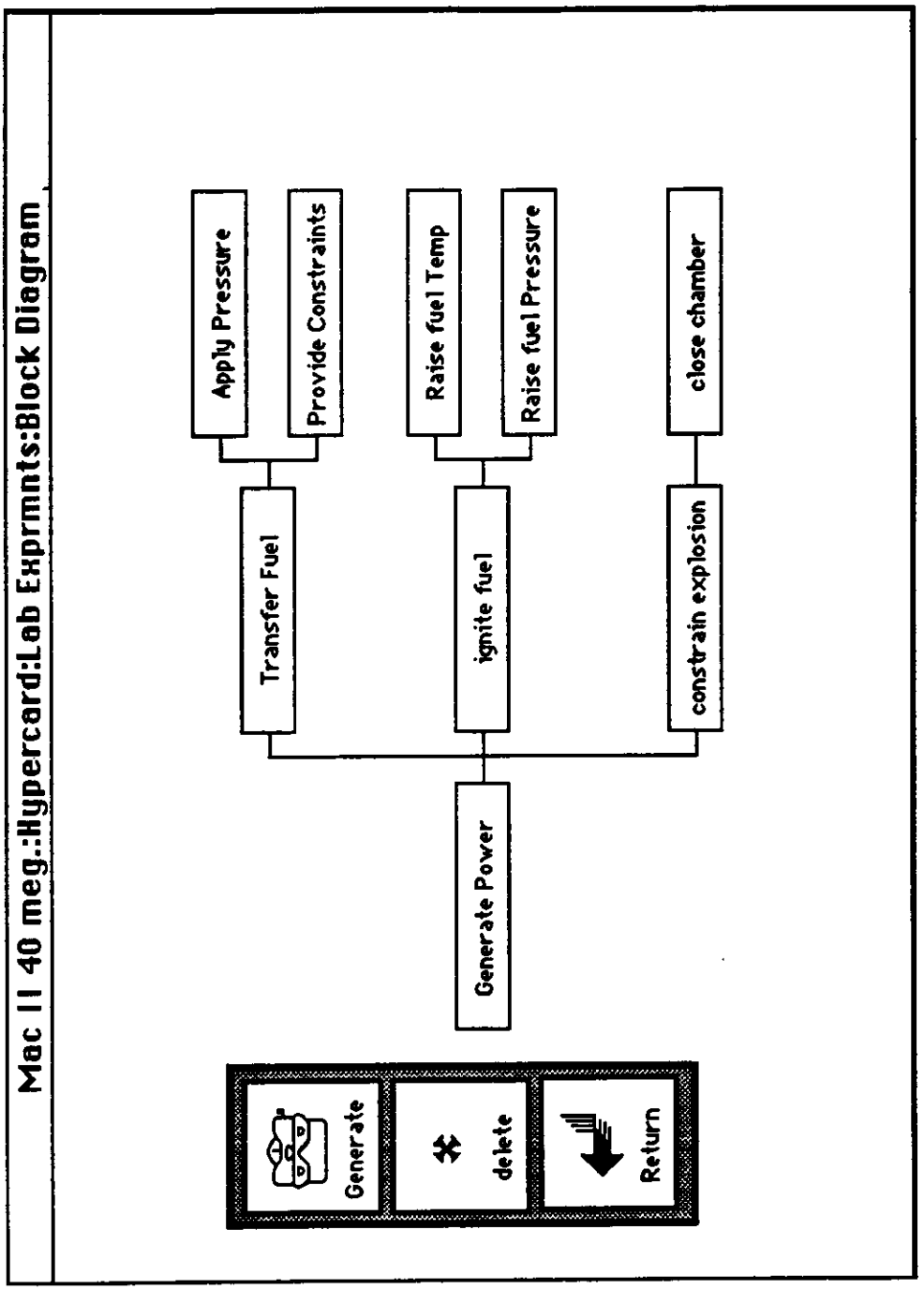


Figure 5. Function Block Diagram Generation Template.