SYSTEM ARCHITECTURE TEMPLATE

White Paper

Written by: Jason D. Hintersteiner[1] and Glenn Friedman[2]

Last updated: April 22, 1999

This document provides the standard templates for system architecture tables, along with complete descriptions, for hardware systems[3]. The following template should be used when applying Axiomatic Design (AD) to a system design project. This template is intended to provide a consistent format for the system architecture (SA), which can be used as a standard format for representation and design reviews.

The *mission statement* for any project can be written as a single functional requirement (FR), which asks "*what* is the overall project goal". This FR is addressed by a single design parameter (DP), which asks "*how* is this project goal accomplished". If a system architecture already exists at a higher level, the mission statement is the parent FR and DP from that system architecture.

Going through a process of *decomposition*, each design parameter (starting from the mission statement) is broken down into constituent sub-FRs with corresponding sub-DPs. This decomposition process, known as *zigzagging*, continues top-down from the topmost level (i.e. system) to levels of increasing detail (i.e. subsystems and components). The decomposition should be taken down to levels where the DPs are physical parts (geometries), computer programs (flow charts), and specifications (tolerances, limits, etc.). The hierarchical structure that emerges is known as the system architecture. The format for the system architecture is the subject of this document.

Section 1 discusses the format for representing FRs and DPs, along with design matrices to capture the relationships between the FRs and DPs, at every level of the design hierarchy. Section 2 discusses the format for representing constraints (Cs) on the design, which can originate from management, marketing, and/or higher level design decisions. Note that the Cs apply to lower level FRs and have the effect of limiting the choice of DPs in the decomposition. Also, it is common for constraints at a parent level to dictate the need for one or more sub-level FRs. Section 3 discusses some diagramming conventions, to clarify the hierarchical structure of the FRs and DPs, as well as graphically depict the design matrices to show the order of design tasks.

1 FUNCTIONAL REQUIREMENTS AND DESIGN PARAMETERS

In axiomatic design, problems are formulated first by specifying functional requirements (FRs), which are single phrases that describes each overall task. Each FR should be specified as a *verb*, since it asks "*what* task do you want to accomplish", and should also be specified in a solution-

[1] Axiomatic Design Group, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Rm. 31-261, Cambridge, MA 02139 USA. jdhinter@mit.edu

[2] SVG Lithography Systems, Inc. 901 Ethan Allen Highway, Ridgefield, CT 06877. friedmag@svg.com

[3] Readers should refer to [Hintersteiner & Nain, 1999], shown in the references, for information on representing software systems.

neutral manner (i.e. specify the task itself, not a preconceived notion of how that task will be addressed). Each FR is addressed by a corresponding design parameter (DP), which are single phrases which express the solution or approach for each task. Each DP should be specified as a *noun*, since it asks "*how* do you want to accomplish the task". This is summarized in Table 1.

Table 1: Summary of an FR and a DP.

Functional Requirement (FR)	Design Parameter (DP)
Single phrase that describes the overall project goal.	Single phrase that represents the project solution.

1.1 THE FR/DP TABLE

Table 2: Generic FR / DP table (any level).

	Functional Requirements (FRs)		Design Parameters (DPs)	
	Name	*Description*	*Description*	*Ver*
j .#		***Parent FR***	***Parent DP***	
1	*Process*	Perform physical process #1	Process module #1	T
2	*Process*	Perform physical process #2	(a) Process module #2 (1st alternative) (b) Process module #2 (2nd alternative) (c) Process module #2 (3rd alternative)	In
3	*Process*	Perform physical process #3	Process module #3	De
4	*Transport*	Perform process #4 (transport)	Transport module	Dr
5	*Control*	Schedule and coordinate all local process functions	Command and control algorithm (CCA)	T
6	*Support*	Integrate subassemblies	Support framework	U

Table 2 provides the generic template for listing FRs and DPs. The top row indicates the index at this level, where "#" refers to the index in later rows, and the "φ" indicates the full index of the parent FR/DP. The parent FR and DP are included in the table in order to place the FRs and DPs at this level in context with their parent FR/DP. At the top level, the parent is the defining FR provided by the customer, and the defining DP which is the name of the system. For example, if φ = 3.4a.2, the parent FR is FR.3.4a.2, and the FRs listed in this table are FR.3.4a.2.1, FR.3.4a.2.2, etc.

The FR column has been divided into two columns, for "name" and "description". Here, the "name" is a one or two word summary of the process, control, and support FRs. The template provided shows four process modules, where the last process module is instantiated specifically as a transport process. Hence, for FR.3.2.4a.4, the term "transport" is used as the name. In a manufacturing system, transport processes will always be listed after any physical processes which perform "value-added" activities on the operands. This is because transport processes, in general, will be responsible for transporting operands between physical process modules, and thus will depend on the specific design of those DPs. There can be an ***arbitrary but nonzero*** number of process modules at each level of the hierarchy, depending upon the specific design, and the indices can be renumbered accordingly. These will ***always*** be followed by one control and one support FR.

There may be situations where alternative choices for DPs may exist, such as selecting between two or more possible design alternatives and in cases where a particular tool platform may institute different DPs at different times. In such situations, alternate DPs can be listed as shown in Table 2.

Each alternate DP will, in general, have different sets of sub-FRs and constraints, and so a separate decomposition should be provided for each alternative.

The last column in the table is used for verification codes, so that the designer can specify a code to indicate the verification procedure to be used to ensure that the DP is satisfying its corresponding FR. Verification may be done by testing (T), inspection (I), demonstration (De), drawings (Dr), or proven unchanged technology (U). More complete descriptions of verification procedures can be included in the description for the DP. This column is optional, and may be unnecessary in particular design situations. However, this column is useful in encouraging the designer to think about verification issues.

When writing a system architecture in Microsoft Word, bookmarks with a unique name should be used for the parent index (i.e. "φ"), so that they can be cross-referenced in the DP description titles and in the off-diagonal element descriptions for the design matrix. The bookmarks can contain a cross-reference to the parent's parent, and so a nested set of bookmarks can be created. This is advised so that, if a parent index number is altered, its child indices are automatically updated in the document.

1.2 DP DESCRIPTIONS

A bulleted list description of each DP should be provided underneath the table, in order to clarify what is meant by that DP, as well as any specific design issues that must be considered. This includes providing information such as the vendor (if the DP is purchased externally), relevant operating conditions, and so forth. Verification information may also be included.

The format of this list should be as follows.

- **Process module #1 (DP.j .1):** This module is primarily responsible for performing process #1. This includes...
- **Process module #2 (DP.j .2):** This module is primarily responsible for performing process #2. This includes...
- **Process module #3 (DP.j .3):** This module is primarily responsible for performing process #3. This includes...
- **Transport module (DP.j .4):** This module is primarily responsible for performing transport tasks between some or all of the process modules. This includes...
- **Command and control algorithm (DP.j .5):** This CCA coordinates the process tasks at this level. This includes...
- **Support framework (DP.j .6):** This support framework includes the layout and configuration of the process DPs, electrical and pneumatic supplies, wiring, mechanical support structure, and so forth. This includes...

1.3 DESIGN MATRIX

When the list of FRs and DPs are formulated at a particular hierarchical level, a *design matrix* is used to correlate how the DPs impact the FRs. An axiomatic design matrix equation of the form {FR}=[A]{DP}is constructed and the matrix elements evaluated. Each element A_{ij} (row *i*, column *j*) in the matrix is evaluated by asking, "Can we design (or change the existing design of) DP.j without impacting how we address FR.i?" In the design matrix, a "X" is used to signify that a relationship exists between DP.j and FR.i, and a "O" is used to signify that no relationship exists.

Obviously, the diagonal elements (i=j) should all be "X", since DP.i is chosen to satisfy FR.i. When there are no off-diagonal elements (i.e. every $i \neq j$ has a "O"), the design is *uncoupled*, which indicates that each DP only impacts its corresponding FR, and is independent of all other FRs. This is the most desirable case by the Independence Axiom, though is unlikely to occur in practice. When there are off-diagonal elements, the matrix rows and columns should be reordered in an attempt to achieve a lower-triangular matrix. If the design matrix is lower-triangular, it is a *decoupled* design and still satisfies the Independence Axiom, since DPs can be altered in a specific sequence without needing to iterate the design solution. If there exist any off-diagonal "X" in the upper-triangular portion of the design matrix, the design is *coupled*, indicating that iteration is required to converge on an acceptable design solution to satisfy all of the coupled FRs. This last case violates the Independence Axiom, and should be avoided by choosing alternative DPs whenever possible.

Note that the order of FRs in a lower-triangular matrix indicate the order of importance, with the top matrix row being the most important FR (and the first to decompose).

In general, the control and support FRs will be impacted by the specific design of the process modules, and hence at least the last two rows will be decoupled, as shown. (Note, if a process module is completely passive and requires no active control, there will be a zero in the corresponding column for the control FR). Question marks are used to show the relationships between the process modules, indicating that the process modules can be either uncoupled or decoupled, depending on the specifics of the design. Coupling between the process modules, as well as between the process and control and support modules, is possible though undesirable as it violates the Independence Axiom.

The format for the FRs and DPs in the matrix equation can either be abbreviated descriptions (as shown) or the *full* numerical index can be used, and is left to the preference of the designer. The problem with using the full numerical index, however, is that there is no method currently available in Microsoft Word to automatically update the index numbers if the parent index is altered, and thus they must be updated manually. This should not be a problem, however, in the Axiomatic Design software under development.

$$\begin{bmatrix}\text{Perform Process \#1}\\ \text{Perform Process \#2}\\ \text{Perform Process \#3}\\ \text{Perform Process \#4}\\ \text{Control processes}\\ \text{Integrate subass'y}\end{bmatrix} = \begin{bmatrix} X & O & O & O & O & O\\ ? & X & O & O & O & O\\ ? & ? & X & O & O & O\\ ? & ? & ? & X & O & O\\ X & X & X & X & X & O\\ X & X & X & X & X & X\end{bmatrix}\begin{bmatrix}\text{Process module \#1}\\ \text{Process Module \#2}\\ \text{Process Module \#3}\\ \text{Process Module \#4}\\ \text{CCA}\\ \text{Support Framework}\end{bmatrix}$$

After the design matrix, a remark (i.e. explanation) should be provided for each off-diagonal “X” in the design matrix. In some instances, a description may also be desirable for an off-diagonal “O”, especially in cases where the relationship is a “O” only under certain conditions, or where a “O” appears in an *on-diagonal* term (meaning that the DP does not satisfy its corresponding FR). If a “?” or “*” is used within the design matrix (in cases where the relationship between an FR and a DP is either variable or unknown), an explanation should also be provided.

There are currently two acceptable formats for these descriptions, either in text or tabular format. It doesn t matter which format is followed, so long as one format is used consistently in a document. Please note that indices can be grouped in cases where the same description is applicable to multiple off-diagonal elements.

1.3.1 TEXT FORMAT

The text format for making remarks on off-diagonal terms is as follows:

Off-diagonal Terms:
Here, the index (*i*, *j*) refers to the element corresponding to FR. .*i* and DP. .*j*.

(2, 1): xxx

(3, 1): xxx

(3, 2) – (4, 2): xxx

(3, 4): Coupling results under conditions where… .

(4, 1), (4, 3): xxx

(5, 1) – (5, 4): The CCA is responsible for coordinating the activities and interactions between these process modules.

(6, 1) – (6, 5): The support framework design depends on the design of each process DP and the requirements specified by the design of the CCA.

1.3.2 TABULAR FORMAT

The tabular format for making remarks on off-diagonal terms is shown in Table 3:

Table 3: Tabular format for comments on off-diagonal terms in the design matrix.

i	j	Remarks (FR.j .i, DP.j .j)
i	j	jth design parameter affects how system performs ith functional requirement: e.g. some specific example is included hear. Additional comments go here.

2 CONSTRAINTS

Error! Reference source not found. is the current version of the generic template for constraints. Each constraint gets a unique index, indicating the level of the decomposition to which these constraints correspond. The title bar for the table includes a cross-reference to the current parent index (i.e. “φ“). The table also includes columns for the full index of the parent constraints for each constraint. At the top level, the constraints can come from external customers (i.e. marketing) or internal departments (i.e. management). Each constraint has a one or two word "name" summarizing the nature of the constraint, followed by a full description.

The columns after the description indicate which FRs at this level are impacted by each constraint, by means of a check mark or a question mark, if the relationship is uncertain. In the future Axiomatic Design Software, this may be expanded to include a method of adding comments, so that the relationship between the Cs and FRs can be captured more thoroughly. The last column is provided for verification codes, in order to specify how verification will be performed to ensure that the constraint is being satisfied. (These codes are discussed in the previous section.) As with the impact boxes, the future Axiomatic Design Software should allow these to be commented in greater detail in a separate window.

In order to properly identify the importance and stringency of different constraints at a particular level of the hierarchy, it is useful to categorize the constraints. These classifications appear at every level of the system hierarchy, maintaining the fractal representation of the system architecture.

- **Critical performance specifications:** These are the set of constraints which are most critical for the system to be considered a success, and form the metrics by which a system will be judged by its customers. Generally, these constraints will have specific values which must be achieved or exceeded in order for the system to be acceptable to the customer. Examples of this type of constraint include throughput specifications and specific process performance metrics.

- **Interface Constraints:** These are the set of constraints which describe how the system must interact with its environment This includes specification of all interfaces between the system and the environment (including operators and maintenance personnel), as well as specific features or options that particular customers may desire (e.g. a custom layout so that the system will fit into a preexisting facility). Generally, these constraints include what types of operands / inputs the system must handle, as well as the parameters and features that must be incorporated into the design of specific components in the system. This category also includes constraints on human and equipment safety, as well as acceptable impacts on the environment.

- **Design Restrictions and Limitations:** The categories above capture constraints which are externally imposed by the customer, by industry standardization, and by government regulation. Several constraints emerge, however, due to choices and tradeoffs made elsewhere in the design of the system, including acceptable factors of risk. For example, the choice to use a particular robot for an application may lead to limitations on where that robot can reach, and thereby restrictions as to where accessible stations need to be placed. In addition, vibrations induced by the robot may dictate requirements for vibration isolation of other components in the system. If a different robot or another type of mechanism is selected, such requirements may not be necessary.

- **Global Constraints:** These are the set of constraints which apply, to some greater or lesser degree, to every component in the system. Generally, these constraints will state overall goals, such as minimizing footprint or maximizing availability. While some of these constraints may have limits (e.g. do not exceed a footprint of 100 sq. ft.), the are usually more negotiable, and can be "massaged" as necessary to address other issues in the design. Typically, the better that these constraints can be achieved, the lower the ultimate cost of ownership will be for the customer.

Global constraints will also include so-called "external constraints", which include conforming to national and international laws, industry standards and safety regulations.

- **Project Constraints:** These are the set of constraints dictated by marketing and management which impact the effectiveness and timetable of the design effort. Thus, this category incorporates such items as the schedule for design reviews, project deadlines, specific test procedures, development budgets, and staffing and resource limitations.

These constraints can be summarized in a *constraint table*, which provides a unique index number for every constraint, the source (i.e. index number, marketing, and/or management) of that constraint, a description of the constraint, and a chart indicating which FRs at this level of the hierarchy this constraint impacts. An example template for the constraint table is shown in Table 4.

Table 4: Generic constraint table.

Constraints on FR.j.# Index: C.j-#				**Impacts FR.j.#**						
	Parent	**Name**	**Description**	**1**	**2**	**3**	**4**	**5**	**6**	**Ver**
			-- Critical Performance Specifications --							
1	Marketing	*Throughput*	Meet throughput specifications	✓	✓	✓	✓	✓	✓	T
2	Marketing	*Performance*	Meet process spec. (arbitrary # of instances)	?	?	?	?	?	?	T
3	Marketing	*Performance*	Meet transport spec. (arbitrary # of instances)	?	?	?	?	?	?	T
			-- Interface Constraints --							
4	Marketing	*Operand*	Handle customer specified operand (arbitrary # of instances)	?	?	?	?	?	?	I
5	Marketing	*Interface*	Handle customer specified interface (arbitrary # of instances)	?	?	?	?	?	?	I
			-- Design Restrictions & Limitations --							
6	Marketing	*Feature*	Include specified components & features (arbitrary # of instances)	?	?	?	?	?	?	I
7	Mgmt.	*Inventory*	Use specific components from inventory	?	?	?	?	?	?	I
			-- Global Constraints --							
8	Marketing	*Quality*	Maximize availability / reliability (maximize MTBF and MTTF, minimize MTTR)	✓	✓	✓	✓	✓	✓	I
9	Marketing	*Geometry*	Minimize size (footprint)	✓	✓	✓	✓	✓	✓	I
10	Marketing	*Integration*	Integrate tool with factory environment (host computer, air and water supply, facilities, etc.)	✓	✓	✓	✓	✓	✓	I
11	Marketing Mgmt	*Flexibility*	Provide maximum flexibility to accommodate individual customer needs	✓	✓	✓	✓	✓	✓	Dr
12	Marketing	*Cleanliness*	Maintain appropriate cleanliness rating (industry-dependent)	✓	✓	✓	✓	✓	✓	Dr
13	Marketing	*Geometry*	Minimize weight (industry-dependent)	✓	✓	✓	✓	✓	✓	I
14	Mgmt	*Maintenance*	Make serviceable (easy access to components)	✓	✓	✓	✓	✓	✓	I
15	Marketing	*Human*	Make tool "user-friendly" (ergonomics and software interfaces)	✓	✓	✓	✓	✓	✓	I
16	Marketing Mgmt	*Cost*	Minimize costs (design, manufacturing, operational, maintenance, etc.)	✓	✓	✓	✓	✓	✓	I
17	Mgmt	*Change*	Integrate maximum amount of existing technology (minimize redesign of proven components, use off-the-shelf equipment)	✓	✓	✓	✓	✓	✓	U
18	Marketing Mgmt	*Assembly* *Test*	Provide ease of testability (make components compatible with standard & customer-defined tests)	✓	✓	✓	✓	✓	✓	T I
19	Mgmt.	*Manufacture*	Optimize components for ease of manufacture and assembly (DFM, DFA)	✓	✓	✓	✓	✓	✓	Dr I
20	Marketing	*Standards*	Conform to industry and safety standards (arbitrary # of instances)	✓	✓	✓	✓	✓	✓	I D
			-- Project Constraints --							
21	Marketing	*Interface*	Use budget and resources	✓	✓	✓	✓	✓	✓	I
22	Mgmt	*Time*	Meet project development timetable	✓	✓	✓	✓	✓	✓	I
23	Marketing Mgmt	*Test standards*	Meet test standards	✓	✓	✓	✓	✓	✓	I

3 DIAGRAMS[4]

Once the decomposition is complete, it is useful to have graphical depictions of the design. These diagrams can offer clues into implementation issues, as well as highlighting specific areas of the design which may require further development and decomposition.

The first step is to generate *tree diagrams* that demonstrate the layout of the design hierarchy. These diagrams are used to identify the leaves (i.e. endpoints) of the design. Assuming that the number of FRs and the number of DPs are equal at each level of the hierarchy, the tree diagrams should reveal the same pattern for the FRs and the DPs. If the number of FRs and DPs are not equal, then this diagram may indicate that certain FRs are not being adequately addressed in the design. Examples of the tree diagrams are shown in Figure 1 and Figure 2.

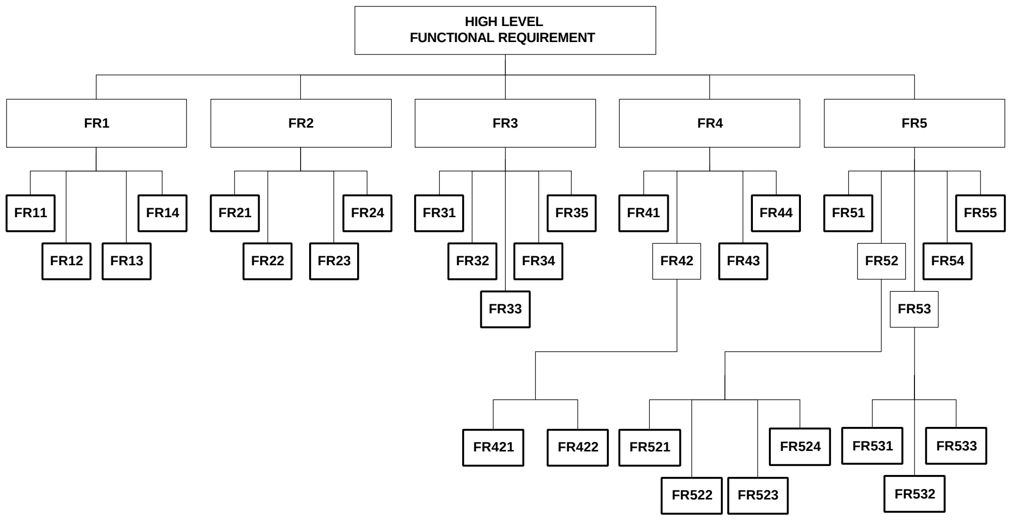

Figure 1: Tree diagram representing the functional requirements (FRs).

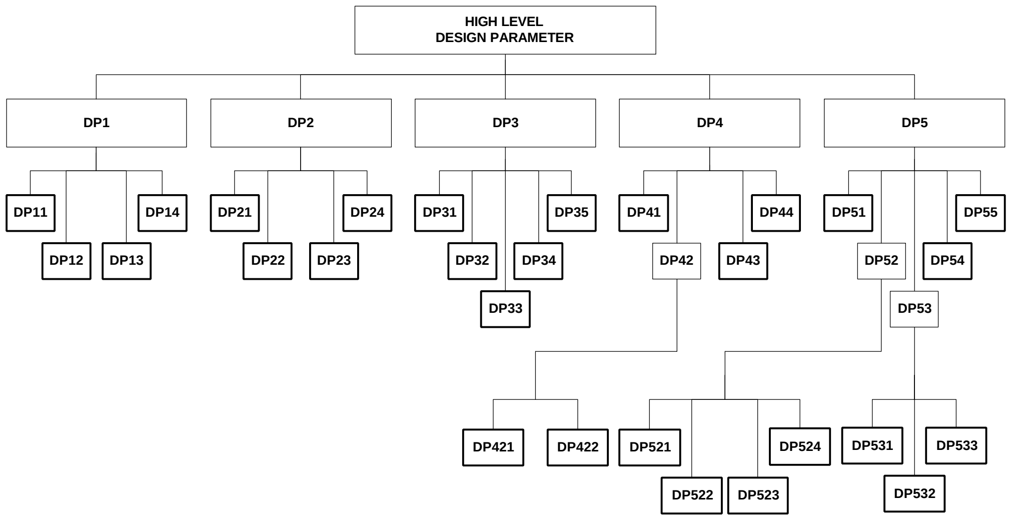

Figure 2: Tree diagram representing the design parameters.

[4] Parts of this section also appear in Hintersteiner, J. D. (1998) ***Axiomatic Design and Network Performance Analysis for Applications in Home-Based Health Care***. MIT Master's Thesis. February, 1998. Pgs. 93-96.

The leaves of the tree can be transformed into independent modules, which represent the non-zero diagonal elements of the design matrices relating the FRs to the DPs. These modules are then joined by junctions in a *module-junction structure diagram*, which represents the relationship between them (i.e. uncoupled, decoupled, or coupled). The module-junction structure diagram should roughly maintain the pattern of the tree diagram, as shown in Figure 3. There are three types of junctions which can appear in the module-junction structure diagram, as specified in Table 5.

Once the module-junction structure diagram is specified, it can be used to generate a *design flow diagram*, which shows how design information must flow through the system design process. Thus, it represents the order in which modules must be designed in order to satisfy the overall functional requirements of the system, as shown in . The determination of parallel, sequential, and feedback loops for the information flow is dictated by the junctions in the module-junction structure diagram, as shown in Table 5. This information flow is shown in Figure 4.

Table 5: Junction types and relationships for system architecture diagrams.

Symbol	Type	Relationship	Flow Diagram Representation
S	Summation	Uncoupled	Parallel summation of modules (order of design does not matter)
C	Control	Decoupled	Sequential processing of modules (order of design is critical)
F	Feedback	Coupled	Feedback loop of sequentially processed modules (design iteration required)

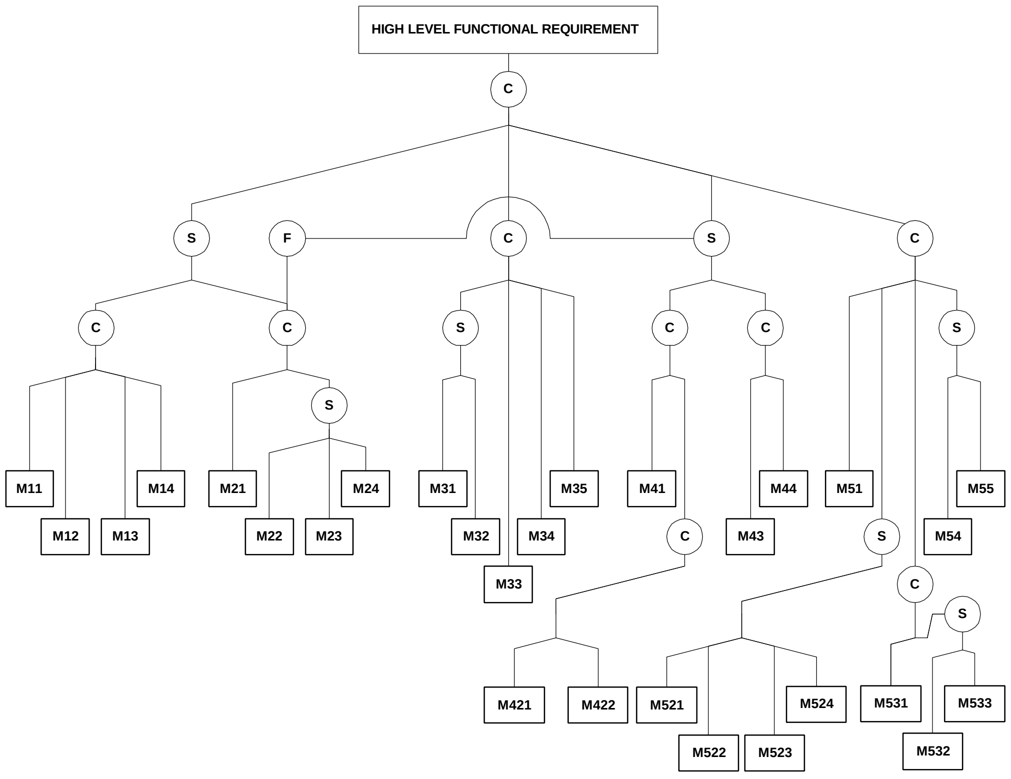

Figure 3: Module-junction structure diagram. Note that, in this example, the feedback (F) junction indicates coupling between FR.2 (M2) and FR.4 (M4). This will dictate a feedback loop in the design flow diagram.

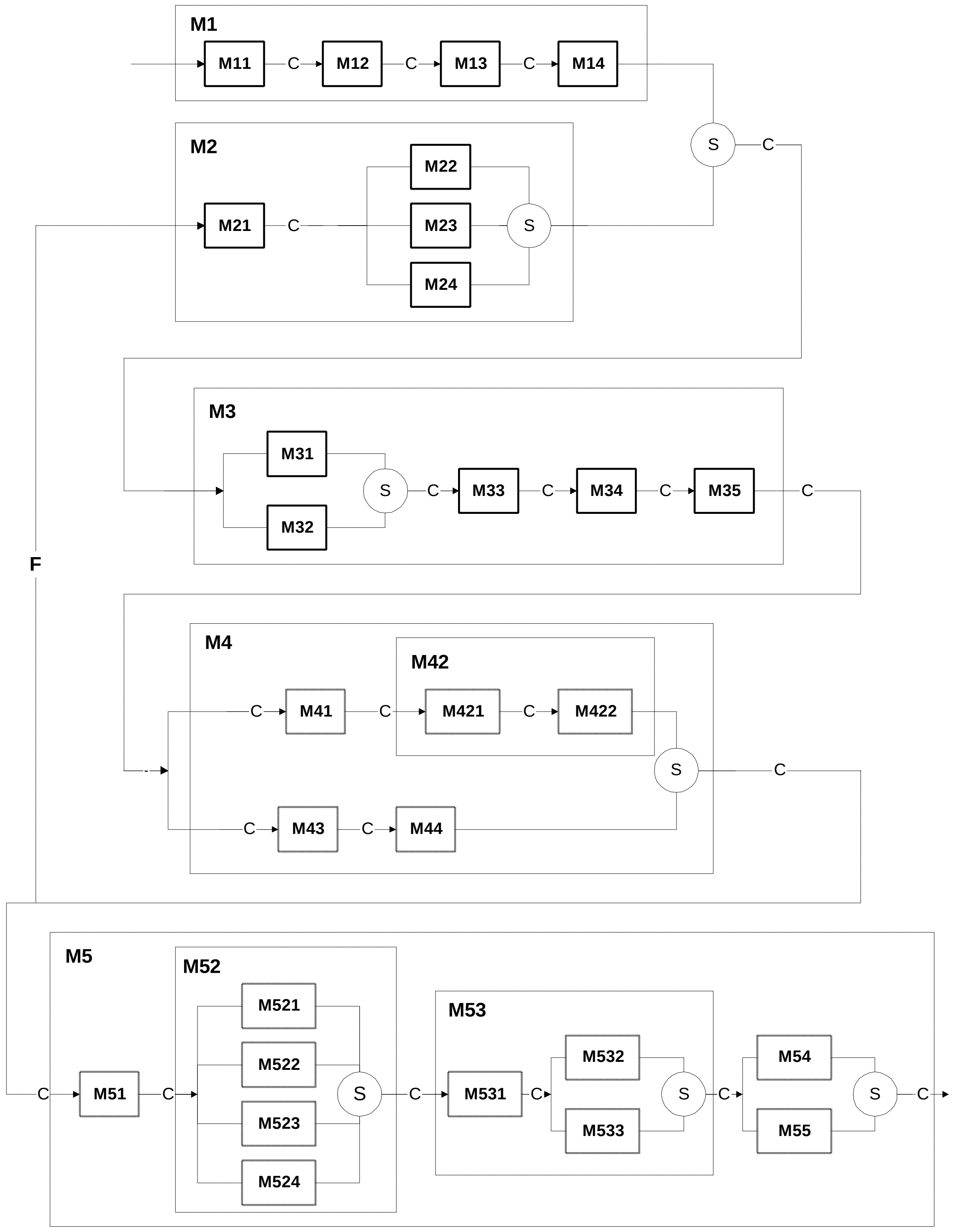

Figure 4: Design flow diagram. Note that the junctions in the module-junction structure diagram are referenced here, in conjunction with parallel (S), sequential (C), and feedback (F) information flow.

4 FURTHER REFERENCES

For more information about Axiomatic Design and its applications to systems, the reader is referred to the following published reference sources. Please also visit the web site for the MIT Axiomatic Design Group (http://axiom.mit.edu/) and the web site for Axiomatic Design Software, Inc. (http://www.axiomaticdesign.com/) for the latest information and publications on this topic.

[Hintersteiner, 1999] Hintersteiner, J. D. "A Fractal Representation for Systems." *Proceedings of the 1999 International CIRP Design Seminar*, Enschede, the Netherlands, March 24-26, 1999.

[Hintersteiner & Nain, 1999] Hintersteiner, J. D. and Nain, A. "Integrating Software into Systems: An Axiomatic Design Approach." *Proceedings of the 3rd International Conference on Engineering Design and Automation*, Vancouver, B. C. Canada. August 1-4, 1999.

[Hintersteiner & Tate, 1998] Hintersteiner, J. D. and Tate, D. "Command and Control in Axiomatic Design Theory: Its Role and Placement in the System Architecture." *Proceedings of the 2nd International Conference on Engineering Design and Automation*, Maui, HI. August 9 – 12, 1998.

[Suh, 1990] Suh, N. P. The Principles of Design. *Oxford University Press*, NY. 1990.

[Suh, 1999] Suh, N. P. Axiomatic Design: Advances and Applications. To be published by *Oxford University Press*, NY. 1999.

[Tate, 1999] Tate, D. "A Roadmap for Decomposition: Activities, Theories, and Tools for System Design." *Ph.D. Thesis*, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA. February, 1999.