Hardware, Software, Firmware Allocation of Functions in Systems Development

by

David D. Webster

Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Systems Engineering

APPROVED:

Scott F. Midkiff, Chairman

Donald R. Drew

September, 1987

Blacksburg, Virginia
The top-down development methodology is, for the most part, a well defined subject. There is, however, one area of top-down development that lacks structure and definition. The undefined topic is the hardware, software, and firmware allocation of functions. This research addresses this deficiency in top-down system development. The key objective is the restructuring of the hardware, software, and firmware process from a subjective, qualitative decision process to a structured, quantitative one.

Factors that affect the hardware, software, and firmware allocation process are identified. Qualitative data on the influence of the factors on the allocation process are systematized into quantitative information. This information is used to develop a model to provide a recommendation for implementing a function in hardware, software, or firmware. The model applies three analytical methods: 1) the analytic hierarchy process, 2) the general linear model, and 3) the second order regression technique. These three methods are applied to the quantified information of the hardware, software, firmware allocation process. A computer based software tool is developed by this research to aid in the evaluation of the hardware, software, and firmware allocation process. The software support tool assists in data collection. Future application of the support tool will enable the capture and documentation of expert knowledge on the hardware, software, and firmware allocation process. The improved knowledge base can be used to improve the model which in turn will improve the system development process, and resulting system.
Acknowledgements

I extend my appreciation to James Blackmon and Barry Corson for nominating me for the Warner Fellowship, to Rear Admiral Calvert and NEAI for selecting me for this Fellowship, and to Senator Warner and the State of Virginia for sponsoring the fellowship. I thank Dean Blanchard, Linda Scott, and Loretta Tickle for their advice and administrative aid.

I thank those experts who provided technical input: David Maddox, Tom Smith, Brian Koster, Lenny Burke, Joseph Cagnazzi, Marshal Greenspan, and of course Traci Massey.

My deepest respect and gratitude are extended to Dr. Scott F. Midkiff who provided the single most important influence and support throughout this research. It would not have been possible to accomplish this research within the aggressive time frame without his guidance and professionalism.

Finally, I thank those that helped and supported me through my research trauma: Susan Stoff from AT&T Bell Labs, the girl most likely to be on the telephone, Ken Goad the man with the code, Tom Strandberg the man of many languages, Jolyn, Ron, Jenny, Warren, Tina, Patrick, Vicky, Kathy, Cassey, Drew, and MOM & DAD. THANK YOU.
# Table of Contents

Chapter 1. Introduction ........................................................................ 1

1.1 Research Objective ........................................................................ 1

1.2 Research Motivation ....................................................................... 1

1.3 Exploratory Nature of Research .................................................. 2

1.4 Contribution of Research ............................................................ 3

1.5 Summary of Thesis ....................................................................... 3

Chapter 2. HW/SW/FW Allocation Background .................................. 5

2.1 Introduction .................................................................................... 5

2.2 Allocation in the Top-Down Design Process ................................ 6

  2.2.1 Complex System Development .............................................. 6

  2.2.2 Top-Down System Development ........................................... 8

  2.2.3 Allocation to Hardware, Software, and Firmware .................. 10

  2.2.4 Top-Down Development Example ....................................... 10

    2.2.4.1 System Requirements .................................................... 10

    2.2.4.2 Subsystem Partitioning ................................................. 11

    2.2.4.3 HW/SW/FW Allocation ............................................... 12
List of Illustrations

Figure 1. Top-down decomposition and bottom-up composition of a system. ................ 9
Figure 2. Example of Top-Down Development. .................................................. 14
Figure 3. Model Development Overview. .......................................................... 26
Figure 4. Data Collection Types [19]. .............................................................. 28
Figure 5. Reducing Qualitative Data to Quantitative Data [19]. ......................... 29
Figure 6. HW/SW/FW Allocation Model Building Specifics. .............................. 30
Figure 7. Feedback of Model Building. ............................................................ 35
Figure 8. AHP Level 0 and 1 Matrices. ............................................................ 38
Figure 9. AHP Stepwise Overview. .................................................................. 41
Figure 10. HW/SW/FW Allocation Process Decision Break Down. ..................... 42
Figure 11. HW/SW/FW Allocation Level 0 Completed Matrix. ............................ 43
Figure 12. HW/SW/FW Allocation Process Level 1 Matrices Example. ................. 44
Figure 13. AHP Synthesis for HW/SW/FW Allocation Process ......................... 45
Figure 14. HW/SW/FW Allocation Tool Main Menu. ......................................... 47
Figure 15. HW/SW/FW Allocation Tool AHP Analysis Menu Screen. ................. 48
Figure 16. HW/SW/FW Allocation Tool AHP Comparative Judgement Input Screen. .................................................................................................................. 49
Figure 17. HW/SW/FW Allocation Tool AHP Flow Chart: Level 0 Matrix .......... 50
Figure 18. HW/SW/FW Allocation Tool AHP Level 1 Matrices Input Screen. .......... 51
Figure 19. HW/SW/FW Allocation Tool AHP Flow Chart: Level 1 Matrix.......... 52
Figure 20. HW/SW/FW Allocation Tool AHP Output Screen. ............................ 53
Figure 21. GLM Factor Input Screen. ................................................................. 58
List of Illustrations
Chapter 1. Introduction

1.1 Research Objective

The objective of this thesis is to quantify, structure, and model the hardware/software/firmware (HW/SW/FW) allocation process of avionics system development. The research results include a tool to be used for data acquisition, data reduction, and model building. An initial baseline data collection, data reduction, and model building is performed.

1.2 Research Motivation

The motivation for this research is the need to quantify, structure, and document the current qualitative hardware/software/firmware allocation process of avionics system development. Quantifying the HW/SW/FW allocation process will improve the structure, quality, and consistency of the process. Structuring the quantified information into a model will assist systems engineers in the
HW/SW/FW allocation process that occurs in top-down avionics system developments. The model will also provide documentation of the HW/SW/FW allocation process to aid in the independent verification and validation (IV&V) of systems development. In addition, it will provide a historical record for future system maintenance and updates as well as aid in the development of new avionics systems. The recognized requirement for achieving these objectives are discussed in detail in Chapter 2, Section 6 "Prior Work".

1.3 Exploratory Nature of Research

There are two generally accepted types of research: 1) exploratory and 2) confirmatory [15]. Exploratory research covers investigation into relatively unknown and uncharted subject areas. Confirmatory research on the other hand is concerned with verifying and validating already quantified subject areas. This research is exploratory due to the current unstructured nature of the HW/SW/FW allocation process. To date, no research has been published which discusses the structuring of the HW/SW/FW allocation process. Much of the current decision process is subjective in nature, based on past experience, and intuitive judgement. Both of these decision methods are important, however, it is the main goal of this research to capture and document the current decision methods to assist in structuring the HW/SW/FW allocation process for future system development projects.

Each system engineer has judgements, as well as biases developed from personal experience. Developing a method to document and analyze these decision attributes is the first step, or the exploratory step, to structuring the HW/SW/FW allocation process. This exploratory research sets the foundation for data collection, data reduction, and model building.
1.4 Contribution of Research

This research is the first step towards structuring the present unstructured process of HW/SW/FW allocation. This research captures the factors that affect HW/SW/FW allocation. Categories are defined and the initial boundaries of the decision factors are determined. The research applies three analytical methods, based on other decision making research, to the HW/SW/FW allocation process. Two of these three analytical methods are implemented as a software package. The tool built under this research provides a vehicle for future data collection and data reduction. The third analytical method develops an experimental design to collect data for the development of a second order regression without confounding the main effects. A foundation is set by this research for creating a data base to improve the HW/SW/FW allocation process through adding structure to the process. In addition, the research develops a vehicle for documenting the HW/SW/FW decision process. This is important for several reasons: 1) it records the allocation of the functions, 2) it enables analysis of the allocation to aid in future development efforts, and 3) it provides information for system maintenance and updates.

1.5 Summary of Thesis

Chapter 2 provides the background for the HW/SW/FW allocation process. It discusses complex systems and top-down development methods. A survey of literature pertaining to the HW/SW/FW allocation process is presented. Chapter 2 also provides a detailed description of the HW, SW, and FW allocation process and develops the initial categories and associated factors which affect the HW/SW/FW allocation decision process. Chapter 3 discusses model development theory and applies the theory to the HW/SW/FW allocation process. Chapters 4, 5, and 6 describe the analytic methods used to build the model. These three chapters introduce the methods, apply them to the
HW/SW/FW allocation process and provide a walk-through of using the computer tool implementation of the analytic methods. Chapter 7 describes the tool input and output results of the model's analytical methods including: analysis of the results, and comparison of the results from the different methods. Chapter 8 summarizes this work and recommends future research.
Chapter 2. HW/SW/FW Allocation Background

2.1 Introduction

This chapter describes HW/SW/FW allocation in the context of the system development process and the factors that affect the allocation. The allocation of functions for hardware (HW), software (SW), and firmware (FW) implementation is a crucial part of the design process in the development of complex systems. The allocation sets the foundation requirements for the separate, but concurrent, HW, SW, and FW developments. The decision to implement a function in HW, SW, or FW has system cost, schedule, and performance ramifications. There are advantages and disadvantages inherent in the utilization of each HW, SW, and FW media and the allocation can determine if a design can meet its operational and contractual requirements. Despite the importance of effective allocation of system functions to HW, SW, or FW, this decision is usually based on intuition. This chapter examines the factors that influence the allocation decision and the trade-offs

---

1 This chapter is work originally reported in Midkiff and Webster [14].
realized through specific allocation. Additionally, this chapter lays a foundation for forming the allocation procedure into a structured knowledge-based decision process.

The chapter first describes the allocation process in the context of top-down development methodologies for complex systems such as those undertaken by the Department of Defense (DOD). Section 2.7 examines the properties of HW, SW, and FW and Section 2.8 describes the technical, programmatic, and environmental factors that influence the allocation process. Only by understanding the properties of the media, the attributes of a development project, and the interactions between the two, can a structured methodology be proposed to guide the allocation process.

2.2 Allocation in the Top-Down Design Process

The allocation of functions is one step in a top-down design methodology required for the development of complex systems. This section describes the top-down design process, the importance of HW, SW, and FW allocation, and prior research.

2.2.1 Complex System Development

The DOD embarks on many projects each year to develop complex systems that perform specialized functions in critical scenarios. Avionics such as radar, display processors, and electronic warfare systems are examples of this type of development project. The performance and development constraints place stringent requirements on designs. The systems are used for real-time applications and therefore must meet certain performance criteria. The systems utilize extensive high-speed input and output between components within a system and between different systems. Since the systems are often mission-critical and used in aircraft, there are also reliability, weight, size, and
power constraints. Policy and standardization requirements that enable communication between systems and decrease life-cycle costs are also imposed on the design.

There are two fundamental types of requirements placed on a complex system development project: operational and contractual. The operational requirements define the capability, performance, and effectiveness of the system. The capability is what the system can do, the performance is how well the system accomplishes the capability, and the effectiveness is the probability that the performance and capability standards are met on a given mission. As an illustration, a navigational system may be required to have the capability to navigate from point X to point Y, with a performance of less than 1 mile of error every 1000 miles. The effectiveness required may be that the system can navigate from X to Y within the acceptable error limit 99.5 percent of the time.

Contractual requirements consider policy attributes such as standardization, schedule, and methodology. Standardization is a concept driven by the goal to decrease total life-cycle cost. The standardization policy can impose harsh technical requirements and constraints on the development and can impact performance and capability. For example, a requirement to use standard embedded processors in a system restricts the optimization of the processor hardware for a particular task. One cannot both optimize performance and standardize the implementation of a set of systems with different types of tasks. The contractual requirements may also set specific schedules, therefore limiting the available technical alternatives. The schedule usually necessitates concurrent HW, SW, and FW development. Successful concurrent development requires discipline and the use of structured design methodologies. Methodology requirements can dictate the development practices that enforce structured system design, including a requirement for top-down development and traceability.
2.2.2 Top-Down System Development

A project to develop a complex system begins with specifications including operational and contractual requirements. The goal of top-down development is to refine and decompose the specifications into manageable pieces for development. The separate pieces can then be integrated from bottom-up to form a unified system. The top-down decomposition and bottom-up composition development process is depicted in Figure 1.

The first step is to convert the operational requirements into functions that achieve the necessary capability, performance, and effectiveness. This functional decomposition of the top-level requirements is a difficult and critical activity in system development since it is the foundation for the entire design. A typical and important contractual requirement of the decomposition process is that it provide documented traceability. Specifically, each function should be related to an operational or contractual requirement. The function-to-requirement relation must be explicitly defined and maintained at all levels of decomposition. The overall system function definition must describe the interaction between the different functions. Dependencies between functions can be total, partial, or infrequent. Data flow diagrams (DFD) define the interplay, information transfer, and dependencies of the functional entities. The use of DFD's is sometimes a contractual requirement. The DFD, or the current functional decomposition in general, can be further decomposed as necessary.

The functions are grouped into subsystems in accordance with contractual specifications. Each subsystem is viewed as a single entity. The information transfer between subsystems must be identified from the function decomposition. Each subsystem consists of a list of decomposed functions and the corresponding input and output with other subsystems. All operational and contractual requirements are implemented by the set of subsystem specifications and their interactions. Subsystem functions can be decomposed into lower level functions.
It is at the point of subsystem decomposition that functions are allocated to HW, SW, or FW. Each subsystem function (sub-function) must be assigned to HW, SW, or FW implementation.

The sub-function requirements follow directly from the higher level requirements and state what the sub-function must do. A design document then specifies how the requirements will be achieved. The sub-function implementation is specified during the detailed design phase, for example by a program design language or hardware description language. After implementation the sub-function must be tested and integrated with other sub-functions. Finally, the subsystems must be integrated and system test must be performed.
2.2.3 Allocation to Hardware, Software, and Firmware

The near impossible task of developing systems that meet DOD requirements is accomplished by using state-of-the-art HW, SW, and FW components. These components must be optimized, so that when integrated the system will meet its requirements. The achievable performance is usually close to the minimum acceptable required performance. The HW, SW, and FW interfaces must be identified and designed to optimize performance. The ability to optimize the interplay is possible only if careful attention is given to task requirement analysis and the system function allocation between the HW, SW, and FW.

2.2.4 Top-Down Development Example

An example of top-down development for a aircraft navigation requirement is presented in this section. The example develops a requirement and decomposes it into HW/SW/FW components.

2.2.4.1 System Requirements

The systems requirement for the example includes both operational requirements and contractual requirements.

The operational requirements include the 1) capability, 2) performance, and 3) effectiveness of the end system. The capability requirement is to navigate from point X to point Y. The performance is required to have a course error of less than one mile for every one thousand miles or 99.9%. The capability and performance is required to be achievable with 99.5% effectiveness. In other words the system should be capable to navigate from point X to point Y with a course error of less than 0.1% for one thousand miles 99.5% of the time.
Contractual requirements include 1) policy, 2) standards, 3) development, and 4) documentation specifications. Policy dictates that the navigation information be available on all displays. The Navy Standard Inertial Navigation System is required to be used for standardization purposes. The top-down development methodology is required to be used in the design process. Full DOD-STD-2167 documentation is required on software and firmware designs and level three drawings are required on all hardware designs.

These are the requirements of the system. The requirements must now be decomposed and partitioned into subsystems in accordance with the top-down methodology requirement. The decomposition must not violate any of the contractual requirements.

2.2.4.2 Subsystem Partitioning

The requirements are parcelled-out to subsystem components to reduce the size and complexity of the development components. The system is broken down into more manageable elements. The subsystems of the navigation example are 1) the inertial navigation subsystem, 2) the mission computer subsystem, 3) the flight control subsystem, and 4) the display subsystem.

The inertial navigation subsystem is responsible for providing XYZ location coordinates. The coordinates are relative to the NAVSTAR Global Navigation System 1, 2, or 3. The coordinates must be updated every 0.1 seconds in order to meet performance requirements.

The mission computer is required to store and maintain the “waypoints” to be traveled. It is also required to note deviance from the course and notify the other subsystems of course adjustments. The mission computer monitors and coordinates the other subsystems.
The flight control system is responsible for the aircraft control surfaces. This is basically the inflight adjustment, but also covers degradation reallocation to backup systems. The flight control subsystem also measures the wind speed using its external aircraft sensors.

The display subsystem is the human interface subsystem. It needs to display a map to the operator(s). The aircraft position information must also be provided to the operator(s). Finally, the projected and intended flight paths are presented.

All subsystems have contractual requirements applied to them such as schedule constraints, volume and weight constraints, power constraints, off-the-shelf component requirements, and documentation requirements.

2.2.4.3 HW/SW/FW Allocation

The display subsystem is chosen for further breakdown and allocation to hardware, software, and firmware implementation. The display requirements or tasks are decomposed further into 1) maintain display formats, 2) maintain equipment, and 3) manage displays. The managing of the displays is decomposed into configure displays, manage display information, and display information. There are several types of display information capability functions to be allocated to hardware, software, and firmware. These functions are listed here with the corresponding HW, SW, or FW implementation.

1. Display Text - software
2. Display graphic information - software
3. Generate Graphics - firmware
4. Select Video Recorder - hardware
5. Combine Video and Graphics - hardware

Thus the allocation of functions is made to hardware, software and firmware.
2.2.4.4 Example Summary

An example of the step-by-step process of top-down development for an aircraft navigation system was provided. The example showed how the system requirements are decomposed into several layers of lower level elements. The decomposition was discussed to the point of HW/SW/FW allocation in systems development. decomposition in the subsystem leading to a functional level for HW/SW/FW allocation. The example of the top-down process described in this section is depicted in Figure 2.

2.2.5 Prior Work

Top-down development and total system design methodologies are widely accepted and practiced techniques for complex system developments typical of DOD projects.

DOD complex system attributes and their development characteristics are described by Alexandridis [1]. He emphasizes that DOD mission critical systems require integration of hardware and software to enable meeting requirements. Six general requirements are specified: 1) real time throughput, 2) increased speed and parallel processing, 3) adaptability, 4) reuseability, 5) hardware/software synergy, and 6) integrated support methods. Alexandridis states the need “to determine which functions should be implemented in software and which in hardware to obtain maximum system capability, modularity, and adaptability, along with minimum system development, maintenance, and enhancement costs.”

Many of the complex systems are embedded in larger systems. Wagreich [23] discusses embedded system characteristics and support tool requirements. These characteristics include: stringent performance requirements, software development on a host different than the target machine, untested target hardware, time critical interfaces, and late technology choices. The life cycle considerations
Figure 2. Example of Top-Down Development.
for these systems are also described. The life cycle factors include: strict system response times, high tolerance to external errors, graceful degradation, physical constraints, development "time" budgets, traceability requirement, and hardware design schedule slips affecting the software schedule. The complexity of the system characteristics and the life cycle considerations can be aided by a support tool environment. Tools required include: representation tools, analytic and traceability tools, documentation tools and configuration management tools. Wagriech states a specific requirement to link the hardware and software support environments. A recommendation is made for a decision support system "to give advice to the user on how to proceed with development."

Roman [16] defines three activities of the design process, these include identification of need, solution development, and design implementation. A total system design (TSD) focus is stressed. TSD contains six stages: 1) problem definition, 2) system development, 3) software design, 4) machine design, 5) circuit design, and 6) firmware design. Roman emphasizes the fact that requirements must lead to the design specification and vice versa. This an important and necessary concept. Although not stated by Roman, documenting the hardware/software/firmware allocation process is a crucial part of this requirement to track design. Roman does state that, "The growing area of requirements specification still needs a broader formal foundation, more automation, new development methods, and a higher level of integration into the overall design process." That is the integration of the hardware, software and firmware developments which begins at the allocation.

In a separate paper, Roman [17] discusses the "total system" as the integrated hardware and software implementation. The stages and phases of the TSD are discussed. A few hardware/software trade-offs are described. Roman states that "The problem embodied in hardware-software trade-offs is that the system functions need to be allocated between hardware and software components in a way that satisfies all system constraints." The paper mentions some of the complexity involved in the hardware-software partitioning process, including: new design versus off-the-shelf components, risk, development costs, maintenance costs, and support systems. Two tools to support the software development phases, SREM and SARA, are discussed [2].
Turn [22] reinforces the TSD process. He states that, "Acquisition of software and hardware separately with the hope of integrating them later in complex systems does not work in complex systems." Turn describes a management approach to the development of complex systems. A warning is provided against developing hardware and software in a vacuum. Turn believes that there exists a requirement to consider the system as an integrated, interactive hardware/software system during development. A major supporting factor is the risk of specification changes in development caused by design constraints. This exists partially because physics defines the hardware boundaries for performance, however, software performance boundaries are unclear. The paper outlines the computer system development life cycle, but the life cycle depicted does not contain the hardware/software/firmware allocation. However, contained in the list of design review questions is "Have relevant hardware-software trade-offs been adequately performed?". This implies that the allocation of functions to hardware-software is recognized as an important decision process but that they were unclear on how to manage it.

Hardware and software partitioning is discussed by Loy [13] and a justification for a process support tool is made. Loy emphasizes that the system partitioning phase is often approached in a "haphazard" manner. He explains that the first step is to develop a logical model that depicts function requirements independent of hardware or software implementation. From this step trade-offs need to be considered and "a system configuration that clearly maps the software components onto the hardware environment, and highlights the interfaces between all system components" needs to be developed. Loy describes the software partitioning process using several different methods. The trade-offs for configuration are generalized to fit into four general categories: 1) technology, 2) implementation resources, 3) policy and management, and 4) processing needs. For each general category a list of factors is presented including: processor speeds, specialized hardware, staff attributes, support tools, scheduling, subsystem interfaces, and time constraints. Loy's concluding remarks include: "the need for a system partitioning methodology to bridge the critical gap between analysis and design." and "we have requirements analysis and software design tools but we are lacking in system partitioning tools."
The Defense Management College, System Engineering, Management Guide [5] guide provides instruction on the top-down design process. The process consists of function analysis and breakdown from system to subsystem to components to hardware, software, and firmware. Although the necessity to perform the breakdown is described nothing is said as to how one partitions functions to hardware, software, and firmware. It is concluded that a gap exists in the top-down design/partitioning process.

Work on top-down development has focused on the procedures followed and the decisions made during design, but has largely ignored the task of providing structured methodologies to guide decision making related to allocation of functions to HW, SW, and FW. Design and analysis tools, such as SREM and SARA exist that allow a developer to determine if a particular decomposition and function allocation will, with some degree of certainty, meet the capability and performance requirements of the system. What is needed, however, is a methodology and a tool that bridges the gap between analysis and design. The tool would provide a decision support system to advise the system developer, provide documentation, and provide traceability of requirements.

This research address the gap between design and analysis and the need for a tool and a methodology to support the allocation of functions to HW, SW, or FW in top-down design. The remainder of the chapter examines the properties of HW, SW, and FW, and the attributes of a development project that impact the allocation decision.

2.3 Properties of Hardware, Software, and Firmware

The properties of hardware, software, and firmware are obviously critical to the allocation process. The properties of each are discussed with respect to the HW/SW/FW allocation process.
Functions implemented in *hardware* are implemented using physical or tangible circuits or devices. The hardware can be, and usually is, controlled by software or firmware logic. Hardware is inflexible to adaptation. Modifying the function requires changing the physical circuits or devices.

Functions implemented in *software* are programs composed of high-level language or machine instructions and data definitions that direct the hardware to perform computations or control functions. The machine readable representation of the program is executed from read/write memory. Software is adaptable by changing the instructions in the program. Software can be reprogrammed in a system without physically altering the system. The function changes can be effected at the operational level of maintenance.

Function implementation in *firmware* is similar to implementation in software except that the program representation resides in some form of read-only-memory (ROM). Often the program in firmware is developed using machine-level instructions that allow the simultaneous control of multiple hardware units. This provides execution speed at the cost of development complexity. Firmware cannot be altered at the operational level of maintenance, however a physical change to the system, beyond possibly replacing ROM, is not required as in hardware.

These definitions resolve the "gray" areas that exist between hardware, software, and firmware. Hardware is differentiated from both software and firmware by its physical attribute. Software and firmware are distinguished by the support level at which modification can be made to the logic. Firmware requires opening the system container and pulling the cards with firmware components for modification using electric or ultra-violet erasing devices. For software the programs can be changed by reprogramming without taking the system apart.
2.4 Factors Affecting Allocation

The factors that affect hardware, software, and firmware can be derived from the operation and contractual requirements of the system and from the development environment. There are three categories of factors: technical, programmatic, and environmental. Technical attributes describe the requirements of the system or end-product being developed. Technical factors typically follow from operational requirements and include constraints such as speed and reliability. Programmatic factors represent attributes of the development process rather than of the end-product. Constraints imposed by contractual requirements such as development cost, schedule, and standards are programmatic issues. Environmental factors describe the environment or conditions under which the development is being performed and include factors such as facility availability and skill distribution among available personnel.

2.4.1 Technical Factors

Technical factors define the system capability, performance and effectiveness. Descriptions of technical factors that affect HW, SW, and FW allocation are provided below. It is important to note that the factors pertain to specific functions rather than the system. For example, a fast system could utilize certain slow functions.

- **Speed/Throughput**: The speed at which the function must execute or the amount of information that the function must process in a given period of time. Typically direct HW implementation provides the fastest operating speed and SW the slowest.
• **Adaptability:** The required ease with which the function can be modified. Modifications may be unplanned requirements changes or preplanned product improvements. SW is the most adaptable medium, while HW is the least adaptable.

• **Accuracy:** The precision or acceptable deviation required of the function.

• **Reliability:** The period of time during which the function must operate correctly, i.e. provide the required capability and performance.

• **Maintainability:** The ease with which the function can be kept in operation. Maintenance is usually simpler for a standardized processor executing SW or FW than for a specialized HW unit.

• **Testability:** The ease with which the capability and performance of the function can be tested.

• **Interfaces:** The quantity and complexity of the interactions required with other functions.

• **Compatibility:** Requirements that the system be compatible with previous systems. These requirements may dictate the use of a particular type of implementation or it may impose implicit capability or performance requirements.

• **Technology Age:** The confidence of successful implementation based on prior experience with the same or similar function.

• **Safety:** The impact of the function on mission-critical or life-sustaining operations of the system.

• **Size/Weight:** Constraints on the physical size, volume, or weight of the components that implement the function.
• **Power:** Constraints on the power consumed by the components that implement the function.

Each factor is qualitatively related to the choice of selecting to implement a function in hardware, software, or firmware. For instance high speed requirements push implementation towards hardware. High adaptability requirements tend to emphasize software design. A moderate to high speed requirement and a relative adaptability forecast leads one to develop a function in firmware that optimizes the speed available through firmware.

### 2.4.2 Programmatic Factors

Programmatic factors represent contractual requirements and overall system development concerns. Although most programmatic factors relate to the system as a whole, the top-down development process assigns requirements to individual functions so as to meet contractual requirements. The following list describes programmatic factors that may affect the allocation of functions to HW, SW, or FW.

- **Schedule:** The time allotted for function design, implementation, and test. The important consideration is the aggressiveness of the schedule for a given function rather than the absolute length of time.

- **Development Cost:** The expected cost of the development.

- **Life-Cycle Cost:** Limits on the estimated cost of the system throughout its life-cycle.

- **Risk:** The acceptable probability that the function will not meet requirements on time and within budget.
• **Documentation**: The type and quantity of the documentation required on the function design effort.

• **Development Practices**: Methodologies and practices dictated for the development. HW, SW, and FW developments may be required to adhere to certain common practices at the higher levels of design, but differing methodologies at lower levels. The requirements may be general, for example to use top-down design, or specific, for example to use a specific hardware description language such as the DOD's VHDL.

• **Standard Technologies/Languages**: Requirements to use a specific hardware technology or programming language. For example, SW may need to be written in a particular language such as the DOD's ADA.

• **Standard Components**: Requirements to use standard or “off-the-shelf” components. Standard HW components may include processors, such as the AYK-14 Navy standard computer, and standard SW may include library programs.

• **Testing**: The types and completeness of tests that must be passed to fulfill contractual agreements.

• **Security Classification**: Security classification required of the function. SW and FW are the easiest to duplicate, while HW and FW are non-volatile.

Similar to the technical factors, qualitative data is available on the relative importance of the factors to the HW/SW/FW allocation process. The relationship of the factors can be determined through subjective questions. For instance, “How would a tight, moderate, or generous schedule affect the HW/SW/FW allocation process?” This type of question aids in collecting qualitative data on the factors.
2.4.3 Environmental Factors

Environmental factors are the supporting resources for design and implementation. For instance, it may be technically and programmatically desirable to implement a function in HW, but facilities may not be available to support the HW development. Definitions of environmental factors are listed below.

- **Personnel Demographics**: The specialization, experience, cost, and size of the available work force. Clearly a circuit designer and a programmer are not interchangeable. It may also be that the cost of employing a SW designer is not the same as that of employing a FW designer, for example.

- **Support Tools/Equipment**: Tools and equipment available to support development. The tools may include computer-aided design and computer-aided software engineering packages and hardware and software development systems.

- **Facility Availability and Cost**: The availability and cost of facilities required for implementation. For example, hardware development may require the timely availability of integrated circuit fabrication facilities.

- **Organizational Structure**: The structure of the organization doing development. For example, if the development organization develops software internally, but an external support group designs any hardware, the risk, development time, and development cost associated with hardware implementation may be greater than that associated with software implementation.

Determination of the environmental factor relations to the HW/SW/FW allocation process can be made through subjective evaluation. For example the support tools factor is an evaluation of the corporate or project assets. Specifically, it is an evaluation of the quality of the hardware computer
aided design or CAD tools, the quality of the software computer aided software engineering or CASE tools and the quality of firmware development tools.

2.5 Summary

This chapter set the foundation to discuss the hardware, software, and firmware allocation process. A description of complex systems for which the HW/SW/FW allocation process is applicable was provided. The top-down design methodology for complex system development which entails HW/SW/FW allocation was discussed and an example was provided. A survey of prior work was given to place this research in context with other work on similar subjects as well as to provide justification of the importance of structuring the HW/SW/FW allocation. The definitions of hardware, software, and firmware were presented to alleviate any discrepancies in definition due to personal experiences. Finally, the three categories used to collect factors that affect the HW/SW/FW allocation process were discussed and the initial list of factors and their descriptions were provided. Focus is now switched to model development.
Chapter 3. Model Development

3.1 Introduction

This chapter discusses model development. Initially a general process model development is presented. The general process is then applied to develop a model for the HW/SW/fw allocation process.

3.2 Theory

Theory on data collection, data reduction, and model building is presented in this section. The discussion is a general overview and survey of key topics. Figure 3 provides an overview of the model building process.
3.2.1 Data Collection

Decision model building requires data collection on the subject of the decision process. Data is necessary to obtain an understanding of the characteristics, attributes, factors, and criterion of the decision process. Data collection methods are categorized by Kosecoff and Fink into five types [11]. The five types are described below.

1. **Interview.** Interactive discussions that are held face-to-face or over the telephone.

2. **Questionnaire.** Non-interactive data collection which applies a series of written questions that can be in a variety of response formats, such as multiple choice, open-ended answers, rating and ranking scales, and semantic differentials.

3. **Observation.** An analysis of the system or situation as it currently exists.

4. **Document Analysis.** Evaluation on documents on the process such as system development reports, engineers notes, informal meeting and program review minutes as well as published literature.
5. Critical Incident Review. Analysis of the "out of the ordinary" event or situation that is unexpected.

There are advantages and disadvantages with each of the data collection methods. These trade-offs among the different data collection types are discussed in detail in Kosecoff and Fink [11] and are conveniently summarized by Singhal [19] in Figure 4.

3.2.2 Data Reduction

Data reduction is used to consolidate the initial data collection and aid in structuring future data collection. Initial data collection is broad in scope and general in nature. This data needs to be systematized. Meyers [15] discusses two methods to systematize data: 1) categorization through development of rules and 2) quantification through assignment of numeric values or weights to the categories and their elements. The flow process from impressionistic and descriptive data to quantitative data is summarized by Singhal [19] in Figure 5. The path from qualitative data to quantitative data requires the collection of data. The data collection aids in the validation of results. There is a fine line between data collection and data reduction.

3.2.3 Model Building

The stages of data reduction can be developed into a model when the data and information mature. After data has been categorized into systematic data, it is quantified using scales, ratings, and weights. A model is formed to structure the qualitative data to quantified information path. Kurstedt [12] has developed a nine-step method for qualitative modeling adapted from Glaser's [8] constant comparative method. The nine-steps are summarized below.

1. Brainstorm and list all possible influencing factors and interdependences.
<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interview</td>
<td>Permits in-depth probing.</td>
<td>Expensive and time-consuming.</td>
</tr>
<tr>
<td></td>
<td>Permits discussion of sensitive issues.</td>
<td>Inter- and intra-rater reliability may be difficult to obtain.</td>
</tr>
<tr>
<td>Face-to-face</td>
<td>Good for eliciting new ideas.</td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questionnaire</td>
<td>Anonymity encourages truthful response to embarrassing questions.</td>
<td>May need follow-up to obtain adequate number of responses.</td>
</tr>
<tr>
<td></td>
<td>Can be administered to large groups at low cost.</td>
<td>May require respondents to draw distinctions that don't exist.</td>
</tr>
<tr>
<td></td>
<td>Reduces judgmental data into a manageable form.</td>
<td>May produce no response if poorly designed (complex instructions; lengthy; ambiguous).</td>
</tr>
<tr>
<td></td>
<td>Produces objectified data.</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>Permits first hand observation of events.</td>
<td>Observer can change the environment.</td>
</tr>
<tr>
<td></td>
<td>Skilled observers can obtain insightful facts.</td>
<td>Inter- and intra-observer reliability may be difficult to obtain.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Analysis</td>
<td>Unobtrusive.</td>
<td>Documents may be disorganized, unavailable, or too voluminous.</td>
</tr>
<tr>
<td></td>
<td>Can be inexpensive.</td>
<td>Only formal communications are likely to be recorded.</td>
</tr>
<tr>
<td></td>
<td>No new data collection required.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Critical Incident</td>
<td>Exposes facets which are otherwise not obvious.</td>
<td>Can be difficult to interpret.</td>
</tr>
<tr>
<td></td>
<td>Focuses only on particularly important or revealing events</td>
<td>Insiders may not notice situations which appear very unusual to an outsider.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Data Collection Types [19].

2. Separate the factors of step 1 into categories through the application of rules and defining properties.

3. Define the boundaries of the categories. This is accomplished by comparing the factors within the categories and across categories.

4. Consolidate the category properties.

5. Evaluate the categories by using new items.

6. Combine categories into higher level divisions.

7. Iterate and reiterate to refine the model.

8. Develop the theory that relates the categories.

9. Test and validate the model through experimentation and data collection.
3.3 Application

This section addresses how the model development theory is applied to developing a model for the HW/SW/FW allocation process. The data collection, data reduction, and model building are discussed. In actuality these are not separate and distinct phases. Rather they are tightly integrated elements. Data collection is used for data reduction, data reduction is used for model building, model building and data reduction are used for data collection, etc. Figure 6. presents a diagram for model building for the HW/SW/FW allocation process.
3.3.1 Data Collection

All five data collection methods were considered for the HW/SW/FW allocation process data collection. Some of the methods were determined to be impractical for this research but are suggested for future research on this subject. Other methods were useable directly or in a modified form. Each method is briefly analyzed with respect to this research on the HW/SW/FW allocation process. The methods used by this research are discussed in detail.

Interview

The interview was used both for preliminary and specific in-depth data collection. The interviews for preliminary data collection were of a "brainstorming" nature. The effort was to identify as many factors affecting the HW/SW/FW allocation process as possible. The interviews also help to categorize and determine the more important factors. The preliminary interviews laid a foundation for further data collection.

Interviews were used in the latter stages of model development to collect data. These interviews were pin-pointed to get information for the model. The data collection in these interviews is con-
cise, dealing with specific scales and weighting. In fact, the interview is used to walk-through a questionnaire that was developed.

Questionnaire

A questionnaire was developed to collect information on fifteen factors that were reduced from analyzing the factors collected by interviewing and brainstorming. The questionnaire is provided in Appendix A. For each factor, the questionnaire requests that the probability of implementing a function in HW, SW, and FW be determined for each of three levels (high, moderate, and low) of the fifteen factors. A 1 to 5 weighting scheme was employed where:

- 1 implies almost certain or 95% probability
- 2 implies more than likely or 75% probability
- 3 implies uncertain or 50% probability
- 4 implies more than likely not or 35% probability
- 5 implies almost certainly not or 10% probability

The questionnaire also requests a rating of importance be given to each factor. The importance ratings are not important, moderately important, and very important with respect to the HW/SW/FW allocation process. A request was also made to rank the fifteen factors from 1 to 15 without repeating any of the numbers. The questionnaire participants were also asked to provide comment on the factors as well as provide any additional factors. The questionnaire was sent to major avionics developers and used for interview guidance.

Observation

Observation was not used as a data collection method for this research. It would require monitoring the HW/SW/FW allocation process during several system development projects. The tool developed will facilitate future data collection through observation. The HW/SW/FW allocation tool developed by this research provides a method for documenting allocation decisions over time.
Document Analysis

Data collection was performed using limited document analysis. Specifically, published literature was used. However, little data is available on the HW/SW/FW allocation process. "In fact one would think that hardware, software, and firmware were mutually exclusive, unrelated elements."[24]. System development reports, engineers notes, and meeting minutes were unavailable. A system engineer applying this research would have access to this information and should use it to improve the accuracy and quality of the model.

Critical Incident Review

Critical incident review is not applicable to the HW/SW/FW allocation process since it is not a day-to-day business transaction. An exception would be if something failed to perform adequately because it was developed in HW, SW, or FW due to a misrated factor.

3.3.2 Data Reduction

Data reduction was first applied to narrow the HW/SW/FW allocation factors that were collected from brainstorming, interviews, and document analysis. Meyer's [15] methods of categorizing and quantifying were applied to the reduced set of factors. Singhal's [19] path from qualitative to quantitative data presented in Figure 5 was applied to the collected data. Initial impressionistic data from interviews, research, and informal conversations were reduced by categorization through analytic induction into systematic data. The questionnaire and follow-up interviews applied rating scales, ranks, and weights to the systematic data to quantify it. In this manner the data on the HW/SW/FW allocation process progressed from impressionistic data to systematic data, to quantified data.

Data reduction using statistical packages can provide analysis of variance information and regression analysis. The statistical validity of the data reduction using the software data package can be in-
creased through experimentally designed data collection. Chapter 6 provides a discussion of experimentally designed data collection.

3.3.3 Model Building

The HW/SW/FW model includes a systems engineer, factors and factor attributes, functions and function factor characteristics, and some analytical techniques. The inputs to the model are contained in the expert judgement of the systems engineer. The factors that affect the HW, SW, and FW allocation are rated on the implementation of a function in HW, SW, and FW for three levels of the factor: high, moderate, and low. A very important, moderately important, or not important rating is given to each factor by the systems engineer. The n factors are ranked from 1 to n on the relative importance of the factor to the HW/SW/FW allocation process. The system expert also provides the strength of each factor for each function.

This research structures the inputs to the model and manipulates the data by applying analytical methods to provide HW/SW/FW implementation recommendations. The structuring of the data input systematizes the information on the factors affecting the HW/SW/FW allocation and the functions to be allocated. Two analytical techniques are applied to the structured data to model the HW/SW/FW allocation process. The two analytic techniques are the analytic hierarchy process (AHP) [18] and the general linear model (GLM) [20]. The analytic methods manipulate the information to provide a recommendation of HW, SW, or FW implementation. The decision process is modeled using these two methods. The AHP is an analytic method based on pairwise comparative judgements of the factors affecting a decision and the possible decision outcomes. The GLM is a linear regression analysis technique. Weights are applied to the decision factors and summed in accordance with function characteristics to obtain a composite indication for HW, SW, and FW implementation. The two methods are presented and described in detail in Chapters 4 and 5. In this manner the model enables consistent, structured data collection and data reduction.
Inherent in model building is a feedback process involving the system engineer, analytical methods application, model recommendations, function implementation, and analysis. The model weights are fine tuned through expert interaction until the model represents the project and organization attributes. In this manner expert knowledge is captured by the model. As more data and information is collected the knowledge base is increased and the model can be improved. Figure 7 depicts the continuous flow feedback process. Model iteration and revision is a crucial element of model building and improvement.

3.4. Model Implementation

The HW/SW/FW allocation model is implemented as a computer based tool. The IBM-PC is the chosen machine because of its widespread use. This attribute of the IBM-PC makes the software package transportable to many companies and academic institutions. Turbo Pascal [21] is the implementation language of the support software package. The HW/SW/FW allocation model is a procedural process. The procedural characteristic makes it suitable for Turbo Pascal which is a powerful, user friendly, IBM-PC based software language. Symbolic processing and expert system/inference languages are less suited to the HW/SW/FW allocation process tool.

The tool is designed as a menu driven program. The menus provide helpful instructions on using the tool. It is possible to go back and forth between menus and submenus. However, once a procedure has been started reversal is difficult and requires exiting the procedure and reentry. Procedures allow for flexibility through modification of model parameters. "Automatic" routines are available to give the user a "first cut" in which to commence modification.
3.5. Summary

This chapter discussed model development. Concepts were described for data collection, data reduction, and model building. These concepts were addressed with respect to the HW/SW/FW allocation process. Data collection was conducted using questionnaires, interviews, and document analysis. Data reduction techniques include categorization and quantification using weight, rank, rate, and relative comparison scales. HW/SW/FW allocation model building involved application of the AHIP and GLM techniques. The application of the model development concepts to the HW/SW/FW allocation process was presented.
Chapter 4. The Analytic Hierarchy Process (AHP)

4.1 AHP Introduction

This chapter examines the analytic hierarchy process (AHP) discussed by Saaty [18]. The AHP model is applied to the HW/SW/FW allocation process. Finally the implementation of the application of AHP as an IBM-PC based software package is described.

4.2 AHP Method Description

The AHP is a process by which complex problems are decomposed into pairwise comparisons. This AHP process is based on the principle of comparative judgement. The AHP method uses comparisons to convert qualitative and quantitative data into quantified information. AHP consists of three phases: 1) decomposition, 2) comparative judgement and 3) synthesis of priorities.
The decomposition phase breaks down a decision into its components. This decomposition is very similar to the break down concept of top-down design used in avionics development. The decision is decomposed into the possible outcomes or choices. The factors or criterion used in differentiating the decision alternatives are also determined.

The AHP comparative judgement phase analyzes the decision factors and the decision alternatives. The relative importance to the decision of one factor over another factor is required by the AHP. It is also necessary to obtain for each factor the relative likelihood of selecting a decision alternative based on the factor. This factor specific decision alternative weight is obtained through the pairwise comparison of the alternatives by factors.

Matrices are developed to make the comparative judgements of the factors and the alternatives. Figure 8 shows the two levels of matrices required by the AHP process. The first level, or level 0, is an \( nxn \) matrix \( A \) for pairwise comparisons of the decision factors. Both the row and column indices represent the 1 to \( n \) factors. The matrix element \( a_{ij} \) is the relative importance of factor \( i \) over factor \( j \). The specific question to be asked when determining the value for each \( a_{ij} \) element is: "How much more important is factor \( i \) than factor \( j \) to the decision objective?" The comparison process is clarified using the following quantitative example.

Suppose the volumetric size of a black box is being considered. If \( V_a \) is the volume of box \( a \) and \( V_b \) is the volume of box \( b \), then \( \frac{V_a}{V_b} \) is the comparative judgement of box \( a \) over box \( b \). The comparative judgement of box \( b \) over box \( a \) is therefore \( \frac{V_b}{V_a} \) or the reciprocal of box \( a \) over box \( b \). Note that the comparative judgement of box \( a \) to itself is 1.

The reciprocal property of comparative judgement dictates that a decision with \( n \) factors requires \( \frac{n \times (n - 1)}{2} \) judgements to complete the matrix.
Figure 8. AHP Level 0 and 1 Matrices.
The second level, or level 1, matrices are a set of \( n \) matrices, one for each of the \( n \) factors. These matrices are used to comparatively judge each of the \( k \) decision alternatives relative to factor \( i \). The matrices are \( k \times k \) where the row and column indices are the \( k \) decision alternatives. Element \( b_{im} \) in matrix \( B_i \) is the likelihood of choosing alternative 1 over alternative \( m \) with respect to factor \( i \). Similar to the level 0 matrix, \( \frac{k \times (k - 1)}{2} \) comparative judgements are required to fill out each matrix.

The level 0 and level 1 matrices are used to develop relative priorities for the \( n \) factors and the \( k \) decision alternatives. The priorities are calculated by manipulating the eigenvectors of the matrices. The eigenvector for the level 0 matrix is calculated. The resulting vector is \( \{Ev_1, Ev_2, Ev_3, ..., Ev_n\} \). The level 0 eigenvector elements are normalized by dividing each \( Ev_i \) element by the sum of the elements. The result is \( n \) values representing the \( L0_j \) priorities of importance for the \( n \) factors to the decision process. For each of the \( n \) \( B_i \) level 1 matrices the \( \{ev_1, ev_2, ev_3, ..., ev_k\} \) eigenvector is calculated. The \( ev_i \) elements of each level 1 eigenvector are normalized to obtain a set of \( n \times k \) \( L1_{ij} \) priorities which represent the relative importance of alternative \( i \) with respect to factor \( j \).

The priority synthesis phase combines the level 0 and level 1 priorities to obtain an overall or composite priority for the decision alternatives. The synthesis is accomplished by summing each level 1 alternative priority, with respect to a particular factor \( j \), multiplied by the factor’s level 0 priority as follows.

\[
Y_i = \sum_{i=1}^{k} \sum_{j=1}^{n} [L0_j \times L1_{ij}]
\]

where:
- \( Y_i \) = the priority of decision alternative \( i \),
- \( k \) = the number of decision alternatives,
- \( n \) = the number of decision factors,
- \( L0_j \) = the level 0 priority of factor \( j \),
- \( L1_{ij} \) = the level 1 priority of alternative \( i \) with respect to factor \( j \).
The composite priority of each decision alternative is the percentage rating of making that choice. In this manner AHP supports the structured decision approach. A stepwise overview of the AHP structured decision approach is shown in Figure 9.

### 4.3 AHP HW/SW/FW Application

The steps provided in Figure 9 are applied to the HW/SW/FW allocation process.

**Step 1.** The overall goal of the decision process is the optimal recommendation for allocating a function to HW, SW, or FW implementation.

**Step 2.** The decision alternatives are HW implementation, SW implementation, and FW implementation. Therefore $k = 3$ decision alternatives.

**Step 3.** Through data collection and data reduction as discussed in Chapter 3, three categories are defined. Category 1 is technical factors which have been reduced to 9 factors. Category 2 is programmatic factors which contains 2 factors. The third category is environmental and it contains 1 factor. The number of factors relevant to the HW/SW/FW allocation process has been reduced to 12. Therefore $n = 12$. These categories and factors are described in detail in Section 2.4. Figure 10 presents a graphical look at the HW/SW/FW allocation process decision.

**Step 4.**

**Level 0:** A $12 \times 12$ (nxn) matrix is constructed where the $n$ factors are represented in the rows and columns. Each pair of factors is compared by asking, "How much more important is factor $i$ over factor $j$ in the HW/SW/FW allocation process?" That value is placed in matrix element $a_{ij}$. The reciprocal of $a_{ij}$ is placed in $a_{ji}$. An example of a completed level 0 matrix is presented in Figure 11.

**Level 1:** There are $n$ matrices in level 1. Each matrix is $3 \times 3$: HW/SW/FW by HW/SW/FW. The $b_{im}$ elements are determined by asking, "For factor $i$, how much more advantageous is decision al-
Step Procedure
1. Define the decision objective.
2. Determine the alternative outcomes of the decision.
3. Decompose the decision into the factors affecting the decision.
4. Construct level 0 and level 1 pairwise comparisons.
5. Apply eigenvector calculations to the matrices and determine the priorities.
6. Combine the priorities for composite decision alternative priorities.
7. Analyze the priorities and make a decision.

Figure 9. AHP Stepwise Overview.

Step 5. The eigenvectors are calculated for the level 0 matrix and the n level 1 matrices. Each element of the level 0 matrix eigenvector is divided by the sum of the eigenvector elements. This results in n priorities, one for each of the factors. The same procedure is applied to the n level 1 matrices.

Step 6. The AHP priority synthesis equation discussed earlier is applied to the level 0 and level 1 priorities. The synthesis process is demonstrated graphically in Figure 13.

Step 7. Finally, in step seven the resulting composite priorities are evaluated. The decision alternative with the highest priority is the recommendation of the AHP model. The significance of the recommendation over the other decision alternatives should be checked.
4.4 AHP Tool Implementation

The AHP method is programmed as an IBM-PC based software tool. This section describes some of the AHP steps that must be accomplished before using the tool. The input mechanics for AHP data and information are described. Output of the AHP program and general analysis of the output "solutions" are provided.
**AHP MODEL**

**DECISION FACTORS:**

1. throughput
2. adaptability
3. accuracy
4. reliability
5. safety
6. interface
7. test confid
8. error tol
9. schedule
10. devel cost
11. risk
12. documents

**LEVEL 0 MATRIX:**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1/2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1/2</td>
<td>1/3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1/2</td>
<td>1/2</td>
<td>1</td>
<td>3</td>
<td>1/2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1/2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1/5</td>
<td>1/3</td>
<td>1/3</td>
<td>1</td>
<td>1/4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1/2</td>
<td>1/4</td>
<td>3</td>
<td>1/2</td>
</tr>
<tr>
<td>5</td>
<td>1/2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1/4</td>
<td>1/2</td>
<td>1</td>
<td>1</td>
<td>1/4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>1/3</td>
<td>3</td>
<td>1/2</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1/7</td>
<td>1</td>
<td>1</td>
<td>1/5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1/6</td>
<td>1/2</td>
<td>1/4</td>
<td>1/2</td>
<td>1/6</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1/2</td>
<td>1/5</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>9</td>
<td>1/3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1/3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1/2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1/2</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1/2</td>
<td>1/5</td>
<td>1/3</td>
<td>1</td>
<td>1/3</td>
<td>5</td>
<td>1</td>
<td>1/4</td>
<td>1/8</td>
<td>1</td>
<td>1/4</td>
</tr>
<tr>
<td>12</td>
<td>1/3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1/3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1/2</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 11. HW/SW/FW Allocation Level 0 Completed Matrix.

Chapter 4. The Analytic Hierarchy Process (AHP)
## AHP MODEL

### LEVEL 1 MATRICES:

<table>
<thead>
<tr>
<th></th>
<th>throughput</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1/5</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1/4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>test confid</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>adaptability</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>1/4</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>2</td>
<td>1/2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>error tol</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>accuracy</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>1/3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>1/3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1/3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>schedule</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
<td>1</td>
<td>1/4</td>
<td>1/2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>error tol</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td>1/4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>reliability</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1/4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1</td>
<td>1/4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>safety</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1/2</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1/2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>risk</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
<td>1</td>
<td>1/3</td>
<td>1/2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>reliability</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>1/4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>1</td>
<td>1/4</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>interface</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1/2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>1/2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>documents</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>documents</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td>1/2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>1/2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

---

Figure 12. HW/SW/FW Allocation Process Level 1 Matrices Example.

---

Chapter 4. The Analytic Hierarchy Process (AHP)
4.4.1 Preliminary Requirements

The AHP computer based tool implementation requires that a fair amount of data collection and data reduction have been done. This is required because there are subjective evaluations that are not implemented in the program which must be entered during a program session. In other words steps 1 through 3 of the AHP method must already be addressed. The tool user therefore has a list of factors affecting the decision. The user needs to either have the pairwise relationships of the factors and the decision alternatives for each factor, or be capable of making the judgements in real time during the computer session.
4.4.2 AHP Tool Input

The information required by the AHP implementation is a list of decision factors, their rank from 1 to n, the ability to make a comparison judgement between factors, and the ability to make a subjective comparison between HW, SW, and FW implementation for each factor. The first item required to initialize the tool is the list of factors and the associated rankings. The user selects menu item 2, "AHP Analysis", from the tool's main menu. The main menu is depicted in Figure 14.

The AHP analysis menu screen is presented in Figure 15. The automatic AHP selection uses the general linear model (GLM), chapter 5, factor and function input to calculate the level 0 and level 1 matrices and the resulting composite priorities for hardware, software, and firmware implementation. The level 0 \( a_{ij} \) matrix elements are determined by dividing \( \text{factor}_i(rank \times rate) \) by \( \text{factor}_j(rank \times rate) \). The level 1 \( b_{im} \) matrix elements are the 1 to 5 weight for HW, SW, and FW of the factor i high, moderate, or low level divided by the corresponding value for factor j.

For specifics on factor and function GLM inputs see Chapter 5.

Another possible selection from the AHP analysis menu is "AHP Level 0 Matrix". The level 0 matrix procedure provides the means to manually set up the relational matrix for level 0. The level 0 matrix screen requests whether a new level 0 matrix be created or the automatically generated level 0 matrix modified. Figure 16 is the screen which the tool displays to allow the input of comparative judgements for a new file. After the \( \frac{n \times (n - 1)}{2} \) comparisons have been made, the completed matrix is displayed and modification is requested. The modification process continues until the operator feels the level 0 matrix is complete. Figure 17 provides a flow diagram of the level 0 matrix procedure.

The "AHP Level 1 Matrices" selection from the AHP analysis menu allows for the modification of the automatically generated level 1 matrices or the creation of new matrices. The program brings each factor up on the level 1 matrix screen as shown in Figure 18. If the automatically generated
AHP function level 1 matrices are used then the values for the level 1 matrix elements will be presented. The operator is requested to modify existing or input new judgmental comparisons between decision alternatives in the same manner as the level 0 matrix. When all of the level 1 matrices have been completed the user signifies that the AHP input is complete then the AHP model can commence calculations using the modified AHP matrices. Figure 19 presents the flow diagram of the level 1 matrix procedure.
4.4.3 AHP Tool Output

The AHP model calculates the eigenvectors of the matrices, determines level 0 and level 1 priorities and then combines the priorities. The outputs of the model are the composite priorities for HW, SW, and FW implementation. This output is supported by displaying of priorities of each factor and the priorities of each decision alternative for each factor. Figure 20 provides an example of the output screen.
### AHP Factor Matrix Input

#### Scale of Relative Importance

<table>
<thead>
<tr>
<th>Intensity</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equal Importance</td>
</tr>
<tr>
<td>3</td>
<td>Moderate Importance</td>
</tr>
<tr>
<td>5</td>
<td>Essential or Strong</td>
</tr>
<tr>
<td>7</td>
<td>Very Strong Importance</td>
</tr>
<tr>
<td>9</td>
<td>Absolute Importance</td>
</tr>
<tr>
<td>2, 4, 6, 8</td>
<td>Intermediate Values</td>
</tr>
</tbody>
</table>

How much more important is _____________ than _____________ with respect to implementing a function in Hardware, Software, or Firmware? _____________ (Use above rating scale)

---

**Figure 16.** HW/SW/FW Allocation Tool AHP Comparative Judgement Input Screen.

---

**4.4.4 Analysis of AHP Tool Output**

The output information provides a suggestion for HW, SW, or FW implementation of the function. The supporting priorities allow the systems engineer to evaluate the strength of the underlying priorities. An understanding can be obtained about why the model gave the results it did.
Now an- 

factor Iilc M l y 
or modify 

old 

Display factor comparison 
input screen and input factor comparisons

For factor i 
and factor j 
modify aij & aji

Modify matrix ?

Yes

No

Save Matrix

END

Figure 17. HW/SW/FW Allocation Tool AHP Flow Chart: Level 0 Matrix
AHP Function __f£t_________ Matrix Input

Factor: throughput

Relational Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1 = Hardware
2 = Software
3 = Firmware

Do you wish to modify an element? _____ (Y=yes, N=no)

Figure 18. HW/SW/FW Allocation Tool AHP Level 1 Matrices Input Screen.
Figure 19. HW/SW/FW Allocation Tool AHP Flow Chart: Level 1 Matrix.
4.5 *AHP Summary*

This chapter described the AHP model. It presented a step-by-step approach to performing AHP. The AHP steps were applied to the HW/SW/FW allocation process. A tool to implement the AHP model for the HW/SW/FW allocation process was developed. A walk through of the AHP computer implementation was conducted. This tool can be used to collect and reduce future data. Through the modification of input factors and weights the impact of the factors on the HW/SW/FW allocation process can be evaluated. A factor that has a consistently low priority may be desirable to delete from the factor list. In this manner the HW/SW/FW database is built, the data reduction process is structured, and the HW/SW/FW allocation process model is refined.
Chapter 5. General Linear Model (GLM)

5.1 Introduction

The chapter examines the general linear model (GLM) [20]. The GLM is applied to the HW/SW/FW allocation process. The HW/SW/FW process application of GLM is then implemented as a computer based tool.

5.2 GLM Method Description

Many have researched and investigated decision making behavior. Much of this research is concerned with constructing models of the human decision making process. The typical condition of this research is that the decision process is adequately modeled by a linear model of the relationship between the decision alternatives and the decision factors. The general linear additive model that can be applied to decision processes is of the form:
\[ Y = \sum_{i=1}^{l} [\beta_i \times X_i] \]

where:

- \( Y \) = the decision alternative,
- \( \beta_i \) = the weight of factor \( i \),
- \( X_i \) = the specific dimension value for factor \( i \).

Since the HW/SW/FW allocation process is a decision process, the general linear additive equation can be applied to the HW/SW/FW allocation process.

### 5.3 GLM HW/SW/FW Allocation Application

The questionnaire of Appendix A used to collect data on the HW/SW/FW allocation process is applied to the general linear model. The \( \beta_i \) values of the GLM equation are calculated from the rank and the rating of importance of each factor \( i \). The function has a high, moderate, or low characteristic for each factor which corresponds to a 1 to 5 weight given to HW, SW, and FW based on the factor level. The \( X_j \) variable is this 1 to 5 value for function \( j \) characteristic level of factor \( i \). For hardware allocation the equation is:

\[ H_j = \sum_{i=1}^{n} [\beta_i \times X_{ij}] \]

where:

- \( H_j \) = the hardware indication for function \( j \),
- \( \beta_i = \frac{W_i}{3} \times [15 - R_i] \),
- \( X_{ij} = \frac{H(a_j)}{5} \),
- \( W_i \) = the weight of importance of factor \( i \),
- \( R_i \) = the 1 to \( n \) rank of factor \( i \),
\[ H(a_i) = \text{the 1 to 5 value for factor } i \text{ at characteristic level } a \text{ of function } j. \]

The \( X_u \) values are characteristics of the function decomposed from the system requirements. \( X_u \) is relative to the other functions of the system, i.e. a fast function in a slow system, or an inexpensive function in an expensive system. The application of GLM provides a regression value for each HW, SW, and FW implementation. These IIW, SW, and FW values are each divided by their sum to obtain a percentage recommendation for HW, SW, or FW allocation.

### 5.4 GLM Tool Implementation

An IBM-PC computer based tool is developed to implement the GLM procedure as applied to the HW/SW/FW allocation process. The input to the and output of the tool is described in the following sections.

#### 5.4.1 GLM Tool Preliminary Requirement

The tool requires the user to be equipped with a list of factors affecting the IIW/SW/FW decision process. Attributes of these factors with respect to the IIW/SW/FW allocation decision are required. The attributes include the probability of implementing a function with this factor at three strength levels, an importance rating, and an overall ranking. The user also needs to have the characteristic level of each factor for each function. With this set of data the operator is ready to run the GLM tool.
5.4.2 GLM Tool Input

The GLM implementation requires the input of factors that affect the HW/SW/FW allocation process along with attribute ratings for importance of the factor at three different levels (low, moderate, and high). An overall rating of importance (very important, moderately important, or not important) for the factor is also required. Finally, a rank of the factor relative to the other factors is necessary. The format for the factor data input screen is provided in Figure 21. The flow diagram for factor input is presented in Figure 22.

Attributes of the functions to be allocated to HW, SW, and FW are required. The functions are rated on the level (high, moderate, or low) for each factor. Figure 23 presents the function input data screen. The name and factor characteristics are the required input. The flow diagram for the function input procedure is shown in Figure 24.

5.4.3 GLM Tool Output

A strict weighting scheme manipulates weights that are assigned to the characteristic trade-off factors of the HW/SW/FW allocation process and to the function attributes. For each function, decomposed from the requirements, a rating for HW, SW, and FW implementation is calculated from the assigned factor and function weights. For example, the HW rating for function $j$ is calculated using the following equation:

$$HW_j = \sum_{i=1}^{n} \left( \frac{H_i(a_j)}{5} \right) \left( \frac{W_i}{3} \right) (15 - R_i)$$

where:

- $n$ = the number of trade-off factors,
- $a_j$ = level of factor $i$ for function $j$,
- $H_i(a_j)$ = factor $i$ weight for the $a_j$ level of HW,
### General Linear Model (GLM) Tool Input Screen

<table>
<thead>
<tr>
<th>Level</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moderate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Weight Input:** VI, MI, or NI

**Ranking Input:** 1 to 15

**Helpful Messages:**

- Type "help" for Instructions
- Type "quit" to Exit
- Type "next" for next factor
- Type "del" to delete factor

---

**Figure 21.** GLM Factor Input Screen.

- \( W_i \) = relative importance of factor \( i \),
- \( R_i \) = rank importance of factor \( i \).

Software and firmware ratings are calculated by replacing \( H_i(a_i) \) with \( S_i(a_i) \) or \( F_i(a_i) \) weighting respectively. A percentage is calculated by dividing each rating by the sum of the three ratings. The output from the tool is a percentage suggestion for implementing each function in HW, SW, and FW. The output screen is presented in Figure 25.

### 5.4.4 GLM Tool Output Analysis

The output presents the relative strength of the HW, SW, and FW implementation for each function. Depending on the strength of the recommended (highest) choice one may choose to modify the factors and repeat the GLM procedure to determine if different feasible weighting schemes re-
inforce the previous recommendation. If reinforcement is not the case, then additional modification can be done or an intuitive decision for allocation can be made. The tool allows for documentation of such decisions for future reference, which may aid in a future decision.
### FUNCTION INPUT SCREEN

Enter: H for High
M for Moderate
L for Low

<table>
<thead>
<tr>
<th>Function Name</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>__________</td>
<td>schedule</td>
</tr>
<tr>
<td>adaptability</td>
<td>__________</td>
<td>devel cost</td>
</tr>
<tr>
<td>accuracy</td>
<td>__________</td>
<td>risk</td>
</tr>
<tr>
<td>reliability</td>
<td>__________</td>
<td>documents</td>
</tr>
<tr>
<td>safety</td>
<td>__________</td>
<td></td>
</tr>
<tr>
<td>interface</td>
<td>__________</td>
<td></td>
</tr>
<tr>
<td>test confid</td>
<td>__________</td>
<td></td>
</tr>
<tr>
<td>error tol</td>
<td>__________</td>
<td></td>
</tr>
</tbody>
</table>

Type "quit" to Exit  
Type "help" for Instructions  
Type "del" to delete factor  
Type "next" for next factor

---

**Figure 23. GLM Function Input Screen.**

### 5.5 GLM Summary

The GLM method of decision making was discussed. The form of the model was explained and applied to the HW/SW/FW allocation decision process. This HW/SW/FW allocation application model was developed into a computer based tool. The tool was presented and a step of the program walk-through was performed.
New function file or modify old

Display function input screen and input function characteristics

Quit?

Save function file

END

Modify

Display function and function characteristics and allow modification

Quit?

Save function file

END

Figure 24. GLM Function Input Flow Diagram.
FUNCTION: fft

throughput: M
adaptability: H
accuracy: L
reliability: M
safety: M
interface: H
test confid: L
error tol: M
schedule: H
devel cost: H
risk: H
documents: M

Method 1 Results:
Hardware: 23.276
Software: 47.390
Firmware: 29.333

Figure 25. GLM Result Output Screen.
Chapter 6. Second Order Model

6.1 Introduction

This chapter discusses the use of second order regression to analyze the HW/SW/FW allocation process. The theory of second order regression is described along with experimental design theory. The theory is applied by developing an experimental design for the HW/SW/FW allocation process data collection. The data collection is not performed due to the time constraints of this research. It follows from the lack of data that the second order regression (SOR) analysis method is not implemented as a procedure in the HW/SW/FW allocation process software support tool.

6.2 Second Order Model Description

The linear model is considered by many to encompass most of the variance of the decision making process. It is felt that it adequately describes and models the decision process. Eihorn and Hogarth
[7], however, support the belief that higher order interactions may be significant to the decision process. Williges [25] states "Usually a second-order polynomial approximation is adequate to account for most human performance data." In this research human performance is the decision making process. The SOR model is of the form:

\[
Y = \beta_0 + \sum_{i=1}^{k} [\beta_i \times X_i] + \sum_{i=1}^{k} [\beta_{k+i} \times X_i^2] + \epsilon
\]

where:
- \( Y \) = the decision,
- \( \beta_i \) = the beta weight or importance of factor i,
- \( X_i \) = the function specific information on factor i,
- \( \epsilon \) = the random error.

In this equation, \( X_i^2 \) is the interaction of the factor with itself. The \( X_i \times X_j \) terms are the interaction between factor i and factor j.

Obtaining data on the second order interaction, \( X_i \) interacting with \( X_j \), is difficult using happenstance data. Usually data that is not collected using experimental design has confounded second order effects. Therefore it is recommended that an experimental design be used to collect data. Specifically a fractional factorial design is developed for future data collection. This enables data collection without confounding the main effects or the second order interactions or at least to consciously plan for confounding specific factors.
6.3 Experimental design

There are several different methods for developing experiments for data collection. The human factors community is doing extensive research in and application of experimental designed data collection. The goal is to collect data that is statistically valid without confounding significant factors. A tutorial on experimental design data collection is not conducted in this thesis. However, several valuable resources references are Box, Hunter, and Hunter [3], Cochran and Cox [4], Myers [15], and Williges [25]. These references form the foundation for the development of a fractional factorial experiment design for the HW/SW/FW allocation process.

6.4 HW/SW/FW Allocation Experimental Design

The experiment designed for HW/SW/FW allocation process data collection uses eight factors, each with two levels. The eight factors and their associated levels are presented in Figure 26. A complete experimental data collection design using these eight factors would require \(2^8\) or 256 treatment conditions. The design developed here blocks the design twice to reduce the number of treatments. This creates a one fourth replicate fractional factorial design. The number of treatments for the blocked experiment is \(2^4\) or 64 treatment combinations. The relationships used to block the experiment are, from Figure 26, \(ABCD\) and \(EFGH\).

The blocking relationships are applied to break-up the experiment combinations through the use of modulo 2 notation. Modulo 2 is used because there are two levels for each factor, 0 and 1. Each blocking effect is set equal to 0 or 1, in this example both interaction one, \(I_1\), and interaction two, \(I_2\), are set to 0. The interaction of the two blocking effects, \(I_1 + I_2\), does not need to be set equal to 0 or 1 directly because it is a consequence of assigning \(I_1\) and \(I_2\).
A = Throughput
   \( a_0 = \text{HIGH} \)
   \( a_1 = \text{LOW} \)

B = Adaptability
   \( b_0 = \text{HIGH} \)
   \( b_1 = \text{LOW} \)

C = Accuracy
   \( c_0 = \text{HIGH} \)
   \( c_1 = \text{LOW} \)

D = Reliability
   \( d_0 = \text{HIGH} \)
   \( d_1 = \text{LOW} \)

E = Safety
   \( e_0 = \text{HIGH} \)
   \( e_1 = \text{LOW} \)

F = Interfaces
   \( f_0 = \text{HIGH} \)
   \( f_1 = \text{LOW} \)

G = Risk
   \( g_0 = \text{HIGH} \)
   \( g_1 = \text{LOW} \)

H = Schedule
   \( h_0 = \text{HIGH} \)
   \( h_1 = \text{LOW} \)

Figure 26. Experiment Factors and Levels
The application of the blocking effects $I_1 = 0$ and $I_2 = 0$ involves determining all of the combinations of the eight factor levels that add to 0 in modulo 2. Two sets are developed, one for $I_1 = 0$ and one for $I_2 = 0$. The intersection of the two blocks provides the 64 combinations of factor levels for the experiment treatments. Replacement of the 1's and 0's by $a_n, a_1, b_n, b_1$, etc. levels is the experiment design in factor level form. Figure 27 presents the experiment development using modulo 2 notation and the final experiment factor level treatment combinations.

The subjects of the experiment used to collect data are expected to be experts on the HW/SW/FW allocation process. Each subject is given the 64 scenarios of factor combinations as function attributes. The subjects are then requested to provide a recommended implementation in hardware, software, and firmware for each function. A 1 to 100 value for each implementation is given, the total of which adds to 100.

The data collected can be analyzed using statistical packages with second order regression analysis capability such as SAS [9]. The statistical analysis develops a regression equation in the form of a second order polynomial and analysis of variance (ANOVA) information of the significance of the regression. The regression equation is listed as beta weights, one for each main effect $X$ and one for each interaction $X_A \times X_B$ through $X_G \times H$ and $X_A \times A$ through $X_H \times H$. The equation is formed by substituting the beta values into the SOR model. The ANOVA summary table provides significance data which includes the degrees of freedom (df), the sum of squares, the mean square, and the F-test for each factor, main effect and interaction. An example of possible partial output is provided in Figure 28.
Modulo 2 Representation:  
\[ x_1 + x_2 + x_3 + x_4 = 0 \]
\[ x_5 + x_6 + x_7 + x_8 = 0 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>x_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>x_5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>x_8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 27. Modulo Experiment design Development
Figure 28. Analysis Package Output Example
6.5 Application of SOR

The SOR equation can be used to provide HW/SW/FW allocation recommendations. The $\beta_i$ elements of the equation are obtained directly from the output of the statistical package. An equation is developed for each hardware, software, and firmware medium. For example, if the data of the two factor design presented in Figure 29 is used the SOR equations would be:

\[
HW = 1.5X_A + 0.87X_B - 0.15(X_A \times X_B)
\]

\[
SW = 0.75X_A + 1.81X_B + 0.55(X_A \times X_B)
\]

\[
FW = 0.75X_A + 0.92X_B + 1.59(X_A \times X_B)
\]

The three second order polynomial equations together comprise the SOR model for the HW/SW/FW allocation process.

6.6 Summary

Discussion on the value of second order regression analysis was presented. The requirement for experimental data collection to avoid the confounding of factors and their interactions was described. An experimental design example of data collection was developed for the HW/SW/FW allocation process. The experiment was a $2^4$ one fourth fractional factorial design with eight factors or main effects. It should be noted that the experiment could be expanded to include more factors but the complexity of the design would increase exponentially.
Hardware Regression
\[ \beta_d = 1.50 \]
\[ \beta_s = 0.87 \]
\[ \beta_{ds} = -0.15 \]

Software Regression
\[ \beta_d = 0.75 \]
\[ \beta_s = 1.81 \]
\[ \beta_{ds} = 0.55 \]

Firmware Regression
\[ \beta_d = 0.75 \]
\[ \beta_s = 0.29 \]
\[ \beta_{ds} = 1.59 \]

Figure 29. Two Factor SOR Analytic Package Results
7.1 Introduction

This chapter presents the data collected from questionnaires on the HW/SW/FW allocation process. The data is analyzed using GLM and AIIP in a two phased approach. The first phase is a general analysis of the factors to: 1) obtain priorities of the factors, 2) test the consistency of the model results, and 3) obtain an indication of the response characteristics. The second phase uses an experimental design to obtain information on the selected high priority factors to the HW/SW/FW allocation process. The nature of the analysis is preliminary since only two questionnaire responses have been received at this time. However, the methods and data presented are used to form a baseline for future analysis.
7.2 HW/SW/FW Data Analysis

The GLM and AHP analysis methods are applied to the questionnaire data in a two phased approach. Details of the analysis and the results are presented in the following sections.

7.2.1 Phase 1: General Analysis

Twelve factors are selected from the questionnaire for the general analysis of factor influence on HW/SW/FW allocation. A representative list of the twelve factors and the associated attributes are provided in Figure 30. The data provided in Figure 30 contains insights into the HW/SW/FW allocation decision process. A walk through of some of the factors and their attributes is presented.

Throughput is the highest ranked factor. It is also rated as very important to the HW/SW/FW allocation process. The tendencies of throughput are strong. High throughput requirements for a function indicates that the function should be implemented in hardware. However, low throughput requirements emphasize software implementation. The FW implementation is uncertain at all throughput levels.

Adaptability is ranked number two and has a moderate importance rating. A high adaptability requirement for a function emphasizes a recommendation of software implementation. Low adaptability requirements identify a recommendation for firmware implementation. The hardware weights range from "uncertain" to "certainly not" for the adaptability levels.

Accuracy is ranked eleven, with a rating of moderately important. All levels of accuracy recommend probable software implementation. The firmware and hardware weights range in the "uncertain" and "probably not" sector of the implementation scale.
### FACTORS:

1. **throughput**
   - Level: HDW, SFW, FMW
   - Rating: VI
   - Rank: 1

2. **adaptability**
   - Level: HDW, SFW, FMW
   - Rating: MI
   - Rank: 2

3. **accuracy**
   - Level: HDW, SFW, FMW
   - Rating: MI
   - Rank: 11

4. **reliability**
   - Level: HDW, SFW, FMW
   - Rating: MI
   - Rank: 4

5. **safety**
   - Level: HDW, SFW, FMW
   - Rating: MI
   - Rank: 9

6. **interface**
   - Level: HDW, SFW, FMW
   - Rating: MI
   - Rank: 5

7. **test confid**
   - Level: HDW, SFW, FMW
   - Rating: NI
   - Rank: 13

8. **error tol**
   - Level: HDW, SFW, FMW
   - Rating: NI
   - Rank: 12

9. **schedule**
   - Level: HDW, SFW, FMW
   - Rating: NI
   - Rank: 14

10. **devel cost**
    - Level: HDW, SFW, FMW
    - Rating: NI
    - Rank: 11

11. **risk**
    - Level: HDW, SFW, FMW
    - Rating: MI
    - Rank: 9

12. **documents**
    - Level: HDW, SFW, FMW
    - Rating: MI
    - Rank: 6

---

Figure 30. HW/SW/FW Allocation Factors and Attributes
A system look at all the factors and factor attributes shows that, in general, this systems engineer favors software implementation. This hypothesis is made because the most common factor level recommendation is "probably software" or a weight of two in the software column. The only strong hardware and firmware implementation recommendations are for high throughput and low adaptability respectively. The general analysis of factors allows the engineer to evaluate possible biases.

The three functions used for the preliminary analysis are 1) fast fourier transform (FFT), 2) add integer (ADDINT), and 3) division (DIV). FFT is characteristic of high levels for all factors. ADDINT is characteristic of moderate levels for all factors. DIV is characteristic of low levels for all factors. The three functions represent the extremes of all high, all moderate, and all low for the factors. The examination of the extremes identifies some boundaries of the analytic techniques for the HW/SW/FW allocation process. The GLM and AIIP analytical methods are applied to each function. Figure 31 presents the GLM result output and Figure 32 presents the AHP result output.

The output of the two models are found to be consistent in the sense that the same recommendation is made for each respective function. In fact, all three functions were recommended for software allocation. However, for the FFT function, software implementation was rated 38 compared to 35 and 27 for hardware and firmware, respectively. This is not a very large differential between HW, SW, and FW. Therefore hardware may also be considered for implementation depending on the entire set of function results and the systems engineer's expert judgement.

The priorities of the factors that are calculated by the AIIP method can be used to understand more about the factors and the associated attributes. The factor priorities provide an overview of each factor importance to the HW/SW/FW allocation process relative to the other factors. Special attention is given to the high and low priority factors. High priorities such as throughput, adaptability and reliability play major roles in the decision process. Throughput for instance accounts for 28.5% of the decision process. Throughput, adaptability, and reliability together account for over 50% of the HW/SW/FW allocation decision process. The systems engineer should be conscious of the factor priorities when assigning the factor characteristics of the functions. The low priority
FUNCTION: fft

throughput: H
adaptability: H
accuracy: H
reliability: H
safety: H
interface: H
test confid: H
error tol: H
schedule: H
devel cost: H
risk: H
documents: H

Method 1 Results:
Hardware: 34.518
Software: 38.862
Firmware: 26.620

FUNCTION: addint

throughput: M
adaptability: M
accuracy: M
reliability: M
safety: M
interface: M
test confid: M
error tol: M
schedule: M
devel cost: M
risk: M
documents: M

Method 1 Results:
Hardware: 27.239
Software: 40.264
Firmware: 32.497

FUNCTION: div

throughput: L
adaptability: L
accuracy: L
reliability: L
safety: L
interface: L
test confid: L
error tol: L
schedule: L
devel cost: L
risk: L
documents: L

Method 1 Results:
Hardware: 25.183
Software: 41.229
Firmware: 33.588

Figure 31. HW/SW/FW Allocation GLM Results for General Analysis
Figure 32. HW/SW/FW Allocation AHP Results for General Analysis

factors such as: test confidence, error tolerance, and schedule with priorities of 1.9%, 2.5%, and
1.3% respectively; should be considered for elimination from the factor list in order to refine the
model and reduce the process complexity.
7.2.2 Phase 2: Experimental Design

Three high priority factors are used to develop an experimental data analysis. The three factors are presented in Figure 33 along with their associated attributes. The possible combinations or treatment conditions for the three factor by three level situation is $3^3$ or 27 treatments. For this representative example of factor analysis, the treatments are blocked by the three way interaction $ABC$, which reduces the number of treatment conditions to $3^3$ or 9. The $ABC$ interaction is set equal to 0 modulo 3. Modulo 3 notation is used due to the three levels for each factor. The resulting set of factor level combinations is presented in Figure 34 along with the treatment combinations converted to factor level notation. Figures 33 and 34 provide the input to the GLM and AHP models.

The GLM procedure is applied to each function. Function one HW, SW, and FW recommendation weights are calculated using the GLM equation discussed in Chapter 5 and as presented below.

$$HW_1 = (\frac{5.0}{1.0})(\frac{3.0}{1.0})(16 - 1) + (\frac{5.0}{5.0})(\frac{3.0}{2.0})(16 - 2) + (\frac{5.0}{5.0})(\frac{3.0}{2.0})(16 - 4) = 264$$

$$SW_1 = (\frac{5.0}{5.0})(\frac{3.0}{1.0})(16 - 1) + (\frac{5.0}{1.0})(\frac{3.0}{2.0})(16 - 2) + (\frac{5.0}{2.0})(\frac{3.0}{2.0})(16 - 4) = 195$$

$$FW_1 = (\frac{5.0}{3.0})(\frac{3.0}{1.0})(16 - 1) + (\frac{5.0}{2.0})(\frac{3.0}{2.0})(16 - 2) + (\frac{5.0}{3.0})(\frac{3.0}{2.0})(16 - 4) = 157.5$$

These HW, SW, and FW values are normalized by dividing each value by the sum of the values. The normalized weights for function one are presented in Figure 35 along with the resulting GLM recommendation weights for the other eight functions.

The application of the AHP method requires the pairwise comparisons of the factors and the comparisons of the HW, SW, and FW alternatives. Since the experts that responded to the questionnaire are not available to work with the support tool the comparisons are developed "auto-
FACTORS:

1. throughput
   Level HDW SFW FMW
   H  1  5  3
   M  3  2  3
   L  4  1  3
   Rating: VI
   Rank:  1

2. adaptability
   Level HDW SFW FMW
   H  5  1  2
   M  4  3  2
   L  3  4  1
   Rating: MI
   Rank:  2

3. reliability
   Level HDW SFW FMW
   H  5  2  3
   M  3  3  3
   L  2  4  3
   Rating: MI
   Rank:  4

Figure 33. Factors of Experimental Design Analysis

matically" from the factor information as presented in Chapter 4. For function one throughput is compared to adaptability using the level 0 equation for element $a_j$ as shown below.

$$a_j = \frac{[(16 - 1)\left(\frac{3.0}{1.0}\right)]}{[(16 - 2.0)\left(\frac{3.0}{2.0}\right)]} = 2$$

and

$$a_{ji} = \frac{1.0}{(a_j)} = \frac{1}{2}$$
Factors

A = > Throughput
B = > Adaptability
C = > Reliability

Blocking Interaction:
A × B × C
\[ x_1 + x_2 + x_3 = 0 \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>A_{hB_C}</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>A_{hB_mC}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>A_{mB_C}</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>A_{hB_mC}</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>A_{mB_mC}</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>A_{hB_mC}</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>A_{hB_mC}</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>A_{mB_mC}</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>A_{hB_C}</td>
</tr>
</tbody>
</table>

\[ 0 = \text{ high} \]
\[ 1 = \text{ moderate} \]
\[ 2 = \text{ low} \]

Figure 34. Experimental Design Analysis Treatments
<table>
<thead>
<tr>
<th>FUNCTION: one</th>
<th>FUNCTION: four</th>
<th>FUNCTION: seven</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: H</td>
<td>throughput: H</td>
<td>throughput: L</td>
</tr>
<tr>
<td>adaptability: H</td>
<td>adaptability: L</td>
<td>adaptability: M</td>
</tr>
<tr>
<td>reliability: H</td>
<td>reliability: M</td>
<td>reliability: H</td>
</tr>
</tbody>
</table>

Method 1 Results:
- Hardware: 46.991
- Software: 28.653
- Firmware: 24.355

<table>
<thead>
<tr>
<th>FUNCTION: two</th>
<th>FUNCTION: five</th>
<th>FUNCTION: eight</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: H</td>
<td>throughput: M</td>
<td>throughput: M</td>
</tr>
<tr>
<td>adaptability: M</td>
<td>adaptability: L</td>
<td>adaptability: M</td>
</tr>
<tr>
<td>reliability: L</td>
<td>reliability: H</td>
<td>reliability: M</td>
</tr>
</tbody>
</table>

Method 1 Results:
- Hardware: 54.768
- Software: 17.439
- Firmware: 27.793

<table>
<thead>
<tr>
<th>FUNCTION: three</th>
<th>FUNCTION: six</th>
<th>FUNCTION: nine</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: M</td>
<td>throughput: L</td>
<td>throughput: L</td>
</tr>
<tr>
<td>adaptability: H</td>
<td>adaptability: H</td>
<td>adaptability: L</td>
</tr>
<tr>
<td>reliability: L</td>
<td>reliability: M</td>
<td>reliability: L</td>
</tr>
</tbody>
</table>

Method 1 Results:
- Hardware: 21.769
- Software: 49.320
- Firmware: 28.912

Figure 35. Experimental Design Analysis GLM Results
The rest of the elements are calculated in the same manner. The level 1 matrices compare for each factor the HW, SW, and FW alternatives relative to the function. The throughput of function one is high therefore, hardware and software are compared at the high level using the equation presented in Chapter 4 to obtain level 1 matrix element $b_{lm}$ as presented below.

$$b_{lm} = \frac{5}{1} = 5$$

$$b_{mi} = \frac{1}{b_{lm}} = \frac{1}{5}$$

The other matrices elements are calculated in the same manner. Figure 36 provides the level 0 matrix for the model and level 1 matrices for each function by factor using rounded values. Rounded values are used to simplify the display. Note that reliability appears completely indecisive in function 4, 6, and 8, however, these are fractions rounded to 1, so there is some effect due to reliability.

The eigenvector priorities are calculated for the matrices and manipulated as described in Chapter 4 to obtain the composite HW, SW, and FW priorities. The composite priorities are calculated using the level 0 and level 1 priorities for function one as follows.

$$HW_1 = [(0.536 \times 0.652) + (0.250 \times 0.118) + (0.214 \times 0.194)] \times 100 = 42.027$$

$$SW_1 = [(0.536 \times 0.130) + (0.250 \times 0.588) + (0.214 \times 0.484)] \times 100 = 32.062$$

$$FW_1 = [(0.536 \times 0.217) + (0.250 \times 0.294) + (0.214 \times 0.323)] \times 100 = 25.911$$

The same AHP procedures are applied to all nine functions. Figure 37 presents the composite priority results for all nine functions.

The results obtained through applying the GLM and AHP analytic methods are consistent. Both of the techniques provided the same HW, SW, and FW recommendations for the nine functions.
Figure 36. Data Sample Level 0 and Level 1 AHP Matrices.
Figure 37. Experimental Design Analysis AIIP Results

Figure 38 presents the functions, their factor characteristics, and the model’s recommendation for implementation.

The combination of factors and factor levels enabled the evaluation of models over a range of statistically independent function characteristics. Possible biases, in the model, become apparent if the result is that all the functions are allocated to one of the HW, SW, or FW media or if none of the functions are allocated to one of the HW, SW, or FW media. In this data sample of nine functions three are allocated to hardware, five are allocated to software, and one is allocated to firmware. It appears that there may be a bias against firmware and a bias for software. A larger data sample could be used to affirm the bias hypothesis or the systems engineer could beware of the possibility of the bias in either the assigning of function characteristics or in the system under development i.e.
the system could by nature be software intensive. If biases are found that the systems engineer feels
are too pronounced then the model parameters can be modified to correct for the deficiency.

7.3 Model Adaptability

The systems engineer should evaluate the allocation recommendations and determine if the sug-
gestions of the models are consistent with the rough allocation scheme the system engineer has in
mind. The factor weights could then be modified accordingly and the experiment treatments run
again. For example Figure 38 displays the modified model for the data sample of this chapter.
The system engineer is intending to alleviate the software bias. The throughput factor is modified
in two ways: 1) the software strength of the moderate level throughput weight is changed from two
to three and 2) the FW uncertainty is modified form 3 to 2 for a moderate throughput requirement.
The strength of software implementation for low adaptability is modified from 4 to 5. The result
for GLM analysis with the new model is presented in Figure 39. The distribution is now three
hardware, four software, and two firmware as compared to three hardware, five software, and one
firmware. Function eight is changed from SW to FW allocation due to the model refinement.
Though this process the engineer is tuning the model factors to his or her project and experience.
It is in this manner the HW/SW/FW allocation process model is improved through a refined
knowledge base. It is a good HW/SW/FW allocation model operating procedure to run a sample
batch of functions and modify the model factors before running a full scale application.

The use of the GLM to obtain the AHP comparative judgments provides a first cut on the matrices.
The AHP model can be refined by modifying the level 0 and level 1 matrices in accordance with
the expert judgement of the systems engineer and the insights gleaned from the initial function
treatment combinations. Figure 40 demonstrates a possible refinement for the sample data set as
compared to Figure 36. The results of the AHP model are presented in Figure 41 as compared to
FACTORS:

1. throughput
<table>
<thead>
<tr>
<th>Level</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
   Rating: VI
   Rank: 1

2. adaptability
<table>
<thead>
<tr>
<th>Level</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>3</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
   Rating: MI
   Rank: 2

3. reliability
<table>
<thead>
<tr>
<th>Level</th>
<th>HDW</th>
<th>SFW</th>
<th>FMW</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>L</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
   Rating: MI
   Rank: 4

Figure 38. Model Improvement to Remove Biases.

the automatic AHP results in Figure 37. The comparison of the automatic method with the improved AHP model shows function allocations based on an improved knowledge base.
<table>
<thead>
<tr>
<th>FUNCTION: one</th>
<th>FUNCTION: four</th>
<th>FUNCTION: seven</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: H</td>
<td>throughput: H</td>
<td>throughput: L</td>
</tr>
<tr>
<td>adaptability: H</td>
<td>adaptability: L</td>
<td>adaptability: M</td>
</tr>
<tr>
<td>reliability: H</td>
<td>reliability: M</td>
<td>reliability: H</td>
</tr>
</tbody>
</table>

Method 1 Results:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.991</td>
<td>28.653</td>
<td>24.355</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNCTION: two</th>
<th>FUNCTION: five</th>
<th>FUNCTION: eight</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: H</td>
<td>throughput: M</td>
<td>throughput: M</td>
</tr>
<tr>
<td>adaptability: M</td>
<td>adaptability: L</td>
<td>adaptability: M</td>
</tr>
<tr>
<td>reliability: L</td>
<td>reliability: H</td>
<td>reliability: M</td>
</tr>
</tbody>
</table>

Method 1 Results:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>54.768</td>
<td>17.439</td>
<td>27.793</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNCTION: three</th>
<th>FUNCTION: six</th>
<th>FUNCTION: nine</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput: M</td>
<td>throughput: L</td>
<td>throughput: L</td>
</tr>
<tr>
<td>adaptability: H</td>
<td>adaptability: H</td>
<td>adaptability: L</td>
</tr>
<tr>
<td>reliability: L</td>
<td>reliability: M</td>
<td>reliability: L</td>
</tr>
</tbody>
</table>

Method 1 Results:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.792</td>
<td>44.610</td>
<td>31.599</td>
</tr>
</tbody>
</table>

---

Figure 39. Model Improvement Results.
**Figure 40. AIIP Model Improvement.**

### Chapter 7. Model Results

<table>
<thead>
<tr>
<th>DECISION FACTORS</th>
<th>LEVEL 0 MATRIX:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. throughput</td>
<td>1 2 3</td>
</tr>
<tr>
<td>2. adaptability</td>
<td>1 1 1</td>
</tr>
<tr>
<td>3. reliability</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: one</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: two</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: three</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: four</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: five</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: six</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function: seven</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL 1 MATRICES:</td>
</tr>
<tr>
<td>1. throughput</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>2. adaptability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
<tr>
<td>3. reliability</td>
</tr>
<tr>
<td>HNW 1 2 3</td>
</tr>
<tr>
<td>SFM 5 1 2</td>
</tr>
<tr>
<td>FNM 2 1 3</td>
</tr>
</tbody>
</table>
### AHP Results

#### Function: one

| N | 0.468 | 0.311 | 0.111 |
| S | 0.123 | 0.508 | 0.540 |
| F | 0.230 | 0.309 | 0.297 |

#### Function: four

| N | 0.468 | 0.311 | 0.111 |
| S | 0.123 | 0.508 | 0.540 |
| F | 0.230 | 0.309 | 0.297 |

#### Function: seven

| N | 0.143 | 0.196 | 0.163 |
| S | 0.571 | 0.311 | 0.540 |
| F | 0.286 | 0.493 | 0.297 |

Hardware Composite: 43.216
Software Composite: 31.536
Firewire Composite: 38.714

#### Function: two

| N | 0.366 | 0.511 | 0.111 |
| S | 0.186 | 0.243 | 0.168 |
| F | 0.217 | 0.347 | 0.249 |

Hardware Composite: 49.090
Software Composite: 21.355
Firewire Composite: 29.515

#### Function: five

| N | 0.366 | 0.511 | 0.111 |
| S | 0.450 | 0.223 | 0.163 |
| F | 0.310 | 0.012 | 0.540 |

Hardware Composite: 50.019
Software Composite: 21.355
Firewire Composite: 29.515

#### Function: three

| N | 0.566 | 0.311 | 0.111 |
| S | 0.413 | 0.228 | 0.492 |
| F | 0.260 | 0.607 | 0.196 |

Hardware Composite: 36.808
Software Composite: 36.808
Firewire Composite: 37.192

#### Function: six

| N | 0.566 | 0.311 | 0.111 |
| S | 0.413 | 0.228 | 0.492 |
| F | 0.260 | 0.607 | 0.196 |

Hardware Composite: 36.530
Software Composite: 36.530
Firewire Composite: 36.012

#### Function: nine

| N | 0.143 | 0.210 | 0.333 |
| S | 0.571 | 0.311 | 0.540 |
| F | 0.286 | 0.493 | 0.287 |

Hardware Composite: 16.140
Software Composite: 48.643
Firewire Composite: 35.156

### 7.4 Summary

The questionnaire data and analysis methods were evaluated in this chapter. It was found that GLM and AHP provide consistent recommendations. Results of experimentally designed treatment combinations show that the factor attributes contain a possible prejudice against firmware implementation and for software implementation. The user needs to evaluate his or her prejudices and consider factor attribute modification to remove the bias. The model modifications to improve the model include 1) factor and factor attribute modification and 2) AHP level 1 and level 2 matrices refinement through modifying the comparisons made from GLM input data. The recom-
mended use of the support tool includes pretests of the model using experimentally designed treatment combinations. From the pretest results the model can be modified and the AHP matrices can be improved. This model refinement is crucial to capturing the expert knowledge of the system engineer.
Chapter 8. Conclusion

8.1 Introduction

This chapter provides an overview of the results accomplished on the HW/SW/FW allocation process by this research. The work performed is summarized, research contributions are discussed, and possibilities for future research on the HW/SW/FW allocation process are presented.

8.2 Research Overview

The objective of this research was to quantify, structure and model the HW/SW/FW allocation process of avionics system development. The research was motivated by the unstructured, intuitive nature of the present HW/SW/FW allocation process. Little research has been performed on the HW/SW/FW allocation process subject. Therefore, the research was exploratory in nature.
A background was laid for discussion of the HW/SW/FW allocation process. The allocation process is only applicable to complex systems which have concurrent or near concurrent hardware, software, and firmware developments. A description of the top-down methodology used to develop the complex systems was presented with emphasis on the placement of the HW/SW/FW allocation process into the "big picture". Hardware, software, and firmware definitions and descriptions were presented to alleviate ambiguities among schools of thought.

A literature review was performed. This review placed this research on HW/SW/FW allocation in context with other related research. It also provided justification for the value of the research. Attributes of the HW/SW/FW allocation process and influencing trade-off factors were collected. Three categories were used to facilitate the factor acquisition process: 1) technical, 2) programmatic and 3) environmental.

A model development plan was constructed. The plan consisted of data collection, data reduction, and model building. Both theory and application of the three components of model development to the HW/SW/FW allocation process were provided. AHP, GLM, and SOR are three analytic methods used to build the HW/SW/FW allocation model.

The AHP is an analytic technique which uses pairwise comparative judgements to analyze a decomposed decision process. The decision is decomposed into factors affecting the decision and the alternative decision outcomes. Matrices were used to perform comparisons between the HW/SW/FW allocation factors and the decision alternatives. Eigenvector priority calculations were applied to obtain composite hardware, software, and firmware implementation priorities.

The GLM is a linear regression analysis of the decision process. Beta weights were developed for each HW/SW/FW allocation factor from systematized data. These beta weights were applied to the GLM to generate an equation for each hardware, software, and firmware implementation recommendation. Although much research supports the use of the GLM to model decision making, other researchers believe that the second order interactions are significant. This research addresses
the dilemma by designing a data collection experiment which does not confound the second order interactions of the trade-off factors. The data collected from the experiment can be processed using a statistical analysis package to develop a second order polynomial regression. Similar to GLM, a regression equation is generated for each hardware, software, and firmware implementation recommendation.

The AHP and GLM analytical methods were developed into a computer based support tool. The tool can be used for data collection, data reduction, and model building. Data collected on the HW/SW/FW allocation factors was applied to the tool. A general overview of the factor affects on the allocation and the priorities of the factors was conducted and evaluated. The general evaluation was followed by a detail investigation into high priority factors. An experimental design was applied using the tool. The output of the tool provided results to analyze a range of factor combination affects on the HW/SW/FW allocation process.

Conclusions are drawn from this research. The research verified that the HW/SW/FW allocation process is a subjective, qualitative decision process. The allocation factor priorities varied by company, project, and system engineer. It is concluded that in order to structure the HW/SW/FW allocation process, the tool must be tailored to the particular company, project, and system engineer. However, once the model has captured the specific information it can be used with little or no modification on similar projects in the future. The expert knowledge of the system engineers is collected and documented to aid the HW/SW/FW allocation process.

8.3 Research Contributions

There is a lack of research into the allocation of functions to hardware, software, and firmware in complex system top-down developments. This research is a first effort to structure the subjective
HW/SW/FW allocation process. A foundation is set by this thesis for future research into the allocation process. Factors affecting the allocation of functions to HW, SW, or FW implementation have been identified, and categorized. A methodology for systematizing and quantifying the factors has been developed. Analytic techniques were applied to the systematized data in order to provide HW, SW, and FW implementation recommendations. Experimental design theories were used to analyze the model results and to prepare a foundation for future data collection. Second order regression models will be possible using the developed experimentally design treatment combinations for data collection.

A software support package was developed to facilitate HW/SW/FW allocation process modeling. The software tool allows for flexibility in systematizing the factor data. The tool enables the addition and deletion influencing factors. Future efforts to quantify and structure the HW/SW/FW allocation process now has a starting point through the model developed by this research and the corresponding software support tool.

8.4 Future Work

This research sets the foundation for studying the HW/SW/FW allocation process of system development. A tool has been developed to collect, reduce, and analyze HW/SW/FW allocation data. The initial trade-off factors have been identified and applied to create an analytical model of the decision process.

Research efforts on improving the model are needed. The research would emphasize data collection and tool application. Figure 42 depicts the scenario of the research. The feedback to improve the knowledge parameters is crucial to model improvement. This type of data collection would involve expert interaction with the tool. Experimental design techniques could be applied to help capture
statistically valid data. The result of this research efforts would be an improved HW/SW/FW allocation model and a data base to support the decision process.

Research could also be done on expanding the support tool. The capability to handle inference rules could be developed. This would allow for GO/NO-GO determinants and maximum/minimum parameter specifications. Graphics capabilities could be added to the model. The graphics could illustrate the trade-off factor relationships and allocation capacities.

Further research could integrate the HW/SW/FW allocation support tool or model with other design tool packages. The design packages would include computer aided system engineering (CASE) and computer aided design (CAD) support tools. Application to the reliability and maintainability life cycle cost tool (RAMCAD) is also foreseeable.
Figure 42. Decision Support Model Improvement
Bibliography


Appendix A: Questionnaire
Hardware/Software/Firmware Allocation Questionnaire

Background

The purpose of this questionnaire is to collect information on the hardware/software/firmware function allocation in system development. The objective of the questionnaire is to determine general tendency in implementing a function in hardware, software, or firmware on the basis of 15 factors. Procedure

Please evaluate the following factors relative to whether you would perform the function in hardware, software, or firmware. Explanation of the factors are provided on page 9. Three levels for each factor are presented. It is requested that for each factor level, you provide your expert opinion on the probability that you would implement it in hardware, software, or firmware using the following probability rating scheme:

<table>
<thead>
<tr>
<th>Level</th>
<th>Qualitative</th>
<th>Quantitative</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Almost Certain</td>
<td>95%</td>
</tr>
<tr>
<td>2</td>
<td>More Than Likely</td>
<td>75%</td>
</tr>
<tr>
<td>3</td>
<td>Uncertain</td>
<td>50%</td>
</tr>
<tr>
<td>4</td>
<td>More Than Likely Not</td>
<td>35%</td>
</tr>
<tr>
<td>5</td>
<td>Almost Certain Not</td>
<td>10%</td>
</tr>
</tbody>
</table>

Please circle the appropriate response (your opinion) for each factor level under each implementation medium.

For each factor please indicate the importance of the factor by circling the appropriate response (your opinion), using the following rating definitions:

VI - Very Important
MI - Moderately Important
NI - Not Important

Any comments can be made in the space provided or on an attached sheet. Comments are welcome and will be appreciated.
### 1. Throughput/Speed

<table>
<thead>
<tr>
<th></th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Medium</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Slow</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

---

### 2. Adaptability

<table>
<thead>
<tr>
<th></th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

---

### 3. Accuracy

<table>
<thead>
<tr>
<th></th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Nominal</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

---
<table>
<thead>
<tr>
<th>4. Reliability</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

<table>
<thead>
<tr>
<th>5. Safety Factor</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

<table>
<thead>
<tr>
<th>6. Technology Stability</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Unstable</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:
### 7. Interface

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

### 8. Testing Confidence

<table>
<thead>
<tr>
<th>Confidence</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:

### 9. Allowable Error Tolerance

<table>
<thead>
<tr>
<th>Tolerance</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Overall value of Factor: VI MI NI

Specific Comments or Justification:
<table>
<thead>
<tr>
<th>Factor</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. Schedule</td>
<td>Tight</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td></td>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td></td>
<td>Lacks</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI MI NI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>11. Development Cost</td>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Expectation</td>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI MI NI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factor</th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>12. Requirement Risk</td>
<td>High</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td></td>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI MI NI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware</td>
<td>Software</td>
<td>Firmware</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>13. Security</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td>秘密</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI</td>
<td>MI</td>
<td>NI</td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>14. Required Documentation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensive</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Moderate</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Minimal</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI</td>
<td>MI</td>
<td>NI</td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Hardware</th>
<th>Software</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>15. Minimize Life Cycle Cost</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
<td></td>
</tr>
<tr>
<td>Overall value of Factor:</td>
<td>VI</td>
<td>MI</td>
<td>NI</td>
</tr>
<tr>
<td>Specific Comments or Justification:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relative Factor Rating

Please rate the following factors on a relative scale of importance with 1 being the most important and 15 being the least important. Each factor should be rated number only once. Factors:

<table>
<thead>
<tr>
<th>Factor</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptability</td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
</tr>
<tr>
<td>Safety Factor</td>
<td></td>
</tr>
<tr>
<td>Technology Stability</td>
<td></td>
</tr>
<tr>
<td>Interface Requirement</td>
<td></td>
</tr>
<tr>
<td>Testing Confidence</td>
<td></td>
</tr>
<tr>
<td>Allowable Error</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td></td>
</tr>
<tr>
<td>Development Cost Expectation</td>
<td></td>
</tr>
<tr>
<td>Requirement Risk</td>
<td></td>
</tr>
<tr>
<td>Security Classification</td>
<td></td>
</tr>
<tr>
<td>Required Documentation</td>
<td></td>
</tr>
<tr>
<td>Life Cycle Cost</td>
<td></td>
</tr>
</tbody>
</table>

Appendix A: Questionnaire
General Comments/Request for Other Factors

Please provide your comments on this questionnaire. Specifically a request is made for other factors that in your opinion are important in the hardware/software/firmware decision process.
Clarification of Factors

1. **Throughput/Speed** - The function timing performance requirement relative to the state-of-the-system.

2. **Adaptability** - Planned or anticipated modification. Includes possibility of unplanned required change and preplanned product improvement (P3I).

3. **Accuracy** - The precision constraint on the function, including tolerance in deviation acceptable.

4. **Reliability** - The requirement for sustaining function operation over a period of time.

5. **Safety Factor** - The relative safety of flight critical or life sustaining critical attribute of the function.

6. **Technology Stability** - The confidence of successful function implementation based on previous industry experience. Is the function been proved? Is the organization familiar with the planned implementation?

7. **Interface Requirement** - The amount of interaction with other functions. Including the complexity of the interaction with other functions.

8. **Testing Confidence** - The significance of the guarantee that the function will meet the specified accuracy with a certain precision.

9. **Allowable Error** - The required probability of the function being performed as specified.

10. **Schedule** - The aggressiveness of the time allotted to function design, implementation, and test.

11. **Development Cost Expectation** - The relative expected dollar value of the function development. Specifically, if one has more money would hardware, software, or firmware be preferred.

12. **Requirement Risk** - The confidence of meeting the performance requirement of the function as directly related to function complexity, schedule constraints, and extraneous requirements.

13. **Security Classification** - The security classification level of the function. How does the classification effect the allocation of a function to hardware, software, or firmware in order to protect the sensitive information.

14. **Required Documentation** - The type and quantity of documentation required on the function design effort.

15. **Life Cycle Cost** - In order to minimize the cost to design, test, operate, and maintain the function which implementation medium would be chosen.
Appendix B: Hardware, Software, Firmware Allocation Tool Code

```pascal
(* HSF_ALLOC is the main program for the hardware, software, and firmware allocation support tool. *)
(* It sets up the main menu, allows operator procedure selection, and performs the procedure. *)

PROGRAM HSF_ALLOC;

CONST null = ' ';

TYPE
  string80 = string[80];
  matrix = ARRAY[1..16,1..16] OF real;
  eigmat = ARRAY[1..16] OF real;
  fxeig = ARRAY[1..3] OF real;
  fxmat = ARRAY[1..3,1..3] OF real;

(* Factor attribute record *)
attrb = RECORD
  name: string80;
  hw: ARRAY[1..3] OF integer;
  sw: ARRAY[1..3] OF integer;
  fw: ARRAY[1..3] OF integer;
  hwavg: real;
  swavg: real;
  fwavg: real;
  wgt: integer;
  rank: integer;
END;

(* Function characteristic record *)
funct = RECORD
  name: string80;
  attribz ARRAY[1..16] of integer;
END;

(* GLM result record *)
meth1 = RECORD
  name: string80;
  hdw: real;
```

Appendix B: Hardware, Software, Firmware Allocation Tool Code 109
(F AHP Factor record F)

ahp_fact = RECORD
  name: string80;
  mat_fx: fxmat;
END;

(F AHP Function record F)

ahp_fx = RECORD
  name: string80;
  factor: ARRAY[1..16] OF ahp_fact;
END;

(F AHP results record F)

ahp_rlt = RECORD
  name: string80;
  pr1_fact: eigmat;
  pr1_fx: ARRAY[1..16] OF fxel;
  hdu: real;
  sfu: real;
  fmu: real;
END;

VAR
  funct_num: integer;
  finish: boolean;

(F Initialize the programs file representations F)

wgtfile: FILE OF attrib;
fx_file: FILE OF funct;
m1_file: FILE OF method;
afac_f1le: FILE OF matrix;
aftx_f1le: FILE OF ahp_fx;
apr1_f1le: FILE OF eigmat;
ahp_f11: FILE OF ahp_rlt;

PROCEDURE DRAWBOX(x1•y1•x2•y2: integer);
  (F Draw a Box F)
  var i: integer;
  begin
    gotoxy(x1•y1);
    for i := x1 to x2 do write("-'");
    for i := y1+1 to y2 do
      begin
        gotoxy(x1•i); write('`
        gotoxy(x2•i); write('`
      end;
    gotoxy(x1•y2);
    for i := x1 to x2 do write('-');
  end;  (* End Draw Box F *)

PROCEDURE DISPLAY;
  (* DISPLAY presents the HW, SW, and FW *)
  (* allocation main menu procedure options *)
  begin
    clrscr;
    drawbox(5,1,40,20);
  end;
PROCEDURE SELECT(var funct_num: integer;
n: integer);
(* SELECT enables the selection of a procedure from the menu and includes validity check of input. *)

VAR
   response: string[80];
   code: integer;
   accept: boolean;

BEGIN
  gotoxy(10,23);
  write('Enter Function Number:');
  gotoxy(35,23);
  repeat
    gotoxy(35,23);
    clrscr;
    gotoxy(35,23);
    read(response);
    val(response, funct_num, code);
    accept:=true;
    if code <> 0 then accept:=false;
    if funct_num < 1 then accept:=false;
    if funct_num > n then accept:=false;
    if accept = false then
      begin
        gotoxy(35,24);
        clrscr;
        write('Invalid input: try again.');
      end;
  until accept = true;
end; (* Select Procedure *)

(*$I A: PRT.INC*)
(*$I A: GLN.INC*)
(*$I A: ANP.INC*)

PROCEDURE HELP;
(* HELP displays the main menu help screen. *)
VAR
  line: string80;
  dump: text;
BEGIN
  ASSIGN(dump,'MM.HLP');
  RESET(dump);
  CLRSCR;
  GOTOXY(2,2);
  WHILE NOT(EOF(dump)) DO
    BEGIN
      READLN(dump, line);
      WRITE(line);
    END;
  GOTOXY(4,25); WRITE('Press any key to continue.');
  REPEAT UNTIL KEYPRESSED;
  CLOSE(dump);
END;

BEGIN
  PROCEDURE PERFORM(funct_num: integer);
    (* PERFORM uses the main menu selection to *)
    (* call for the performance of the selected *)
    (* procedure. *)
  begin
    gotoxy(1,25);
    case funct_num of
      1: SET_HEIGHT; (* Set Height Procedure *)
      2: IN_DATA; (* Input Procedure *)
      3: METHONE; (* Method 1 Procedure *)
      4: AHP_METH; (* Method 2 Procedure *)
      5: PRINT; (* Print Choices Screen *)
      6: finish := true; (* Quit Procedure *)
      7: Help; (* Help Procedure *)
    end;
    (* Perform Procedure *)
  end;

END. (* MAIN PROGRAM *)

(* MAIN PROGRAM *)

(* Run HSF allocation software support tool.*)
(* 1. Display main menu *)
(* 2. Select procedure *)
(* 3. Perform procedure *)
(* 4. Repeat until finished *)
BEGIN (* MAIN *)
  finish := false;
  repeat
    display;
    select(funct_num, 7);
    perform(funct_num);
  until finish = true;
END. (* MAIN *)

PROCEDURE FACT_HELP;
  (* FACT_HELP displays the factor help screen *)

VAR
  line: string80;
  dump: text;
BEGIN
ASSIGN(dump,'FACT.HLP');
RESET(dump);
CLRSCR;
GOTOXY(2,2);
WHILE NOT(EOF(dump)) DO
BEGIN
READLN(dump,line);
WRITELN(line);
END;
GOTOXY(4,25); WRITE('Press any key to continue.');
REPEAT UNTIL KEYPRESSED;
CLOSE(dump);
END!

PROCEDURE FXT_HELP;
(* FX_HELP displays the function input help screen *)
(*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
TYPE
string80 = string[80];
VAR
line: string80;
dump: text;
BEGIN
ASSIGN(dump,'FX.HLP');
RESET(dump);
CLRSCR;
GOTOXY(2,2);
WHILE NOT(EOF(dump)) DO
BEGIN
READLN(dump,line);
WRITELN(line);
END;
GOTOXY(4,25); WRITE('Press any key to continue.');
REPEAT UNTIL KEYPRESSED;
CLOSE(dump);
END!

PROCEDURE SET_NGHT;
(*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
PROCEDURE Wgt_Screen;
(*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
VAR
index: integer;
finl: boolean;
ext: boolean;
modify: boolean;
factor: attrib;
value: string80;
(*Wrst: 1111111111111111111111111111111111111111111111111111111111111111*)
Appendix B: Hardware, Software, Firmware Allocation Tool Code 113
BEGIN
CLRSCR;
GOTOXY(2,0); WRITE('SET MODEL TRADE-OFF FACTOR WEIGHTS');
GOTOXY(49,7); WRITE('1 => Almost Certainly  => 95%');
GOTOXY(49,8); WRITE('2 => More Than Likely  => 75%');
GOTOXY(49,9); WRITE('3 => Uncertain  => 50%');
GOTOXY(49,10); WRITE('4 => More Than Likely Not  => 35%');
GOTOXY(49,11); WRITE('5 => Almost Certainly Not  => 5%');
GOTOXY(2,23); WRITE('Type "help" for Instructions');
GOTOXY(2,24); WRITE('Type "quit" to Exit');
GOTOXY(40,23); WRITE('Type "next" for next factor');
GOTOXY(40,24); WRITE('Type "del" to delete factor');
GOTOXY(4,3);
CLRSCR;
WRITE('Factor Name: '); 
GOTOXY(17,3); WRITE(''); 
GOTOXY(6,5); WRITE('Level'); 
GOTOXY(4,7); WRITE('High'); 
GOTOXY(4,9); WRITE('Moderate'); 
GOTOXY(6,11); WRITE('Low'); 
GOTOXY(17,5); WRITE('Hardware'); 
GOTOXY(27,5); WRITE('Software'); 
GOTOXY(37,5); WRITE('Firmware'); 
GOTOXY(18,7); WRITE(''); 
GOTOXY(28,7); WRITE(''); 
GOTOXY(38,7); WRITE(''); 
GOTOXY(18,9); WRITE(''); 
GOTOXY(28,9); WRITE(''); 
GOTOXY(38,9); WRITE(''); 
GOTOXY(18,11); WRITE(''); 
GOTOXY(28,11); WRITE(''); 
GOTOXY(38,11); WRITE(''); 
GOTOXY(4,14); WRITE('Weight'); 
GOTOXY(12,16); WRITE(''); 
GOTOXY(4,16); WRITE('Ranking'); 
GOTOXY(12,16); WRITE(''); 
GOTOXY(49,13); WRITE('VI => Very Important'); 
GOTOXY(49,14); WRITE('MI => Moderately Important'); 
GOTOXY(49,15); WRITE('NI => Not Important'); 
GOTOXY(20,16); WRITE('Input: VI, MI, or NI'); 
GOTOXY(20,16); WRITE('Input: 1 to 15'); 
GOTOXY(2,18); WRITE('Helpful Messages: '); 
END;

PROCEDURE Get_Elem;
(* Get_Elem retrieves factors from wgtfile for modification. One factor is retrieved on each call. The factor information is written to the wgt_screen. *)

VAR
\( x: \text{integer}; \)  
\( y: \text{integer}; \)  
\( i: \text{integer}; \)

BEGIN
    READ(wgtfile,factor);  
    SEEK (wgtfile,index);  
    gotoxy(2,2); write(index);  
    index := index + 1;  
    GOTOXY(4,3);  
    CLREOL();  
    WRITE('Factor Name: ',factor.name);  
    x := 19; y := 7;  
    FOR i := 1 TO 3 DO  
        BEGIN  
            GOTOXY(x,y); WRITE(factor.hw[i]);  
            GOTOXY(x+10,y); WRITE(factor.sw[i]);  
            GOTOXY(x+20,y); WRITE(factor.fw[i]);  
            y := y + 2  
        END;  
    GOTOXY(4,14);  
    (* Convert integer to character for display *)  
    IF factor.wgt = 1 THEN WRITE('Weight VI')  
    ELSE IF factor.wgt = 2 THEN WRITE('Weight MI')  
    ELSE IF factor.wgt = 3 THEN WRITE('Weight NI')  
    ELSE WRITE('Weight error')  
    GOTOXY(4,16);  
    WRITE('Ranking ',factor.rank,' ');  
    modify := true;  
END;

PROCEDURE Del_Fact;  
(* Del_Fact enables the deletion of the factors. *)  
(* The entire wgtfile file is written over. *)  
VAR  
    temp_wgt: FILE OF atrib;  
BEGIN  
    ASSIGN(temp_wgt,'temp.fil');  
    REWRITE(temp_wgt);  
    RESET(wgtfile);  
    WHILE NOT(EOF(wgtfile)) DO  
        BEGIN  
            READ(wgtfile,factor);  
            IF (index <> FILEPOS(wgtfile)) THEN  
                WRITE(temp_wgt,factor);  
        END;  
    RESET(temp_wgt);  
    REWRITE(wgtfile);  
    WHILE NOT(EOF(temp_wgt)) DO  
        BEGIN  
            READ(temp_wgt,factor);  
            WRITE(wgtfile,factor);  
        END;  
    CLOSE(temp_wgt);  
END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
PROCEDURE Get_Name;
BEGIN
VAR
accept: boolean;
value: string80;
BEGIN
REPEAT
accept := true;
GOTOXY(17,3); READ(value);
IF (value = 'quit') OR (value = 'QUIT')
THEN fin := true
ELSE IF (value = 'help') OR (value = 'HELP') THEN
BEGIN
FACT_HELP;
accept := false;
END
ELSE IF (value = 'next') OR (value = 'NEXT')
THEN BEGIN
next := true;
WRITE(wgtfile,factor);
END
ELSE IF (value = 'del') OR (value = 'DEL') THEN
BEGIN
next := true;
DE1_Fact;
index := index -1;
SEEK(wgtfile,index);
END
ELSE IF (value = null) THEN
IF (modify <> true) THEN
accept := false
ELSE
BEGIN
GOTOXY(17,3); WRITE(factor.name,'___');
accept := true;
END
ELSE BEGIN
CLREOL;
factor.name := value;
END;
UNTIL accept = true;
END;
PROCEDURE Get_hw;
BEGIN
VAR
accept: boolean;
value: string80;
BEGIN
REPEAT
accept := true;
GOTOXY(17,3); READ(value);
IF (value = 'quit') OR (value = 'QUIT')
THEN fin := true
ELSE IF (value = 'help') OR (value = 'HELP') THEN
BEGIN
FACT_HELP;
accept := false;
END
ELSE IF (value = 'next') OR (value = 'NEXT')
THEN BEGIN
next := true;
WRITE(wgtfile,factor);
END
ELSE IF (value = 'del') OR (value = 'DEL') THEN
BEGIN
next := true;
DE1_Fact;
index := index -1;
SEEK(wgtfile,index);
END
ELSE IF (value = null) THEN
IF (modify <> true) THEN
accept := false
ELSE
BEGIN
GOTOXY(17,3); WRITE(factor.name,'___');
accept := true;
END
ELSE BEGIN
CLREOL;
factor.name := value;
END;
UNTIL accept = true;
END;
VAR
  x: integer;
  y: integer;
  i: integer;
  accept: boolean;
  value: string80;
BEGIN
  x := 19;
  y := 7;
FOR i := 1 TO 3 DO
  IF (fin1<> true) AND (next<> true) THEN BEGIN
    accept := true;
    GOTOXY(x,y); READ(value);
    IF (value = 'quit') OR (value = 'QUIT') THEN fin1 := true
    ELSE IF (value = null) AND (modify = true) THEN
      BEGIN
        accept := true;
        GOTOXY(x,y); WRITE(factor.hw[1], '__');
      END
  END
  GOTOXY(6,19); WRITE('ERROR: TRY AGAIN');
  accept := false;
UNTIL accept = true;
  y := y + 2;
  factor.hwavg := factor.hwavg + factor.hw[1];
END;
  factor.hwavg := factor.hwavg/3.03
END;

PROCEDURE Get_sw;
  (* Get_sw inputs "factor.sw[1..3]" weights for (*)
  (* 1 = high, 2 = moderate, and 3 = low factor (*)
  (* levels. The procedure supports quit, help, (*)
  (* and next commands. Quit will not save factor (*)
  (* modifications. A null or return indicates no (*)
  (* weight changes to the existing factor (*)
  (*)
  (*)
  (*)
  (*)
  *)
VAR
  x: integer;
  y: integer;
  i: integer;
  accept: boolean;
  value: string80;
BEGIN
FOR i := 1 TO 3 DO
  IF (finish <> true) AND (next <> true) THEN BEGIN
    REPEAT
      accept := true;
      GOTOXY(x, y); READ(value);
      GOTOXY(6, 19); CLREOL; (* Clear any error messages*)
      IF (value = 'quit') OR (value = 'QUIT')
        THEN FINI := true
      ELSE IF (value = null) AND (modify = true) THEN BEGIN
        accept := true;
        GOTOXY(x, y); WRITE(factor.sw[i],', _')
      END
      (* Convert character to integer *)
      ELSE IF value = '1' THEN factor.sw[i] := 1
      ELSE IF value = '2' THEN factor.sw[i] := 2
      ELSE IF value = '3' THEN factor.sw[i] := 3
      ELSE IF value = '4' THEN factor.sw[i] := 4
      ELSE IF value = '5' THEN factor.sw[i] := 5
      ELSE BEGIN
        GOTOXY(6, 19); WRITE('ERROR: TRY AGAIN')
        accept := false;
      END
    UNTIL accept = true;
  y := y + 2;
  factor.swavg := factor.swavg + factor.sw[i];
  END;
  factor.swavg := factor.swavg/3.0;
END;

PROCEDURE Get_fw;
(* *)
(* Get_fw inputs "factor.fw[1..3]" weights for *)
(* 1 = high, 2 = moderate, and 3 = low factor *)
(* levels. The procedure supports quit, help, *)
(* and next commands. Quit will not save factor *)
(* modifications. A null or return indicates no *)
(* weight changes to the existing factor *)
(* *)
VAR
  x: integer;
  y: integer;
  i: integer;
  accept: boolean;
  value: string80;
BEGIN
  x := 39;
  y := 7;
  FOR i := 1 TO 3 DO
    IF (finish <> true) AND (next <> true) THEN BEGIN
      REPEAT
        accept := true;
        GOTOXY(x, y); READ(value);
        GOTOXY(6, 19); CLREOL; (* Clear any error messages*)
        IF (value = 'quit') OR (value = 'QUIT')
          THEN FINI := true
        ELSE IF (value = null) AND (modify = true) THEN
          (* *)
BEGIN
    accept := true;
    GOTOXY(x,y); WRITE(factor.fw[1],'__');
END

(* Convert character to integer *)
ELSE IF value = '1' THEN factor.fw[1] := 1
ELSE IF value = '2' THEN factor.fw[1] := 2
ELSE IF value = '3' THEN factor.fw[1] := 3
ELSE IF value = '4' THEN factor.fw[1] := 4
ELSE IF value = '5' THEN factor.fw[1] := 5
ELSE BEGIN
    GOTOXY(6,19); WRITE('ERROR: TRY AGAIN');
    accept := false;
END;
UNTIL accept = true;
y := y + 2;
factor.fwavg := factor.fwavg + factor.fw[1];
END;

factor.fwavg := factor.fwavg/3.0;
END;

PROCEDURE Get_Hgt;
(* Get_Hgt inputs "factor.Hgt" rates. *)
(* The procedure supports quit, help, *)
(* and next commands. Quit will not save factor *)
(* modifications. A null or return indicates no *)
(* weight changes to the existing factor *)
(* *)
BEGIN
    x := 14;
y := 14;

    IF (fini<> true) AND (next<> true) THEN BEGIN
        REPEAT
            accept := true;
            GOTOXY(x,y); READ(value);
            GOTOXY(6,19); CLREOL; (* Clear any error messages *)
            IF (value = 'quit') OR (value = 'QUIT')
                THEN FINI := true
            ELSE IF (value = null) AND (modify = true) THEN
                BEGIN
                    accept := true;
                    GOTOXY(x,y);
                (* Convert integer to character *)
                IF factor.wgt = 1 THEN WRITE('VI ')
                ELSE IF factor.wgt = 2 THEN WRITE('MI ')
                ELSE IF factor.wgt = 3 THEN WRITE('NI ')
                ELSE WRITE('Height error');
            END
        END
        (* Convert character to integer *)
        ELSE IF (value = 'VI') OR (value = 'vi')
            THEN factor.wgt := 1
        ELSE IF (value = 'MI') OR (value = 'mi')
            THEN factor.wgt := 2
    END
END;
ELSE IF (value = 'NI') OR (value = 'nl')
THEN factor.wgt := 3
ELSE BEGIN
    GOTOXY(6,19); WRITE('ERROR: TRY AGAIN');
    accept := false;
END;
UNTIL accept = true;
END;

PROCEDURE Get_rank;
BEGIN
VAR
    x: integer;
y: integer;
    accept: boolean;
    value: string80;
BEGIN
    x := 16;
y := 16;
    IF (fini<> true) AND (next<> true) THEN BEGIN
        REPEAT
            accept := true;
            GOTOXY(x,y); READ(value);
            IF (value = 'quit') OR (value = 'QUIT')
            THEN FINI := true
            ELSE IF (value = null) AND (modify = true) THEN
                BEGIN
                    accept := true;
                    GOTOXY(x,y); WRITE(factor.rank,'__');
                END
            ELSE IF value = '1' THEN factor.rank := 1
            ELSE IF value = '2' THEN factor.rank := 2
            ELSE IF value = '3' THEN factor.rank := 3
            ELSE IF value = '4' THEN factor.rank := 4
            ELSE IF value = '5' THEN factor.rank := 5
            ELSE IF value = '6' THEN factor.rank := 6
            ELSE IF value = '7' THEN factor.rank := 7
            ELSE IF value = '8' THEN factor.rank := 8
            ELSE IF value = '9' THEN factor.rank := 9
            ELSE IF value = '10' THEN factor.rank := 10
            ELSE IF value = '11' THEN factor.rank := 11
            ELSE IF value = '12' THEN factor.rank := 12
            ELSE IF value = '13' THEN factor.rank := 13
            ELSE IF value = '14' THEN factor.rank := 14
            ELSE IF value = '15' THEN factor.rank := 15
            ELSE BEGIN
                GOTOXY(6,19); WRITE('ERROR: TRY AGAIN');
                accept := false;
            END;
        UNTIL accept = true;
    END;
END;

BEGIN
(* Procedure SET_WGT .....................................................*)

Appendix B: Hardware, Software, Firmware Allocation Tool Code

120
ASSIGN(wgtfile, 'model.wgt');

(* Request type of session: new input or modifications *)
CLRSCR;
GOTOXY(2,2); WRITE('Do you wish to input new functions or?');
GOTOXY(2,3); WRITE('modify the existing function file ?');
GOTOXY(2,4); WRITE('(* modify, I=input new *)');
GOTOXY(3,4); READ(value);

(* If new input clear wgtfile file. *)
(* If modification then reset wgtfile. *)
IF (value[1] = 'I') OR (value[1] = 'I') THEN
  REWRITE(wgtfile)
ELSE
  RESET(wgtfile);

fini := false; (* Initialize variables *)
index := 0;

REPEAT
  modify := false; (* Initialize variables *)
  next := false;
  factor.hwavg := 0.0;
  factor.swavg := 0.0;
  factor.fwavg := 0.0;
  Ngt_Screen; (* Print blank screen for weights *)
  IF NOT(EOF(wgtfile)) THEN (* Retrieve stored factor element *)
    Get_ELEM; (* Write factor attributes to screen *)
    Get_name; (* Input factor.name *)
    Get_hw; (* Input factor.hw[1..3] weights *)
    Get_sw; (* Input factor.sw[1..3] weights *)
    Get-fw; (* Input factor.hw[1..3] weights *)
    Get_wgt; (* Input factor.wgt rate *)
    Get_rank; (* Input factor.rank *)
    Ngt_Screen;
    IF (fini<> true) AND (next<> true) THEN
      WRITE(wgtfile, factor);
  UNTIL (fini = true);
  CLOSE(wgtfile);

END; (* PROCEDURE SET_WGHT................................. *)

(**************************************************************************

(* PROCEDEURE IN_DATA; *)

PROCEDURE IN_DATA;

(* Procedure IN_DATA manages the GLM FUNCTIONS. *)
(* The procedure allows input of new functions, *)
(* modification of existing functions, and deletion *)
(* in the fx_file file which represents the disk *)
(* file funct.dta *)

(**************************************************************************

VAR
  numatt: integer;
  i: integer;
  index: integer;
  fini: boolean;
  accept: boolean;
  modify: boolean;
  next: boolean;

Appendix B: Hardware, Software, Firmware Allocation Tool Code 121
PROCEDURE Del_Funct;

PROCEDURE FX_Scrn;

VAR

BEGIN

ASSIGN(temp_fx,'temp.fil');
REWRITE(temp_fx);
RESET(fx_file);
WHILE NOT(EOF(fx_file)) DO
BEGIN
READ(fx_file,fx);
IF (index <> FILEPOS(fx_file)) THEN
WRITE(temp_fx,fx);
END;
RESET(temp_fx);
REWRITE(fx_file);
WHILE NOT(EOF(temp_fx)) DO
BEGIN
READ(temp_fx,fx);
WRITE(fx_file,fx);
END;
CLOSE(temp_fx);
END;

VAR

att: integer;
BEGIN
CLRSCR;
GOTOXY(2,0); WRITE('FUNCTION INPUT SCREEN');
GOTOXY(2,25); WRITE('Type "help" for Instructions');
GOTOXY(2,24); WRITE('Type "quit" to Exit');
GOTOXY(50,25); WRITE('Type "next" for next factor');
GOTOXY(50,24); WRITE('Type "del" to delete factor');
GOTOXY(2,3); WRITE('Enter: H for High');
GOTOXY(2,6); WRITE(' M for Moderate');
GOTOXY(2,5); WRITE(' L for Low');
GOTOXY(5,7); WRITE('Function Name');
GOTOXY(25,7); WRITE('_');
att := 0;
y := 9;
RESET(wgtfile);
WHILE NOT(EOF(wgtf1le))) DO
BEGIN
  READ(wgtf1le,factor);
  att := att + 1;
  IF att = 9 THEN y := 9;
  IF att > 8 THEN x := 45 ELSE x := 5;
  GOTOXY(x,y); WRITE(factor.name);
  x := x + 20;
  GOTOXY(x,y); WRITE('____');
  y := y + 2;
END;
END; (*End Write Screen*)

PROCEDURE Get_Elem;
 (* Get_Elem retrieves functions from fx_file for modification. One function is retrieved on each call. The function information is written to the FX_screen. *)

VAR
  x: integer;
  y: integer;
  att: integer;
BEGIN
BEGIN
  READ(fx_file,fx);
  SEEK(fx_file,index);
  gotoxy(2,2); WRITE(index);
  index := index + 1;
  GOTOXY(27,7); WRITE(fx.name);
  att := 0;
  y := 9;
  RESET(wgtf1le);
  WHILE NOT(EOF(wgtf1le))) AND (att < 17) DO BEGIN
  READ(wgtf1le,factor);
  att := att + 1;
  IF att = 9 THEN y := 9;
  IF att > 8 THEN x := 45 ELSE x := 5;
  GOTOXY(x,y); WRITE(factor.name);
  x := x + 20;
  GOTOXY(x,y);
  (* Convert integer to character for display *)
  IF fx.attrib[att] = 1 THEN WRITE('H')
  ELSE IF fx.attrib[att] = 2 THEN WRITE('M')
  ELSE IF fx.attrib[att] = 3 THEN WRITE('L')
  END;
END;
END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
ELSE WRITE('error');

y := y + 2;
END;
modify := true;
END;
END;

INest 1

PROCEDURE Get_FXname;
(*
(* Get_FXname inputs "fx.name". It supports quit, *
(* help, next, and delete commands. A null or return *
(* input indicates no changes to the existing data. *
(* *
(*
(*
(*
INest 1 *)

VAR
x: integer;
y: integer;
accept: boolean;
value: string80;
BEGIN
x := 27;
y := 7;
REPEAT
accept := true;
GOTOXY(x,y); READ(value);
IF (value = 'quit') OR (value = 'QUIT')
THEN fini := true
ELSE IF (value = 'help') OR (value = 'HELP') THEN
BEGIN
FXT_HELP;
accept := false;
index := index -1
CLRSCR;
FX_Scrn /
IF NOT(file) THEN
Get_Elem
ELSE modify := false;
END
ELSE IF (value = 'next') OR (value = 'NEXT')
THEN BEGIN
next := true;
WRITE(file,fx);
END
ELSE IF (value = 'del') OR (value = 'DEL') THEN
BEGIN
next := true;
Del_Funct;
index := index -1
SEEK(file,index);
END
ELSE IF (value = null) THEN
IF (modify <> true) THEN accept := false
ELSE
BEGIN
accept := true;
GOTOXY(x,y); WRITE(fx.name,'___');
END
ELSE fx.name := value;
UNTIL accept = true;
END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code 124
PROCEDURE Get_Attrib

(* Get_Attrib inputs the level of each factor for the functions: high, moderate, and low. *)

VAR
    x: integer;
    y: integer;
    att: integer;
    accept: boolean;
    value: string80;

BEGIN
    att := 0;
    y := 7;
    IF (fini <> true) AND (next <> true) THEN
        REPEAT
            y := y + 2;
            att := att + 1;
            IF att = 9 THEN y := 9;
            IF att > 8 THEN x := 67
                         ELSE x := 27;

        REPEAT
            accept := true;
            GOTOXY(x,y); READ(value);
            IF (value = 'quit') OR (value = 'QUIT')
                THEN fini := true
            ELSE IF (value = 'H') OR (value = 'h')
                THEN fx.attrib[att] := 1
            ELSE IF (value = 'M') OR (value = 'm')
                THEN fx.attrib[att] := 2
            ELSE IF (value = 'L') OR (value = 'l')
                THEN fx.attrib[att] := 3
            ELSE IF (value = null) AND (modify = true) THEN
                BEGIN
                    GOTOXY(x,y);
                    IF fx.attrib[att] = 1 THEN
                        WRITE('H')
                    ELSE IF fx.attrib[att] = 2 THEN
                        WRITE('M')
                    ELSE IF fx.attrib[att] = 3 THEN
                        WRITE('L');
                    accept := true;
                END
            ELSE
                BEGIN
                    WRITE('___ ERROR');
                    accept := false;
                END;
            END;
        UNTIL accept = true;
    UNTIL (att = numatt) OR (fini = true);

END (* Procedure IN_DATA.......................................................... *)

ASSIGN(wgtfile,'model.wgt');  (* Set-up function factors *)
RESET(wgtfile);
numatt := FILESIZE(wgtfile);
ASSIGN(fx_file,'funct.dta');
/* Check for existence of factors in ugtfile file */
CLRSCR;
IF EOF(ugtfile) THEN
BEGIN
  GOTOXY(4,6);
  WRITE('ERROR: no factors on which to rate function. ');
  GOTOXY(4,6);
  WRITE('Press any key to return to menu. ');
  REPEAT UNTIL KEYPRESSED;
END
ELSE

/* Request type of session: input new or modify */
BEGIN
CLRSCR;
GOTOXY(2,2); WRITE('Do you wish to input new functions or ');
GOTOXY(2,3); WRITE('modify the existing function file ? ');
GOTOXY(2,4); WRITE(' (M=modify, I=input new) ');
GOTOXY(3,4); READ(value);

/* If new input clear fx_file file. */
/* If modification then reset fx_file. */
IF (value[1] = 'I') OR (value[1] = 'I') THEN
  REWRITE(fx_file);
ELSE
  RESET(fx_file);
FOR i := 1 TO 15 DO
  fx.attrib[i] := 0;  /*initialize attribute array to zero*/
index := 0;  /* initialize variables */
modify := false;
finit := false;
REPEAT
  next := false;
  FX_Scn;
  IF NOT(EOF(fx_file)) THEN
    (* Print the function input screen *)
    Get_ELEM
    /* Get function & write factor characteristics */
    ELSE modify := false;
    Get_FXname;
    /* Input or modify fx.name */
    Get_Attrib;
    /* Input or modify factor level */
  (* Write function input and modifications to file unless function aborted *)
  IF (finit <> true) AND (next <> true) THEN
    WRITE(fx_file,fx);
  UNTIL finit = true;
END;
CLOSE(fx_file);
CLOSE(ugtfile);
END;  /* PROCEDURE IN_DATA. */

PROCEDURE METHONE;
/* METHOD performs the GLM calculation. */
/* */
VAR
att: integer;
wgtrec: attrb;
fx_rec: funct;
ml_rec: method;
hsf_tot: real;

BEGIN (# Method One #)

ASSIGN(ml_file,'meth1.rst');
REWRITE(ml_file);

ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);

ASSIGN(fx_file,'funct.dta');
RESET(fx_file);

(\# Check for existence of factor \#)
CLRSCR;
IF EOF(wgtfile) THEN
BEGIN
GOTOXY(4,6); |
WRITE('ERROR: no factors on which to calculate Hdw, Sfw, Fmw weights');
GOTOXY(4,6); |
WRITE('Press any key to return to menu.\');
REPEAT UNTIL KEYPRESSED;
END
ELSE IF EOF(fx_file) THEN
BEGIN
GOTOXY(4,6); |
WRITE('ERROR: no functions on which to calculate Hdw, Sfw, Fmw weights');
GOTOXY(4,6); |
WRITE('Press any key to return to menu.\');
REPEAT UNTIL KEYPRESSED;
END
ELSE
REPEAT
(\# For each function \#)
READ(fx_file,fx_rec);
att := 1;
(\# Initialize variables \#)
ml_rec.hdw := 0.0;
ml_rec.sfw := 0.0;
ml_rec.fmw := 0.0;
RESET(wgtfile);
READ(wgtfile,wgtrec);

(\# For each function calculate the Hdw, Sfw, and \#)
(\# Fmw weights using the GLM equation \#)
WHILE NOT(EOF(wgtfile)) DO
BEGIN

ml_rec.hdw := ml_rec.hdw +
(5.0/wgtrec.hw[fx_rec.attrib[att]]) *
(3.0/wgtrec.wgt) *
(16.0·wgtrec.rank);

ml_rec.sfw := ml_rec.sfw +
(5.0/wgtrec.sw[fx_rec.attrib[att]]) *
(3.0/wgtrec.wgt) *
(16.0·wgtrec.rank);

ml_rec.fmw := ml_rec.fmw +
(5.0/wgtrec.fw[fx_rec.attrib[att]]) *
(3.0/wgtrec.wgt) *
(16.0·wgtrec.rank);

END

Appendix B: Hardware, Software, Firmware Allocation Tool Code
att := att + 1;
READ(wgtfile, attrec); ("Next factor")
END;

("Normalize the Hdw, Sw, Fm weights and convert to percentages")
hsf_tot := ml_rec.hdw + ml_rec.sw + ml_rec.fmw;
ml_rec.hdw := 100 * ml_rec.hdw/hsf_tot;
ml_rec.sw := 100 * ml_rec.sw/hsf_tot;
ml_rec.fmw := 100 * ml_rec.fmw/hsf_tot;

("Display results for each function one at a time")
WRITE(ml_file, ml_rec);

CLRSCR;
GOTOXY(3, 8); WRITE('FUNCTION NAME: ', fx_rec.name);
GOTOXY(5, 10); WRITE('Hardware: ', ml_rec.hdw);
GOTOXY(5, 12); WRITE('Software: ', ml_rec.sw);
GOTOXY(5, 14); WRITE('Firmware: ', ml_rec.fmw);

IF NOT(EOF(fx_file)) THEN
    BEGIN
        GOTOXY(5, 18); WRITE('Press any key to for next function');
        REPEAT UNTIL KEYPRESSED;
    END;
ELSE
    BEGIN
        GOTOXY(5, 18); WRITE('Press any key to return');
        REPEAT UNTIL KEYPRESSED;
    END;
UNTIL EOF(fx_file);

CLOSE(wgtfile);
CLOSE(fx_file);
CLOSE(ml_file);
END; ("END PROCEDURE METHOD ONE")

PROCEDURE EIGENFAC(var x: elgmat; ("eigenvector matrix: initial & final")
a: matrix; ("n x n matrix to find eigenvector of")
n: integer); ("number of rou & columns in matrix A")

("EIGENFAC Calculates the eigenvector")
("for the level 0 factor matrices")
("using POWER.")

CONST
TOL = 0.001;

VAR
y: ARRAY[1..16] OF real; ("Array workspace variable")
yscale: real;
temp: real;
estold: real;
estnew: real;
i: integer;
j: integer;
m: integer;
iterm: integer;
lambda: real;
itr: integer;

BEGIN

Appendix B: Hardware, Software, Firmware Allocation Tool Code 128
\[ m := 10; \]

\[
\text{FOR } i := 1 \text{ TO } n \text{ DO}
\]
\[
\begin{align*}
\& x[i] := 1; \\
\& \text{itr} := 1;
\end{align*}
\]

\((^* \text{ Calculate the initial EIGENVALUE approximation } ^*)\)

\[
\text{FOR } i := 1 \text{ TO } n \text{ DO}
\]
\[
\begin{align*}
\& \text{BEGIN} \\
\& \quad y[i] := 0.0; \\
\& \quad \text{FOR } j := 1 \text{ TO } n \text{ DO} \\
\& \quad \quad y[i] := y[i] + a[i,j] \cdot x[j]; \\
\& \quad \text{END}; \\
\& \quad \text{temp} := 0.0; \\
\& \quad \text{yscale} := 0.0; \\
\& \text{END};
\end{align*}
\]

\[
\text{yscale} := \text{SQRT}(\text{yscale});
\]

\((^* \text{ Power method iteration with scaling } ^*)\)

\[
\text{IF } (\text{yscale} = 0.0) \text{ THEN}
\]
\[
\begin{align*}
\& \text{BEGIN} \\
\& \quad \text{CLRSCR}; \\
\& \quad \text{GOTOXY}(10,10); \text{ WRITELN('ERROR: ZERO MATRIX');} \\
\& \quad \text{WRITE('Press any key to return');} \\
\& \quad \text{REPEAT UNTIL KEYPRESSED;}
\& \text{END};
\end{align*}
\]

\[
\text{ELSE}
\]
\[
\begin{align*}
\& \text{BEGIN} \\
\& \quad \text{REPEAT}
\end{align*}
\]
\[
\begin{align*}
\& \quad \text{estold} := \text{temp}; \\
\& \quad \text{itr} := \text{itr} + 1; \\
\& \quad \text{FOR } i := 1 \text{ TO } n \text{ DO} \\
\& \quad \quad x[i] := y[i]/\text{yscale}; \\
\& \quad \text{FOR } i := 1 \text{ TO } n \text{ DO}
\end{align*}
\]
\[
\begin{align*}
\& \quad \text{BEGIN} \\
\& \quad \quad y[i] := 0.0; \\
\& \quad \quad \text{FOR } j := 1 \text{ TO } n \text{ DO} \\
\& \quad \quad \quad y[i] := y[i] + a[i,j] \cdot x[j]; \\
\& \quad \quad \text{END}; \\
\& \quad \quad \text{temp} := 0.0; \\
\& \quad \quad \text{yscale} := 0.0; \\
\& \quad \text{END};
\end{align*}
\]
\[
\begin{align*}
\& \quad \text{FOR } i := 1 \text{ TO } n \text{ DO}
\end{align*}
\]
\[
\begin{align*}
\& \quad \text{BEGIN} \\
\& \quad \quad \text{temp} := \text{temp} + y[i] \cdot x[i]; \\
\& \quad \quad \text{yscale} := \text{yscale} + y[i] \cdot y[i]; \\
\& \quad \quad \text{YES} \\
\& \quad \quad \text{END}; \\
\& \quad \text{yscale} := \text{SQRT}((\text{yscale})); \\
\& \quad \text{estnew} := \text{temp};
\end{align*}
\]
\[
\begin{align*}
\& \text{UNTIL } (\text{ABS}(\text{estnew} - \text{estold}) < \text{tol}) \text{ OR } (\text{itr} > m);
\end{align*}
\]
lambda := estnew;

temp := 0.0;
FOR i := 1 TO n DO
    temp := temp + x[i];
END;

FOR i := 1 TO n DO
    x[i] := x[i]/temp;
END;

CLRSCR;
GOTOXY(10,10); WRITE(lambda); FOR i := 1 TO n DO
    BEGIN
        GOTOXY(15,i); WRITE(x[i]/temp); END;
END;
GOTOXY(10,20); WRITE('PRESS ANY KEY TO CONTINUE');
REPEAT UNTIL KEYPRESSED;
END;

BEGIN

m := 10;
FOR i := 1 TO n DO
    x[i] := 1;
    itr := 1;
END;

BEGIN

m := 10;
FOR i := 1 TO n DO
    x[i] := 1;
    itr := 1;
END;

PROCEDURE EIGENFXT(var x: fxe1g; (N eigenvector matrix: initial & final )
a: fxmatj; (N 3 x 3 matrix to find eigenvector of )
n: integer); (N number of row & columns in matrix A )

BEGIN

m := 10;
FOR i := 1 TO n DO
    x[i] := 1;
    itr := 1;
END;

END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
yscale := 0.0;

FOR i := 1 TO n DO
  BEGIN
    temp := temp + y[i] * x[i];
    yscale := yscale + y[i] * y[i];
  END;

yscale := SQRT(yscale);

(* Power method iteration with scaling *)

IF (yscale = 0.0) THEN
  BEGIN
    CLRSCR;
    GOTOXY(10,10); WRITELN('ERROR: ZERO MATRIX');
    WRITE('Press any key to return');
    REPEAT UNTIL KEYPRESSED;
  END
ELSE
  BEGIN
    REPEAT
      estold := temp;
      itr := itr + 1;
      FOR i := 1 TO n DO
        x[i] := y[i]/yscale;
      FOR i := 1 TO n DO
        BEGIN
          y[i] := 0.0;
          FOR j := 1 TO n DO
            y[i] := y[i] + a[i][j] * x[j];
        END;
        temp := 0.0;
        yscale := 0.0;
        FOR i := 1 TO n DO
          BEGIN
            temp := temp + y[i] * x[i];
            yscale := yscale + y[i] * y[i];
          END;
        yscale := SQRT(yscale);
        estnew := temp;
      UNTIL (ABS(estnew - estold) < tol) OR (itr > m);

lambda := estnew;

temp := 0.0;
FOR i := 1 TO n DO
  temp := temp + x[i];
FOR i := 1 TO n DO
  x[i] := x[i]/temp;

(*
CLRSCR;
GOTOXY(10,10); WRITE(lambda);
FOR i := 1 TO n DO
  BEGIN
    GOTOXY(15,i); WRITE(x[i]/temp:2:3);
  END;
GOTOXY(10,20); WRITE('PRESS ANY KEY TO CONTINUE');
REPEAT UNTIL KEYPRESSED; *)
PROCEDURE Convert_prt(x: real);
    "Convert_prt converts the real x value to a character string for display."
BEGIN
    IF x <= 1.0/9.0 THEN WRITE('1/9')
    ELSE IF x <= 1.0/8.0 THEN WRITE('1/8')
    ELSE IF x <= 1.0/7.0 THEN WRITE('1/7')
    ELSE IF x <= 1.0/6.0 THEN WRITE('1/6')
    ELSE IF x <= 1.0/5.0 THEN WRITE('1/5')
    ELSE IF x <= 1.0/4.0 THEN WRITE('1/4')
    ELSE IF x <= 1.0/3.0 THEN WRITE('1/3')
    ELSE IF x <= 1.0/2.0 THEN WRITE('1/2')
    ELSE IF x <= 1.0 THEN WRITE('1')
    ELSE IF x <= 2.0 THEN WRITE('2')
    ELSE IF x <= 3.0 THEN WRITE('3')
    ELSE IF x <= 4.0 THEN WRITE('4')
    ELSE IF x <= 5.0 THEN WRITE('5')
    ELSE IF x <= 6.0 THEN WRITE('6')
    ELSE IF x <= 7.0 THEN WRITE('7')
    ELSE IF x <= 8.0 THEN WRITE('8')
    ELSE WRITE('9');
END;

PROCEDURE AHP_Scrn;
    "AHP_Scrn displays the generic AHP rating schema for pairwise comparative judgements"
BEGIN
    CLRSCR;
    DRAWBOX(2,4,50,12);
    GOTOXY(2,1); WRITE('AHP Factor Matrix Input');
    GOTOXY(2,3); WRITE('Scale of Relative Importance');
    GOTOXY(4,5); WRITE('Intensity');
    GOTOXY(30,5); WRITE('Definition');
    GOTOXY(9,6); WRITE('1');
    GOTOXY(25,6); WRITE('Equal Importance');
    GOTOXY(9,7); WRITE('3');
    GOTOXY(25,7); WRITE('Moderate Importance');
    GOTOXY(9,8); WRITE('5');
    GOTOXY(25,8); WRITE('Essential or Strong');
    GOTOXY(9,9); WRITE('7');
    GOTOXY(25,9); WRITE('Very Strong Importance');
    GOTOXY(9,10); WRITE('9');
    GOTOXY(25,10); WRITE('Absolute Importance');
    GOTOXY(4,11); WRITE('2,4,6,8');
    GOTOXY(25,11); WRITE('Intermediate Values');
END;

PROCEDURE CH_to_Real(VAR x: real;
    VAR accept: boolean;
    value: string80);
    "CH_to_Real changes the input string"
IF(value = '1') THEN x := 1.0
ELSE IF(value = '2') THEN x := 2.0
ELSE IF(value = '3') THEN x := 3.0
ELSE IF(value = '4') THEN x := 4.0
ELSE IF(value = '5') THEN x := 5.0
ELSE IF(value = '6') THEN x := 6.0
ELSE IF(value = '7') THEN x := 7.0
ELSE IF(value = '8') THEN x := 8.0
ELSE IF(value = '1/2') THEN x := 1.0/2.0
ELSE IF(value = '1/3') THEN x := 1.0/3.0
ELSE IF(value = '1/4') THEN x := 1.0/4.0
ELSE IF(value = '1/5') THEN x := 1.0/5.0
ELSE IF(value = '1/6') THEN x := 1.0/6.0
ELSE IF(value = '1/7') THEN x := 1.0/7.0
ELSE IF(value = '1/8') THEN x := 1.0/8.0
ELSE IF(value = '1/9') THEN x := 1.0/9.0
ELSE accept := false;
END;

PROCEDURE Mod_Matrix(var i,j: integer; var fin1:boolean);
BEGIN
  (* Request modification to matrix: yes or no. If yes then input i=row and j=col values for element to modify. If no then return no modification code -1 *)
  VAR
    value: string80;
    accept: boolean;
    valid: boolean;
    code: integer;
  BEGIN
    (* Request matrix modification = 1 if desired get i,j element *]
    GOTOXY(2,24);
    WRITE('Do you wish to modify an element? (Y=yes, N=no, or quit)');
    GOTOXY(4,24); WRITE('Y=yes, N=no, or quit');
    REPEAT
      valid := true;
      GOTOXY(39,24); READ(value);
      IF(value = 'QUIT') OR (value = 'quit') THEN
        fin1 := true
    (* If matrix modification requested get i & j, row & col values *)
    ELSE IF(value[1] = 'y') OR (value[1] = 'Y') THEN
      BEGIN
        GOTOXY(2,24); CLREDL;
        GOTOXY(2,23);
        WRITE('Which a(i,j) element would you like to change? ');
        GOTOXY(4,24); WRITE('i = ___ j = ___');
      (* Read and check validity of i value than convert to integer *)
      REPEAT
        accept := true;
      END;
  END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code

133
GOTOXY(10,24); READ(value);
IF(value = 'QUIT') OR (value = 'quit') THEN
  fin1 := true
ELSE IF(value = 'help') OR (value = 'HELP') THEN
  accept := false
ELSE
  BEGIN
    val(value, code);
    IF (code <> 0) OR (1 > FILE$IZE(wgfile)) THEN
      BEGIN
        GOTOXY(35,24);
        WRITE('ERROR, try again!!');
        accept := false;
      END;
    END;
UNTIL (accept = true) OR (f1n1 = true);

IF (fin1 <> true) THEN
  REPEAT
    accept := true;
    GOTOXY(20,24); READ(value);
    IF(value = 'QUIT') OR (value = 'quit') THEN
      fin1 := true
    ELSE IF(value = 'help') OR (value = 'HELP') THEN
      accept := false
    ELSE
      BEGIN
        val(value, code);
        IF (code <> 0) OR (j > FILESIZE(wgfile)) OR (j = 1) THEN
          BEGIN
            GOTOXY(35,24);
            WRITE('ERROR, try again!!');
            accept := false;
          END;
        END;
    END;
UNTIL (accept = true) OR (fin1 = true);
END

(* If no modification requested return NO modification code -1 *)
ELSE IF (value = 'n') OR (value = 'N') THEN
  i := -1
ELSE
  BEGIN
    GOTOXY(35,24);
    WRITE('ERROR, try again!!');
    valid := false;
  END;
UNTIL (valid = true) OR (fin1 = true);
END;

(******************************************************************************)
PROCEDURE AHP_FACTOR;
(******************************************************************************)

(* AHP_FACTOR develops the level 0 *)
(* factor comparative judgement matrix *)
(******************************************************************************)

VAR
  i: integer;
  j: integer;
  empty: boolean;
  fin1: boolean;
  done: boolean;
  value: string80;
  mat_fact: matrix;
  factor: attr;

Appendix B: Hardware, Software, Firmware Allocation Tool Code 134
PROCEDURE Query(x: integer; y: integer);

(* Query displays the generic factor *)
(* pairwise comparison question format *)

BEGIN
  GOTOXY(x,y); WRITE('How much more important is _________ than');
  GOTOXY(x,y+2); WRITE('___________ with respect to implementing');
  GOTOXY(x,y+4); WRITE('a function in Hardware, Software, or');
  GOTOXY(x,y+6); WRITE('Firmware? ______ (Use above rating scale)');
END;

PROCEDURE IN_AFACI;

(* IN_AFAC writes the factors in the query *)
(* question in a priority order by high rank. *)
(* The procedure reads the comparison value as *)
(* a character string and converts it to a real *)
(* number. *)

VAR
  factor1: attrib;
  factor2: attrib;
  k: integer;
  m: integer;
  i: integer;
  j: integer;
  index: integer;
  accept: boolean;
  fini: boolean;
  level0: boolean;

BEGIN
  (* Initialize ARRAYs to 1.0 *)
  FOR i := 1 TO 16 DO
    FOR j := 1 TO 16 DO
      mat_fact[i,j] := 1.0;
  (* Initialize variables *)
  level0 := true;
  fini := false;
  index := FILESIZE(wgtfile) - 1;
  (* Make the n(n-1)/2 comparisons *)
  (* 1. Determine for each pair of factors the priority by rank for query display *)
  FOR k := 0 TO (index - 1) DO
    IF fini = false THEN
      BEGIN
        SEEK(wgtfile,k);
        READ(wgtfile,factor1);
        FOR m := (k+1) TO index DO
          IF fini = false THEN
            BEGIN
              SEEK(wgtfile,m);
              READ(wgtfile,factor2);
              GOTOXY(33,14); WRITE('___________');
              GOTOXY(5,16); WRITE('Firmware? ______ (Use above rating scale)');
              IF (factor1.rank < factor2.rank) THEN
                BEGIN
                  GOTOXY(33,14); WRITE(factor1.name);
                  GOTOXY(5,16); WRITE(factor2.name);
                END;
            END;
          END;
        END;
      END;
    END;
END;
BEGIN

ELSE

BEGIN

GOTOXY(33,14); WRITE(factor2.name);
GOTOXY(5,16); WRITE(factor1.name);
i := m+1;
j := k+1

END

ELSE

BEGIN

GOTOXY(33,14); WRITE(factor2.name);
GOTOXY(5,16); WRITE(factor1.name);
i := m+1;
j := k+1

END

END

END

PROCEDURE Dis_fact_mat;

(# Dis_fact_mat displays the level0 factor #)
(# matrix. Factors are listed by number. #)
(# If the matrix is blank (new) blank lines #)
(# will be displayed for each i,j element #)

VAR

i: integer;
j: integer;
x: integer;
y: integer;
index: integer

BEGIN

RESET(afac_fil);
RESET(ugtfil);

(# Initialize variables #)
empty := true;
index := FILESIZE(ugtfil);
y := 2;
x := 1;

(# Determine if any factors are still in the file for modification #)
IF NOT(EOF(afac_fil)) THEN
BEGIN

READ(afac_fil,mrat_fact);

END

Appendix B: Hardware, Software, Firmware Allocation Tool Code
empty := false;
END;

CLRSCLR;
GOTOXY(2,1); WRITE('AHP Factor Matrix Input');

(* List the factors and provide number, 1 ... n, representation *)
FOR i := 1 TO index DO
BEGIN
IF i = 5 THEN
BEGIN
x := 10; y := 2;
END;
IF i = 9 THEN
BEGIN
x := 36; y := 2;
END;
IF i = 13 THEN
BEGIN
x := 54; y := 2;
END;
READ(weight_file,factor);
GOTOXY(x,y); WRITE('AHP Factor Matrix Input');
END;
BEGIN
x := 7;
y := 8;

(* Print out the level 0 matrix *)
GOTOXY(2,7); WRITE('Relational Matrix');

(* 1. Display column indices 1 = 1 ... n *)
FOR i := 1 TO index DO
BEGIN
GOTOXY(x,y); WRITE(i);
x := x + 4;
END;

(* 2. For each row display index number and border symbol *)
FOR i := 1 TO index DO
BEGIN
x := 3;
y := y + 1;
GOTOXY(x,y); WRITE(i,'');
x := 6;
END;

(* 3. For each column of the row display a space for number *)
(* input or the value of the i,j element of the matrix. *)
FOR j := 1 TO index DO
BEGIN
GOTOXY(x,y);
IF empty = true THEN
WRITE('___')
ELSE Convert_prt(mat_fact[i,j]);
x := x + 4;
END;
END;

PROCEDURE Lev0_Mod;
BEGIN
(* Lev0_Mod enables modification of the *)
(* level 0 factor matrix. It checks for *)
(* valid factor indices. The display screen *)
(* AHP Factor Matrix Input *)
(* Relational Matrix *)
(* 1 = 1 ... n *)
(* i *)
(* x = x + 4 *)
(* 3 *)
(* 6 *)
(* i,j *)
(* empty = true *)
(* ___ *)
(* Convert_prt(mat_fact[i,j]) *)
(* x = x + 4 *)
END;
END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
VAR
  factor1: attri;
  factor2: attri;
  accept: boolean;
BEGIN
  (* Check that the i and j indices are within the factor range. *)
  IF (i <= FILESIZE(wgtfile)) AND (j <= FILESIZE(wgtfile)) THEN BEGIN
    (* Display comparative judgement weighting scale *)
    AHP_Screen;
    (* Get factors to modify from the wgtfile file *)
    SEEK(wgtfile,i-1); READ(wgtfile,factor1);
    SEEK(wgtfile,j-1); READ(wgtfile,factor2);
    (* Display the comparative judgment question *)
    QUERY(4,14);
    GOTOXY(33,14); WRITE(factor1.name);
    GOTOXY(35,16); WRITE(factor2.name);
    (* Read in the value of the comparison *)
    REPEAT
      accept := true;
      GOTOXY(22,20); WRITE('______');
      GOTOXY(24,20); READ(value);
      IF(value = 'QUIT') OR (value = 'quit') THEN
        fini := true
      ELSE IF(value = 'help') OR (value = 'HELP') THEN
        accept := false
      ELSE IF(value = null) AND (empty = false) THEN
        accept := true
    END
    (* Convert character value to real *)
    ELSE CH_to_Real(mat_fact[i,j],accept,value);
    mat_fact[j,i] := 1.0/mat_fact[i,j];
    UNTIL accept = true;
  END
  (* Indices out of range *)
  ELSE WRITE('ERROR');
END;
BEGIN (* Procedure AHP_Fact.................................*)
  done := false;
  ASSIGN(sfac_fil,'fact.ahp');
  ASSIGN(wgtfile,'model.ugt');
  RESET(wgtfile);
  (* Check for the existence of at least two factors to compare *)
  CLS;
  IF (FILESIZE(wgtfile) < 2) THEN BEGIN
    GOTOXY(4,6); WRITE('ERROR: no factors to compare.');
    GOTOXY(4,6); WRITE('Press any key to return to AHP menu');
    REPEAT UNTIL KEYPRESSED;
  END;
END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
BEGIN
  CLRSCR;
  GOTOXY(2,2); WRITE('Do you wish to input a new factor matrix or');
  GOTOXY(2,3); WRITE('modify the current "fact.ahp" file?');
  GOTOXY(2,4); WRITE('M=modify, I=input new');
  GOTOXY(2,5); WRITE('note: choose modify to exit');
  GOTOXY(3,5); READ(value);

  IF(value[1] = 'I') THEN
    BEGIN
      RENWRITE(fac_fil);
      AHP_Screen;
      QUERY(4,14);
      IN_AFact;
      END;
  END;

  IF(value[1] = 'I') THEN
    BEGIN
      REPEAT
        Dis_fact_mat;
        Mod_matrix(i,j,finit);
        IF i = -1 THEN
          done := true
        ELSE
          Lev0_mod;
          RESET(fac_fil);
          WRITE(fac_fil,mat_fact);
          UNIL done = true;
        END;
      END;

  END;

END

PROCEDURE AHP_FUNCT;

VAR
  index: integer;
  fact: integer;
  i: integer;
  j: integer;
  fini: boolean;
  empty: boolean;
  done:boolean;
  fx: ahp_fx;
  factor: attrib;
  fx_rec: funct;
  medium: ARRAY[1..31] of string80;
  state: string80;
  accept: boolean;
  all: boolean;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
PROCEDURE QUEST(x: integer; y:integer);

  (# Quest displays the generic function, #)
  (# factor hardware, software, and firmware #)
  (# pairwise comparison question format. #)
BEGIN
  GOTOXY(x,y-2); WRITE('Function: ');
  GOTOXY(x,y); WRITE('For the factor how more advantageous');
  GOTOXY(x,y+2); WRITE('is implementation than ');
  GOTOXY(x,y+6); WRITE(' (Use above rating scale)');
END;

PROCEDURE Get_Val;

  (# Get_Val gets the comparative value input #)
  (# and converts it to real #)
VAR
  accept: boolean;
  value: string80;
BEGIN
  (# Read in the comparative judgement. #)
  REPEAT
    accept := true;
    GOTOXY(22,22); WRITE(' ');
    GOTOXY(24,22) READ(value);
    IF(value = 'QUIT') OR (value = 'quit') THEN
      accept := true
    ELSE IF(value = null) AND (empty = false) THEN
      accept := true
    ELSE
      BEGIN
        CH_to_Real(fx.factor[2*fact].mat_fx[i,j],accept,value);
      END;
    END;
    (# Input error detected. #)
    IF accept = false THEN
      BEGIN
        GOTOXY(30,22) WRITE('ERROR: try again');
      END;
  UNTIL accept = true;
END;

PROCEDURE Dis_fx_mat;

  (# Dis_fx_mat displays the level 1 function #)
  (# factor hardware, software, and firmware #)
  (# matrix. Hdw, sw and fww are represented #)
  (# by number. The matrices are presented one #)
  (# at a time. #)
  (# If the matrix is blank (new) blank lines #)
  (# will be displayed for each i,j element #)
VAR
BEGIN

(* Display function and factor name for the level 1 matrix *)
CLRSR;
GOTOXY(2,1); WRITE('AHP Function _____________ Matrix Input');
GOTOXY(17,1); WRITE(fx.name);
SEEK(wgtfile,fact-1);
READ(wgtfile,factor);
GOTOXY(2,4); WRITE('Factor: ',factor.name);
GOTOXY(2,6); WRITE('Relational Matrix');

(* Display row and column indices. *)
GOTOXY(4,8); WRITE('1_');
GOTOXY(12,8); WRITE('2_');
GOTOXY(20,8); WRITE('3_');
GOTOXY(4,9); WRITE(' ');
GOTOXY(4,10); WRITE('1');
GOTOXY(2,11); WRITE('1');
GOTOXY(4,12); WRITE('1');
GOTOXY(2,13); WRITE('2');
GOTOXY(4,14); WRITE('2');
GOTOXY(2,15); WRITE('2');
GOTOXY(4,16); WRITE('2');
GOTOXY(2,17); WRITE('3');
GOTOXY(4,18); WRITE('3');

(* Display identity of 1, 2, 3 representations *)
GOTOXY(40,11); WRITE('1 = Hardware');
GOTOXY(40,14); WRITE('2 = Software');
GOTOXY(40,17); WRITE('3 = Firmware');

x := 8;
y := 11;

(* For each column of the row display a space for number *)
(* input or the value of the i,j element of the matrix. *)
FOR i := 1 to 3 DO
BEGIN
  FOR j := 1 to 3 DO
  BEGIN
    GOTOXY(x,y);
    IF (empty = true) THEN
      WRITE('__________');
    ELSE
      Convert_Prt(fx.factor[fact].mat_fx[i,j]);
    x := x + 8;
    END;
  y := y + 3;
  x := 8;
  END;
END;

(*NEST 1 111111111111111111111111111111111111111111111111111111111111111111 *)
PROCEDURE IN_AFXT;
(* IN_AFXT writes hardware, software, firmware *)
(* in the quest question in a priority order by *)
(* high factor level average (hwavg, swavg, & fwavg *)
(* The procedure uses procedure Get_val to read the *)

Appendix B: Hardware, Software, Firmware Allocation Tool Code
VAR
  i: integer;
  j: integer;
BEGIN

/* Display function name */
  GOTOXY(14,14); WRITE(fx_rec.name);
/* Display factor name */
  READ(wgtfile,factor);
  GOTOXY(14,16); WRITE(factor.name);

/* Prioritize h/w, s/w, and f/w by average of factor level values */
/* Present in quest in priority order. Get_val of comparison. */
IF factor.hwavg > factor.swavg THEN
  BEGIN
    GOTOXY(8,18); WRITE('hardware');
    GOTOXY(39,18); WRITE('software');
    i := 1;
    j := 2;
  END
ELSE
  BEGIN
    GOTOXY(8,18); WRITE('software');
    GOTOXY(39,18); WRITE('hardware');
    i := 2;
    j := 1;
  END;
Get_val;
IF fini <> true THEN
  BEGIN
    IF factor.hwavg > factor.fwavg THEN
      BEGIN
        GOTOXY(8,18); WRITE('hardware');
        GOTOXY(39,18); WRITE('firmware');
        i := 1;
        j := 3;
      END
    ELSE
      BEGIN
        GOTOXY(8,18); WRITE('firmware');
        GOTOXY(39,18); WRITE('hardware');
        i := 3;
        j := 1;
      END;
    Get_val;
  END;
IF fini <> true THEN
  BEGIN
    IF factor.swavg > factor.fwavg THEN
      BEGIN
        GOTOXY(8,18); WRITE('software');
        GOTOXY(39,18); WRITE('firmware');
        i := 2;
        j := 3;
      END
    ELSE
      BEGIN
        GOTOXY(8,18); WRITE('firmware');
        GOTOXY(39,18); WRITE('software');
        i := 3;
        j := 2;
      END;
    Get_val;
  END;
END.
PROCEDURE Lev1_Mod(1, j: integer);

  (* Lev1_Mod enables the modification of *)
  (* the level 1 matrices. For a specific *)
  (* function factor matrix the i,j element *)
  (* is modified. *)

BEGIN

(* For the factor display the AHP screen and request the comparison of *)
(* medium i and medium j. Get the value of the comparison. *)
SEEK(wgtfile, fact-1);
READ(wgtfile, factor);
AHP_Scrn;
QUEST(4,16);
GOTOXY(14,14); WRITE(fx.name);
GOTOXY(14,16); WRITE(factor.name);
GOTOXY(8,18); WRITE(medium[i]);
GOTOXY(39,18); WRITE(medium[j]);
Get_Val;
END;

PROCEDURE Get_Hod_Fx;

(* Get_Hod_Fx allows the selection of a *)
(* level 1 matrices set to view and modify *)
(* based on function selection including *)
(* all the functions. Returns index of Fx *)

VAR
  accept: boolean;
  i: integer;
  code: integer;
  value: string80;

BEGIN

(* Request selection of level 1 matrix to work with. *)
CLRSCR;
REPEAT
  accept := true;
  all := false;

  GOTOXY(4,2); WRITE('Which function would you like ');
  WRITE('to modify ?');
  RESET(afx_tfil);

(* Display the list of functions and a number representation *)
(* for each (maximum of 80 functions) *)
FOR i := 1 TO FILESIZE(afx_tfil) DO BEGIN
  READ(afx_tfil, fx);
  IF i < 20 THEN BEGIN
    GOTOXY(8,i+3); WRITE(i, ' = ', fx.name);
  END ELSE IF i < 40 THEN BEGIN
    GOTOXY(18, i+3); WRITE(i, ' = ', fx.name);
  END
END

Appendix B: Hardware, Software, Firmware Allocation Tool Code
ELSE IF 1 < 60 THEN
BEGIN
  GOTOXY(28,i+3); WRITE(1,' = ',fx.name);
END;
ELSE IF 1 < 80 THEN
BEGIN
  GOTOXY(38,i+3); WRITE(1,' = ',fx.name);
END;
END;

GOTOXY(8,FILESIZE(afxt_f11)+1+3)
WRITE(FILESIZE(afxt_f11)+1,' = All Functions');
GOTOXY(8,FILESIZE(afxt_f11)+2+3)
WRITE(FILESIZE(afxt_f11)+2,' = None (Quit)');

(* Get number of function for level 1 matrix to view *)
IF FILESIZE(afxt_f11) < 20 THEN
BEGIN
  GOTOXY(4,FILESIZE(afxt_f11)+4+3);WRITE(' ( input the number of the choice )');
  GOTOXY(5,FILESIZE(afxt_f11)+4+3); READ(value);
END ELSE
BEGIN
  GOTOXY(4,25); WRITE(' ( input the number of the choice )');
  GOTOXY(5,25); READ(value);
END;

(* Validate selection and convert from character to real *)
VAL(value,index,code);
IF (code <> 0) OR (index > (FILESIZE(afxt_f11)+2)) THEN
BEGIN
  GOTOXY(4,3); WRITE('ERROR: try again');
  accept := false;
END ELSE IF (index = (FILESIZE(afxt_f11) + 2)) THEN
  finl := true
ELSE IF (index = (FILESIZE(afxt_f11) + 1)) THEN
BEGIN
  all := true;
  index := 0;
END ELSE
index := index - 1;
UNTIL (accept = true) OR (finl = true);
END;

(*NEST 1

PROCEDURE New_Fx;

(* New_Fx enables the input of new function level 1 *)
(* factor hdw, sfw, & fsw matrices. For each GLM *)
(* function, the GLM factors are presented along with *)
(* the comparative judgement request. *)

VAR
  fact: integer;
BEGIN

(* Initialize ARAYS to 1.0 *)
FOR fact := 1 TO 16 DO
  FOR i := 1 TO 3 DO
    FOR j := 1 TO 3 DO
      fx.factor[fact].mat_fx[i,j] := 1.0;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
(* For each selected function, read the function from the GLM function file *)
WHILE (NOT(EOF(fx_file))) AND (fini = false) DO
BEGIN
  READ(fx_file,fx_rec);
  fx.name := fx_rec.name;
  empty := true;
  AHP_Scrn;
  RESET(wgtfile);
END;

(* For each factor display comparative judgement question and input comparison *)
FOR fact := 1 TO FILESIZE(wgtfile) DO
  IF fini <> true THEN
  BEGIN
    QUEST(4,16);
    IN_AFXT;
  END;
END;

(* Store the functions in factor hw, sw, fw matrices in afxt_file file *)
WRITE(afxt_file,fx);
END;

BEGIN (*AHP_FUNCTION........................................................*)

(* Initialize variables *)
fini := false;
all := false;
medium[1] := 'Hardware';

(* Set up files *)
ASSIGN(afxt_file,'fxt.ahp');
RESET(afxt_file);

ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);

ASSIGN(fx_file,'funct.dta');
RESET(fx_file);

(* Check for existing factors *)
CLRSCR;
IF EOF(wgtfile) THEN
BEGIN
  GOTOXY(4,6); WRITE('ERROR: no factors to compare !!');
  GOTOXY(4,6); WRITE('Press any key to return to AHP menu');
  REPEAT UNTIL KEYPRESSED;
END
ELSE
BEGIN

(* Request session type or state as new input or modify existing *)
CLRSCR;
GOTOXY(2,2); WRITE('Do you wish to input new level 1 matrices or?');
GOTOXY(2,3) I WRITE('modify the current "fxt.ahp" file?');
REPEAT
  accept := true;
  GOTOXY(2,5); WRITE('M=modify, I=input new');
  GOTOXY(2,6); WRITE('note: choose modify to exit');
  GOTOXY(3,5); READ(state);
END

(* If new file then for each function factor matrices request *)
(* the HW, SW, and FW comparisons and store in Afxt_file file. *)
IF(state[1] = 'I') OR (state[1] = 'I') THEN
Neu_Fx

(* If file modification do nothing now *)
ELSE IF(state[1] = 'M') OR (state[1] = 'm') THEN

(* Invalid state *)
ELSE

BEGIN
  GOTOXY(2,7); WRITE('ERROR: try again');
  accept := false;
END;

UNTIL accept OR fin1;

(* Allow selection of function to modify including all the functions *)
(* If all the functions then increment index until all functions presented *)
(* instead of requesting another function *)
REPEAT
  IF ALL THEN
    index := index + 1
  ELSE
    Get_Mod_fx;
    IF index = FILESIZE(afxt_file) THEN
      fin1 := true;
  END;

(* Allow modification to the level 1 matrices if session is not finished. *)
IF fin1 <> true THEN
  BEGIN
    SEEK(afxt_file,index); 
    RESET(wgtfile); 
  
    empty := true;
  END;

(* Get Level 1 function factor matrices. *)
IF NOT(EOF(afxt_file)) THEN
  BEGIN
    READ(afxt_file,fx);
    empty := false;
  END;

(* For each factor: 1. Display matrix *)
(* 2. Request modification to element i,j *)
(* 3. Make Comparison *)
FOR fact := 1 TO FILESIZE(wgtfile) DO
  IF (fin1 <> true) THEN
    REPEAT
      Dis_fx_mat;
      done := false;
      Mod_matrix(i,j,fin1);
      IF (i = -1) OR (fin1 = true) THEN
        done := true
      ELSE IF (i > 3) OR (j > 3) THEN
        BEGIN
          GOTOXY(4,25);
          WRITE('ERROR: press any key to continue');
          REPEAT UNTIL KEYPRESSED;
        END;
      ELSE
        Lev1_Mod(i,j);
        UNTIL done;
      SEEK(afxt_file,index);
      WRITE(afxt_file,fx);
    END;
  END;

UNTIL fin1;

END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
CLOSE(fx_file);
CLOSE(sfx_file);
CLOSE(wgtfile);

END; (* Procedure AHP_FUNCT.........................................................*)

(*---------------------------------------------------------------------*)
PROCEDURE AHP_CALC;

(* AHP_CALC calculates the AHP recommendations. *)
(* The level 0 and level 1 eigenvectors are *)
(* calculated and normalized into priorities. *)
(* The hardware, software, and firmware composite *)
(* priorities are calculated and displayed one *)
(* function at a time in matrix format. *)

(*---------------------------------------------------------------------*)

VAR
  i: integer;
  j: integer;
  n: integer;
  index: integer;
  mat_fact: matrix;
  mat_fxt: matrix;
  pr1_fxt: eigmat;
  fx: ahp_fx;
  factor: attrib;
  rlt: ahp_rlt;

(*---------------------------------------------------------------------*)
PROCEDURE Initial; (* Initialize priority ARRAYs to 0.0 *)

(*---------------------------------------------------------------------*)
BEGIN
  FOR i := 1 TO 16 DO
    rlt.pr1_fact[i] := 0.0;
  END;
  FOR i := 1 TO 16 DO
    FOR j := 1 TO 3 DO
      rlt.pr1_fx[i][j] := 0.0;
    END;
    rlt.hdu := 0.0;
    rlt.sfu := 0.0;
    rlt.fmu := 0.0;
  END;

(*---------------------------------------------------------------------*)
PROCEDURE Calc_wgts;

(* Calculate the eigenvector of the level 0 *)
(* matrix and the level 1 matrices of a particular *)
(* function. *)

(*---------------------------------------------------------------------*)
BEGIN
  index := FILESIZE(wgtfile);
  EIGENFAC(rlt.pr1_fact,mat_fact,index);
  n := 3;
  FOR i := 1 TO index DO
    BEGIN
      EIGENFXT(rlt.pr1_fx[i],factor[i].mat_fxt,n);
      rlt.hdu := rlt.hdu + rlt.pr1_fact[i]*rlt.pr1_fx[i][1];
      rlt.sfu := rlt.sfu + rlt.pr1_fact[i]*rlt.pr1_fx[i][2];
      rlt.fmu := rlt.fmu + rlt.pr1_fact[i]*rlt.pr1_fx[i][3];
    END;
  END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
rit.hdu := r1t.hdu + 100;
rit.sfu := r1t.sfu + 100;
rit.fmu := r1t.fmu + 100;
END;

(*NEST 1

PROCEDURE Disp_rst;

(* Display the AHP results along with the composite priority matrix

(*NEST 1

VAR
index: integer;
x: integer;
y: integer;
BEGIN
index := FILESIZE(wgtfile);
CLRSCR;
x := 1;
y := 2;

(* Display the factor list with their number representation.

FOR i := 1 TO index DO
BEGIN
IF i = 5 THEN
BEGIN
x := 18; y := 2;
END
ELSE IF i = 9 THEN
BEGIN
x := 36; y := 2;
END
ELSE IF i = 13 THEN
BEGIN
x := 54; y := 2;
END;
READ(wgtfile,factor);
GOTOXY(x;y); WRITE(i,' = ',factor.name);
y := y + 1;
END;

(* Display function name for result presentation.

GOTOXY(4,8); WRITE('Function: ',rit.name);
y := 10;
x := 6;

(* Display the column indices (factors) and the level 0 factor priorities

FOR i := 1 TO index DO
BEGIN
GOTOXY(x+2,10); WRITE(i);
GOTOXY(x,11); WRITE(rit.pr1_fact[i]:1:3);
x := x + 6;
END;

(* Display the row indices

GOTOXY(2,13); WRITE('H');
GOTOXY(2,15); WRITE('S');
GOTOXY(2,17); WRITE('F');
y := 13;

(* For each hdu, sfu, and fmu medium type display the n level 1 priorities

Appendix B: Hardware, Software, Firmware Allocation Tool Code

FOR j := 1 TO 3 DO
BEGIN
x := 6;
FOR i := 1 TO index DO
BEGIN
GOTOXY(x,y); WRITE(rlt.pri_fx[i][j]:1:3);
x := x + 6;
END;
y := y + 2;
END;

(* Display the composite results *)
GOTOXY(10,19); WRITE('HDW Composite: ',rlt.hdw:2:3);
GOTOXY(10,20); WRITE('SFW Composite: ',rlt.sfw:2:3);
GOTOXY(10,21); WRITE('FMW Composite: ',rlt.fmw:2:3);
GOTOXY(22,24); WRITE('Press any key to continue');
REPEAT UNTIL KEYPRESSED;
END!

BEGIN (*PROCEDURE AHP_CALC.................................*)

(* Set-up the files *)
ASSIGN(afac_fil,'fact.ahp');
RESET(afac_fil);
READ(afac_fil,mat_fact);
ASSIGN(afxt_fil,'fxt.ahp');
RESET(afxt_fil);
ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);
ASSIGN(ahp_fil,'ahp.rlt');
RESET(ahp_fil);

(* For each function in afxt_fil file: *)
(* 1. Initialize Arrays *)
(* 2. Assign function name to result name *)
(* 3. Calculate priority vectors *)
(* 4. Display results *)
(* 5. Write the results to ahp_fil file *)
WHILE NOT(EOF(afxt_fil)) DO
BEGIN
  Initial;
  RESET(wgtfile);
  READ(afxt_fil.fx);
  rlt.name := fx.name;
  Calc_wgts;
  Disp_rst;
  WRITE(ahp_fil,rlt);
END;
CLOSE(afac_fil);
CLOSE(afxt_fil);
CLOSE(wgtfile);
CLOSE(ahp_fil);
END;

(*PROCEDURE AHP_AUTO:*)

(* AHP_AUTO calculates the comparative *)
(* judgements for the level 0 and level 1 *)
(* matrices from GLM factor and function *)

Appendix B: Hardware, Software, Firmware Allocation Tool Code  149
VAR
  i: integer;
  j: integer;
  n: integer;
  index: integer;
  mat_fact: matrix;
  fx: ahp_fx;
  factor: ahp_factor;
  rlt: ahp_result;
  fx_rec: function;

PROCEDURE Init;

BEGIN
  FOR i := 1 TO 16 DO
    rlt.pr1_fact[i] := 0.0;
  END;

  FOR i := 1 TO 16 DO
    rlt.hdu := 0.0;
    rlt.sfw := 0.0;
    rlt.fmu := 0.0;
  END;
END;

PROCEDURE Calc_wgts;

BEGIN
  index := FILESIZE(wgtfile);

  EIGENFAC(rlt.pr1_fact,mat_fact,index);

  n := 3;
  FOR i := 1 TO n DO
    BEGIN
      rlt.hdu := rlt.hdu + rlt.pr1_fact[i];
      rlt.sfw := rlt.sfw + rlt.pr1_fact[i];
      rlt.fmu := rlt.fmu + rlt.pr1_fact[i];
    END;

  rlt.hdu := rlt.hdu * 100;
  rlt.sfw := rlt.sfw * 100;
  rlt.fmu := rlt.fmu * 100;
END;

PROCEDURE Disp_rst;

BEGIN
  END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code

150
VAR
index: integer;
x: integer;
y: integer;
BEGIN
index := FILESIZE(wgtfile);
CLRSCR;
x := 1;
y := 2;

(* Display the factor list with their number representation. *)
FOR i := 1 TO index DO
BEGIN
  IF i = 5 THEN
    BEGIN
      x := 18; y := 2;
      END
  ELSE IF i = 9 THEN
    BEGIN
      x := 36; y := 2;
      END
  ELSE IF i = 13 THEN
    BEGIN
      x := 54; y := 2;
      END
  END;
  READ(wgtfile,factor);
  GOTOXY(x,y); WRITE(i,' = ',factor.name);
y := y + 1;
END;

(* Display function name for result presentation. *)
GOTOXY(4,8); WRITE('Function: ',rlt.name); y := 10;
x := 6;

(* Display the column indices (factors) and the level 0 factor priorities *)
FOR i := 1 TO index DO
BEGIN
  GOTOXY(x+2,10); WRITE(i);
  GOTOXY(x,11); WRITE(rlt.pr1_fact[i]:1:3);
x := x + 6;
END;

(* Display the row indices *)
GOTOXY(2,13); WRITE('H');
GOTOXY(2,15); WRITE('S');
GOTOXY(2,17); WRITE('F');
y := 13;

(* For each hdw, sw, and fms medium type display the n level 1 priorities *)
FOR j := 1 TO 3 DO
BEGIN
  x := 6;
  FOR i := 1 TO index DO
  BEGIN
    GOTOXY(x,y); WRITE(rlt.pr1_fx[i][j]:1:3);
x := x + 6;
    END;
y := y + 2;
END;

(* Display the composite results *)

Appendix B: Hardware, Software, Firmware Allocation Tool Code
PROCEDURE AUTO_AFAC!

(* AUTO_AFAC uses the GLM factor data to fill N in the level 0 comparative judgement matrix.*)

VAR
  factor1: atrib;
  factor2: atrib;
  k: integer;
  m: integer;
  i: integer;
  j: integer;
  index: integer;
  accept: boolean;
  fini: boolean;
  level0: boolean;

BEGIN

(* Initialize ARRAYS to 1.0 *)
FOR i := 1 TO 16 DO
  FOR j := 1 TO 16 DO
    mat_fact[i,j] := 1.0;

(* Initialize variables *)
level0 := true;
fini := false;
index := FILESIZE(wgtfile) - 1;

(* For each k,m pair of factors, calculate the comparison value *)
(* using the rank and weight of the factors. *)
FOR k := 0 TO (index - 1) DO
  IF fini = false THEN
    BEGIN
      SEEK(wgtfile,k);
      READ(wgtfile,factor1);
      FOR m := (k+1) TO index DO
        IF fini = false THEN
          BEGIN
            SEEK(wgtfile,m);
            READ(wgtfile,factor2);
            mat_fact[k+1,m+1] := ((16·factor1.rank)·(3/factor1.wgt))/
                          ((16·factor2.rank)·(3/factor2.wgt));
            mat_fact[m+1,k+1] := 1/mat_fact[k+1,m+1];
          END;
    END;

(* Clear afac_fil and write level 0 matrix to the afac_fil file *)
REWRI TE(afac_fil);
WRITE(afac_fil,mat_fact);
END;

PROCEDURE AUTO_AFX!

(* AUTO_AFX calculates the level 0 matrices *)
(* for one function in the GLM fx_fil function *)
VAR
  i: integer;
  j: integer;
  fact: integer;

BEGIN

(* Initialize ARRAYS to 1.0 *)
FOR fact := 1 TO 16 DO
  FOR i := 1 TO 16 DO
    FOR j := 1 TO 3 DO
      fx.factor[fact].mat_fx[i,j] := 1.0;

(* Retrieve function *)
READ(fx_file,fx_rec);
fx.name := fx_rec.name;

(* For each function factor calculate the level 1 matrix using *)
(* the hw, sw, & fw weight attributes. *)
RESET(wgtfile);
FOR fact := 1 TO FILESIZE(wgtfile) DO
  BEGIN
    READ(wgtfile,factor);
    fx.factor[fact].mat_fx[2,1] := (factor.hw[fx_rec.attrib[fact]])/
    (factor.sw[fx_rec.attrib[fact]]);
    fx.factor[fact].mat_fx[1,2] := 1.0/(fx.factor[fact].mat_fx[2,1]);
    fx.factor[fact].mat_fx[3,1] := (factor.hw[fx_rec.attrib[fact]])/
    (factor.fw[fx_rec.attrib[fact]]);
    fx.factor[fact].mat_fx[1,3] := 1.0/(fx.factor[fact].mat_fx[3,1]);
    fx.factor[fact].mat_fx[3,2] := (factor.sw[fx_rec.attrib[fact]])/
    (factor.fw[fx_rec.attrib[fact]]);
    fx.factor[fact].mat_fx[2,3] := 1.0/(fx.factor[fact].mat_fx[3,2]);
  END;

(* Store the function's level 1 matrices *)
WRITE(afxt_file,fx);

BEGIN (*PROCEDURE AHP_AUTO.....................................................*)

(* Set up the files *)
ASSIGN(afac_file,'fact.ahp');
ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);
ASSIGN(ahp_file,'ahp.rlt');
ASSIGN(afxt_file,'fxt.ahp');
ASSIGN(fx_file,'funct.dta');
RESET(fx_file);

(* Check for empty GLM factor or function files *)
CLRSCR;
IF (EOF(wgtfile)) OR (EOF(fx_file)) THEN
  BEGIN
    GOTOXY(4,4);
    WRITE('ERROR: either factors or functions are not entered.');
    GOTOXY(4,6); WRITE('Press any key to return to AHP menu');
    REPEAT UNTIL KEYPRESSED;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
BEGIN

ELSE

(* Perform the automatic AHP calculation *)

BEGIN

(* 1. Clear the AHP result file ahp_fil *)
REWRITE(ahp_fil);

(* 2. Calculate the level 0 matrix *)
AUTO_AFAC;

(* 3. Calculate the level 1 matrix for each function *)
REWRITE(afxt_fil);
WHILE NOT(EOF(fx_file)) DO
AUTO_AFX;

(* 4. Reset level 0 and level 1 matrix files for calculations *)
RESET(afxt_fil);
RESET(afac_fil);
READ(afac_fil,mat_fact);

(* 5. Calculate composite priority *)
WHILE NOT(EOF(afxt_fil)) DO
BEGIN
(* 5a. Initialize Arrays *)
Initial;
RESET(wgtfile);
READ(afxt_fil,fx);
rlt.name := fx.name;
(* 5b. Calculate priority vectors *)
Calc_wgts;
(* 5c. Display Results *)
Disp_rst;
(* 6. Store results by function *)
WRITE(ahp_fil,rlt);
END;
END;

CLOSE(afac_fil);
CLOSE(afxt_fil);
CLOSE(wgtfile);
CLOSE(ahp_fil);
CLOSE(fx_file);

END;

PROCEDURE AHP_HELP;

(* AHP_HELP displays the AHP Analysis *)
(* menu help screen. *)

VAR
    line: string80;
    dump: text;
BEGIN
ASSIGN(dump,'AHP.HLP');
RESET(dump);
CLRSCR;
GOTOXY(2,2);
WHILE NOT(EOF(dump)) DO
BEGIN
READLN(dump,line);
WRITELN(line);

Appendix B: Hardware, Software, Firmware Allocation Tool Code 154
PROCEDURE AHP_METH;

(* AHP_METH manages the operator interface *)
(* with the AHP procedures using a menu *)
(* driven routine. *)

VAR

finish: boolean;

PROCEDURE AHP_DISPLAY;

(* AHP_DISPLAY displays the AHP procedure in *)
(* menu format. *)

begin
clrscr;
drawbox(5,1,40,20);
gotoxy(10,2);
WRITE('Hardware/Software/Firmware');
gotoxy(10,3);
WRITE('Allocation AHP Menu');
gotoxy(7,5);
WRITELN('1. Automatic Calc. from GLM');
gotoxy(7,7);
WRITELN('2. Input and Modify Factors');
gotoxy(11,8);
WRITELN('Level 0 Matrix');
gotoxy(7,10);
WRITELN('3. Input and Modify Functions');
gotoxy(11,11);
wr1teln('Level 1 Matrices');
gotoxy(7,13);
WRITELN('4. AHP Calc. w/mod Matrices');
gotoxy(7,15);
WRITELN('5. Quit AHP Selection');
gotoxy(7,17);
WRITELN('6. Get Help');
end;

PROCEDURE GO_AHP(funct_num: integer);

(* GO_AHP uses menu selection to call for *)
(* performance of the selected procedure. *)

begin

goxy(1,25);
case funct_num of
1: AHP_AUTO;
2: AHP_FACTOR;
3: AHP_FUNCT;
4: AHP_CALC;
5: finish := true; (* Quit Procedure *)
6: AHP_Help; (* Help Procedure *)
end;
end; (* Perform Procedure *)

Appendix B: Hardware, Software, Firmware Allocation Tool Code
BEGIN  (* Procedure AHP_METH. *)
    finish := false;

    (* Run AHP menu *)
    (* 1. Display menu *)
    (* 2. Select function *)
    (* 3. Perform AHP function *)
    (* 4. Repeat until finished *)
    REPEAT
        AHP_DISPLAY;
        SELECT(funct_num);
        GO_AHP(funct_num);
        UNTIL finish = true;

END;

PROCEDURE PRT_FACT;
    (* PRT_FACT sends the GLM factors in wgtfile to the printer for hardcopy in appropriate format *)

VAR
    att: integer;
    wgt_rec: atr1b;

BEGIN
    ASSIGN(wgtfile,'model.wgt');
    RESET(wgtfile);
    IF EOF(wgtfile) THEN
        BEGIN
            CLRSCR;
            GOTOXY(4,4); WRITE('No Factors');
            GOTOXY(4,6); WRITE('Press any key to return to print menu');
            REPEAT UNTIL KEYPRESSED;
        END
    ELSE
        BEGIN
            WRITELN(LST);
            WRITELN(LST,' FACTORS:');
            WRITELN(LST);
            att := 0;
            WHILE (NOT(EOF(wgtfile))) AND (att < 17) DO
                BEGIN
                    READ(wgtfile,wgt_rec);
                    att := att + 1;
                    WRITELN(LST,' att', att,' wgt_rec.name');
                    WRITELN(LST,' Level HDW SFN FMN');
                    WRITELN(LST,' H ', wgt_rec.hw[1],
                    ' ', wgt_rec.sw[1],
                    ' ', wgt_rec.fm[1]);
                    WRITELN(LST,' M ', wgt_rec.hw[2],
                    ' ', wgt_rec.sw[2],
                    ' ', wgt_rec.fm[2]);
                    WRITELN(LST,' L ', wgt_rec.hw[3],
                    ' ', wgt_rec.sw[3],
                    ' ', wgt_rec.fm[3]);
                END;
        END;
    END;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
IF (wgt_rec.wgt = 1) THEN
  WRITELN(LST,' Rating: VI')
ELSE IF (wgt_rec.wgt = 2) THEN
  WRITELN(LST,' Rating: MI')
ELSE IF (wgt_rec.wgt = 3) THEN
  WRITELN(LST,' Rating: NI')
ELSE WRITELN(LST,' Rating: error')!
WRITELN(LST,' Rank: ',wgt_rec.rank);
WRITELN(LST); WRITELN(LST); WRITELN(LST);
END;
CLOSE(wgtfile);
END;

PROCEDURE AHP_PRT;
(* AHP_PRT sets up the AHP files for printing *)
BEGIN
  ASSIGN(afac_file,'fact.ahp');
  ASSIGN(wgtfile,'model.ugt');
  ASSIGN(aftxt_file,'fxt.ahp');
  WRITELN(LST); WRITELN(LST,' AHP MODEL'); WRITELN(LST);
END;

PROCEDURE Prt_Real(x: real);
(* Prt_Real converts a real number to a string and sends it to the printer. *)
BEGIN
  IF x <= 1.0/9.0 THEN HRITE(LST,'1/9')
  ELSE IF x <= 1.0/8.0 THEN HRITE(LST,'1/8')
  ELSE IF x <= 1.0/7.0 THEN HRITE(LST,'1/7')
  ELSE IF x <= 1.0/6.0 THEN HRITE(LST,'1/6')
  ELSE IF x <= 1.0/5.0 THEN HRITE(LST,'1/5')
  ELSE IF x <= 1.0/4.0 THEN HRITE(LST,'1/4')
  ELSE IF x <= 1.0/3.0 THEN HRITE(LST,'1/3')
  ELSE IF x <= 1.0/2.0 THEN HRITE(LST,'1/2')
  ELSE IF x <= 1.0 THEN WRITE(LST,' 1 ')
  ELSE IF x <= 2.0 THEN WRITE(LST,' 2 ')
  ELSE IF x <= 3.0 THEN WRITE(LST,' 3 ')
  ELSE IF x <= 4.0 THEN WRITE(LST,' 4 ')
  ELSE IF x <= 5.0 THEN WRITE(LST,' 5 ')
  ELSE IF x <= 6.0 THEN WRITE(LST,' 6 ')
  ELSE IF x <= 7.0 THEN WRITE(LST,' 7 ')
  ELSE IF x <= 8.0 THEN WRITE(LST,' 8 ')
  ELSE WRITE(LST,' 9 ');
END;

PROCEDURE PRT_LEVO;
(* PRT_LEVO prints the level 0 factor matrix *)
(* at the printer including format symbols *)

Appendix B: Hardware, Software, Firmware Allocation Tool Code
VAR
  i: integer;
  j: integer;
  index: integer;
  factor: ahp_fx;
  mat_fact: matrix;
BEGIN
  AHP_PRT;
  RESET(ugtfile);
  RESET(afac_file);
  index := FILESIZE(ugtfile);
  READ(afac_file,mat_fact);
  WRITELN(LST,'  DECISION FACTORS:');
  WRITELN(LST);
  FOR i := 1 TO index DO
    BEGIN
      READ(ugtfile,factor);
      WRITELN(LST,'  ',”i”. ’,factor.name);
    END;
    WRITELN(LST);
    WRITELN(LST,'  LEVEL 0 MATRIX:');
    WRITELN(LST);
    WRITE(LST,' 
    FOR i := 1 TO INDEX DO
    WRITE(LST,' ‘,’ ‘ ‘
    WRITELN(LST);
    WRITE(LST,
    FOR i := 1 TO INDEX-1 DO
    WRITE(LST,' —— ‘
    WRITELN(LST);
    FOR i := 1 TO INDEX DO
    BEGIN
      WRITE(LST," ‘,”i”. ‘
      FOR j := 1 TO INDEX DO
      BEGIN
        WRITE(LST," ‘
        Prt_Real(mat_fact[i,j]);
      END;
      WRITELN(LST);
      WRITELN(LST);
    END;
    CLOSE(afac_file);
    CLOSE(ugtfile);
    CLOSE(afxt_file);
  END;
PROCEDURE PRT_LEV1;
  (* PRT_LEV1 prints the level 1 matrices by *)
  (* function at the printer including format *)
  (* symbols *)
VAR
  i: integer;
  index: integer;
  fx: ahp_fx;
factor: array;

BEGIN
  AHP_PRT;
  RESET(wgtfile);
  RESET(afxt_file);

  index := FILESIZE(wgtfile);
  WHILE NOT(EOF(afxt_file)) DO
    BEGIN
      RESET(wgtfile);
      READ(afxt_file, fx);
      WRITELN(LST,' Function: ', fx.name);
      WRITELN(LST);
      WRITELN(LST,' LEVEL 1 MATRICES: ');
      WRITELN(LST);
      FOR i := 1 TO index DO
        BEGIN
          READ(afxt_file, factor);
          WRITELN(LST, ' ' , i , ' . ' , factor.name);
          WRITELN(LST);
          WRITELN(LST, HDW SFH FMH ');
          WRITE(LST, HDW ' ); Prt_Real(fx.factor[1].mat_fx[1,1]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[1,2]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[1,3]);
          WRITELN(LST);

          WRITE(LST, ' SFH ' ); Prt_Real(fx.factor[1].mat_fx[2,1]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[2,2]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[2,3]);
          WRITELN(LST);

          WRITE(LST, ' FMH ' ); Prt_Real(fx.factor[1].mat_fx[3,1]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[3,2]);
          WRITE(LST, ' '); Prt_Real(fx.factor[1].mat_fx[3,3]);
          WRITELN(LST);
        END;
    END;
  END;
  CLOSE(afac_file);
  CLOSE(wgtfile);
  CLOSE(afxt_file);
END;

PROCEDURE GLM_PRT;
(* GLM_PRT prints the results of the GLM analysis for each function including *)
(* the factor characteristic of the function *)
(*---------------------------------------------------------------------*)
VAR
  att: integer;
  wgt_rec: atr1b;
  fx_rec: funct;
  mi_rec: meth1;
BEGIN
  ASSIGN(fx_file, 'funct.dta');
  RESET(fx_file);
ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);

ASSIGN(mi_file,'meth1.rst');
RESET(mi_file);

WHILE NOT(EOF(fx_file)) DO (* if file not empty examine *)
  BEGIN
    READ(fx_file,fx_rec);
    WRITELN(LST,'FUNCTION: ',fx_rec.name); (*print function name*)
    WRITELN(LST);
    RESET(wgtfile);
    att := 0;
    WHILE (NOT(EOF(wgtfile))) AND (att < 17) DO
      BEGIN
        READ(wgtfile,wgt_rec);
        att := att + 1;
        WRITE(LST,wgt_rec.name);
        IF fx_rec.attrib[att] = 1 THEN
          WRITELN(LST,'H');
        ELSE IF fx_rec.attrib[att] = 2 THEN
          WRITELN(LST,'M');
        ELSE IF fx_rec.attrib[att] = 3 THEN
          WRITELN(LST,'L');
        ELSE WRITE('error');
      END;
    END;
    WRITELN(LST);
  END;
IF NOT(EOF(mi_file)) THEN
  BEGIN
    WRITELN(LST,'Method 1 Results:');
    READ(mi_file,mi_rec);
    WRITELN(LST,'Hardware: ',mi_rec.hdw:2:3);
    WRITELN(LST,'Software: ',mi_rec.sw:2:3);
    WRITELN(LST,'Firmware: ',mi_rec.fw:2:3);
  END;
WRITELN(LST, 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX');
WRITELN(LST);
END;

CLOSE(wgtfile);
CLOSE(fx_file);
CLOSE(mi_file);

END;

(XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX)
PROCEDURE ARLT_PRT;
  (* ARLT_PRT prints the AHP results for each *)
  (* function. The composite hw, sw, and fw *)
  (* priorities are printed along with the *)
  (* combined level 0 & level 1 priority matrix. *)
  (XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX)

VAR
  index: integer;
  i: integer;
  factor: atrib;
  rlt: ahp_rlt;

Appendix B: Hardware, Software, Firmware Allocation Tool Code
BEGIN

ASSIGN(wgtfile,'model.wgt');
RESET(wgtfile);
index := FILESIZE(wgtfile);
ASSIGN(ahp_file,'ahp.rlt');
RESET(ahp_file);

WHILE NOT(EOF(ahp_file)) DO
BEGIN

READ(ahp_file,rlt);
WRITELN(LST,'AHP RESULTS Function: ',rlt.name);
WRITELN(LST);
WRITE(LST,' ');

FOR i := 1 TO index DO
  WRITE(LST,'____',i,'_');
WRITELN(LST);
WRITE(LST,' ');
FOR i := 1 TO index DO
  WRITE(LST,rlt.pri_fact[i][1]:1:3,' ');
WRITELN(LST);
WRITELN(LST);
WRITE(LST,' ');% FOR i := 1 TO index DO
WRITE(LST,rlt.pri_fxn[i][1]:1:3,' ');
WRITELN(LST);
WRITELN(LST);
WRITE(LST,' ');% FOR i := 1 TO index DO
WRITE(LST,rlt.pri_fxn[i][2]:1:3,' ');
WRITELN(LST);
WRITELN(LST);
WRITE(LST,' ');% FOR i := 1 TO index DO
WRITE(LST,rlt.pri_fxn[i][3]:1:3,' ');
WRITELN(LST);
WRITELN(LST);
WRITELN(LST,' Hardware Composite: ',rlt.hdw:2:3));
WRITELN(LST,' Software Composite: ',rlt.sfw:2:3);
WRITELN(LST,' Firmware Composite: ',rlt.fmw:2:3);
WRITELN(LST,' Help Procedure ');}

END;
CLOSE(wgtfile);
CLOSE(ahp_file);
END;

Procedure fx_Help; (* Help Procedure *)
var
  exit: string[80];
  return: boolean;
begin
  clrscr;
Appendix B: Hardware, Software, Firmware Allocation Tool Code 161
gotoxy(10,2);  
writeln('HELP PROCEDURE');  
writeln('Type "exit" to return to DSS Menu.');  
return := false;  
repeat  
  read(exit);  
  if exit = 'exit'  
    then return := true  
    else  
    begin  
      writeln;  
      writeln('Invalid Entry!! Try again.');  
    end;  
  until return = true;  
end;  
(* Print Procedure *)

PROCEDURE PRINT;

(* PRINT manages the operator interface *)
(* with the print options. The procedures *)
(* are menu driven. *)

VAR
  finish: boolean;

PROCEDURE PRINT_DSP;

(* PRINT_DSP displays the PRINT procedure *)
(* options in menu format. *)

begin  
  clrscr;  
  drawbox(5,1,40,20);  
gotoxy(10,2);  
WRITE('Hardware/Software/Firmware');  
gotoxy(10,3);  
WRITE('Allocation Printing Menu');  
gotoxy(7,5);  
WRITE('1. Print Decision Factors');  
gotoxy(7,6);  
WRITE('2. Print AHP LEVEL 0 MATRIX');  
gotoxy(7,7);  
WRITE('3. Print AHP LEVEL 1 MATRIX');  
gotoxy(7,8);  
WRITE('4. Print GLM Results');  
gotoxy(7,9);  
writeln('5. Print AHP Results');  
gotoxy(7,10);  
WRITE('6. Quit Print Selection');  
end;

PROCEDURE GO_PRINT(funct_num: integer);

(* GO_PRINT uses the menu selection to call *)
(* for the performance of the selected procedure *)

begin  
  gotoxy(1,25);  
case funct_num of  
    1: PRT_FACT;  
    2: PRT_LEVO;  
Appendix B: Hardware, Software, Firmware Allocation Tool Code
3: PRT.LEV1;
4: GLM_PRT1;
5: ARLT_PRT1;
6: finish := true; (* Quit Procedure *)
end;
end; (* Perform Procedure *)

BEGIN (* Procedure Print..................................................*)
finish := false;

(* Run PRINT menu *)
(* 1. Display menu *)
(* 2. Select function *)
(* 3. Perform function *)
(* 4. Repeat until finished *)
REPEAT
    PRINT_DSP;
    SELECT(funct_num,6);
    GO_PRINT(funct_num);
    UNTIL finish = true;
END;
The vita has been removed from the scanned document