RASSP Integrated Systems Tool Appnote

Abstract

The goal of the DARPA/Tri-Service-sponsored Rapid Prototyping of Application Specific Signal Processors (RASSP) program is to reduce development and manufacturing time and cost of signal processors by a factor of four. Lockheed Martin's Advanced Technology Laboratories (ATL) RASSP team has developed an integrated systems engineering tool set which forms the basis for a concurrent engineering design environment. This design environment, which consists of Ascent Logic's RDD-100, Lockheed Martin PRICE Systems parametric cost estimation models, and Management Sciences Inc. RAM-ILS tools, provides the integrated product development team with cost and reliability estimation data within a systems engineering toolset. The concurrent engineering design environment is described and an example is provided which demonstrates the value of the tool integration within the design environment. This design environment enables the integrated product development team to estimate the life-cycle costs and reliability early in the design process.

Purpose

Systems engineering decisions early in a project significantly impact schedule and cost. Decisions are typically based on the impact to the current phase of a project, rather than the project's overall life cycle.To help the integrated product development team (IPDT) make these trade-off decisions, the ATL RASSP team developed a concurrent engineering environment consisting of Ascent Logic Corporation', (ALC) RDD-100 tool with Lockheed Martin PRICE Systems' parametric cost estimation models and Management Sciences Inc. (MSI) RAM-ILS tool set. This application note will help you learn more about using these tools in this concurrent engineering environment to provide design, cost, reliability, availability, and maintainability support to the IPDT.

Roadmap

1.0 Executive Summary

- 1.1 Overview
- 1.2 Introduction
- 1.3 RASSP Systems Engineering Process Overview
- 1.4 Design Environment Overview
- 1.5 Example Trade-Off
- 1.6 Summary

2.0 Introduction

3.0 Technical Description

- 3.1 RASSP Systems Engineering Process
  - 3.1.1 Overview
  - 3.1.2 RASSP Systems Process Description
    - 3.1.2.1 Requirement Analysis
    - 3.1.2.2 Functional Analysis
    - 3.1.2.3 System Partitioning
- 3.2 RASSP Integrated System Tool Description
  - 3.2.1 Overview
  - 3.2.2 Individual Tool Description
    - 3.2.2.1 RDD-100
    - 3.2.2.2 PRICE Cost Estimating Tools
    - 3.2.2.3 RAM-ILS
  - 3.2.3 Integrated Tool Description
    - 3.2.3.1 RASSP Schema Extension Overview
    - 3.2.3.2 Integrated Design to Cost
    - 3.2.3.2 RAM Assessment
- 3.3 Benefits Using Integrated Sustem Tools

4.0 Application of Integrated System Tools to SAR Benchmark

- 4.1 SAR Benchmark Description

- 4.2 Requirements Analysis
- 4.3 Functional Decomposition
- 4.4 System Partitioning
    - 4.4.1 Requirements and Functional Allocation
    - 4.4.2 Cost Analysis
    - 4.4.3 Reliability Assessment
- 4.5 Summary

5.0 References

6.0 Appendix

Approved for Public Release; Distribution Unlimited *Dennis Basara*

Next Up Previous contents
Next: 2 Introduction **Up:** Appnotes Index **Previous:** Appnote System Index

RASSP Integrated System Tools Appnote

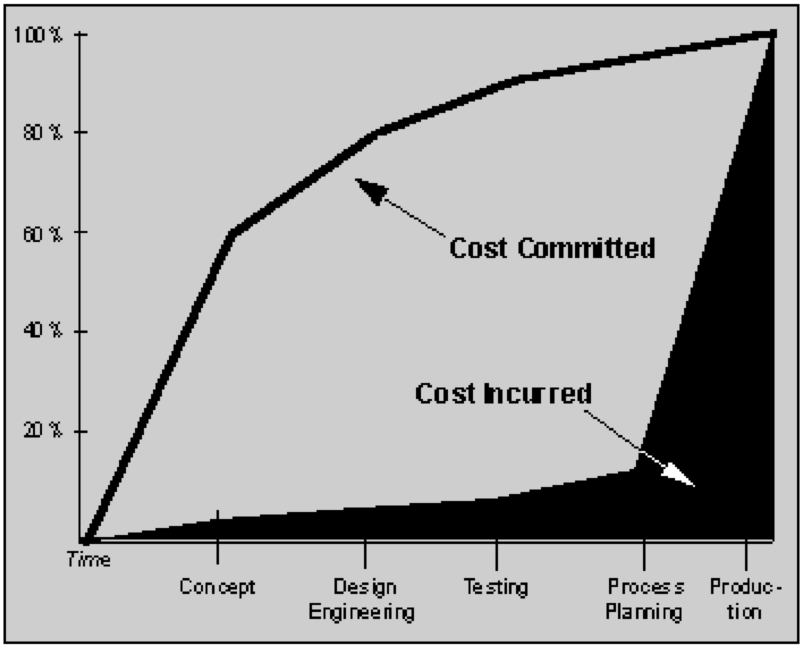
1.0 Executive Summary

1.1 Overview

The goal of the DARPA/Tri - Service sponsored Rapid Prototyping of Application Specific Signal Processors (RASSP) program is to reduce development and manufacturing time and cost of signal processors by a factor of four. Lockheed Martin Advanced Technology Laboratories' (LM/ATL) RASSP team has developed an integrated systems engineering tool set which forms the basis for a concurrent engineering design environment. This design environment, which consists of Ascent Logic's RDD - 100, PRICE Systems parametric cost estimation models, and Management Sciences RAM - ILS tools, provides the integrated product developmentd team with cost and reliability estimation data within a systems engineering toolset. The concurrent engineering design environment is described and an example is provided which demonstrates the value of the tool integration within the design environment. This design environment enables the integrated product development team to setimate the life-cycle costs and reliability early in the design process.

1.2 Introduction

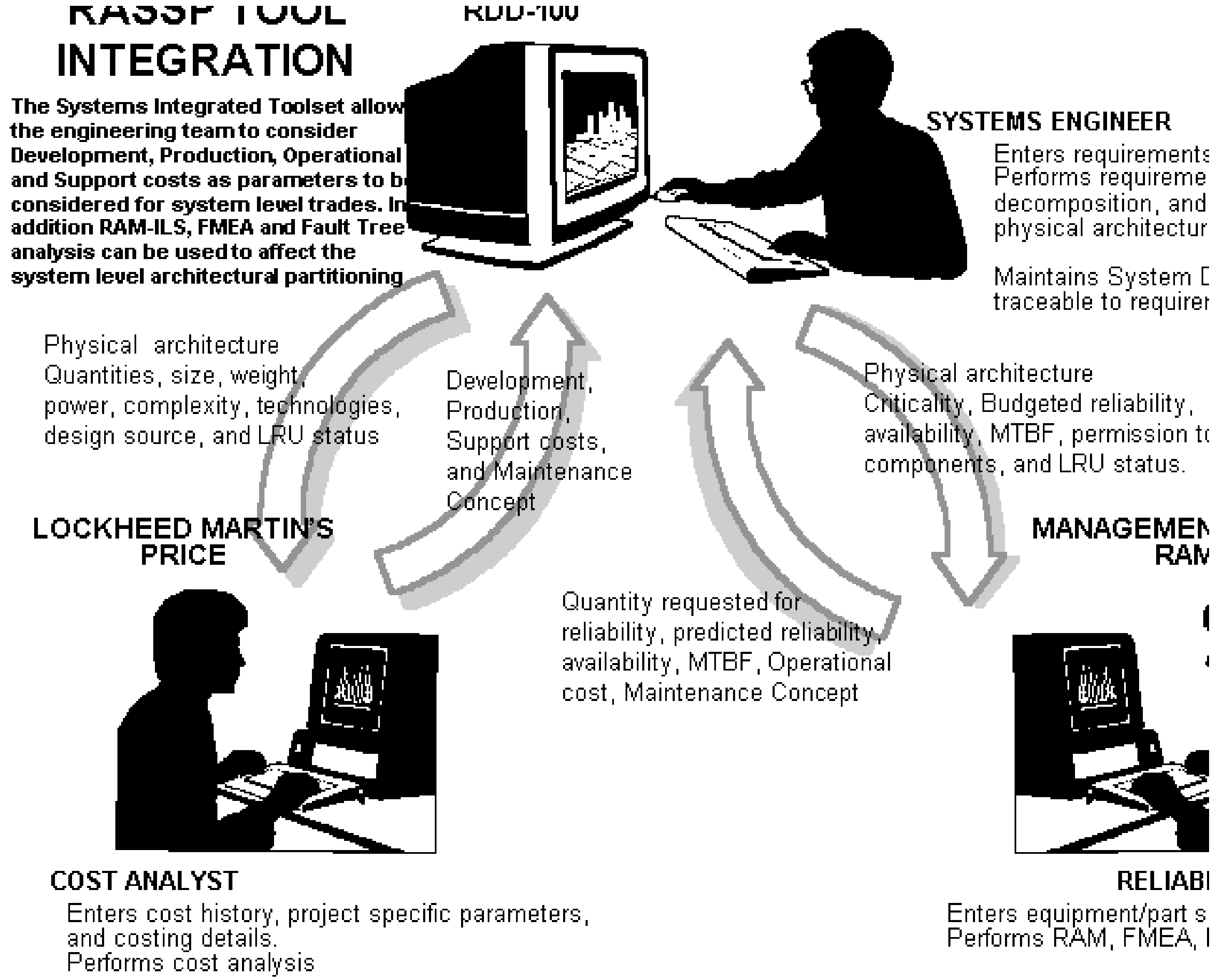
Systems enginerring decisions early in a project significantly impact schedule and cost. Decisions are typically based on the impact to the current phase of a project, rather than the project's overall life cycle. Figure 1 - 1 shows a comparison of cost incurred to cost committed for a typical project. To help the integrated product development team (PDT) make these trade-offs, the ATL RASSP team developed a concurrent engineering environment consisting of Ascent Logic Corporation's (ALC) RDD - 100 tool with PRICE Systems parametric cost estimation models and Management Sciences' (MSI) RAM - ILS tool set, as shown in Figure 1 - 2. Design information is passed among these tools in this concurrent engineering environment to provide design, cost, reliability, availability, and maintainability support to the IPDT.



Source: Business Week 4-30-90

Figure 1 - 1: Typical Project Costs

The RASSP concurrent engineering environment provides the IPDT with the information they need to make decisions early, while making changes is still easy and inexpensive. This environment will allow engineers to make decisions based not only on the current effect of a change, but on the predicted long-term impacts. This information is essential to significantly reducing life-cycle costs.



The Systems Integrated Toolset allow the engineering team to consider Development, Production, Operational and Support costs as parameters to b considered for system level trades. In addition RAM-ILS, FMEA and Fault Tree analysis can be used to affect the system level architectural partitioning

Figure 1 - 2: RASSP Concurrent Engineering Environment

1.3 RASSP Systems Engineering Process Overview

The RASSP design process consists of the signal processing system-level design, architecture selection and detailed hardware/software design, as shown in Figure 1 - 3. The inputs to the RASSP design process are the physical, functional and performance requirements for the signal processing system. These requirements typically are passed down from the platform system level design, which is performed prior to the signal processor design. During the signal processing system design, the requirements are captured and analyzed, the functional behavior of the system is defined and the requirements and functions are allocated to the major subsystems of the signal processor. Hardware/software co-design activities are performed during the architecture selection process and a virtual prototype of the system is developed. Once the processing architectures are determined, the hardware and software are developed and integrated during the detailed design process. Note that these processes are iterative in nature and that feedback between the processes is used whenever required. The focus of this section of the application note is to present an overview of the RASSP systems engineering process.

The RASSP system definition process is a front-end engineering task in which signal processing concepts that meet customer requirements are developed and top-level trade-offs are performed to determine the processing subsystem requirements. Although the same type of functional decomposition and allocation is performed as in the traditional design process, several significant RASSP extensions have been developed which lead to shorter design cycles. Emphasis is placed on understanding the life cycle impact of early design decisions in the RASSP process. Each member of the integrated product development team participates in the system-level tradeoffs to ensure that the complete life cycle is considered during the design process. Model year architecture concepts are used in RASSP designs to ensure that the signal processor can be easily upgraded to support its entire life cycle.

Emphasis is placed on making early design decisions so prototyping activities can begin early in the program to reduce high-risk elements. The output of the system definition process is a set of executable specifications that have the requirements for each processing subsystem in an executable form. The executable specifications support the RASSP concept of reuse and minimize errors due to human interpretation. Traceable system requirements are passed via executable specifications from the system definition process to the architecture design process. As the design progresses, the ability to meet requirements is passed back to the system-level simulations so the impact of lower-level trade-offs are analyzed.

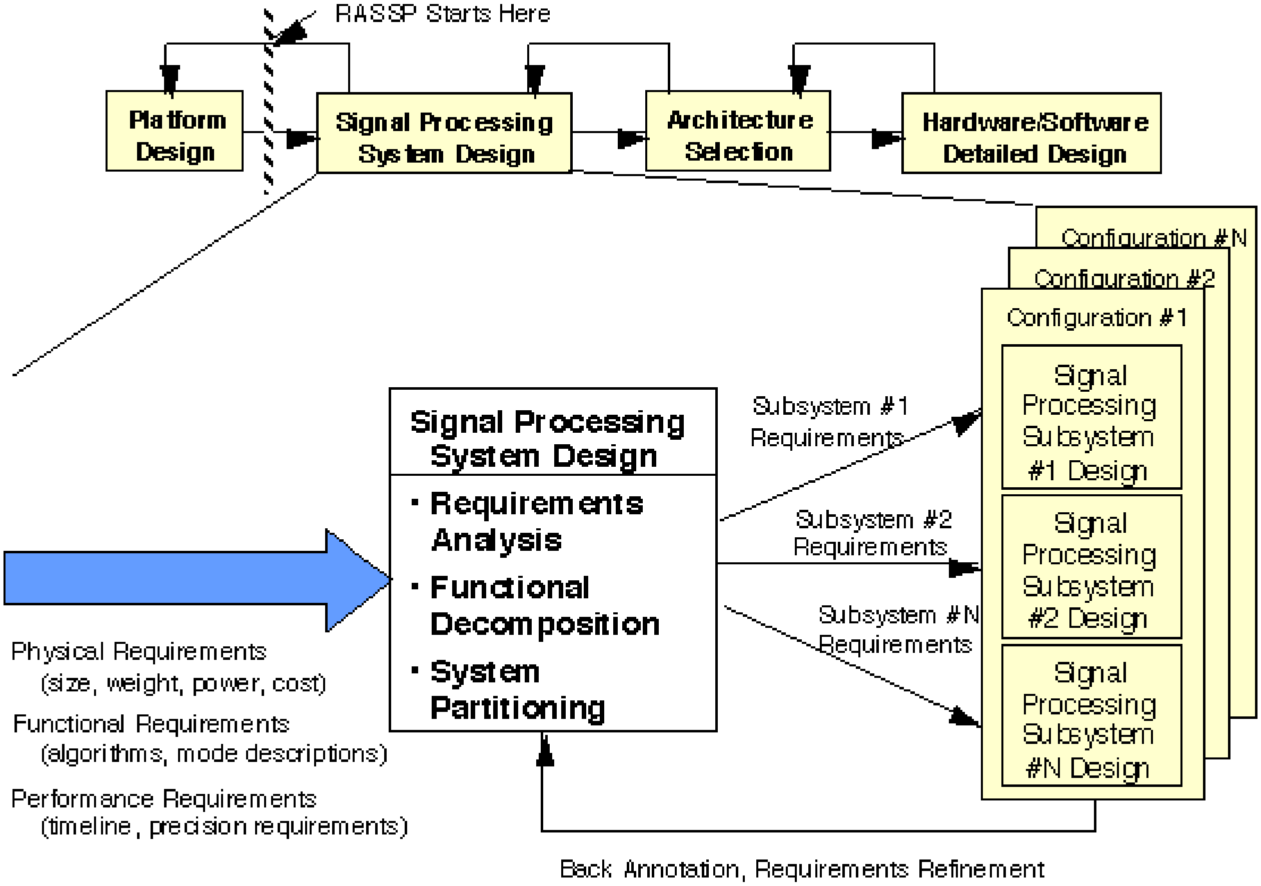

RASSP Starts Here
Platform Design
Signal Processing System Design
Architecture Selection
Hardware/Software Detailed Design
Configuration #N
Configuration #2
Configuration #1
Signal Processing Subsystem #1 Design
Signal Processing Subsystem #2 Design
Signal Processing Subsystem #N Design
Subsystem #1 Requirements
Subsystem #2 Requirements
Subsystem #N Requirements
Signal Processing System Design
• Requirements Analysis
• Functional Decomposition
• System Partitioning
Physical Requirements (size, weight, power, cost)
Functional Requirements (algorithms, mode descriptions)
Performance Requirements (timeline, precision requirements)
Back Annotation, Requirements Refinement

Figure 1 - 3: RASSP Design Methodology

1.4 Design Environment Overview

The RASSP concurrent engineering environment consists of ALC's RDD-100, PRICE System's cost estimating tools and MSI's RAM-ILS toolset, as shown in Figure 1 - 4. The capabilities for each individual tool and for the integrated toolset are described next.

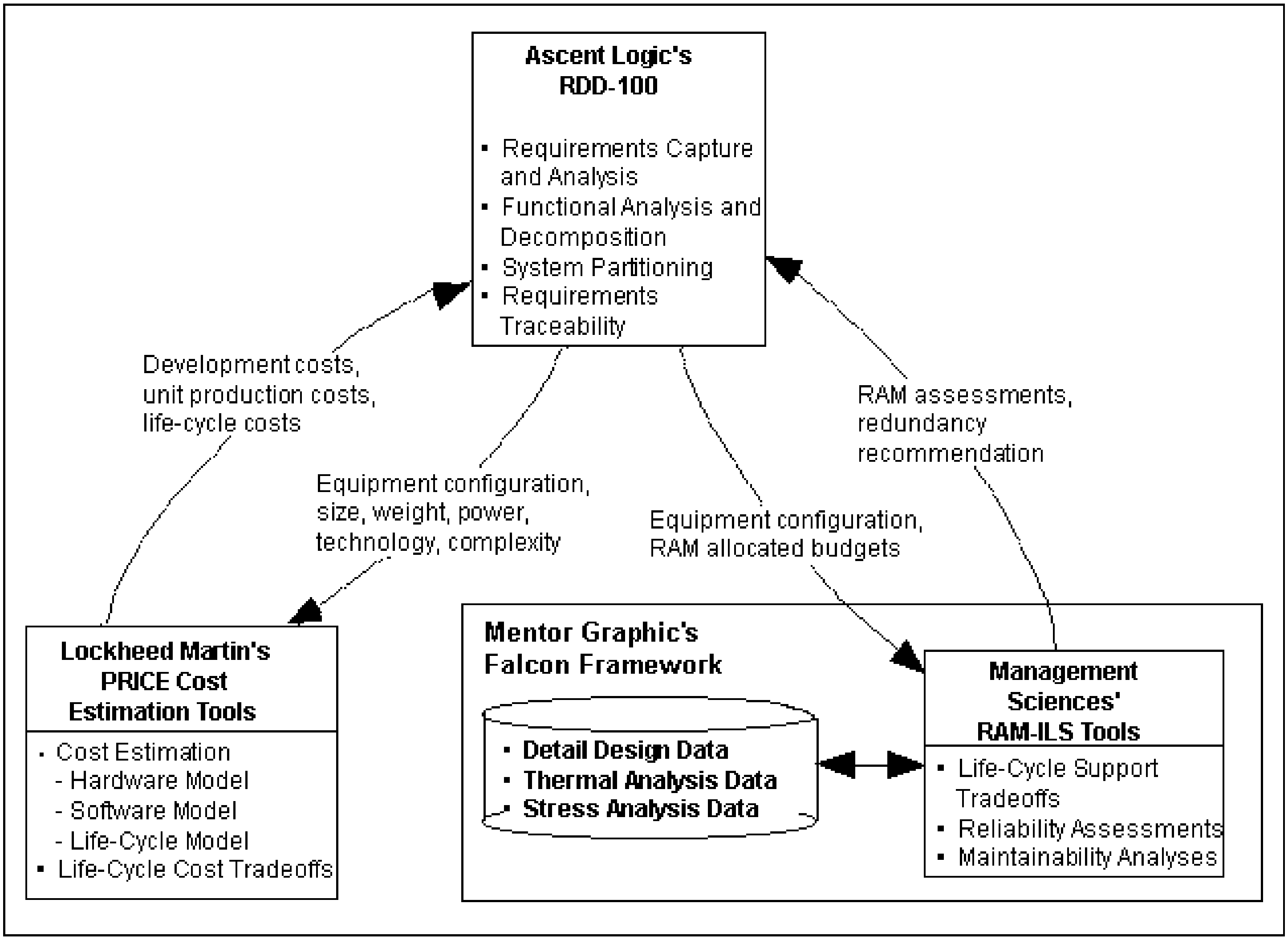

Ascent Logic's
RDD-100
▪ Requirements Capture and Analysis
▪ Functional Analysis and Decomposition
▪ System Partitioning
▪ Requirements Traceability
Development costs, unit production costs, life-cycle costs
Equipment configuration, size, weight, power, technology, complexity
Equipment configuration, RAM allocated budgets
RAM assessments, redundancy recommendation
Lockheed Martin's
PRICE Cost Estimation Tools
▪ Cost Estimation
- Hardware Model
- Software Model
- Life-Cycle Model
▪ Life-Cycle Cost Tradeoffs
Mentor Graphic's
Falcon Framework
▪ Detail Design Data
▪ Thermal Analysis Data
▪ Stress Analysis Data
Management Sciences'
RAM-ILS Tools
▪ Life-Cycle Support Tradeoffs
▪ Reliability Assessments
▪ Maintainability Analyses

Figure 1 - 4: RASSP System Tool Integration

The ATL RASSP team selected Ascent Logic Corporation's RDD-100 tool as the central tool of its integrated toolset. This tool provides requirements analysis, functional analysis, and physical decomposition capabilities. It is an Entity, Relationship, Attribute (ERA) database tool with a substantial graphical data entry user interface. RDD-100's database capability enables it to be the primary data storage tool for the tool set. The ATL RASSP Team defined a set database extensions that support the IPDT through the life of a project.

The RDD-100 tool provides the IPDT with three different views of a system: a requirements view, a functional view, and a physical view. The requirements can be related to the functions and the functions can be allocated to the physical architecture. The interrelation of these three views enables users to automatically generate the lower specification documents from the RDD-100 database. The physical view enables cost analysis and reliability and maintainability analyses.

1.4.1 RDD-100

The ATL RASSP team selected Ascent Logic Corporation's RDD-100 tool as the central tool of its integrated toolset. This tool provides requirements analysis, functional analysis, and physical decomposition capabilities. It is an Entity, Relationship, Attribute (ERA) database tool with a substantial graphical data entry user interface. RDD-100's database capability enables it to be the primary data storage tool for the tool set. The ATL RASSP Team defined a set database extensions that support the IPDT through the life of a project.

The RDD-100 tool provides the IPDT with three different views of a system: a requirements view, a functional view, and a physical view. The requirements can be related to the functions and the functions can be allocated to the physical architecture. The interrelation of these three views enables users to automatically generate the lower specification documents from the RDD-100 database. The physical view enables cost analysis and reliability and maintainability analyses.

1.4.2 PRICE Systems Cost Estimation Models

The ATL RASSP team selected Lockheed Martin PRICE Systems' parametric cost estimation models as the cost analysis tool. These models were originally intended to be used by a cost analyst. PRICE Systems modified them to allow access to the PRICE models through parameters contained within the RDD-100 database and to provide costing information back to this database. The PRICE Systems' tools include a set of four parametric cost estimation models, each with a different specialty areas. Three of the models focus on hardware costing and the fourth model focuses on software costing. These models are summarized below:

- PRICE H: This model specifically addresses the costs associated with development and production of hardware. This tool can use outputs of the PRICE M tool.
- PRICE HL: This model uses data generated by PRICE H and calculates the hardware life-cycle costs, including sparing for a deployment environment.
- PRICE M: This model specifically addresses electronic-module-level hardware development and production costs. It allows engineers to specify individual ASIC and FPGA components to get a detailed cost estimate at the lowest levels.
- Software: This model estimates both development costs and life-cycle support costs for software.

The PRICE models are based on historical models and can be calibrated to match any company's process.

1.4.3 Reliability, Availability, Maintainability: Integrated Logistics Support (RAM-ILS)

Management Sciences' Inc. RAM-ILS tools calculate reliability, maintainability, and availability of a system. This tool set performs mean time between failure (MTBF) and availability calculations using several methods, including Mil-Hdbk-217 and BelCore. If the system doesn't meet the MTBF requirements, RAM-ILS will perform a cost driven trade-off and recommend where redundancy can be added to effectively meet the system MTBF requirement. RAM-ILS is integrated with the Mentor Falcon Framework, which allows it to access the detailed design database to continually improve its estimates as the detailed design progresses.

1.4.4 Integrated Tools

As a part of the RASSP program, RDD-100, PRICE and RAM-ILS have been integrated so design information can be passed among the tools when performing system, costing and reliability analyses. It is through the use of the integrated tools that provides the capabilities needed by the IPDT. The approach used to integrate these tools within the concurrent engineering design environment is to pass data normally resident within one tool to another tool if that data can be used for analyses within the receiving tool. There has been no attempt to build a graphical user interface within any tool for another tool. All data exchanges for these tools are file based.

1.5 Integrated Tools

The following example problem shows how the concurrent engineering environment is applied to a trade-off study.

The ATL RASSP team selected a Synthetic Aperture Radar Digital Signal Processor (SAR-DSP) for a trade-off between two different architecture candidates. A preliminary functional analysis was performed to identify the hardware and software needed by each candidate architecture to satisfy the SAR functional requirements. The Candidate 1 architecture uses a mature technology. As shown in Figure 1-5, this architecture consists of a single-board computer (used as a controller), five processor elements (PE1-5), a cross bar, a fiber interface, and a VME Bus. Each processor element contains four separate computational elements (CE1-4). Also shown in Figure 1-5 is the Candidate 2 architecture. This is similar to the first, except that it uses three state-of-the-art processor elements. In addition, PE2 and PE3 contain only two compute elements rather than four.

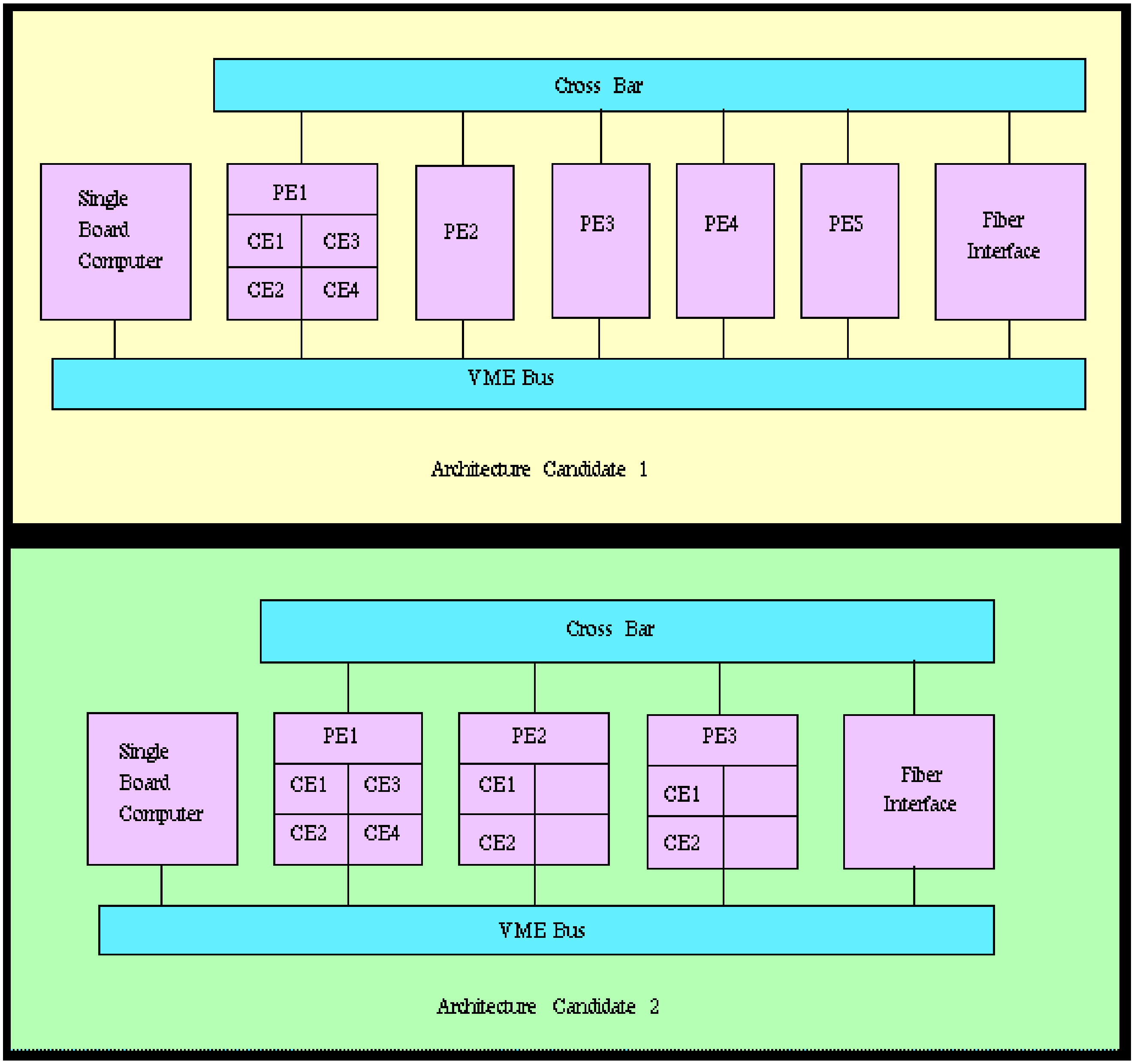


Figure 1 - 5: Two Candidate Architectures for Example Trade-Off

During the development phase, the trade-off is difficult because a mature technology is less expensive per module and is lower risk, while the state-of-the-art technology has fewer modules, is more compact, and consumes less power.

The following tasks were performed when conducting trade-offs between the candidate architectures.

- Requirement Capture and Analysis
- Functional Analysis
- Physical Decomposition
- Preliminary Cost Calculation
- Preliminary Reliability Calculation
- Compute Updated Costing
- Architecture Trade-Off.

Each of these tasks are described below.

1.5.1 Requirements Capture and Analysis

The initial requirements capture and most of the requirements analysis are essentially identical for both candidate architectures. The originating requirements came from a Technical Description Document. The team reworded and reordered these requirements to create a specification for the signal processor. After completing the initial requirements decomposition, the team performed a functional analysis.

1.5.2 Functional Analysis

The functional analysis for both candidate architectures is essentially the same since both architectures are functionally equivalent. The functions are decomposed down to the point where each leaf-level function is allocated to a hardware or software element. At this point, some information about the hardware/software partition may help minimize future changes to the functional decomposition.

1.5.3 Physical Decomposition

The physical decomposition is the only information required to perform cost and reliability analysis. The team developed an equipment/software tree for both candidates that is essentially identical. The primary difference is in the quantities of processor element assemblies. Table 1 - 1 shows an element tree for each of the candidates.

	Architecture 1			**Architecture 2**		
Item	**QTY NHA**	**Design**	**Maturity**	**QTY NHA**	**Design**	**Maturity**
Fiber Interface Assembly	1	-	-	1	-	-
- Data IO Module	1	New	Leading Edge	1	New	Leading Edge
- Fiber Optic Daughter Card	1	COTS	Mature	1	COTS	Mature
- FIR Filter Daughter Card	1	NEW	Leading Edge	1	NEW	Leading Edge
Host Interface	1	COTS	Mature	1	COTS	Mature
Processor Element Assembly	5	-	-	3	-	-
- Mother Board	1	COTS	Leading Edge	1	COTS	Leading Edge
- CE Daughter Card 1	2	COTS	Mature	1 or 0	COTS	Mature
- CE Daughter Card 2	-			1	COTS	SOA
Chassis	1	COTS	Mature	1	COTS	Mature
Backplane Assembly	1	-	-	1	-	-
- VME Backplane	1	COTS	Mature	1	COTS	Mature
- Crossbar	1	COTS	Mature	1	COTS	Mature
COTS : Commercial of the Shelf SOA : State of the Art QTYNHA: Quantity in Next Higher Assembly						

Table 1 - 1: Architecture Candidate Module Complement

While generating the equipment/software tree, the following information is populated in the RDD-100 database for each element:

- Component type
- Component subtype
- Quantity in next higher assembly
- Quantity required for operation
- Redundancy mode
- Budgeted length, width, and depth
- Budgeted weight
- Budgeted power
- Technology
- Technology maturity
- Design source.

Where possible, the team placed the data entry in enumerated lists to guide the IPDT in how to use these fields.

1.5.4 Preliminary Cost Calculation

The team calculated the preliminary cost using the PRICE H, PRICE HL and PRICE S tools. The PRICE tool was configured previously with company-specific calibrations and a deployment environment and scenario. The deployment scenario included two prototypes and 500 production units over a 20-year mission, with 20 organization sites and one depot maintenance site. (An export to PRICE was run from the RDD-100 tool and an import was then run in the PRICE tools.) Table 1-2 shows the calculated costs for Candidate 1. This data was exported from the PRICE tools back to RDD-100. The whole cost analysis and back population is done in less than 1/2 hour. This process allows the IPDT to quickly assess several similar architectures.

Cost Cycle	**Predicted Cost ($M)**
Development Cost	1.9
Production Cost	95.8
Life Cycle Support Cost	36.9
Total Cost	**134.6**

Table 1 - 2: Candidate 1 Preliminary Cost

1.5.5 Preliminary Reliability Calculation

After completing the first costing, an export is performed from RDD-100 to the RAM-ILS tool set. This tool set then calculates the overall MTBCF (mean time between critical failure) and compares it to the budgeted value. In this case, Candidate 1 only achieved a 2069-hour MTBCF for a 2400-hour requirement. Based on a cost trade-off performed within the RAM-ILS tool, this tool recommends that the requirement can be met if a redundant fiber interface is added. RAM-ILS generates the back population results for transfer into the RDD-100 tool. Each component has an attribute "quantity requested for RMA" that indicates where the RAM-ILS tool suggests redundancy. Note that this is just a recommendation from the RAM-ILS tool; systems engineers must determine the feasibility of this recommendation. All the RMA calculations are performed against the original system. If the users believe that this suggestion is proper and feasible given the hardware and software configuration, they change the "quantity in next higher level assembly" within the RDD-100 tool and run the RAM-ILS tool on the new configuration. The final MTBCF for Candidate 1 with the recommended redundancy is 2607 hours.

1.5.6 Cost Updates

At this point, the architecture has changed and more accurate MTBF numbers were available in the database. The team ran the PRICE tools a second time, which provided a more accurate cost assessment, as shown in Table 1 - 3.

Cost Cycle	**Predicted Cost ($M)**
Development Cost	2.0
Production Cost	101.0
Life Cycle Support Cost	39.6
Total Cost	**142.5**

Table 1 - 3: Candidate 1 Updated Costs

1.5.7 Architecture Trade-Off

The team performed similar cost analysis and RMA analysis for Candidate 2. The costing and reliability results for both candidates are shown in Table 1-4. During a typical project, the development costs are the primary criteria used to select the best architecture. Therefore, the life-cycle costs would not be minimized. With the concurrent engineering design environment, the IPDT can pick the most cost-effective solution based on the total life-cycle costs. In the past, Candidate 1 would typically have been selected because there was no easy process to determine life-cycle costs. It is clear from this example that Candidate 2 is the better solution because it is less expensive and more reliable.

With the tools in the concurrent design environment, this information is easily estimated, even during a proposal effort.

Cost Type	**Candidate 1 ($K)**	**Candidate 2 ($K)**
Development Cost	2.0	2.1
Production Cost	101.0	89.1
Life Cycle Support Cost	39.6	29.8
Total Cost	142.5	113.0
MTBCF	2607 hours (Redundancy Required)	3296 hours (No Redundancy)

Table 1 - 4: Trade-Off Table

1.6 Summary

The ATL RASSP team has developed a concurrent engineering environment consisting of three existing computer tools (RDD-100, Price Cost Estimating, and RAM-ILS).

This system design environment quickly provides more detailed and accurate information to the IPDT, and enables them to make better informed decisions early in a system's life cycle and even in the proposal process. Since these early decisions have the largest impact on the overall life-cycle costs of a system, it is important that these decisions be based on all life-cycle costs and not just the cost of the initial development. The tools in this design environment also provide information to support detailed designers throughout the design process.

As shown in the example, it is possible to select the wrong architecture if the decision is only based on the development costs. The life-cycle costs in this example are reduced by over 20% just by understanding these costs early in the development phase. This information is critical in achieving the RASSP goal of a reducing life-cycle costs by a factor of four. The ATL RASSP team is evaluating other technologies to further reduce design-cycle times and costs on the RASSP program.

Although ATL developed the RASSP concurrent system engineering environment to work well in the signal processing domain, many of these concepts can be extended into higher-level systems.

Next Up Previous Contents

Next: 2 Introduction **Up:** Appnotes Index **Previous:**Appnote SYSTEM Index

Approved for Public Release; Distribution Unlimited *Dennis Basara*

RASSP Integrated System Tools Appnote

2.0 Introduction

The goal of the DARPA/Tri-Service sponsored Rapid Prototyping of Application Specific Signal Processors (RASSP) program is to reduce digital signal processor development and manufacturing time and costs by a factor of four. Systems engineering decisions early in the design process significantly impact both the schedule and cost of a project. As shown in Figure 1 - 1, 80 percent of a product's life cycle cost are typically committed by the end of the design development phase. The Lockheed Martin's Advanced Technology Laboratories' (LM/ATL) RASSP team has developed a concurrent engineering design environment which helps the integrated product development team (IPDT) make better trade-offs early in the design process which effectively address life cycle costs. This concurrent engineering environment consists of Ascent Logic Corporation's (ALC) RDD - 100 systems engineering tool, Lockheed Martin's PRICE Systems parametric cost estimating tools and Management Sciences' Inc. (MSI) RAM-ILS tool, as shown in Figure 1 - 2. Design information is passed among these tools in this integrated systems engineering design environment to provide design, cost, reliability, availability and maintainability support to the IPDT. This design environment provides the IPDT with the information they need to make informed and effective decisions early, while making changes is still easy and inexpensive. Engineers are able to determine the impact of design decisions on life cycle costs with this integrated systems engineering environment.

The focus of this application note is to describe the overall RASSP systems engineering methodology, describe how the RASSP integrated system tools support this methodology and to illustrate the benefits of using these tools on an actual application. The intended audience for this application note are program managers, engineering managers and engineers supporting processing system trade-offs who are interested in reducing overall life cycle costs. Each of these groups are interested in a different aspect of this application note. Program managers will be primarily interested how better system level trade-offs can be performed which result in better system designs with lower development costs. Engineering managers will be interested in the overall system engineering process improvements presented in this application note. Engineers will be interested in learning how to perform their jobs more effectively with the use of the integrated RASSP tools. Although the use of these integrated tools have been tailored for signal processing applications on the RASSP program, there are no inherent reasons why these tools can not be extended to work in other domains.

The organization of this application note is as follows. Section 1 contains the executive summary which provides a top level overview for this application note. Section 2 presents the introduction which explains the overall content and organization of the document. Section 3 discusses the RASSP integrated systems engineering process, how the RASSP integrated tools support this process and the benefits for using this process. Section 4 illustrates how the RASSP systems engineering process and integrated tools are used for a Synthetic Aperture Radar (SAR) signal processing example. Section 5 contains a list of reference documents which support this application note.

RASSP Integrated System Tools Appnote

3.0 Technical Description

3.1 RASSP Systems Engineering Process

3.1.1 Overview

The RASSP design process consists of the signal processing system-level design, architecture selection and detailed hardware/software design as shown in Figure 3-1. The inputs to the RASSP design process are the physical, functional and performance requirements for the signal processing system. These requirements typically are passed down from the platform system level design which is performed prior to the signal processor design. During the signal processing system design, the requirements are captured and analyzed, the functional behavior of the system is defined and the requirements and functions are allocated to the major subsystems of the signal processor. Hardware/software co-design activities are performed during the architecture selection process and a virtual prototype of the system is developed. Once the processing architectures are determined, the hardware and software are developed and integrated during the detailed design process. Note that these processes are iterative in nature and that feedback between the processes is used whenever required. The focus of this section of the application note is to describe the RASSP systems engineering process.

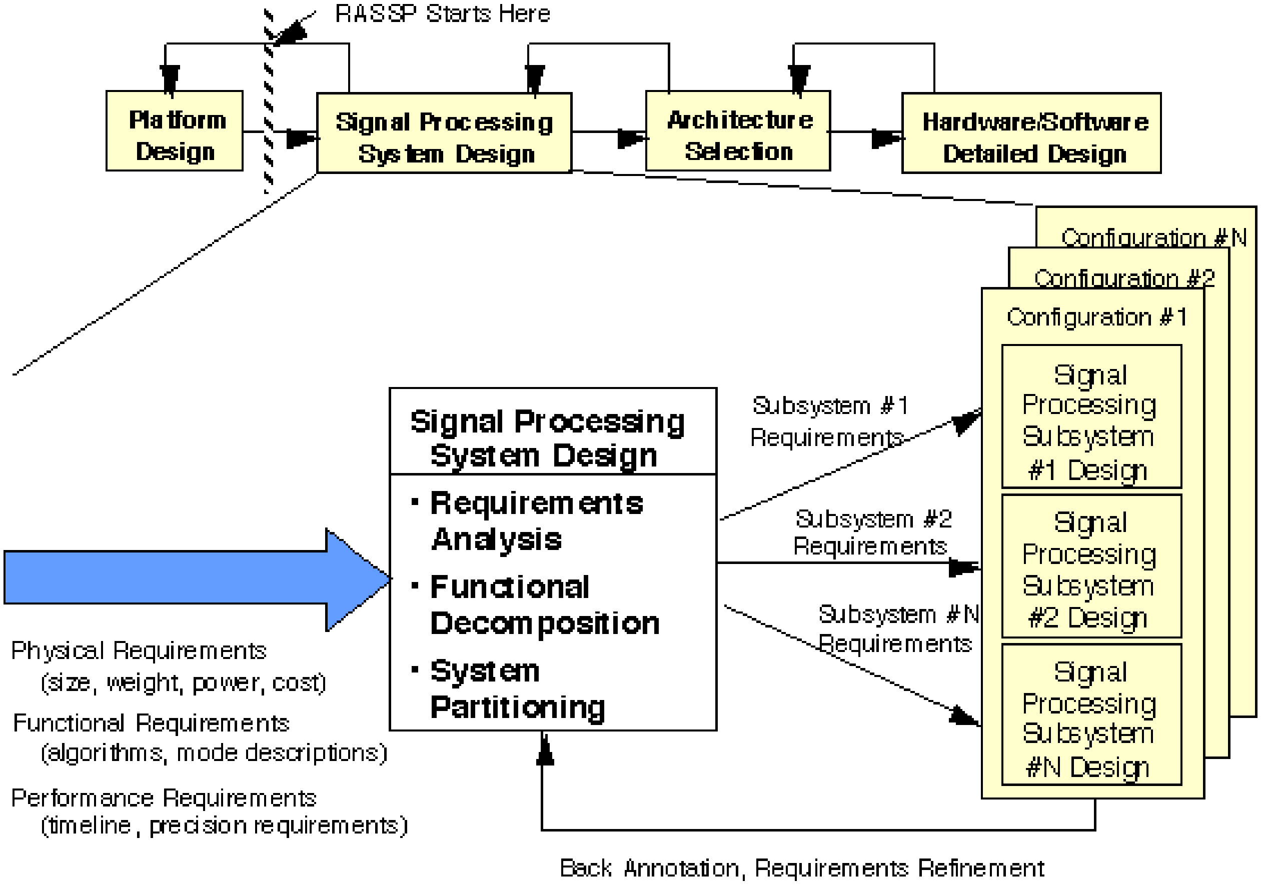

RASSP Starts Here
Platform Design
Signal Processing System Design
Architecture Selection
Hardware/Software Detailed Design
Configuration #N
Configuration #2
Configuration #1
Signal Processing System Design
· Requirements Analysis
· Functional Decomposition
· System Partitioning
Subsystem #1 Requirements
Subsystem #2 Requirements
Subsystem #N Requirements
Signal Processing Subsystem #1 Design
Signal Processing Subsystem #2 Design
Signal Processing Subsystem #N Design
Physical Requirements
(size, weight, power, cost)
Functional Requirements
(algorithms, mode descriptions)
Performance Requirements
(timeline, precision requirements)
Back Annotation, Requirements Refinement

Figure 3 - 1: RASSP Design Methodology

It is recommended that Section 3.1.2 be read to understand what is accomplished during the signal processing system design. Section 3.2.1 entitled "RASSP Integrated System Tool Description Overview" should then be read next to understand the capabilities of these integrated tools. One can then read Section 3.2.2 for more details about the individual tools and Section 3.2.3 for details on their integration. However, both of these sections may be skipped and the reader can move directly to Section 3.3 to learn the benefits of using the integrated toolset. The reader can then progress to Section 4 to see an example how the integrated tools are used on a typical program.

3.1.2 RASSP System Process Description

The RASSP system definition process is a front-end engineering task in which signal processing concepts that meet customer requirements are developed and top-level trade-offs are performed to determine the processing subsystem requirements. Although the same type of functional decomposition and allocation is performed as in the traditional design process, several significant RASSP extensions have been developed which lead to shorter design cycles. Emphasis is placed on understanding the life cycle impact of early design decisions in the RASSP process. Each member of the integrated product development team participates in the system-level tradeoffs to ensure that the complete life cycle is considered during the design process. Model year architecture concepts are used in RASSP designs to ensure that the signal processor can easily be upgraded to support its entire life cycle. Emphasis is placed on making early design decisions so prototyping activities can begin early in the program to reduce high-risk elements. The output of the system definition process is a set of executable specifications that have the requirements for each processing subsystem in an executable form. The executable specifications support the RASSP concept of reuse and minimize errors due to human interpretation. Traceable system requirements are passed via executable specifications from the system definition process to the architecture design process. As the design progresses, the ability to meet requirements is passed back to the system-level simulations so the impact of lower-level trade-offs are analyzed.

The system definition process is shown in Figure 3-2. The inputs to the system definition process include all the customer documentation detailing the processing system specification. Typical signal processing requirements include system mode functional descriptions (search, track, waveforms and algorithms), performance requirements (processing gain, timeline and precision requirements), physical constraints (size, weight, power, cost, reliability, maintainability, testability, etc.), and interface requirements. Top-level tradeoffs are performed by the multidiscipline product development team to determine how the system will operate and what set of subsystems are required. System level functional and timeline simulations are developed to characterize system behavior. The system definition process is iterative, requiring constant interaction with the customer and product development team. The outputs of the system definition process include the functional, performance and physical requirements for each signal processing subsystem.

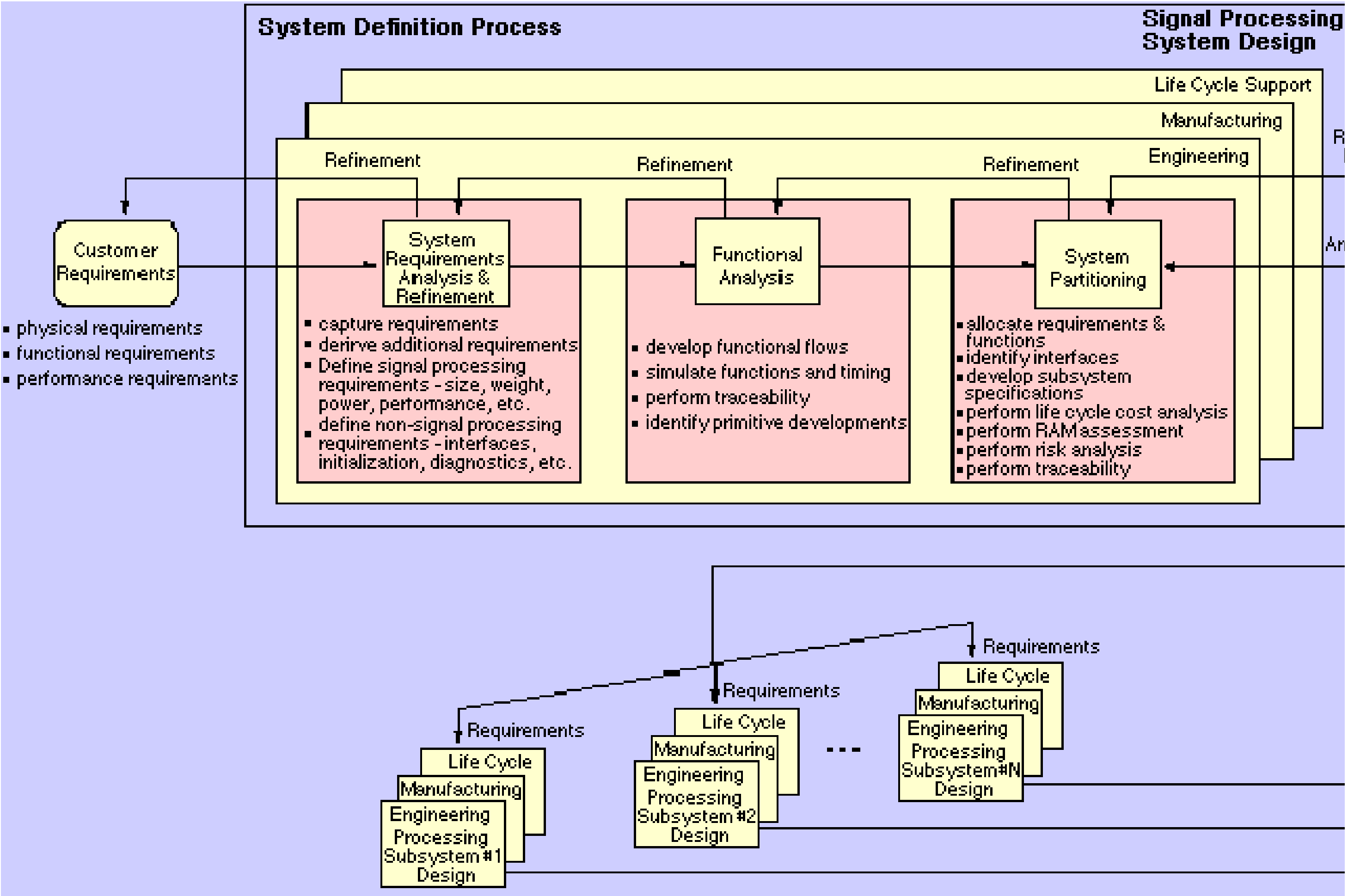

System Definition Process
Signal Processing System Design
Life Cycle Support
Manufacturing
Engineering
Refinement
Refinement
Refinement
Customer Requirements
▪ physical requirements
▪ functional requirements
▪ performance requirements
System Requirements Analysis & Refinement
▪ capture requirements
▪ derirve additional requirements
▪ Define signal processing requirements - size, weight, power, performance, etc.
▪ define non-signal processing requirements - interfaces, initialization, diagnostics, etc.
Functional Analysis
▪ develop functional flows
▪ simulate functions and timing
▪ perform traceability
▪ identify primitive developments
System Partitioning
▪ allocate requirements & functions
▪ identify interfaces
▪ develop subsystem specifications
▪ perform life cycle cost analysis
▪ perform RAM assessment
▪ perform risk analysis
▪ perform traceability
Requirements
Requirements
Requirements
Life Cycle
Manufacturing
Engineering Processing Subsystem #1 Design
Life Cycle
Manufacturing
Engineering Processing Subsystem #2 Design
...
Life Cycle
Manufacturing
Engineering Processing Subsystem #N Design

Figure 3 - 2: System Definition Process

As the subsystem designs progress, key system-level simulations are re-run to ensure that performance is maintained. The subsystem requirements are periodically monitored to make sure that the development risks are appropriately balanced among the subsystems. There is a feedback path back to the system-level from each subsystem design which is used whenever cost-effective subsystem designs cannot be obtained. When subsystem requirements can not be met, analyses are performed to determine a refined partitioning of the system requirements.

3.1.2.1 Requirements Analysis

During requirements analysis a user need is converted into a set of system requirements that satisfies that need. Customer documentation is reviewed and discussions are held with the customer and user to refine the purpose and manner in which users will operate the system. The focus of system requirements analysis is to determine what the system is to do and how the system is to be used. External interfaces to the system are identified. Methods to verify each requirement statement are determined. This process iterates with the functional analysis and system partitioning efforts to assess feasibility and to structure the requirements cost-effectively. This iteration also makes the verification process more accurate and cost effective by eliminating ambiguity in the requirement statements.

The tasks performed during requirements analysis are summarized in Table 3-1. A complete set of customer documents must be used to determine the requirements since the contractor must understand how users intend to use the system. The system is defined in terms of its modes and states, functions and interfaces. Trade-offs establish alternative performance and functional requirements to meet customer needs. Any potential conflicts between the trade-off analysis results and the system requirements are resolved. A work-off plan for all TBD/TBR items that identify the responsible individual, schedule for resolution, risk analysis and key trade-offs to be performed is developed. Traceability of system requirements and decisions ensures that the trade-off decisions made in generating requirements can be tracked and that these requirements are completely and accurately reflected in the final design. Traceability is also used to assess the impact of changes at any level of the system. System requirements are examined to ensure completeness and consistency.

Obtain Relevant Information		System Requirements Assessment		System Definition
● Obtain all relevant customer documentation ● Discuss system requirements with customer ● Obtain data on applicable technology		● Assess functional & performance requirements, operational environment, system constraints & measures of effectiveness		● Define system modes & states ● Define system functions ● Define system configuration items ● Define system interfaces
TBD/TBR Work-off Planning		Requirements Traceability		System Specification Generation
● Identify responsible individual ● Perform risk analysis ● Develop schedule for resolution ● Identify potential trade-offs ● Determine resolution criteria ● Assemble work-off plan ● Conduct internal review		● Capture customer requirements ● Trace system requirements to source documents ● Prepare traceability matrix ● Conduct internal review		● Analyze inputs ● Prepare specification outline ● Incorporate applicable documents & standards ● Prepare preliminary specification ● Conduct review ● Incorporate comments ● Place under configuration control ● Submit for authentication

Table 3 - 1: Requirements Analysis Tasks

The output of the requirements analysis task is the system specification. This specification includes the technical requirements for the system, allocates the requirements to functional areas, documents the design constraints and defines interfaces between functional areas. This specification also contains the necessary performance requirements. Essential physical constraints and requirements for application of any known specific equipment which must be included in the system are included in the specification. The specification can be in either a written format, an executable format or a combination of both formats.

3.1.2.2 Functional Analysis

During functional analysis, the system is decomposed into its functional elements. This analysis is performed by determining what functions are required to implement each system requirement. Functions are described by defining the inputs to the function, the algorithm performed by the function and the outputs of the function. Constraints and timing requirements are established for each function. The top-level functional behavior is modeled to ensure that the functional requirements are met.

The tasks performed during functional analysis are summarized in Table 3-2. The functional identification task translates system requirements, customer heritage and customer rationale into functional block diagrams that are used by subsequent processes to create and evaluate system configurations. This task refines and decomposes the functions identified from the requirements analysis task. Design constraints for each functions are defined. Functional elements within the RASSP reuse library are examined to determine whether existing functional library elements can be used.

Functional Identification		Functional Decomposition
● Analyze system requirements ● Identify system functions ● Develop functional block diagrams ● Identify constraints ● Establish performance timelines ● Evaluate baseline for consistency and completeness ● Perform traceability to requirements ● Provide rationale for functional identification ● Develop alternative functional configurations		● Perform trade-offs to eliminate poor system configurations ● Develop next tier functions ● Develop next tier functional block diagrams ● Identify next tier constraints ● Establish next tier performance timelines ● Perform traceability to upper tier functions and requirements ● Provide rationale for decomposition ● Identify new library primitive elements

Table 3 - 2: Functional Analysis Tasks

Lower level functions, constraints and performance are generated during functional decomposition. This decomposition continues until the functionality can be allocated to a specific subsystem. This more detailed information is used in the subsequent partitioning and architectural trade processes. This functional analysis is performed iteratively to eliminate poor allocation decisions as soon as possible. Functional elements within the RASSP reuse library are examined to determine whether existing library elements can be used at this lower level decomposition. If any of the system functions can not be represented by existing library elements, new primitives are identified for development.

Traceability is established between each functional block and the system requirements.

3.1.2.3 System Partitioning

During system partitioning, candidate system configurations are defined and evaluated to determine which configurations most effectively meet the functional and system requirements. As many configurations as feasibly possible should be evaluated in enough detail to rank the alternatives. A structured method must be followed to quickly identify the feasibility of a specific configuration. The output of the system partitioning process is the set of functional, performance and physical requirements for each subsystem in the baseline configuration.

The tasks performed during system partitioning are summarized in Table 3-3. Functions and constraints are allocated to entities in a specific candidate design during the functional allocation task. This task is completed when all functions are allocated to subsystems and all requirements and constraints are mapped through functions to subsystems. Trade-off analyses assess the risk and life cycle cost for each alternative system configuration. Design decisions and rationale must be documented as the functions are allocated. This task iterates until an allocated baseline is established. Many iterations may be needed to refine the system configuration.

Functional Allocation		Performance Verification
● Create candidate system configurations ● Allocate functions to configuration ● Allocate constraints to configuration ● Identify interfaces ● Perform traceability to functions and requirements ● Provide rationale for allocation ● Identify and quantify risks ● Identify candidate prototyping activities		● Develop metrics ● Develop and configure models ● Estimate unknown parameters ● Execute models ● Evaluate results ● Perform life cycle cost analysis ● Perform reliability and maintainability assessment ● Evaluate trade-offs to eliminate poor configurations ● Reiterate process if completion criteria not met ● Perform traceability ● Develop subsystem requirements ● Support make/buy decision for each subsystem

Table 3 - 3: System Partitioning Tasks

The performance verification task supports system partitioning by providing evaluation criteria to determine which candidate configuration provides the most effective performance. This effort must consider all factors of interest to the product development team: technical performance, risk, life cycle cost, producability, supportability, testability, etc. The performance evaluation must reflect objective, demonstrable evaluation metrics and must assure the customer that sufficient candidates were considered. The behavior of candidate configurations is determined through simulation to ensure all performance requirements are met at the system level. The system partitioning process iterates with the functional allocation process until the performance of a candidate configuration meets the completion criteria established during the requirements analysis process.

The output of the system partitioning process includes the set of functional, performance and physical requirements for each subsystem. This requirements are in the form of an executable requirement and represent the first virtual prototype for the subsystem.

The RASSP architecture selection process transforms the processing requirements for each processing subsystem into a candidate architecture of hardware and software elements. The architecture selection process overlaps with the system definition process during the system partitioning activity. A hierarchical set of simulations is performed at each design level, and the results of these simulations are back annotated in the higher-level simulations to verify that overall performance is maintained.

3.2 RASSP Integrated System Tools Description

3.2.1 Overview

The ATL RASSP team has developed a concurrent engineering environment based upon COTS tools which supports the RASSP systems engineering process. This concurrent engineering environment, which is shown in Figure 3-3, consists of Ascent Logic Corporation's (ALC)RDD-100 system engineering tool, Lockheed Martin PRICE Systems cost estimation tools and Management Sciences' (MSI) RAM-ILS toolset. RDD-100 is used to capture and analyze the requirements, to define the functional behavior of the system, to allocate the requirements and functions to the subsystems, and to provide requirements traceability. PRICE cost estimating tools are used to estimate the development, production and support costs for the processing system. The RAM-ILS tool is used to perform reliability and maintainability analyses.

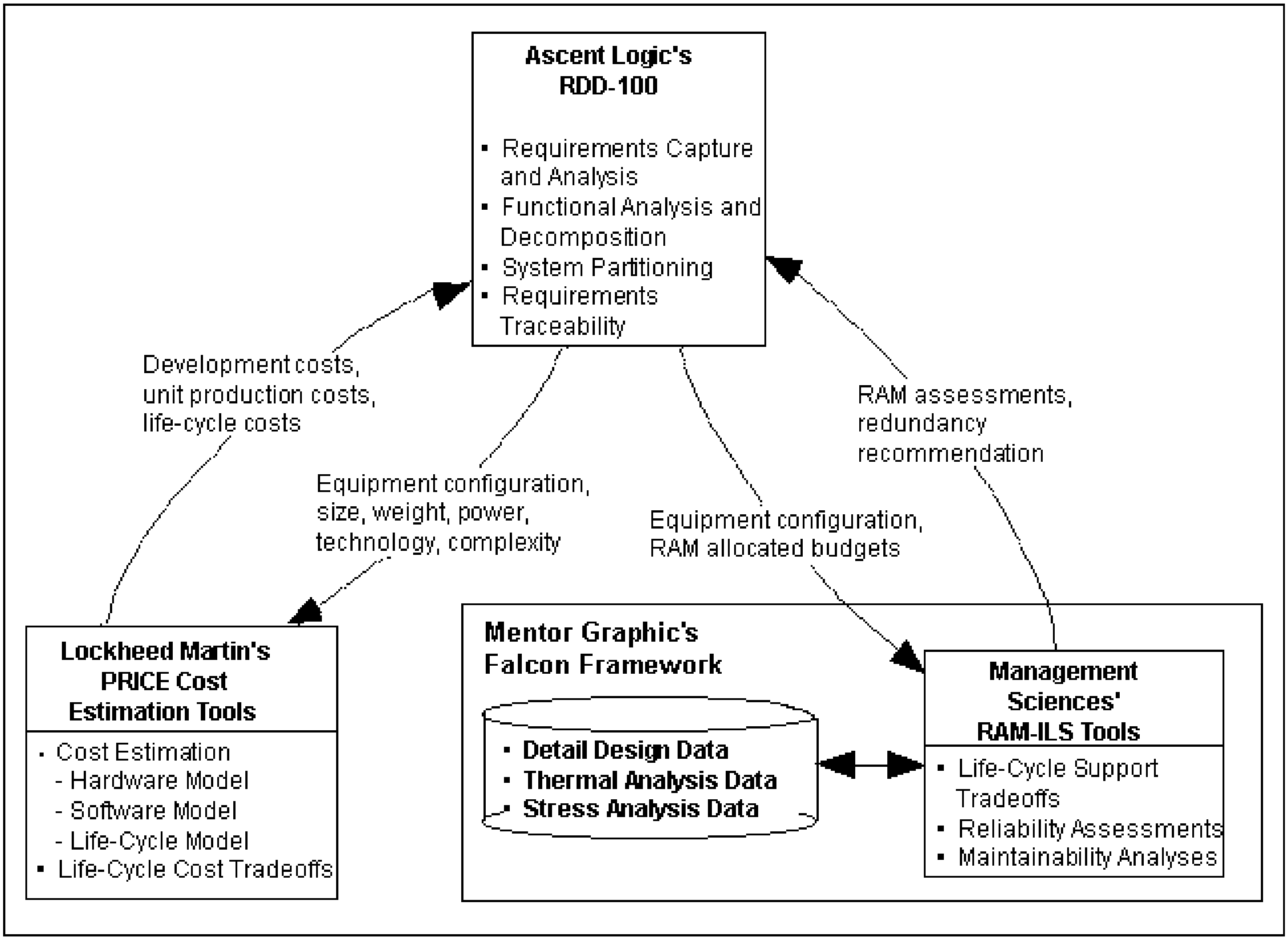

Ascent Logic's RDD-100
▪ Requirements Capture and Analysis
▪ Functional Analysis and Decomposition
▪ System Partitioning
▪ Requirements Traceability
Development costs, unit production costs, life-cycle costs
Equipment configuration, size, weight, power, technology, complexity
Equipment configuration, RAM allocated budgets
RAM assessments, redundancy recommendation
Lockheed Martin's PRICE Cost Estimation Tools
▪ Cost Estimation
- Hardware Model
- Software Model
- Life-Cycle Model
▪ Life-Cycle Cost Tradeoffs
Mentor Graphic's Falcon Framework
▪ Detail Design Data
▪ Thermal Analysis Data
▪ Stress Analysis Data
Management Sciences' RAM-ILS Tools
▪ Life-Cycle Support Tradeoffs
▪ Reliability Assessments
▪ Maintainability Analyses

Figure 3 - 3: RASSP Integrated System Tools

Each tool passes data to another tool through an ASCI file with the appropriate format. The types of data which are passed from one tool to another consist of the data that typically resides in that tool and can be used by the other tool. For example, system engineering data is passed from RDD-100 to the PRICE cost estimating tool. This approach eliminated the need for implementing a GUI interface for PRICE and RAM-ILS in the RDD-100 tool. The types of parameters which are passed from RDD-100 to PRICE include the equipment configuration, size, weight, power, technology and complexity factors. The development, production and support costs are calculated within the PRICE tool and these costs are back annotated into the RDD-100 data base. On the other side of the interface, the equipment configuration, allocated reliability and maintainability budgets, and cost data are passed from RDD-100 to the RAM-ILS toolset. The reliability and maintainability assessment is performed within the MSI toolset and the results of these analyses are back annotated into the RDD-100 data base. In addition, optimizations can be performed within the RAM-ILS toolset when the reliability requirements are not met and the tool can make a recommendation on how redundancy can be added in the system in the most cost effective way to meet the requirements.

Now that you have a basic understanding of the integrated toolset, you may read Sections 3.2.2 and 3.2.3 for more detailed information on the individual tools and their integration. For a shortened version you can skip directly to Section 3.3 to read about the benefits of using these tools in a cooperative manner.

3.2.2 Individual Tool Description

3.2.2.1 RDD-100

Ascent Logic Corporation's (ALC) RDD-100 is a system engineering tool used for capturing requirements, relating the requirements to a behavior model, and allocating the functionality to a physical architecture. The tool supports object-oriented analysis, stimulus response threads, and other analysis techniques. RDD-100 can be used to produce specifications at the system, segment and/or component level. Traceability of all system level parameters from both requirements to functions and from functions to components is maintained within RDD-100, RDD-100 is based upon an entity, relationship and attribute (ERA) database. Entities within the database are the nouns or objects within the system such as requirements, functions and components. The interrelationship between different entity types are defined by an extendible schema within RDD-100. It is through these relationships that traceability is maintained within RDD-100. The base RDD-100 schema has been extended for RASSP to support the integrated costing and reliability analysis and these extensions are described in Section 3.2.3.1.

The typical use of RDD-100 is illustrated in Figure 3-4. Requirements are initially captured, examined and decomposed into lower level requirements during the requirements analysis step. The functionality of the system is then defined using behavior diagrams within RDD-100 during the functional analysis step. Both control and data flow are shown within the same behavior diagram. Every function within the behavior diagram must be traceable to a requirement. The physical architecture consisting of hardware and software components is established during the system partitioning step. Each function in the behavior diagram must be allocated to one component in the system architecture. A traceable path between each function and the component that function is allocated to is maintained within RDD-100.

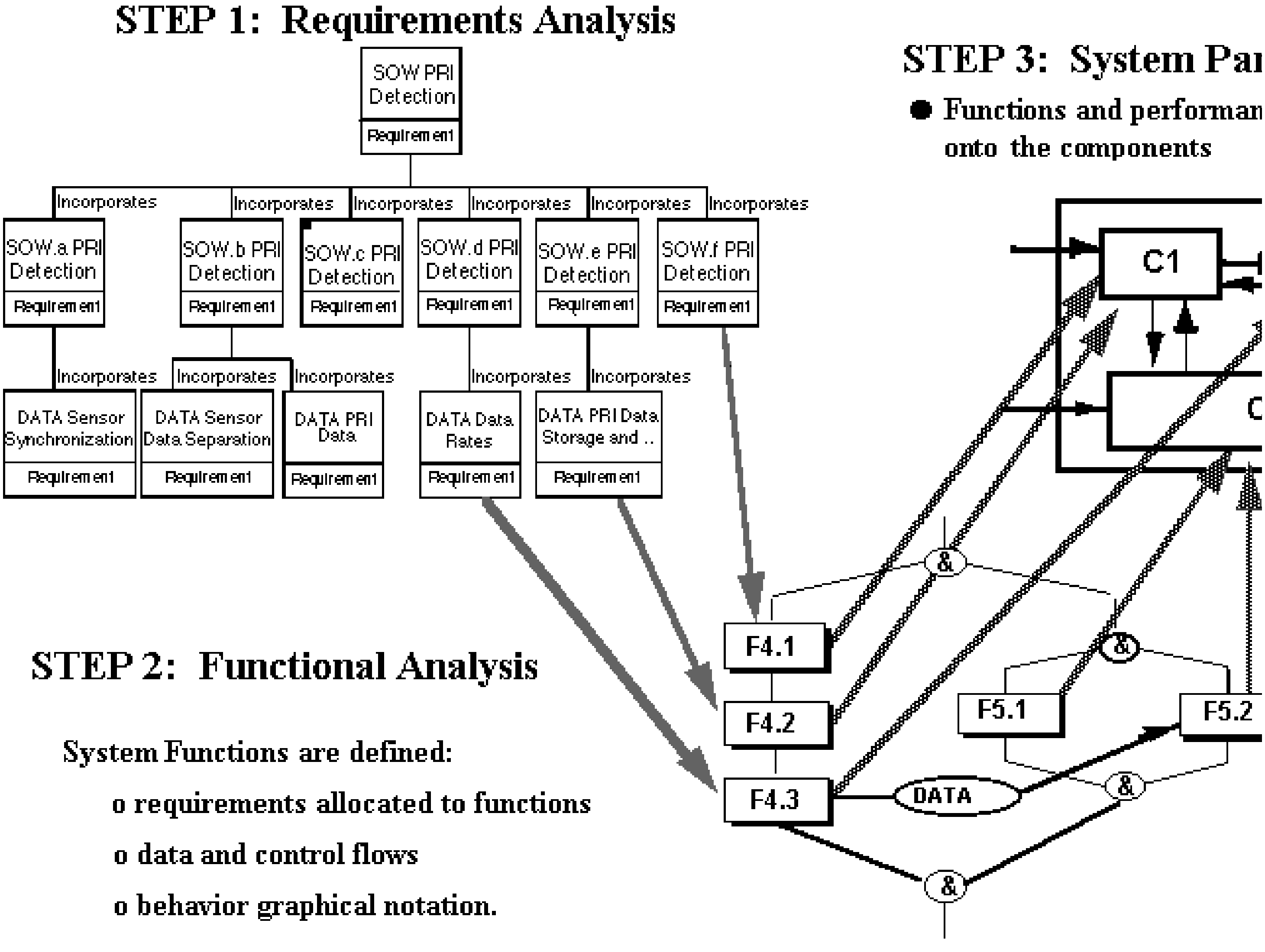


Figure 3 - 4: Typical Use for RDD-100

3.2.2.2 PRICE Cost Estimating Tools

Lockheed Martin's PRICE Systems cost estimating models are advanced Computer Aided Parametric Estimating (CAPE) tools used for calculating cost estimates and schedules for both hardware and software components. Computer-aided parametric estimating tools are parametric cost models that relate physical and empirical system characteristics to the costs and schedules required to develop, produce and maintain the system. These cost estimating relationships (CERs) are embodied within the computer model. PRICE Systems formalizes these relationships by applying regression techniques to hardware and software systems across various industries. In addition, these estimating relationships can be calibrated for a specific organization so the outputs obtained from the PRICE tools reflect that organization's costs.

The PRICE cost estimating tools consist of the hardware, microcircuit, hardware life cycle and software models and each of these models are described below.

PRICE H - Hardware Cost Model : The hardware model is used to estimate the cost and schedule for electronic, electro-mechanical, and structural assemblies. This model incorporates input data concerning weight, size, quantity, process and design sensitivity, and complexity parameters. The hardware model provides cost and schedule outputs for the development and production phases of a program.

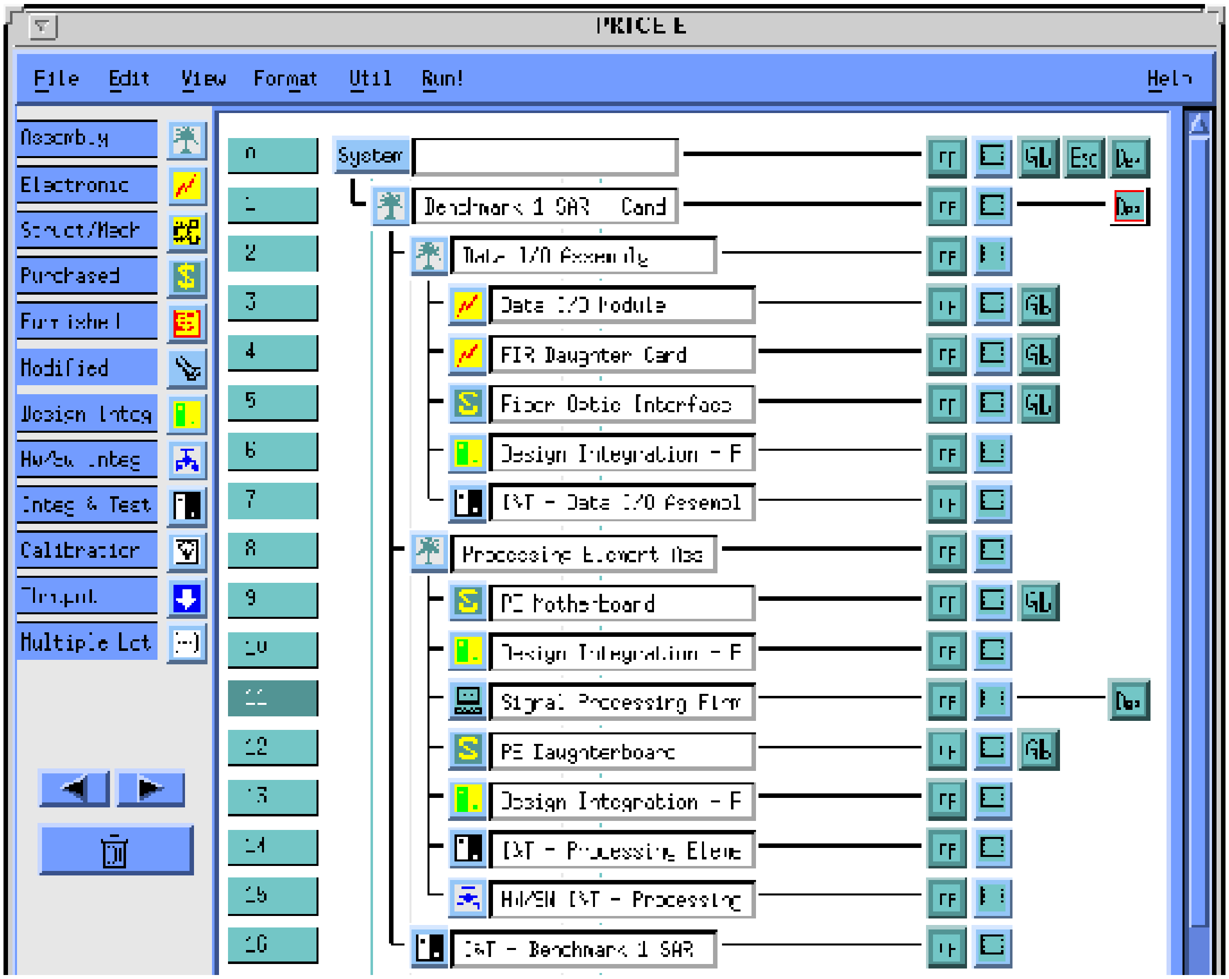
PRICE M - Microcircuit/Module Cost Model : The microcircuit model is used to estimate the cost and schedule for custom microcircuits, printed circuit boards, and electronic modules. The model uses functional relationships based upon parameters such as the number of transistors, percentage of new circuit cells, number of pins, board types and size.

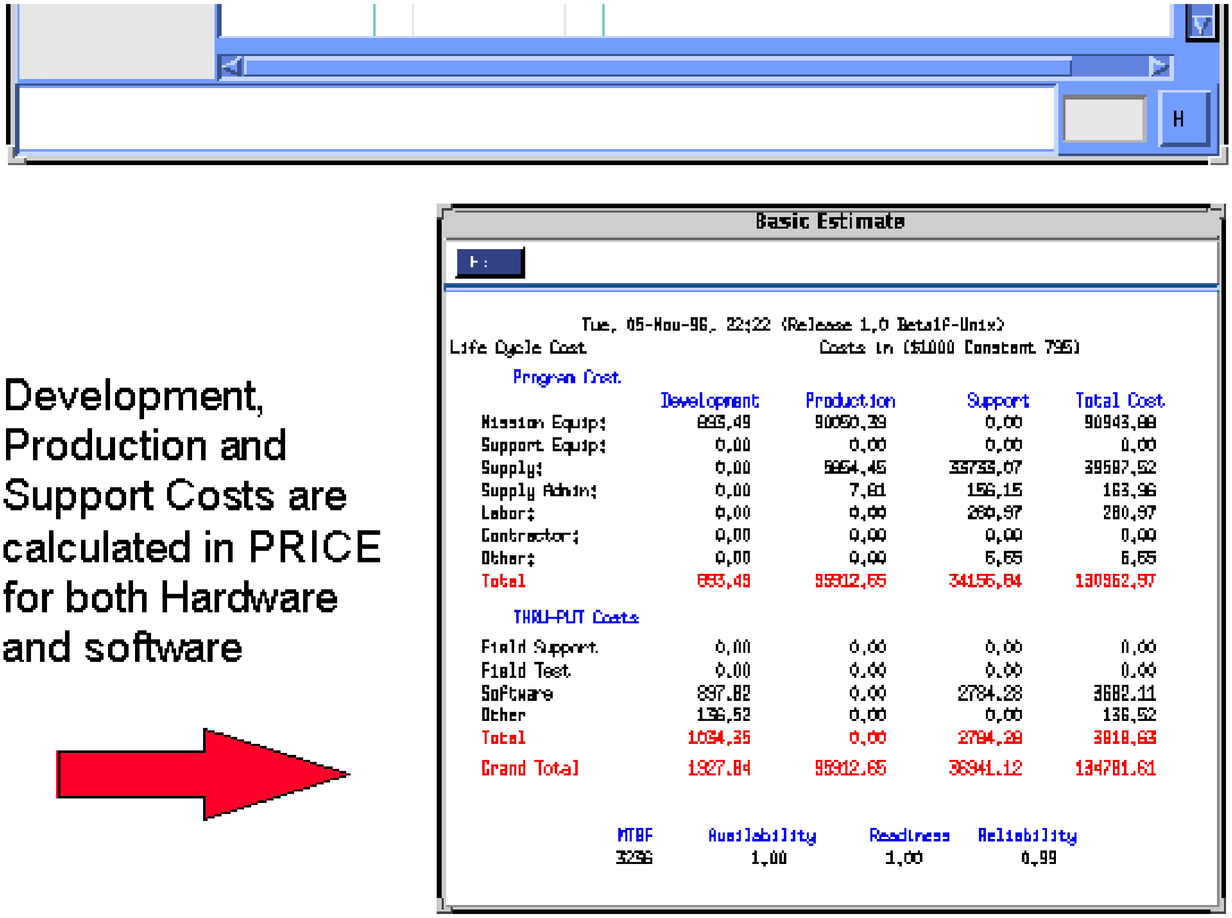
PRICE HL - Hardware Life Cycle Model : The hardware life-cycle model is used to estimate the cost of operating and maintaining hardware systems throughout their deployment. Inputs to the life cycle model include deployment parameters, maintenance concepts, cost, and escalation factors. The life cycle model is a supplement to and works in conjunction with the hardware model.

PRICE S - Software/Software Life Cycle Model : This software model is used to estimate the cost and schedule for the design, development, integration, testing, and support of software. This model uses functional relationships based upon parameters such as function, lines of code, complexity, platform, application, and design reuse to estimate costs.

PRICE Systems refine and update their cost estimating models based upon actual costs being incurred on current products of their users. While the cost models are designed to provide estimates for a typical organization, the models provide the flexibility for the user to tailor and calibrate the models for their specific organization. As a result of calibration, the costs and schedule outputs of the models reflect how a particular organization develops their product.

The PRICE model requires that the system be described as a set of hardware and software components in an equipment breakdown structure (EBS) which is shown in Figure 3-5. Parameters which characterize each component are entered into the model. Global parameters which define labor rates, financial factors and deployment concepts are also required. The PRICE models determines the development, production and support costs for each component in the EBS and these costs are accumulated up the equipment tree to determine the overall system costs.





```
          Tue, 05-Nov-96, 22:22 (Release 1.0 Beta1F-Unix)
Life Cycle Cost                        Costs in ($1000 Constant 795)
     Program Cost
                      Development   Production     Support   Total Cost
   Mission Equip:          893,49     90050,39        0,00     90943,88
   Support Equip:            0,00         0,00        0,00         0,00
   Supply:                   0,00      5864,45    33755,07     39587,52
   Supply Admin:             0,00         7,81      156,15       163,96
   Labor:                    0,00         0,00      280,97       280,97
   Contractor:               0,00         0,00        0,00         0,00
   Other:                    0,00         0,00        6,65         6,65
   Total                   893,49     95912,65    34156,84    130962,97

     THRU-PUT Costs

   Field Support             0,00         0,00        0,00         0,00
   Field Test                0.00         0.00        0.00         0.00
   Software                897.82         0.00     2784.28      3682.11
   Other                   136,52         0,00        0,00       136,52
   Total                  1034,35         0,00     2784,28      3818,63

   Grand Total            1927.84     95912.65    36941.12    134781.61


               MTBF     Availability    Readiness   Reliability
               3296         1,00           1,00        0,99
```

Development, Production and Support Costs are calculated in PRICE for both Hardware and software

Figure 3 - 5: PRICE EBS and Output Report

3.2.2.3 RAM-ILS

Management Sciences Incorporated's (MSI) RAM-ILS toolset provides design assurance management for an overall system design environment. The RAM-ILS toolset consists of synergistic tools within the design framework which measure the quality related aspects of a design. This toolset is used to assess the functional robustness, functional reliability, functional diagnosability, manufacturing process reliability, and deployment reliability issues. The features of the RAM-ILS toolset are summarized in Table 3-4.

Design Assurance Tool	Value Added
Functional Reliability Risk Allocation	Reliability goals for hardware and software
Circuit Based Design Reliability Simulation	Stress derating for component selection
Functional Reliability and Longevity Analysis	Improved performance in operational life cycle
Deployment Life Cycle Cost Tradeoffs	Economic and warranty analysis
Failure Modes and Effects Criticality Analysis	Safety and degraded performance analysis
Diagnosability and Repairability	Maintenance requirements analysis
Mission and Deployment Reliability	Durability, capability and performance analysis
Maintainability and Supportability	Support staff/equipment requirements analysis
Worst Case Analysis (Aging and Degradation)	Parametric degradation analysis
Thermal Damage Analysis	Thermal derating analysis

Table 3 - 4: Features of the RAM-ILS Toolset s

The RAM-ILS toolset can be used for reliability predictions, maintenance analysis, failure modes and effects criticality analysis (FMECA), and success tree analysis. Each of these capabilities are described below.

Reliability Predictions - Reliability predictions are made within the RAM-ILS tool using reliability block diagrams. Failure models for each component in the system can be based upon relevant historical data, specialty computational methods, probability distributions and "similar to" designs. These predictions facilitate trade-off studies when allocating failure rate budgets to system components.

Maintainability Predictions - The RAM-ILS tool can be used to determine the impact of various maintenance concepts on life cycle cost. The tool models various maintenance strategies such as level of repair and fault isolation characteristics. Maintenance diagrams are constructed within the toolset.

Failure Modes and Effects Criticality Analysis (FMECA) - The FMECA portion of the RAM-ILS toolset provides the user with an understanding how the system will perform when it is operating in either a degraded or failed state.

Success Trees - Success trees are modeled within the RAM-ILS toolset to illustrate how a system successfully operates with respect to the interaction of system functions and components. Inverted success trees identify unacceptable critical combinational failures. Success trees can be used to confirm redundancy decisions and identify false redundancy conditions.

The RAM-ILS toolset is integrated with Mentor Graphics Falcon Framework and is illustrated in Figure 3-6. This integration provides a consistent, well-defined, known user interface. As a result, the user does not need to learn another specialized interface to use the RAM-ILS tool. In addition, the Mentor interface provides convenient access to detailed design data.

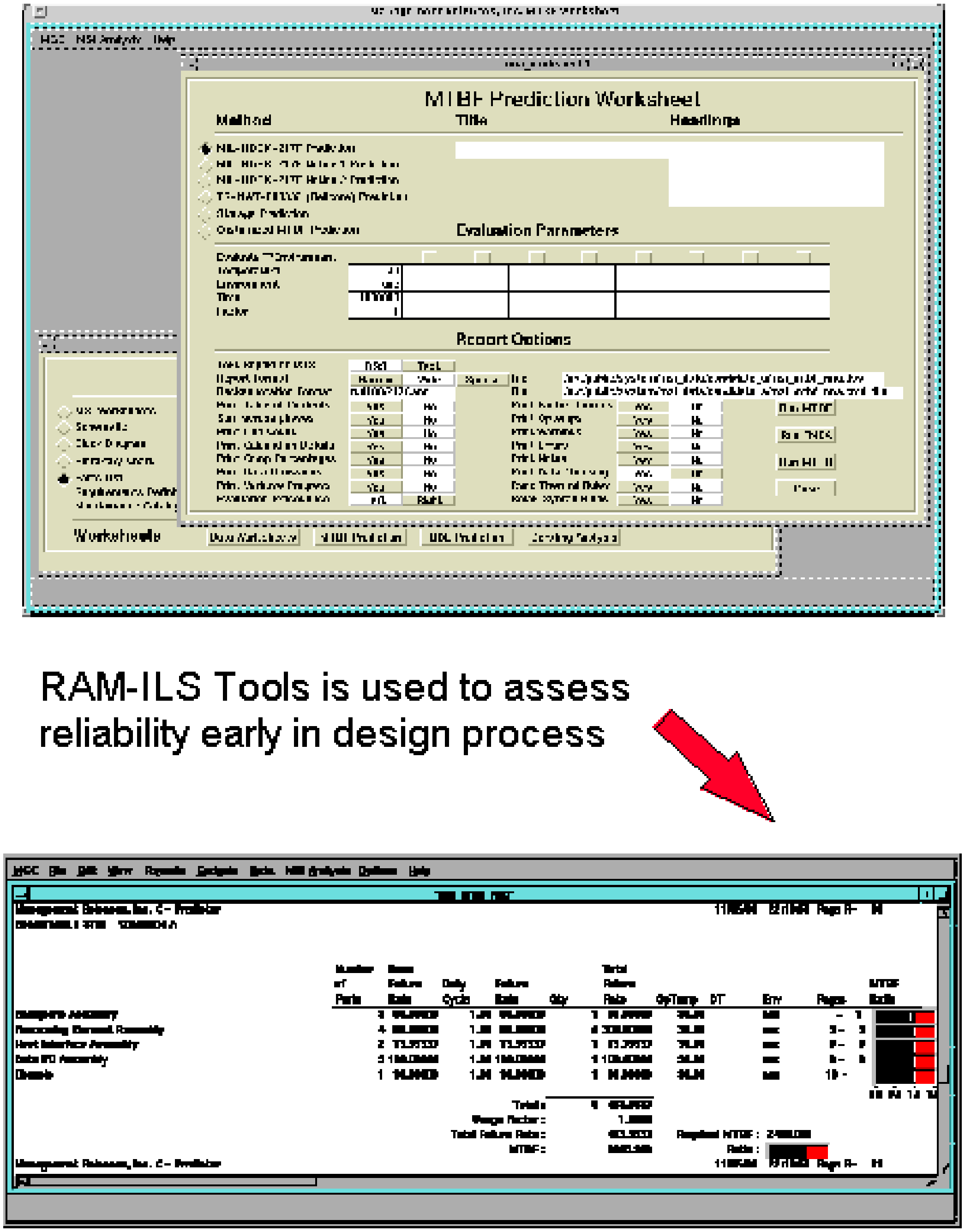

MTBF Prediction Worksheet
Method
Title
Headings
Evaluation Parameters
Report Options
Worksheets
RAM-ILS Tools is used to assess reliability early in design process

Figure 3 - 6: RAM-ILS User Interface

3.2.3 Integrated Tools Description

3.2.3.1 RASSP Schema Extension Overview

The schema within RDD-100 has been extended on the RASSP program to support both cost estimating and specialty engineering. An overview of the RASSP schema extensions is presented in this section to give the reader a basic understanding of these modifications. For more detailed information on the schema extensions see the RDD-100 User's Manual for the Integrated System Engineering RASSP Schema and the "Specification for Ascent Logic Corporation, RDD-100 Schema Extensions (for the RASSP program)". The baseline RDD-100 schema has been extended in six basic areas to support cost estimating and specialty engineering.

- Attributes have been added to the component entity, which describe the physical nature of the component.
- A cost entity has been added to the schema, which contains the development, production and support costs for each component in the system.
- An RMA entity has been added to the schema, which contains both input parameters needed for specialty engineering

analyses and output results from these analyses.
- A life cycle parameter entity has been added which contains parameters that describe how the system is deployed during its life cycle.
- A duplicate component entity has been added to the schema, which identifies multiple instances of the same component within a system.
- An external tool file entity has been added to the schema, which contains file information that the PRICE and RAM-ILS tools need.

Each of these schema extensions is described below.

3.2.3.1.1 Additional Component Attributes

Additional attributes have been added to the component entity in the RASSP schema to characterize the component for cost and reliability assessment. A summary of these additional attributes is given in Table 3-6. Some of these attributes are not applicable to all component types. For example, the attributes which characterize software are not applicable to purely hardware components.

Type of Attribute	Attribute	Comment
Component Characterization Parameters	● Component Type ● Component Subtype ● Design Source ● Percent New Design ● Technology Maturity ● Duplicate Used in Other Assemblies	● User input parameters ● Component type indicates whether component is system, subsystem, hardware, software or firmware ● Component subtype indicates whether component is an assembly, board, module, power supply, etc. ● Design source indicates whether component will be a new design, COTS or furnished
Quantity Parameters	● Quantity in Next Higher Assembly ● Quantity Requested by RMA ● Quantity Required for Operation ● Redundancy Mode	● RAM-ILS tool back annotated quantity requested for RMA which indicates optimizations within the RAM-ILS tool on how system configuration can be changed for a more cost effective system ● Parameters accommodate redundancy in the system
Physical Parameters	● Budgeted Length ● Budgeted Width ● Budgeted Depth ● Budgeted Weight ● Predicted Length ● Predicted Width ● Predicted Depth ● Predicted Weight	● User enters in the budgeted values ● Other tools or user enter in predicted values as design progresses ● Parameters indicate margin between current design and requirement

Power Parameters	● Budgeted Average Power ● Budgeted Maximum Power ● Predicted Average Power ● Predicted Maximum Power	● User enters in the budgeted values ● Other tools or user enter in predicted values as design progresses ● Parameters indicate margin between current design and requirement ● RAM-ILS tool uses power metrics within sensitivity calculations
Sensitivity Parameters	● Volume Sensitivity ● Weight Sensitivity ● Power Sensitivity	● User input parameters ● RAM-ILS uses parameters in sensitivity calculations when trying to optimize system configuration to meet requirements ● Parameters are used to specify relative importance among these sensitivities
Hardware Technology Parameters	● Technology Type 1 ● Equipment Type 1 ● Percentage of Type 1 Technology ● Technology Type 2 ● Equipment Type 2 ● Percentage of Type 2 Technology ● Technology Type 3 ● Equipment Type 3 ● Percentage of Type 3 Technology ● Technology Type 4 ● Equipment Type 4 ● Percentage of Type 4 Technology ● Technology Type 5 ● Equipment Type 5 ● Percentage of Type 5 Technology	● User input parameters ● Up to five different technology/equipment types can be used to characterize component ● Technology type defines the digital technology used to construct component, i.e. VLSI ● Equipment type specifies the application where technology is used, i.e. analog, rf, digital, display, structure, etc. ● Percentages of all technology types should add up to 100 percent
Software Parameters	● Source Lines of Code ● Percent of Memory Utilization ● Percent of Processor Utilization ● Language ● Percent of New Code	● User input parameters ● Parameters indicate the size, language and efficiency of the software component

Software Characterization Parameters	• Percent of Mathematic Code • Percent of String Manipulation Code • Percent of Store and Retrieve Code • Percent of On-line Communication Code • Percent of Operating System & Interactive Code • Percent of User Defined Code • Difficulty of User Defined Code	• User input parameters • Parameters indicate the type of software being developed • Percentage of all code types should add up to 100 percent • Difficulty factor must be defined if user defined code is used

Table 3 - 6: Added RASSP Attributes for Component Entity

3.2.3.1.2 Cost Entity

The attributes in the cost entity primarily contain the development, production and support costs, which are back annotated from the PRICE tool into the RDD-100 database. There must be one cost entity for each component in the equipment tree, which is related to the component by the "costs" relationship. A summary of the cost entity attributes is given in Table 3-7. The budgeted costs in this table are entered by the user and represent the cost requirement. The predicted costs are calculated by the PRICE tool and are back annotated into RDD-100.

Type of Attribute	Attribute	Comment
Development Cost Parameters	• Purchased Item Cost • Budgeted Development Costs • Predicted Development Costs	• Purchased item cost is unit cost of COTS item entered by user • User enters budgeted cost • PRICE calculates predicted development cost
Production Cost Parameters	• Budgeted Amortized Unit Production Cost • Budgeted Unit Production Cost • Budgeted Production Costs • Predicted Amortized Unit Production Cost • Predicted Unit Production Cost • Total Production Quantity • Predicted Production Costs	• User enters budgeted costs • PRICE calculates predicted costs • Total production quantity includes initial spares
Operational Cost Parameters	• Budgeted Operational Costs • Predicted Operational Costs	• User enters budgeted costs • Predicted operational costs are entered by user from either PRICE, RAM-ILS or other source
Support Cost Parameters	• Budgeted Support Costs • Predicted Support Costs	• User enters budgeted costs • PRICE calculates predicted costs
Sensitivity Cost Parameters	• Production Cost Sensitivity • Operational Cost Sensitivity • Support Cost Sensitivity	• User input parameters • RAM-ILS uses parameters in sensitivity calculations when trying to optimize system configuration to meet requirements

		• Parameters are used to specify relative importance among these sensitivities

Table 3 - 7: Cost Attributes

3.2.3.1.3 RMA Entity

The attributes in the RMA entity primarily contain reliability and maintainability metrics which are back annotated from the RAM-ILS tool into the RDD-100 database. There must be one RMA entity for each component in the equipment tree which is related to the component by the "has rma" relationship. A summary of the RMA entity attributes is given in Table 3-8. The budgeted RMA attributes in this table are entered by the user and represent the allocated requirement. The predicted RMA attributes are calculated by the RAM-ILS tool and are back annotated into RDD-100

Type of Attribute	Attribute	Comment
Reliability Parameters	• Allow RMA Quantity Request • Predicted Availability • Predicted Reliability • Budgeted MTBCF • Predicted MTBCF • Budgeted MTBF • Predicted MTBF • Method used for MTBF Prediction • Optimized MTBF • MTBF Optimization Criteria	• User enters budgeted metrics • Allow RMA Quantity Request is a user entered parameter to indicate to the RAM-ILS tool whether redundancy can be considered for this component when optimizing the system • RAM-ILS tool calculates predicted metrics • RAM-ILS can use various methods for MTBF prediction and this method is back annotated into RDD-100 database • RAM-ILS tools is used for sensitivity and optimization studies and the results and criteria used for these studies are back annotated into the optimization metrics • Predicted MTBF is used within PRICE life cycle model
Maintainability Parameters	• Line Replaceable Unit (LRU) • Maintenance Procedure • Budgeted Line MTTR • Predicted Line MTTR • Predicted LRU MTTR at Organization Supply Location • Predicted Module MTTR at Organization Supply Location • Predicted LRU MTTR at Intermediate Supply Location • Predicted Module MTTR at Intermediate Supply Location • Predicted LRU MTTR at Depot Supply Location • Predicted Module MTTR at Depot Supply Location	• User enters budgeted metric • Line Replaceable Unit (LRU) is a user entered parameter to indicate to the RAM-ILS tool whether the component is an LRU • RAM-ILS tool calculates the predicted metrics • Predicted metrics are used as inputs to the PRICE life cycle model
Maintenance Concept Parameters	• Maintenance Concept Requested for Cost Estimating Tool	• User enters requested maintenance concept to direct PRICE which concept to use within the life cycle model

	● Maintenance Concept Used in Cost Estimating Tool	● PRICE selects most cost effective maintenance concept if requested concept field is empty ● PRICE back annotates the maintenance concept used within life cycle model into RDD-100 database

Table 3 - 8: RMA Attributes

3.2.3.1.4 Life Cycle Parameter

The attributes in the life cycle parameter entity characterizes the operational environment and deployment scenario for the product under development. There must be one life cycle parameter entity defined for the system which is related to the system component through the "satisfies" relationship. A summary of the life cycle parameter attributes is given in Table 3-9.

Type of Attribute	Attribute	Comment
Operational Parameters	● Operating Environment ● Environmental Maximum Temperature ● Environmental Minimum Temperature	● User input parameters ● PRICE uses operating environment in cost calculations ● RAM-ILS uses parameters for reliability assessment
System ‘ility Parameters	● Operational Availability ● Inherent Availability ● Reliability	● User input parameters ● System level requirements ● RAM-ILS uses these parameters to determine whether calculated ilities meet the system requirements
Deployment Parameters	● Deployment Quantity ● Prototype Quantity ● Mission Period ● Life Cycle Duration ● On Time Factor	● User input parameters ● PRICE uses parameters for life cycle cost calculations ● RAM-ILS uses parameters for reliability and maintainability assessments
Sensitivity Parameters	● Volume Sensitivity ● Weight Sensitivity ● Power Sensitivity ● Production Cost Sensitivity ● Operational Cost Sensitivity ● Support Cost Sensitivity	● User input parameters ● RAM-ILS uses parameters in sensitivity calculations when trying to optimize system configuration to meet requirements ● Parameters are used to specify relative importance among these sensitivities

Table 3 - 9: Life Cycle Parameter Attributes

3.2.3.1.5 Duplicate Component

A component may be replicated in multiple places within the system equipment tree. Since each instance of a component entity must have a unique name in RDD-100, a means of identifying which components are identical is needed to ensure that the appropriate development, production and support costs are calculated within the PRICE toolset. The duplicate component entity is used to identify families of components that are identical and are built from different parent components. The relationship "includes duplicate" is made from the duplicate component entity to all identical components within a family. Note that when identical components have the same parent within the equipment tree, the attribute entitled "Quantity in Next High Assembly" is used to reflect the number of identical components used in the parent and the use of the duplicate component is not needed for this case.

The attributes for the duplicate component entity are essentially identical to the additional attributes added to the component entity in the RASSP extended schema. The major difference between the attributes is that the various quantity attributes in the component entity have been replaced by the total system quantity attribute in the duplicate component entity. The total system quantity attribute contains the total number of identical components used within one system and this number is calculated and back annotated into the RDD-100 database by running the "Calculate Total System Quantity Report" within RDD-100.

3.2.3.1.6 External Tool File Entity

The attributes in the external tool file entity contains information about file location for both the PRICE and RAM-ILS tool. A summary of the external tool file attributes is given in Table 3-10.

Type of Attribute	Attribute	Comment
PRICE File Parameters	• Cost Analyst File Requested • Sync File Requested • Lock File Requested • Cost File Used • Sync File Used • Lock File Used	• User inputs the requested file names • Specific use of files described in integrated design to cost section • Cost analyst can specify which file to use in lieu of the requested file • Names of actual files used are back annotated into RDD-100 from PRICE
RAM-ILS File Parameter	• RAM-ILS Directory Used	• RAM-ILS tool back annotates this parameter which defines the path name of directory which the RAM-ILS tools uses for RAM analyses

Table 3 - 10: External Tool File Attributes

3.2.3.2 Integrated Design To Cost

3.2.3.2.1 Overview

Traditional cost estimating processes have relied upon estimators and engineers to work within a functionally-oriented organizational process, as shown in Figure 3-7. As a result, the traditional process is slow, inconsistent and generally not repeatable. From the cost estimator's perspective, the costing process is a top-down process that requires the translation of engineering specifications into cost model inputs and an estimating breakdown structure (EBS), and the execution of the model to obtain preliminary results. These results are then iterated back to engineering several times to resolve any questions that arise. From the engineer's perspective, the traditional costing process is a bottoms-up process that requires the creation of an engineering estimate based upon labor hours and a bill of materials supported by vendor quotes. Generally, these two estimates are compared and a resolution is reached via a process that varies from company to company. Although this process may be valuable in providing cost perspectives from two different approaches, this process has proven to be much too slow and costly to be responsive in using cost as a design trade-off parameter in a Design To Cost (DTC) environment.

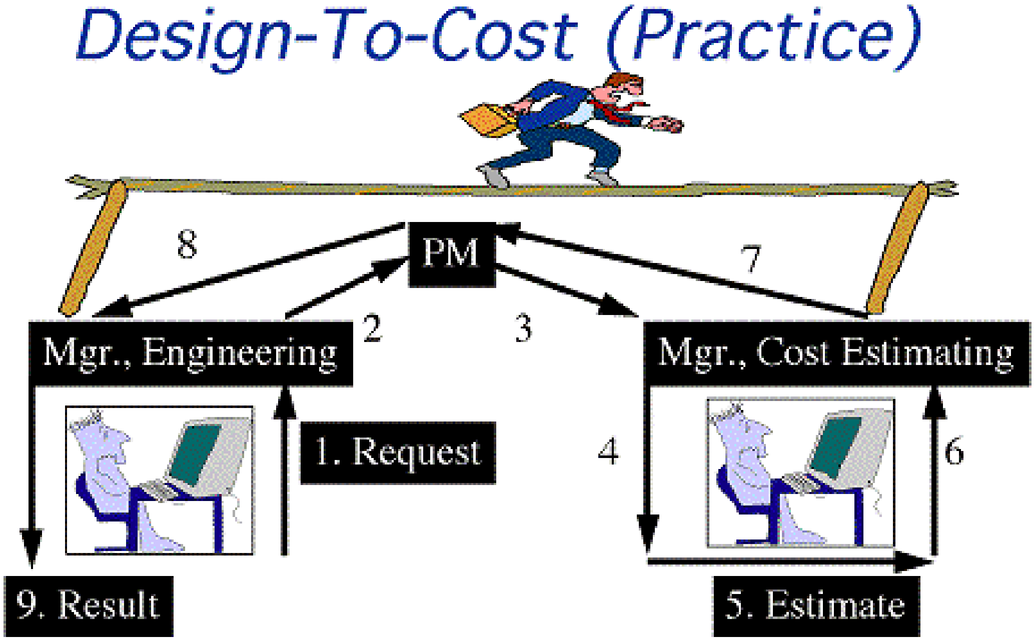


Figure 3 - 7: Traditional Cost Estimating Process

An automated Integrated Design To Cost (IDTC) environment has been developed which enables engineers and cost estimators to work efficiently together using their native tools so many design alternatives can be examined during system concept trade-off studies. As shown in Figure 3-8, the engineer works within his system engineering tool (RDD-100) to enter a physical description of a potential design. This information is then exported out of the system engineering tool and read into the cost estimating tool(PRICE). The data is then translated into cost estimating parameters and merged with information from the cost analyst to produce a complete set of data for the parametric estimating engine. The parametric engine produces a cost and schedule estimate and exports this data back to the system engineering tool. The engineer can then access this cost data within the system engineering tool.

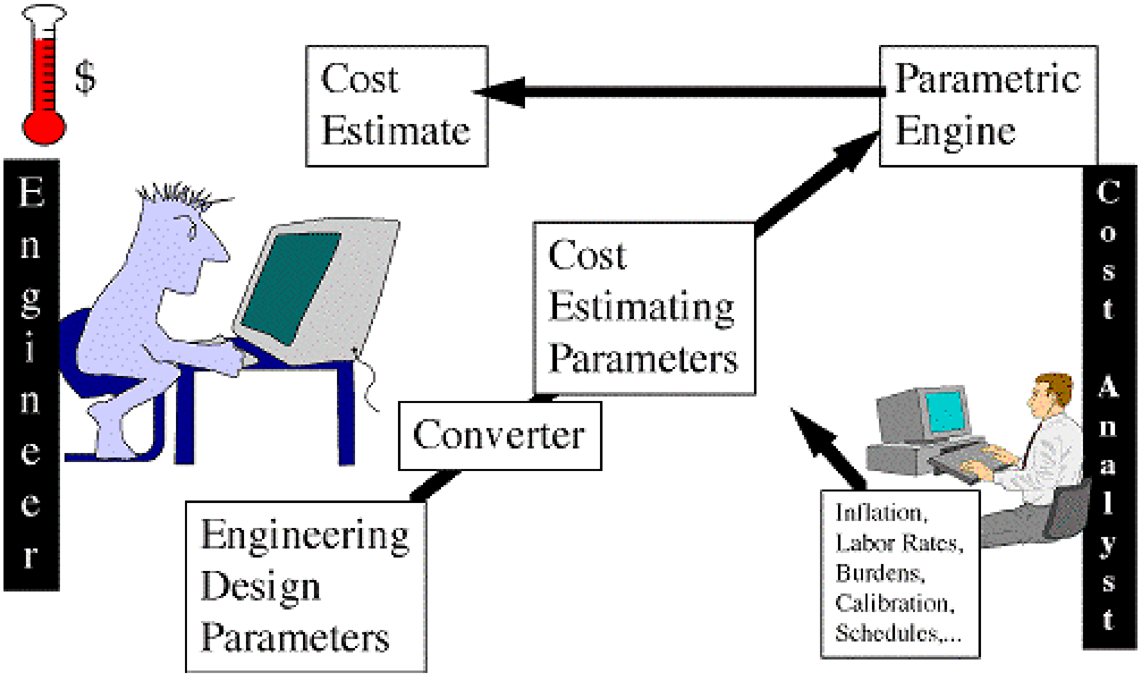

$
Cost Estimate
Parametric Engine
Engineer
Cost Estimating Parameters
Cost Analyst
Converter
Engineering Design Parameters
Inflation, Labor Rates, Burdens, Calibration, Schedules,...

Figure 3 - 8: Integrated Cost Estimating Process

This IDTC estimating process is an improvement to the traditional method in many ways. It is faster, enabling more alternatives to be explored. It is more accurate and repeatable because the estimating relationships are controlled by the estimator, codified into a language script and executed by a computer. Since the relationships are codified, the engineer does not need to meet with the estimator every time a cost estimate is needed. This IDTC process allows the engineer and estimator to work effectively together. A cost estimate can be turned around in minutes instead of days or weeks with this IDTC environment.

The organization of this section is as follows. The physical elements of the IDTC environment are initially described. Then the process to use the IDTC environment is explained.

3.2.3.2.2 IDTC Environment

The RASSP IDTC environment consists of the RDD-100 and PRICE extensions as shown in Figure 3-9. The schema, consistency checks and output reports have been developed for RDD-100 which support the IDTC environment. A cost analyst file, synchronization file, PRICE Rule Language (PRL) import template, and PRL export template have been developed within the PRICE Enterprise toolset to support the automated IDTC environment. The use of each of these elements within the IDTC environment is described below. Note that the parameters calculated by the RAM-ILS tool which are used within PRICE to support cost estimating are not explicitly shown in Figure 3-9, since these parameters are back annotated into the RDD-100 database prior to using them in PRICE.

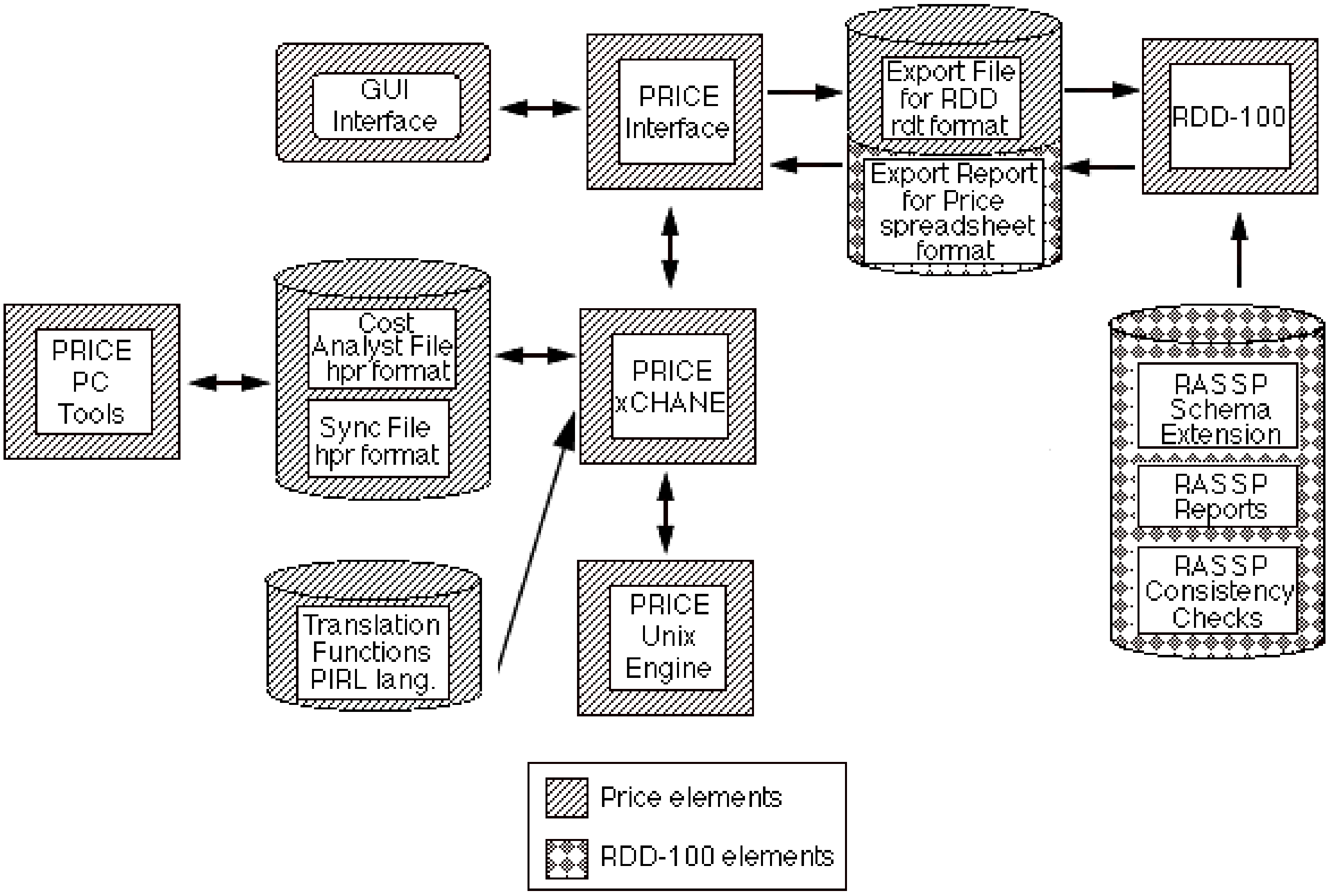


Figure 3 - 9: IDTC Environment

The schema within RDD-100 has been extended on the RASSP program to support both cost estimating and reliability analysis as previously described in section 3.2.3.1. The engineer populates the RDD-100 data base with the system engineering parameters that define the physical configuration of the hardware and software. A consistency report is then executed within RDD-100 to make sure that the database has been sufficiently populated to obtain a cost estimate. An export report is then run within RDD-100 which outputs the system configuration with all the required attributes in the appropriate format (format needed by the PRICE tool is defined in the interface specification) to import into the PRICE cost estimating tools. The cost for each system component is then estimated within the PRICE tool based upon the system engineering parameters and data provided by the cost analyst. The PRICE tool generates an output file in the standard RDD-100 rdt format which contains the development, production and support costs for each system component. This cost data is populated within the RDD-100 data base using the standard RDD-100 import facility.

Cost Analysis File **-** The cost analyst file is used in conjunction with the RDD-100 output file, PRL import template and the synchronization file to establish all of the parameters the PRICE tool needs to perform a cost estimate. The cost analyst file contains default parameters for each component in the system which are missing from translation of the RDD-100 file. The parameters typically defined within the cost analyst file are prototype and production schedule, labor rates, escalation rates and other financial factors that the PRICE tool needs. The information is entered within the cost analyst file on an element type basis. Each component within the PRICE estimating breakdown structure has a particular element type such as electro-mechanical, software and design integration. All components of the same mode type receive the same default parameters contained within the cost analyst file if the parameter is missing from the RDD-100 file and synchronization file.

Synchronization File - The synchronization file is used in conjunction with the RDD-100 output file, PRL import template and the cost analyst file to establish all of the parameters the PRICE tool needs to perform a cost estimate. The synchronization file contains parameters for each component which override any parameter from either the translation of the RDD-100 file or cost analyst file. It is through the use of the synchronization file that the cost analyst can control the cost estimate since any parameter within this file will supersede the same parameter from any other source. The information is entered within the synchronization file on a component name basis which enables each component to have its own parameters within the synchronization file. Since the PRICE interface has been designed to interface with multiple tools, a lock file name is used within the PRICE tool to identify which parameters are active for overriding parameters for each tool interface. It is through the use of the lock file name that the same synchronization file can be used for interfacing PRICE to multiple tools.

PRL Import Template - System engineering parameters within RDD-100 are used to populate attributes within the PRICE toolset. However, there is not a direct one-to-one mapping of the system engineering parameters within RDD-100 to PRICE attributes. A simple illustration of this is that the length, depth and width of a component are entered in RDD-100 for a hardware component, while the PRICE tool only uses volume. As a result, the system engineering parameters must be translated into attributes understood by the PRICE tool. A proprietary interpreted language called PRICE Rule Language (PRL) was developed to translate parameters from other tools into PRICE attributes. As a part of the RASSP program, a PRL import template was written which translates approximately 65 system engineering parameters for each component into about the same number of PRICE attributes. This import template was developed for the signal processing domain, although it may be applicable to digital hardware and software systems. To support other domains, additional translations are required that are unique to that particular domain.

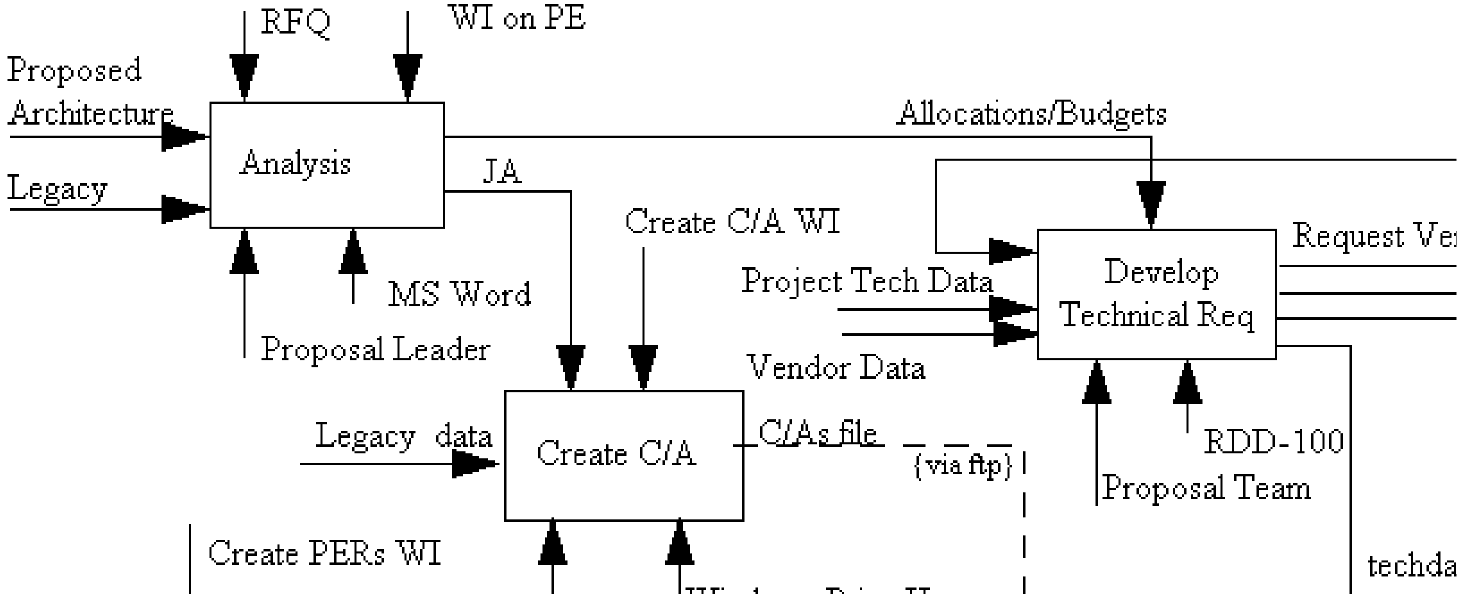
PRL Export Template - Development, production and support costs for each component are back annotated from the PRICE tool into the RDD-100 database. A PRL export template is used to write the cost data out of the PRICE tool into a file with the appropriate format. In this case, the standard RDD-100 rdt format is used so that the data can be imported into RDD-100 with the standard import mechanism.

3.2.3.2.3 IDTC Process

The process in which the IDTC team can effectively implement IDTC is shown in an IDEF3 representation in Figure 3-10. This process diagram is an adaptation of the process used by one of the beta site companies evaluating the RASSP integrated system tools. Each box in this figure represents a process step. The inputs to the process are shown on the left side of the box, the control parameters on the top, the resources needed for the process on the bottom and the output of the process on the right side. The IDTC process consists of six steps which are listed below:

1. Create the parametric estimating relationships;
2. Analyze the programmatic requirements of the project;
3. Create the cost analysis file;
4. Develop and verify system architectures which meet the requirements;
5. Obtain the cost estimate; and
6. Create and maintain the synchronization file.

Each of these process steps is described below. Note that this process is iterative in nature and is repeated as the system design matures, resulting in more accurate cost estimates.



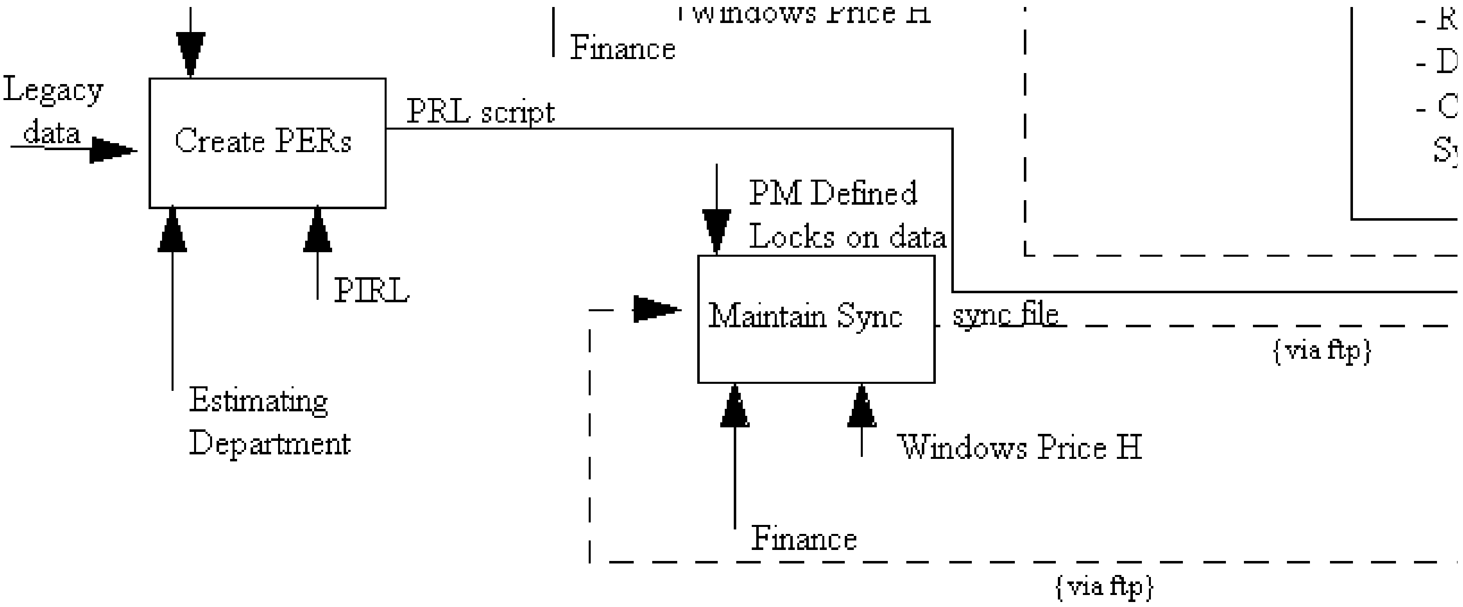


Figure 3 - 10: IDTC Process

Create Parametric Estimating Relationships - The parametric estimating relationships (PER) which are used to convert system engineering data into PRICE cost estimating attributes are established and codified into the PRICE Rule Language (PRL) during this process step. This process step is performed once and should address all product lines and application domains which IDTC is intended to be used. The generation of the PRL import template should be done once for a company and is not typically part of a specific project. The PER's and PRL import template are updated to reflect changes in a company's product line. The cost estimating department is responsible for developing the PER's. The process used to generate these relationships are contained within a company's work instructions. These relationships are developed from legacy data from previous projects. The output of this process step is the PRL import template which codifies the rules to translate RDD-100 system engineering parameters into PRICE attributes. The RDD-100 schema, consistency checks and export reports may need to be modified if additional system engineering parameters are needed to characterize a company's product.

Analyze Project's Programmatic Requirements - The customer requirements are analyzed in this process step to determine whether a company is going to proceed in bidding the program. The program management and proposal response teams perform this process step using the company's legacy data and conceptual baseline architecture for the project. The outputs of this process step include the job instructions for conducting the project and the set of allocated budgets for the program.

Create the Cost Analyst File - The cost analyst file used to supply default parameters while importing system engineering parameters into the PRICE tool is created during this process step. The cost estimating department is responsible for the generation of this file using the PRICE tool. This file contains default parameters which are used to supplement the attributes obtained from the translation of the RDD-100 parameters. This file typically contains prototype and production schedules, labor rates, escalation rates and other financial factors.

Develop and Verify System Architectures - Architecture tradeoffs are performed during this process step to determine the hardware and software elements of the system. This process step is performed by the IPDT team which must consider all aspects of the system life cycle. The IPDT performs this process step using the allocated budgets, project technical information and cost data in determining the composition of the system. The requirements are analyzed, the system functions are decomposed and system architectures are analyzed using RDD-100. This process step is iterative in nature and is repeated for each candidate system design. The output of this process is the system configuration with costing data. The fidelity of the cost data can either be comparative in nature, rough order of magnitude or basis of estimate depending upon the rigor used in developing the parametric estimating relationships.

Obtain the Cost Estimate - The development, production and support costs for a system architecture are determined during this process step. This process step is performed by importing the RDD-100 generated file containing the system engineering parameters using the PRL import template, cost analyst file and synchronization file into the PRICE Enterprise toolset. The outputs of this process step are the cost reports from the PRICE tool and the ASCI file containing the development, production and support costs. This file is used to import the cost data into the RDD-100 database.

Create and Maintain the Synchronization File - The synchronization file used to supply overriding parameters while importing system engineering parameters into the PRICE tool is created during this process step. The cost estimating department is responsible for generating this file using the PRICE tool. This file contains overriding parameters which supersede the values obtained from either the translation of the system engineering data or cost analyst file. This file provides the mechanism that the cost analyst can use to control the cost estimate.

3.2.3.3 RAM (Reliability, Availability and Maintainability) Assessment

3.2.3.3.1 Overview

Traditional RAM engineering processes are disjoint with respect to the product development cycle as shown in Figure 3-11. Reliability engineering has typically been viewed as a necessary evil; a very time consuming effort to be left to specialty engineers. While most engineers have a basic understanding of quality goals and issues, much of the focus in implementing quality has been on the statistical aspects of quality rather than applying reliability knowledge in the engineering design effort. Problems associated with this traditional approach are listed below:

- RAM engineers expend significant effort obtaining numbers for logistics, such as MTBF and MTTR, without really being involved in the early design process.
- Large organizations expend time and effort trying to understand why systems fail, but rarely apply the lessons learned early in a repeatable design process.
- The designer performing the detailed design rarely has a documented list of reliability issues to influence the design.

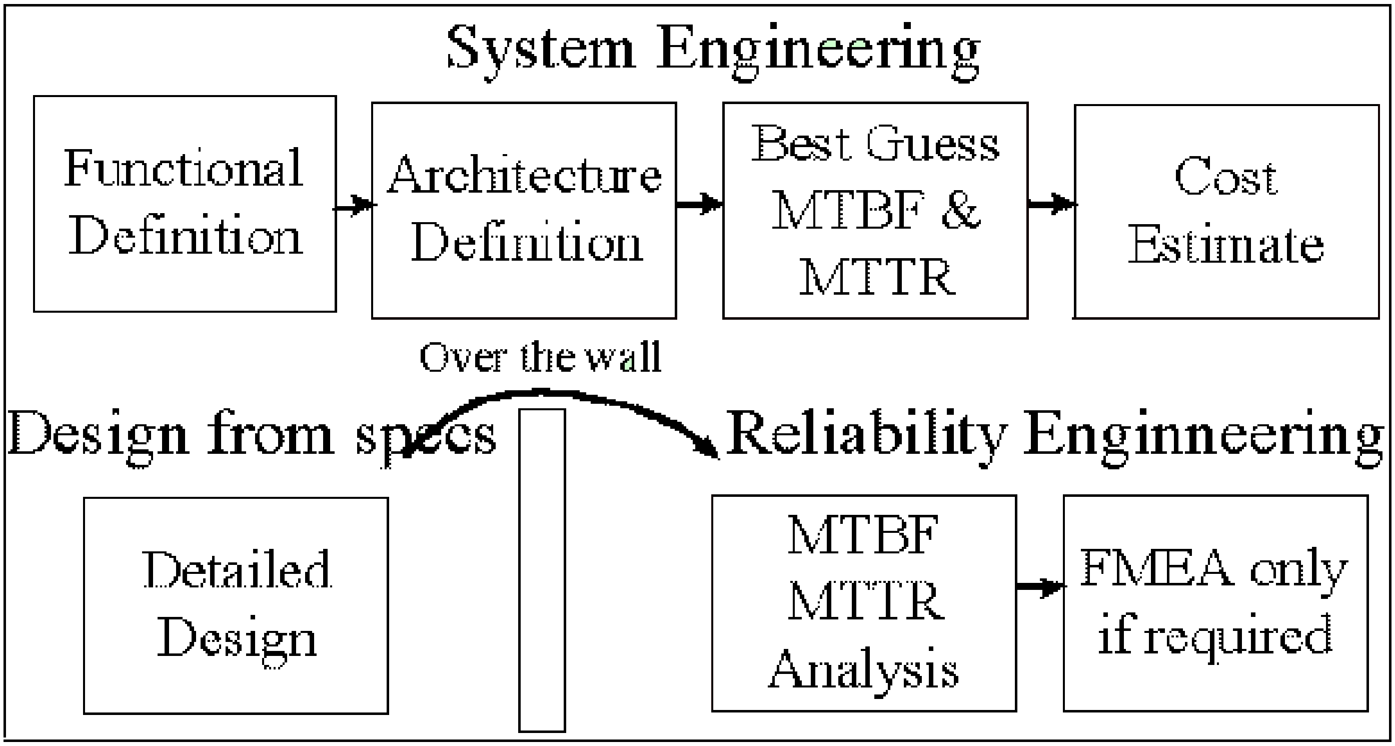
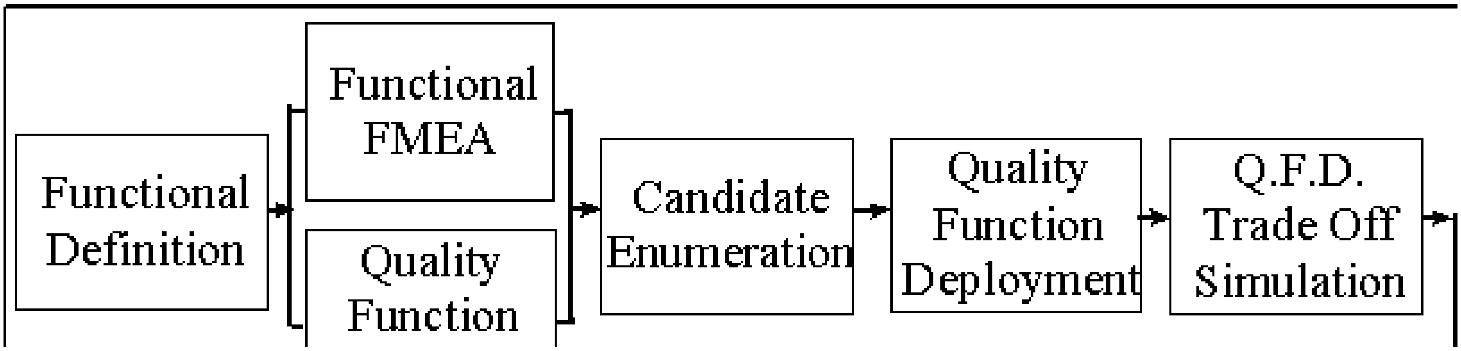


Figure 3 - 11: Traditional RAM Process

Management Sciences, Inc. (MSI) RAM tools provide a new approach to applying traditional "ilities" design techniques early in the design process. The goal of using RAM tools early in the system engineering effort is to reduce the number of nasty surprises encountered later in the product life cycle. It is in this early design phase that the most reliability improvements can be recognized and incorporated cost effectively. Through the early use of the integrated systems tools, reliability requirements and issues are identified and documented cooperatively and concurrently with other design requirements and issues as shown in Figure 3-12. Thus, quality related issues can be applied to the architecture selection process, and issues and requirements can be documented, tracked, and monitored through the product design life cycle.



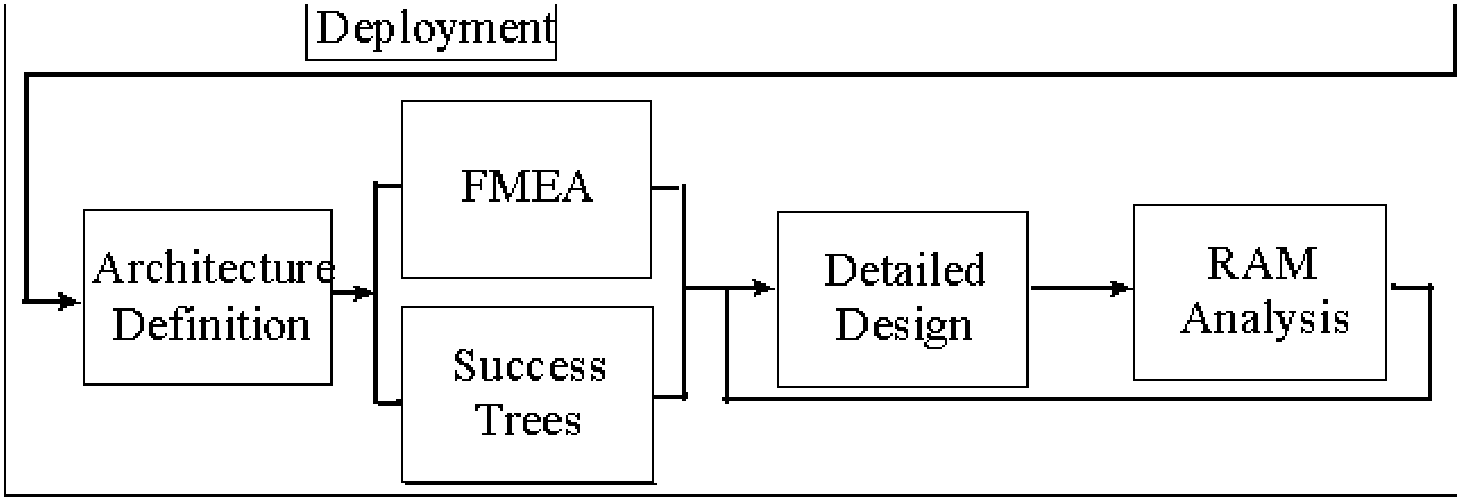


Figure 3 - 12: Integrated RAM Process

The remainder of this section provides a short discussion of the software integration with Ascent Logic's (ALC) RDD-100 tool. This discussion is followed with a description of how the MSI tools support the RASSP systems engineering process. The MSI tools that are discussed include: Quality Function Deployment (QFD), Failure Modes and Effects Analysis (FMEA), and Integrated RAM Analysis. Through reading this section, you will obtain an understanding of the benefits of using these capabilities early in the design process.

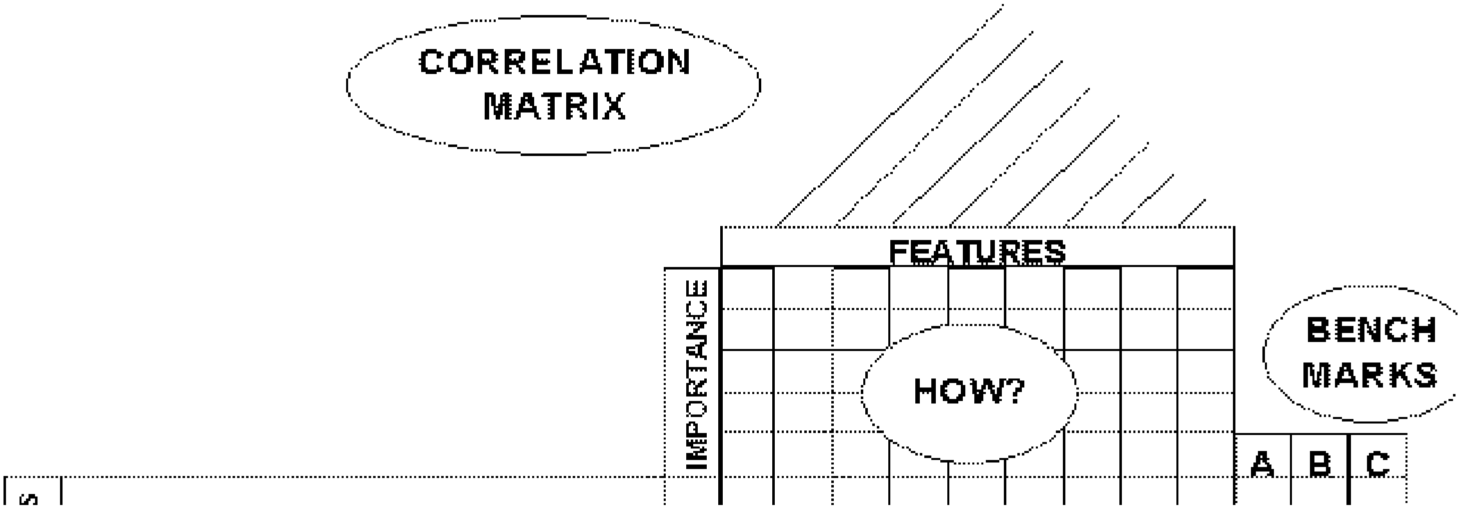
3.2.3.3.2 Software Integration

The RMA entity type has been added to the RDD100 schema (as described previously in Section 3.2.3.1) to support a combination of reliability analysis, documentation, and statistical analysis. This entity type contains statistical values such as MTBF, Reliability, MTTR, and Availability which are used in performing integrated cost estimating and detailed statistical reliability analysis.

The systems engineer populates the RDD-100 data base with the equipment configurations, functional descriptions, interface items and allocated RAM budgets. An export report is run within RDD-100 which outputs the system configuration and its associated attributes in the appropriate format for the MSI RAM toolset. The specialty engineer uses this data within the MSI toolset with detailed CAD data from existing designs to support a variety of RAM analyses such as Quality Function Deployment (QFD), Failure Modes and Effects Analysis (FMEA) and RAM assessment. The predicted RAM attributes calculated by the MSI toolset are back populated within the RDD-100 data base.

3.2.3.3.3 Quality Function Deployment (QFD)

The relationship among the requirements, architecture candidates and customer expectations can be depicted in Quality Function Deployment models. These QFD charts (typically referred to as "House of Quality") show a correlation between what must be done, how to do it, and the relative benefits of each candidate architecture as shown in Figure 3-13. Functions, components and interface items within the RDD-100 data base can be used to populate various elements within the QFD spreadsheet template in the MSI toolset at the indenture level desired. A different chart is created for each function or component within the system. The specialty engineer can then populate the remaining items of this spreadsheet when performing the QFD analysis.



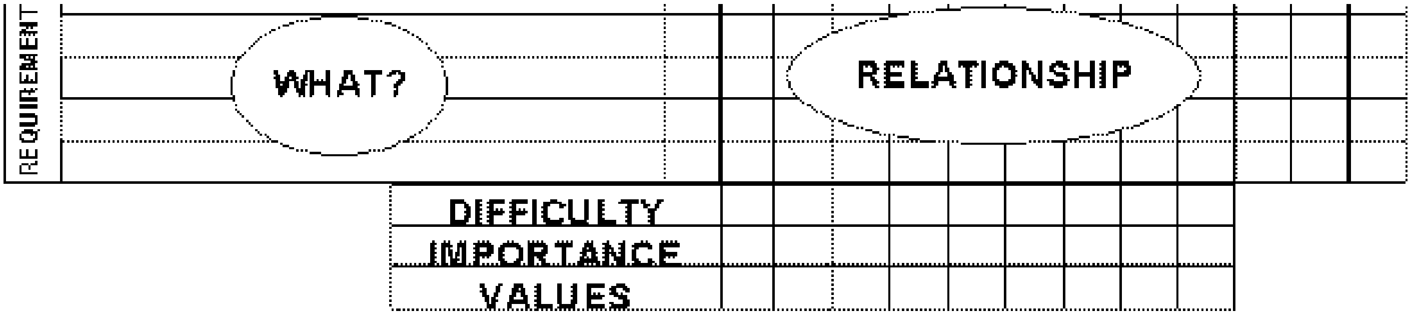


Figure 3 - 13: Quality Function Deployment Template

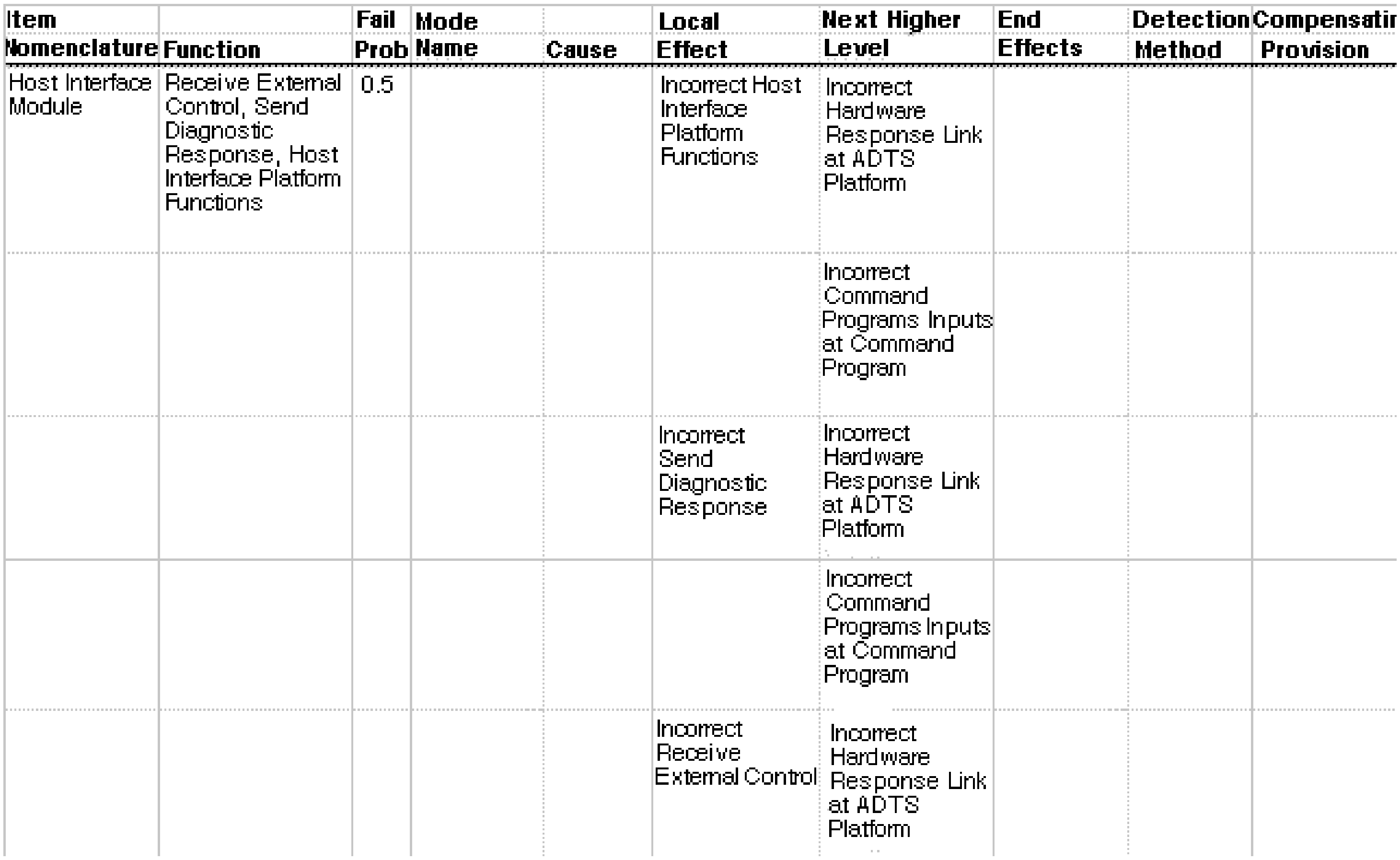
3.2.3.3.4 Failure Modes Effects Analysis (FMEA)

FMEA models define the relationship between components, functions and failures in meeting mission performance requirements. FMEA spreadsheets depict a correlation matrix between how things fail, how the failures will be detected, and what is likely to happen in the event of a failure. The engineer uses this analysis to understand these relationships early in the design and to make changes to the architecture candidate that facilitate a safer, more maintainable, and more reliable system.

Components, functions and interface items defined within the RDD-100 data base are used to initially populate a FMEA data base within the MSI toolset. The specialty engineer can then use this FMEA data base to perform the following tasks:

- Identify potential interface link failure modes and the effects of these failures on the immediate function and mission performance.
- Determine the anticipated method of detecting each failure mode.
- Identify the worst possible consequence of each failure mode.
- Enumerate redundancies or compensating provisions available for each failure mode.
- Identify changes to the RDD system configuration to eliminate or minimize failure modes.

An example of a FMEA spreadsheet populated automatically from the RDD-100 data base is shown in Figure 3-14. Data items that are missing from the worksheet must be manually populated by the engineer.



Item Nomenclature	Function	Fail Prob	Mode Name	Cause	Local Effect	Next Higher Level	End Effects	Detection Method	Compensatir Provision
Host Interface Module	Receive External Control, Send Diagnostic Response, Host Interface Platform Functions	0.5			Incorrect Host Interface Platform Functions	Incorrect Hardware Response Link at ADTS Platform			
						Incorrect Command Programs Inputs at Command Program			
					Incorrect Send Diagnostic Response	Incorrect Hardware Response Link at ADTS Platform			
						Incorrect Command Programs Inputs at Command Program			
					Incorrect Receive External Control	Incorrect Hardware Response Link at ADTS Platform			

Figure 3 - 14: Automated FMEA Worksheet

3.2.3.3.5 Integrated RAM Analysis

RAM models define the relationship between architecture candidates, failure probabilities, availability, and repair times. The components hierarchy and allocated RAM defined within the RDD-100 data base are used to populate the MSI RAM toolset. This toolset can be then used for the following tasks:

- Calculate reliability and availability for designs incorporating redundancy.
- Identify candidate architectures that adversely affect system reliability or availability.
- Model maintenance strategies, such as level of repair and fault isolation characteristics.
- Form basis for more accurate cost estimates for warranty and life cycle cost.
- Identify designs that do not meet RAM requirements.
- Perform sensitivity analysis .
- Determine methods for solving reliability problems.
- Develop strategy to support the detailed design cycle.

The RAM predictions within the MSI toolset are based upon the data populated from RDD-100 and data the user adds within the RAM tools. Typical sources for MTBF and MTTR values used for the system engineering activities are estimates of acceptable loss rate, "similar to" designs, historical data and vendor data. The results of the RAM assessments within the MSI toolset are back annotated in the RDD-100 data base.

3.3 Benefits of Using the RASSP Integrated System Tools

The integrated system tools provide a concurrent engineering environment where tradeoffs considering a product's complete life cycle are performed. Multi-disciplinary design data is stored in one location that the entire IPDT team can assess. As a result, the entire design team uses the same data within their analyses which eliminates the confusion when parameters are maintained in multiple locations. Cost performance tradeoffs are performed using the integrated tools to optimize the system design over multiple disciplines. The integrated tools provide a quick impact analysis capability, as detailed design data can be used to update the reliability and cost predictions.

The integrated system tools provide an efficient process for the engineer to access cost data. Engineers can obtain complete life cycle costs using the system tools without becoming an expert cost analyst. The integrated system tools provide an environment which can be effectively used to implement either Design To Cost (DTC) or Cost as an Independent Variable (CAIV) programs which are being emphasized within DoD. The cost estimation process has been established so the cost analyst is able to control the estimate.

The integrated system tools provide a reliability and maintainability analysis capability throughout the design process. These tools provide the mechanism that allows specialty engineers to be involved early in the design process. The RAM-ILS tool provides capabilities to perform reliability, maintainability, success tree and FMECA analyses. In addition, the system architecture can be optimized to meet reliability requirements in a cost effective fashion with the integrated system tools.

Next Up Previous Contents

Next: 4 Application of Integrated Systen Tools to SAR Benchmark **Up:** Appnotes Index **Previous:**2 Introduction

Approved for Public Release; Distribution Unlimited *Dennis Basara*

Next Up Previous Contents

Next: 5 References **Up:** Appnotes Index **Previous:**3 Technical Description

RASSP Integrated System Tools Appnote

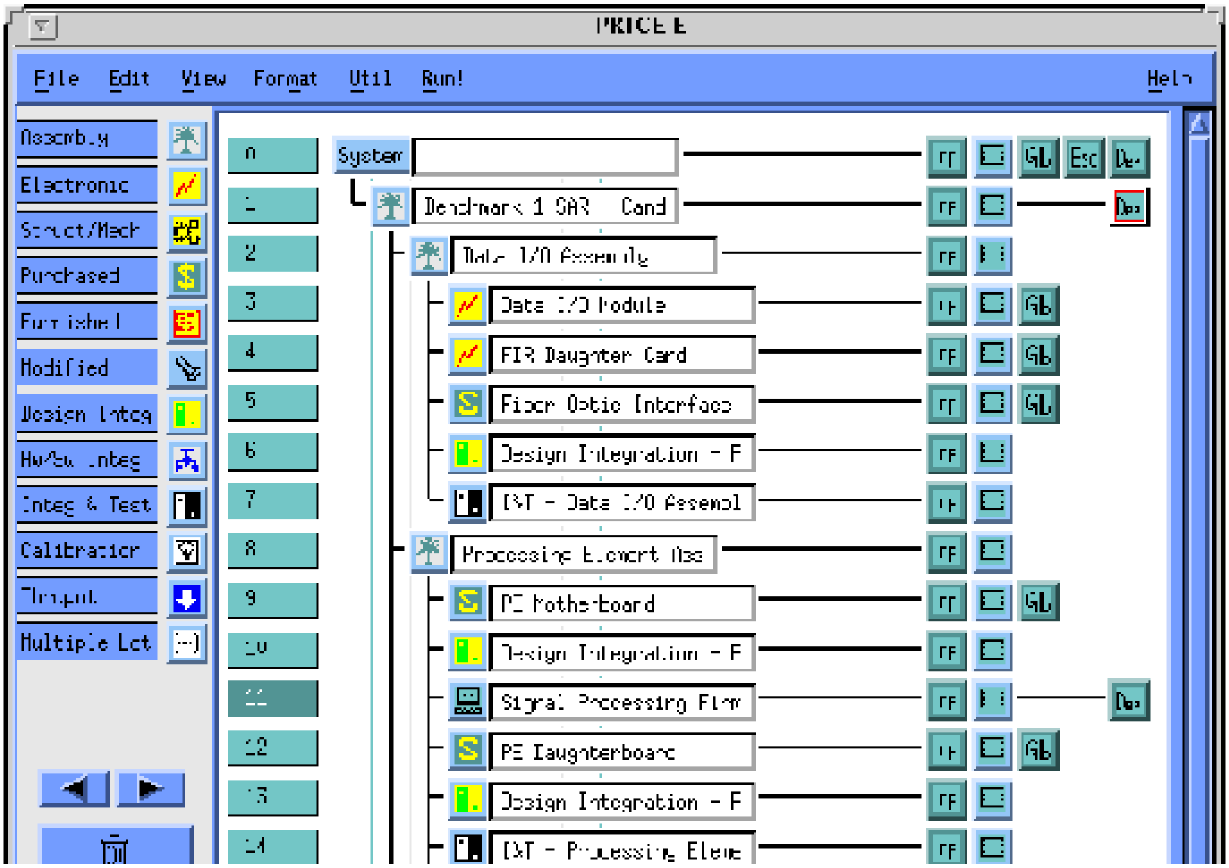
4.0 Technical Description

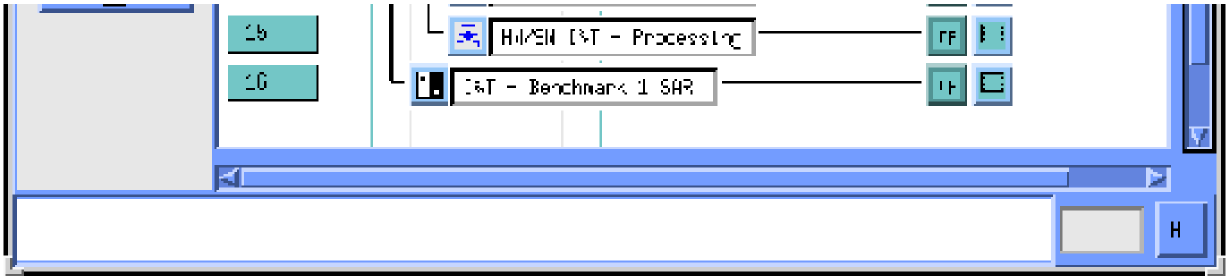
4.1 SAR Benchmark Overview

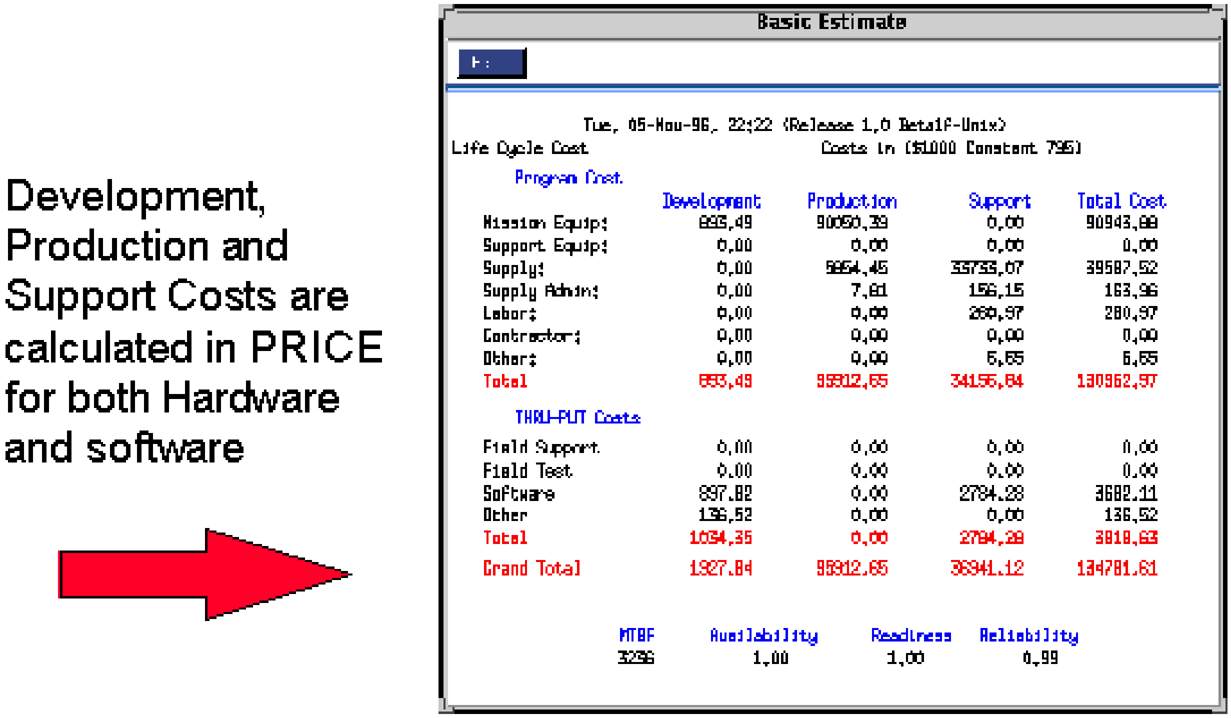
A series of benchmark activities have been performed on the RASSP program to demonstrate how the 4x design improvement are met as the program progressed. The initial benchmark, conducted primarily during 1995, consisted of the development of a signal processor for a Synthetic Aperture Radar (SAR) application. Hardware/software co-design tradeoffs were performed to determine the most cost effective implementation for each function of the SAR processor. The hardware and software were then designed, built and integrated. Although the RASSP system tools had not been integrated in time to directly support this benchmark, these integrated tools were applied to this application after the system had been developed to illustrate how these integrated tools could lead to more cost effective designs early in the design cycle. Additional detailed information may be found in the SAR Case Study.

Note: Several capabilities of the integrated tools described in the previous section will not be illustrated in this example as a complete system level design can not be presented in this application note.

Two of the most prominent architectures for the SAR application form the basis of this example. These two architectures are illustrated in Figure 4 - 1. The number of processing boards required in each architecture was initially determined from the processing throughput requirements, the peak processing capability of each board and an estimate for the processing efficiency for each approach. The first architecture candidate consists of five signal processing boards containing four compute elements per board. These processing boards are based upon mature signal processing technology. The signal processing boards are connected via a crossbar network. Radar data enters through the fiber interface and control is provided by the single board computer. The second architecture candidate uses state-of-the-art signal processing boards. As a result, only three boards are required to perform the same SAR application. These boards have either two or four processing nodes per board. The same types of network interface, radar interface and control computer are used in the second architecture as the first. The integrated RASSP system tools are used in this example to determine the most viable approach between these two architectures. A description of how the individual system tools are used on this application is also included in this example.






Development, Production and Support Costs are calculated in PRICE for both Hardware and software

Basic Estimate

```
            Tue, 05-Nov-96, 22:22 (Release 1.0 Beta1f-Unix)
Life Cycle Cost                               Costs in ($1000 Constant 795)
      Program Cost
                     Development     Production       Support    Total Cost
  Mission Equip:          893,49       90050,39          0,00      90943,88
  Support Equip:            0,00           0,00          0,00          0,00
  Supply:                   0,00        5854,45      33753,07      39587,52
  Supply Admin:             0,00           7,81        156,15        163,96
  Labor:                    0,00           0,00        260,97        280,97
  Contractor:               0,00           0,00          0,00          0,00
  Other:                    0,00           0,00          6,65          6,65
  Total                   893,49       95912,65      34156,84     130962,97

      THRU-PUT Costs

  Field Support             0,00           0,00          0,00          0,00
  Field Test                0.00           0,00          0,00          0,00
  Software                897.82           0,00       2784.28       3682.11
  Other                   136,52           0,00          0,00        136,52
  Total                  1034,35           0,00       2784,28       3818,63

  Grand Total            1927.84       95912.65      36941.12     134781.61


               MTBF     Availability     Readiness   Reliability
               3296         1,00            1,00         0,99
```

Figure 4 - 1: Two Candidate Architectures for SAR Application

4.2 Requirements Analysis

RDD-100 is used to support the requirements analysis task for the SAR benchmark. The system level requirements for the SAR processor are contained within a technical description document. The text for each requirement paragraph of this document is initially parsed into its own requirement within RDD-100. Each requirement paragraph is then refined and decomposed into lower level requirements within RDD-100. Each requirement must be decomposed to a single testable unit so that it can be verified during system acceptance testing. An illustration of the requirements decomposition is shown in Figure 4 - 2 . A graphical representation of one requirement paragraph decomposed into lower level requirements within RDD-100 is shown in this figure. A naming convention which incorporates SOW as part of the requirement name is used so that the source requirements are readily apparent within the database. This top level requirement is decomposed into six lower level requirements as the initial requirement paragraph within the technical description document actually contains multiple requirements. The "incorporates" relationship within the RDD-100 schema is the relationship which links a parent requirement to a child requirement. This requirements decomposition is used to identify and resolve any ambiguous or missing requirements with the customer. The decomposed and refined set of requirements is used to generate the specification for the signal processor.

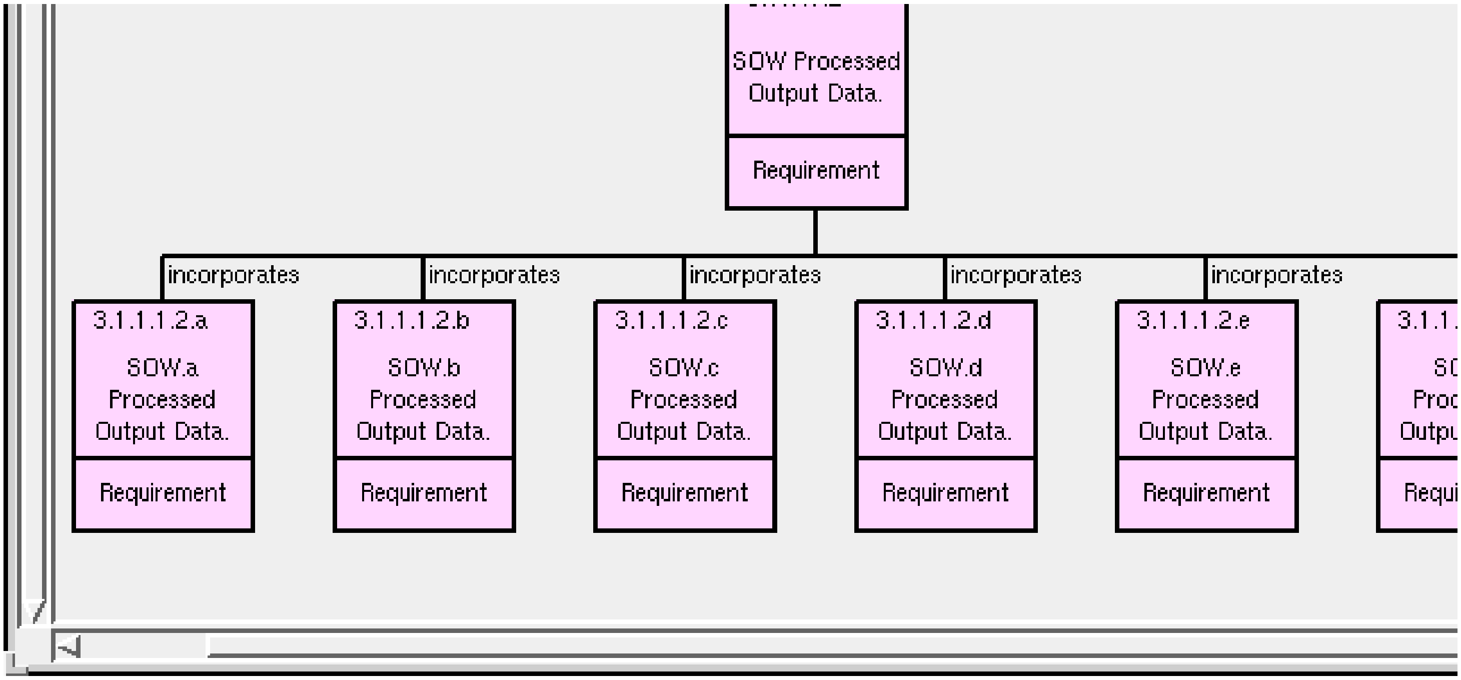


Figure 4 - 2: Requirements Decomposition

4.3 Functional Decomposition

A functional decomposition of the SAR processor is performed after the initial set of requirements are well understood. This decomposition is performed by defining the set of functions, interfaces, control and data flow needed to satisfy the system requirements. The initial functional decomposition is performed without the notion of the physical architecture. However, this process is performed concurrently with the system partitioning task, as elements of the physical architecture impact the physical decomposition. Behavioral simulations (see the Token-Based Performance Modeling application note) of the system are performed during functional decomposition using other complimentary tools/languages to ensure that all system requirements are met.

RDD-100 is used during this task to define the system functionality using an ALC graphical representation called a behavior diagram. The top level behavior diagram showing the interaction of the SAR processor with its external environment is shown in Figure 4 - 3. The rectangular boxes represent functions, while the rectangular boxes with rounded corners indicate data items. The outside world (radar system) and the SAR processor functions appear on parallel branches in this figure since each can operate independently from each other. The flow of data between the SAR processor and its outside world is shown in this figure. The SAR processor receives control and sensor data from the outside world, while it sends output processed SAR and diagnostic data back to the external system. The black square in the upper corner of a box indicates there is a hierarchy of functions contained within that box. The top level system functionality is decomposed into lower level functions until the leaf level function can be allocated to a specific hardware or software item.

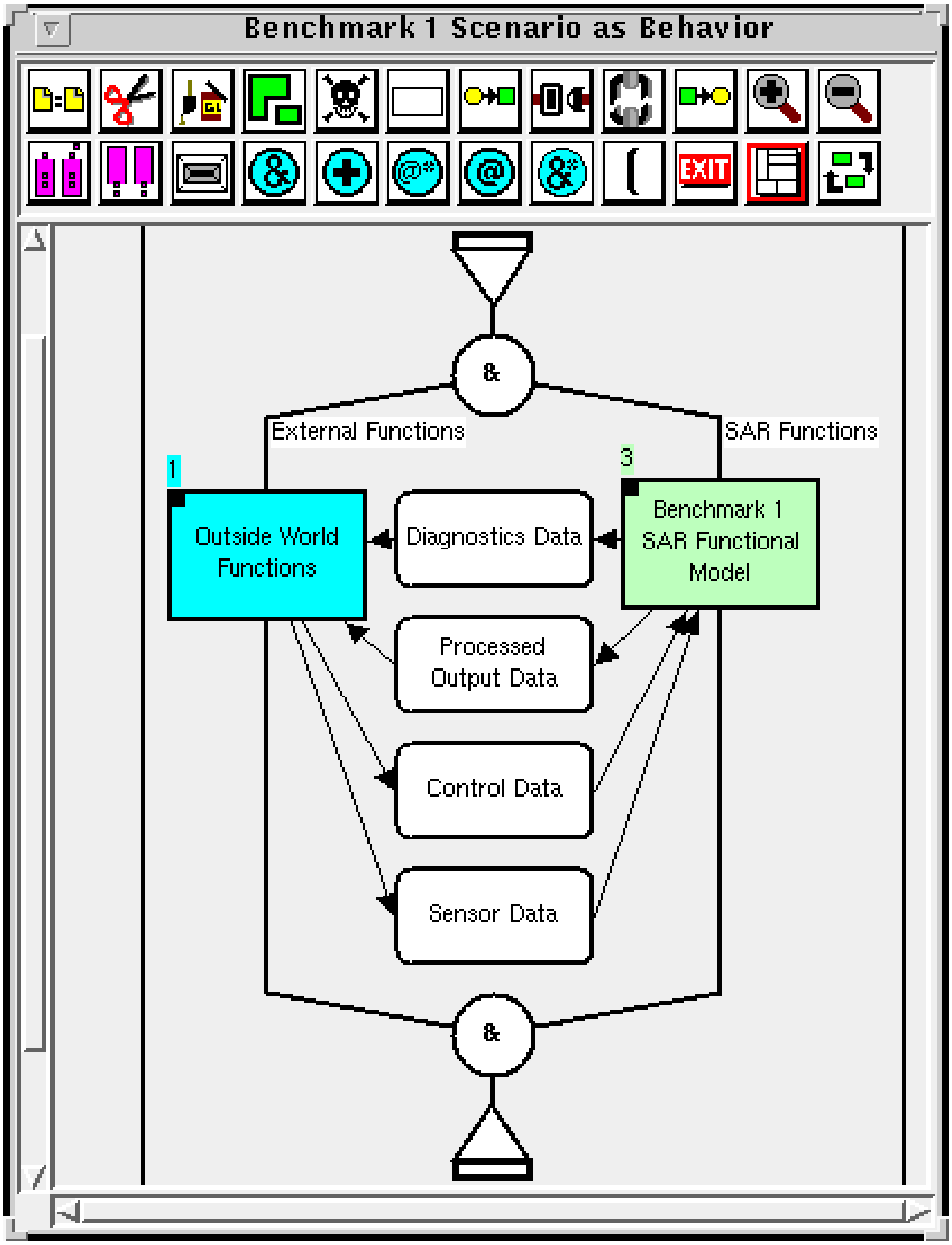
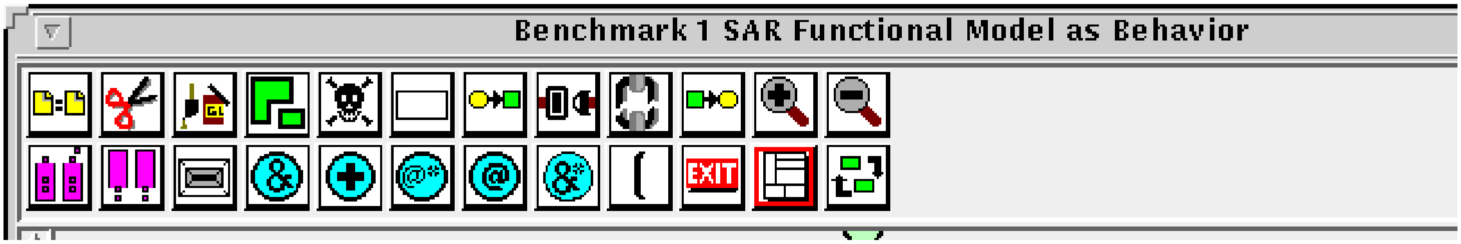


Figure 4 - 3: Top Level Functional View

The next level of functional decomposition for the SAR processor is shown in Figure 4 - 4 (Only a portion of this diagram has been included in the figure due to its large size. A full view of this diagram is found in the SAR case study.) This diagram illustrates the top level functionality the processor and the data items passed among these functions. The functional decomposition for one of the top level functions (host interface platform functions) is shown in Figure 4 - 5. Note that both data and control flow are shown in this behavior graph. Data flow is shown from left-to-right, while control flow, illustrated by the loops in this diagram, is shown from top-to-bottom. The functional decomposition is used to fully characterize the set of functions, interfaces, control and data flow needed to meet system requirements.



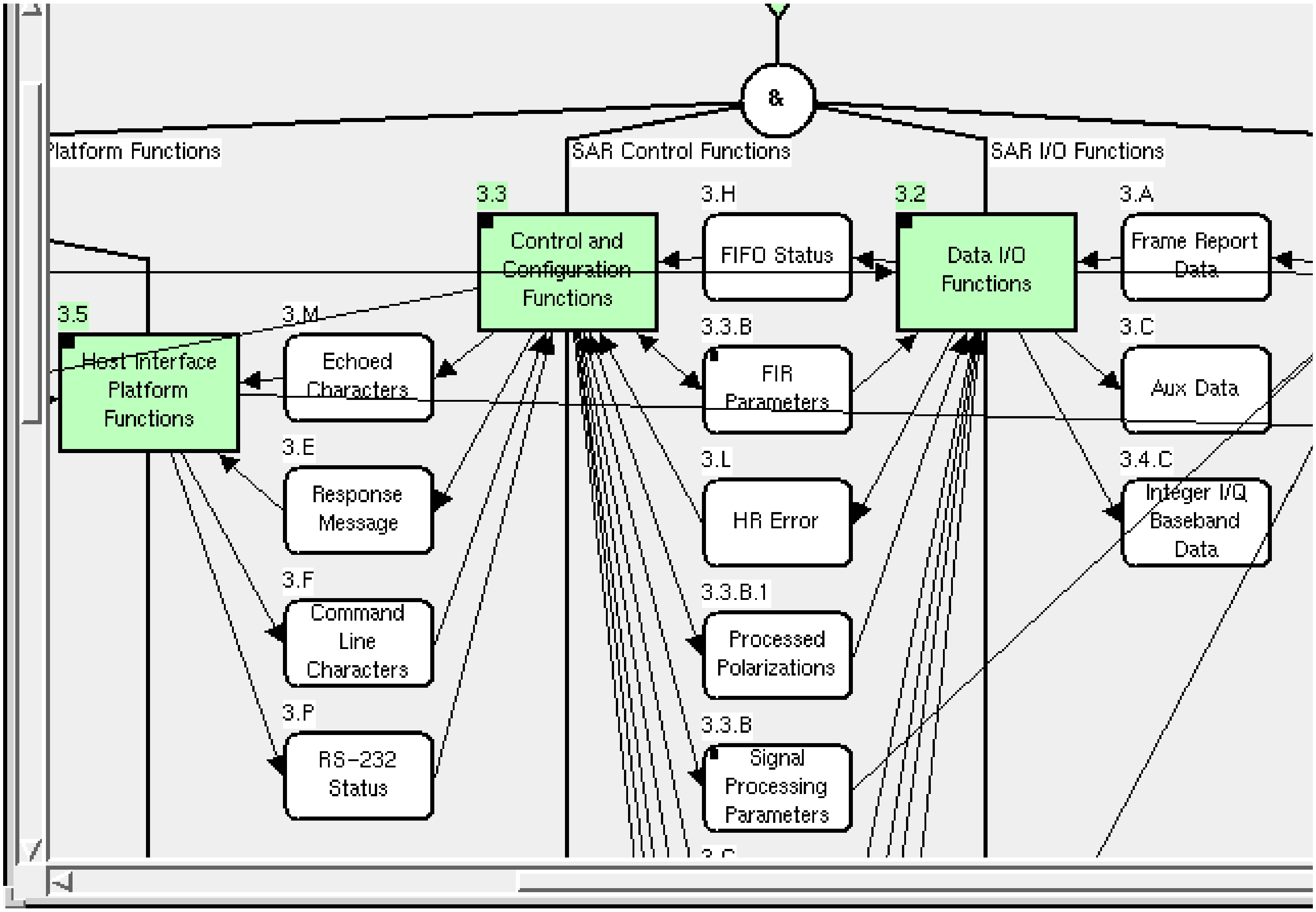
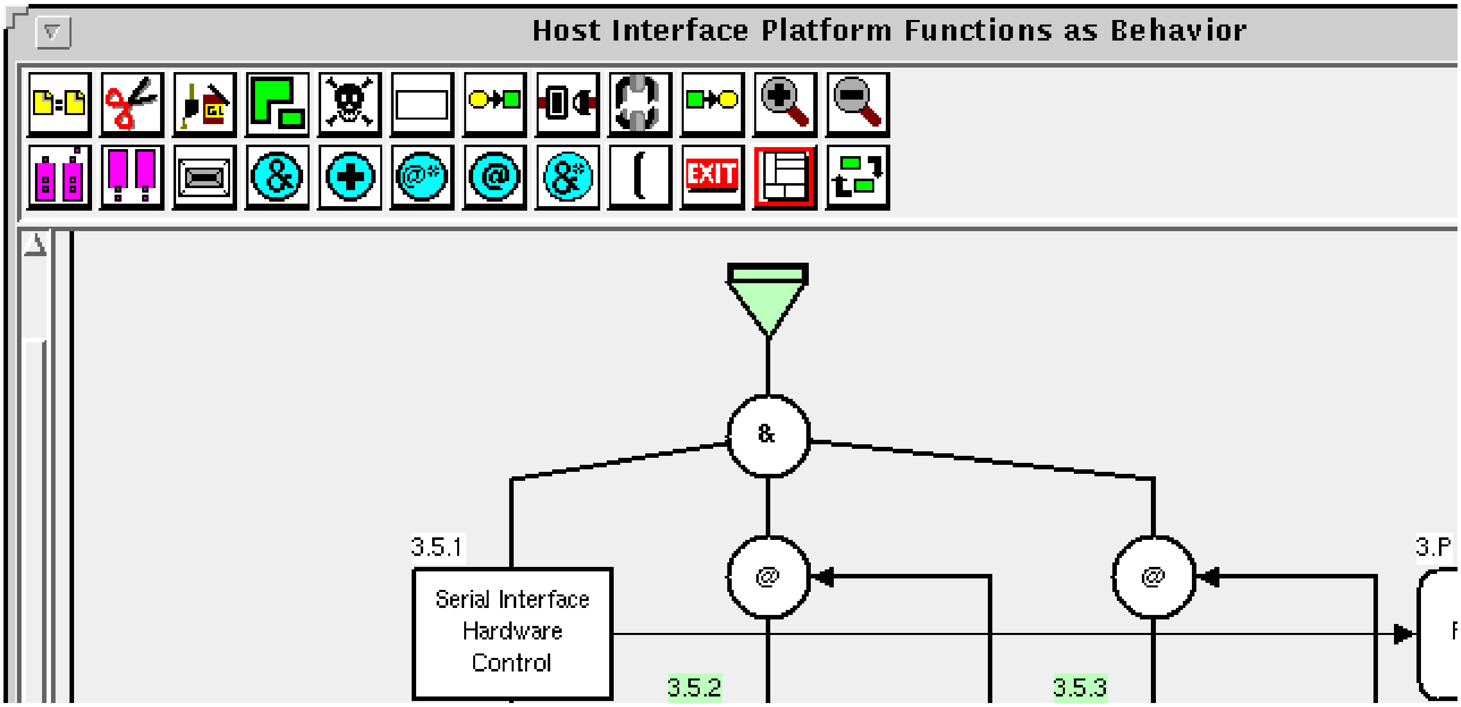


Figure 4 - 4: Top Level SAR Processing Functions


Host Interface Platform Functions as Behavior
&
EXIT
&
3.5.1
Serial Interface Hardware Control
@
@
3.5.2
3.5.3
3.P

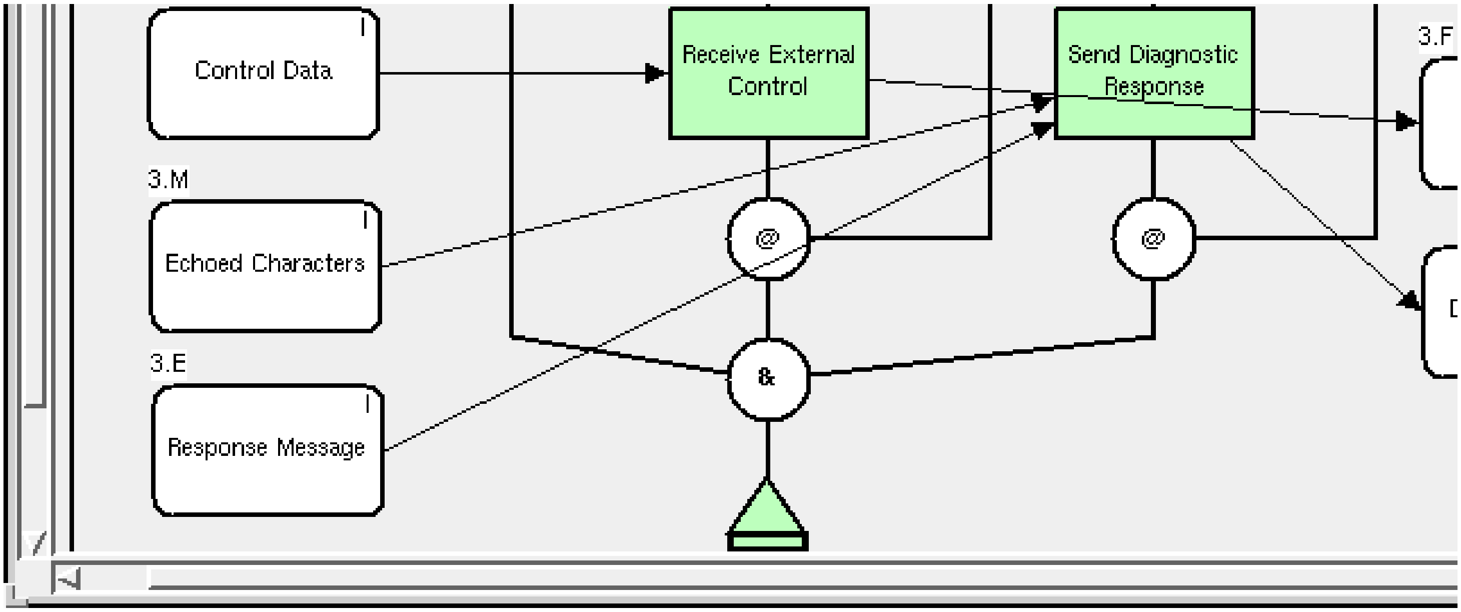
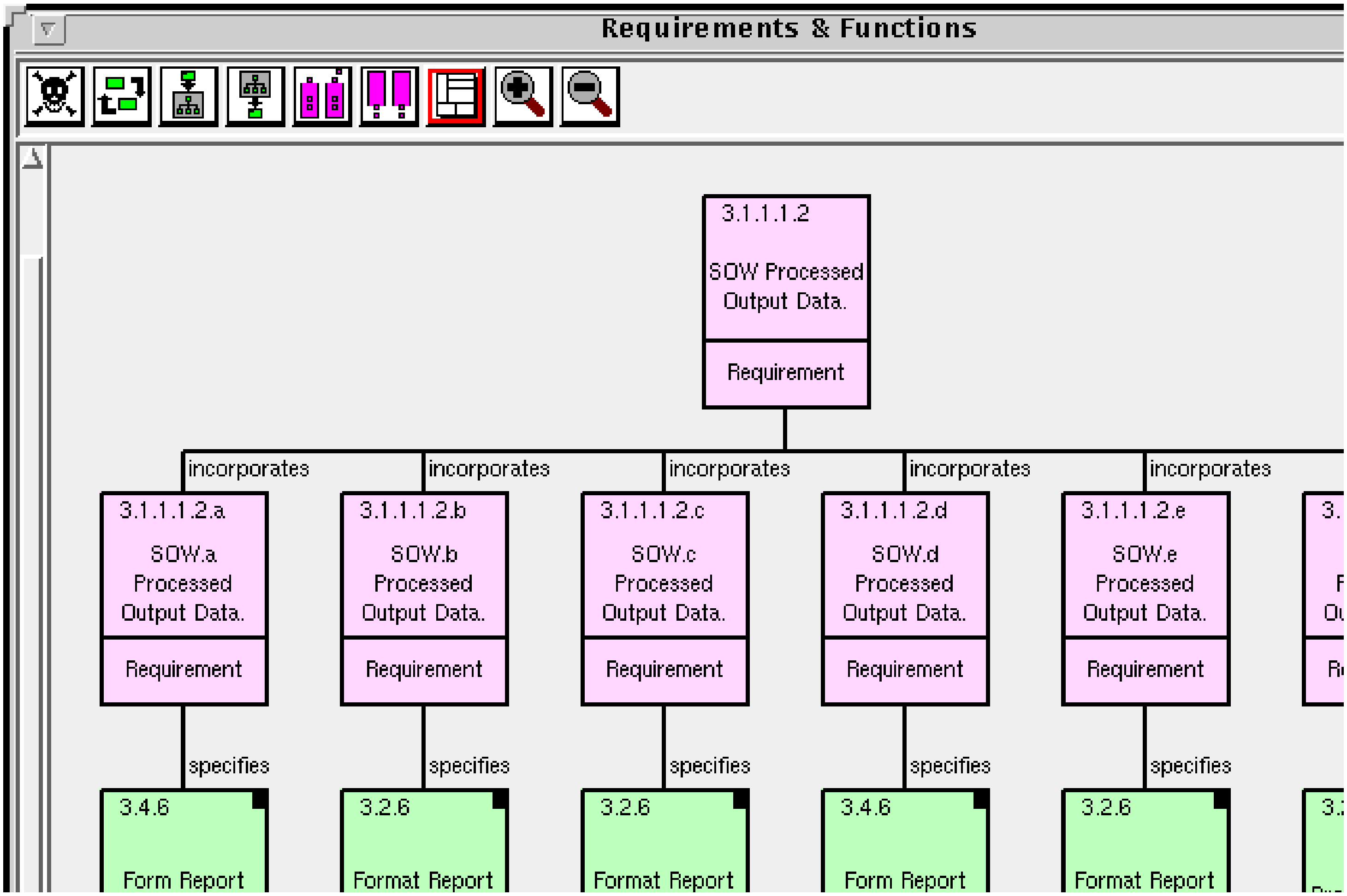


Figure 4 - 5: Host Interface Functions

Traceable links are established in RDD - 100 between the requirements and system functions to ensure that every requirement has been allocated to a function and that every function is directly attributed to a requirement. The links established in RDD - 100 for one of the originating requirement paragraphs are illustrated in Figure 4 - 6. The "specifies" relationship within the RDD - 100 schema is the relationship which links requirements to functions.



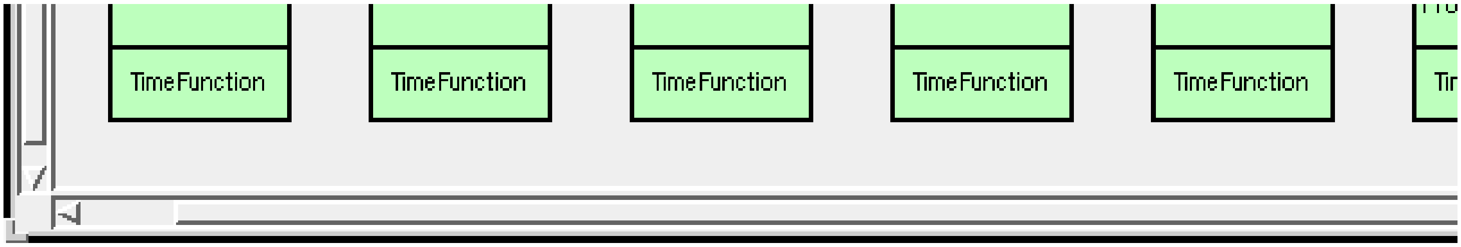


Figure 4 - 6: RDD Links Between Requirements and Functions

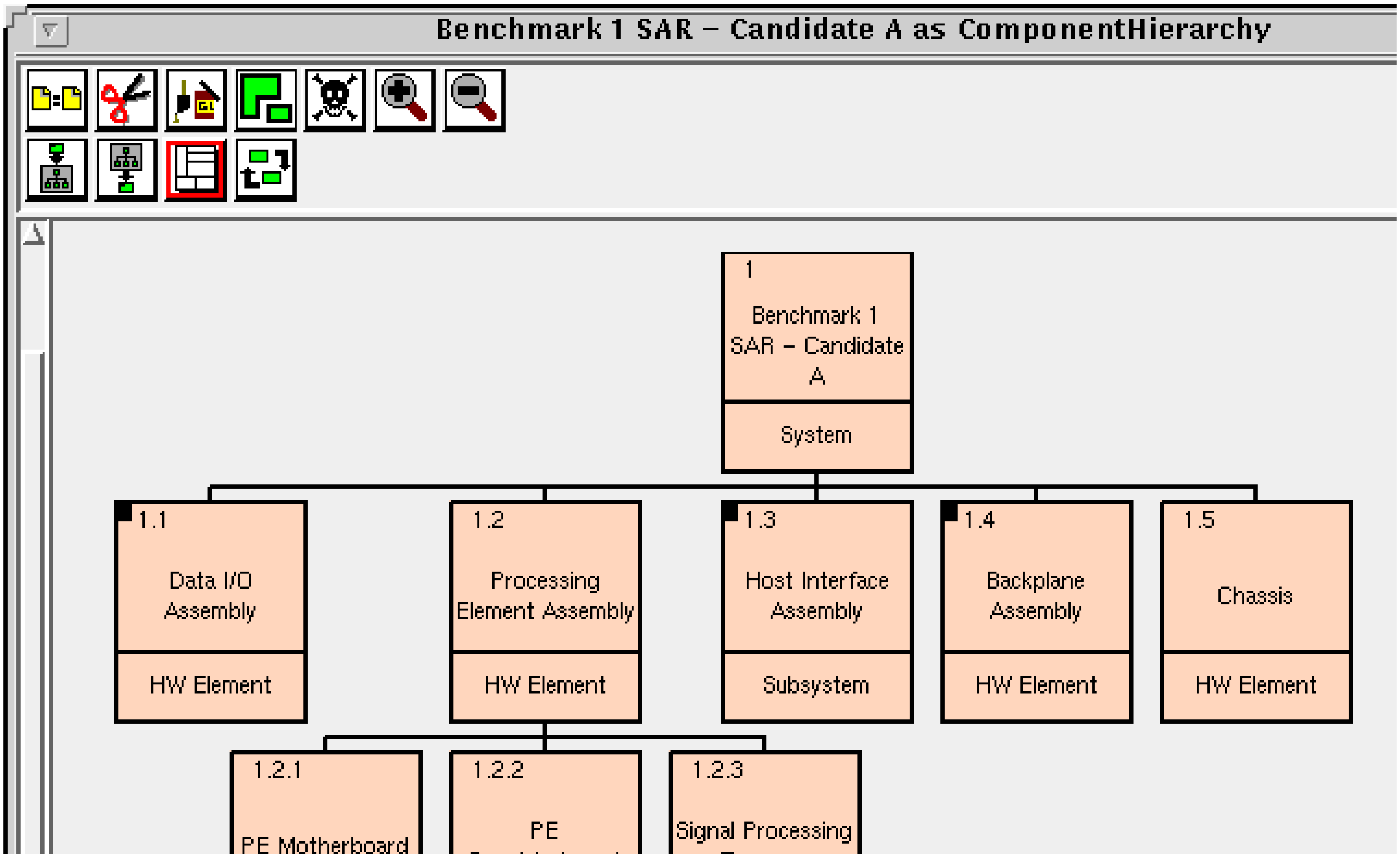
4.4 System Partitioning

System level tradeoffs are performed during the system partitioning task to determine the most effective set of subsystems needed to satisfy the requirements. All aspects of the system's life cycle should be considered when performing these tradeoffs. The system partitioning task is performed concurrently with functional decomposition since the selection of subsystems does impact the functional decomposition.

The use of the integrated tools which support the requirements and functional allocation, cost analysis and reliability tradeoffs is described within this section. Emphasis is placed on illustrating how these three tools can be used cooperatively to perform tradeoffs that consider the entire life cycle.

4.4.1 Requirements & Functional Allocation

The third view of a system within RDD-100 is the physical view. The equipment tree for the first architecture (mature signal processor) in this example is shown in Figure 4 - 7 . Each of the blocks in this figure represents either a hardware or software component in the system. Traceable links are established in RDD-100 between the system functions and components to ensure that every function has been allocated to a component and that every component is directly attributed to a function. The links established in RDD-100 for one of the originating requirement paragraphs are illustrated in Figure 4 - 8 . The "allocated to" relationship within the RDD-100 schema is the relationship which links functions to components. The full set of military specifications can be generated from the information within the RDD-100 database.



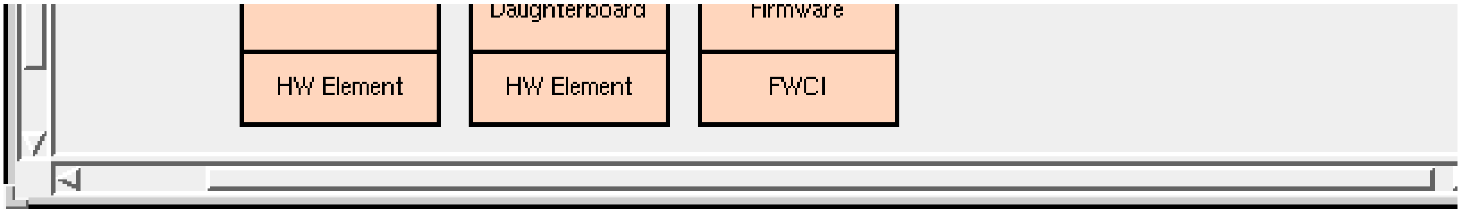


Figure 4 - 7: Equipment Configuration for First Candidate Architecture

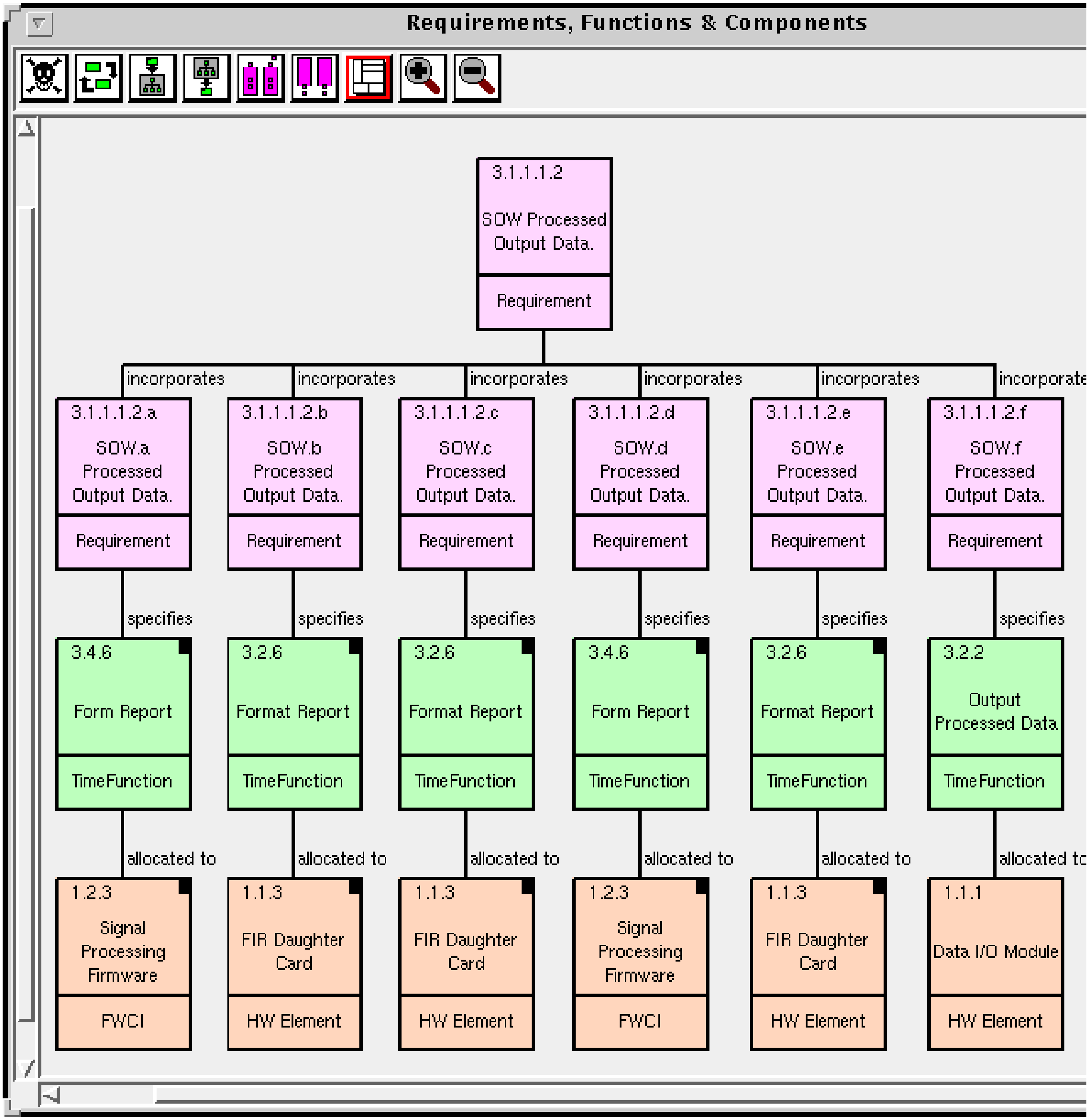


Figure 4 - 8: RDD Links Between Functions and Components

4.4.2 Cost Analysis

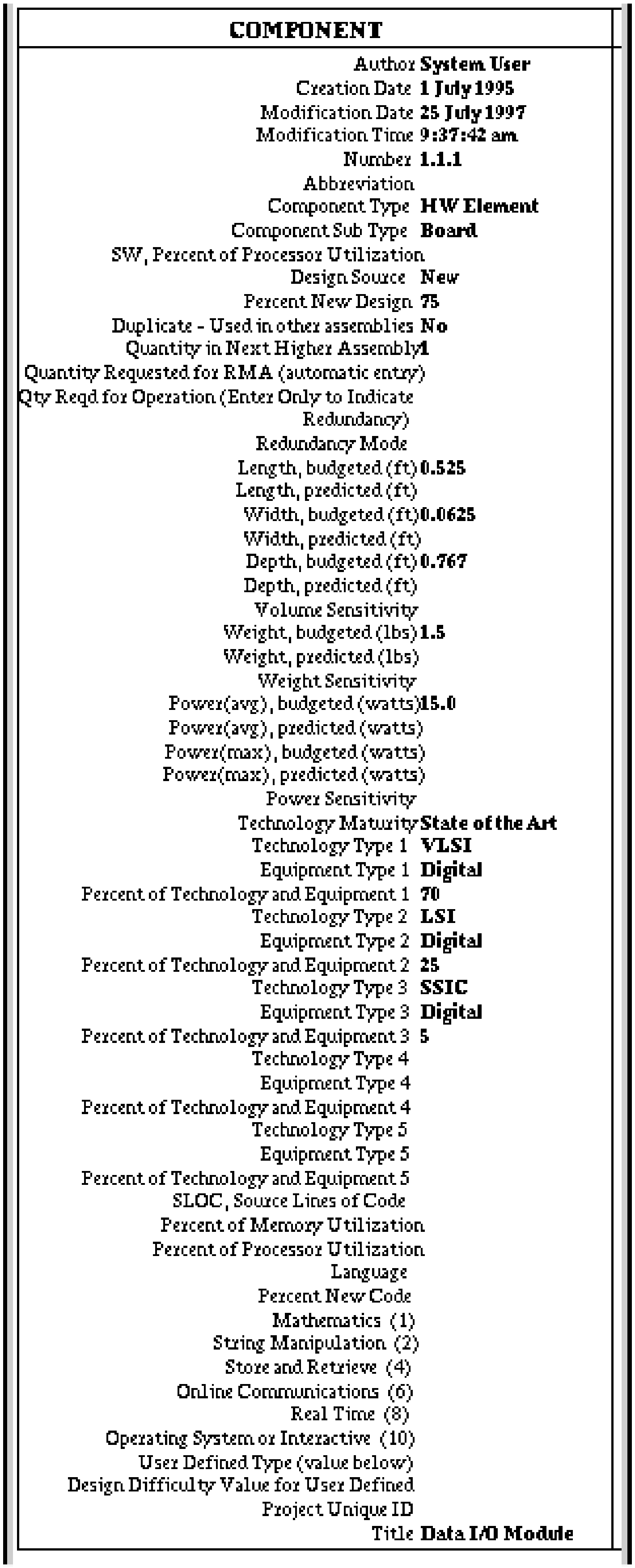
The RDD-100 schema has been enhanced on RASSP so engineers can easily get a cost estimate to support system level tradeoffs. Four different files are needed to generate a cost estimate: a PRICE Rule Language (PRL) import file which contains the translation functions needed to convert system engineering parameters into cost estimating attributes, an output file from RDD-100 which contains the system engineering parameters for the system being costed, a cost analysis file which contains default cost estimating parameters and a synchronization file which contains parameter values which supersede the values obtained from translation of the system engineering parameters. Each of the files used to generate the cost estimate for this example are described below.

PRL import file **-** An initial set of translation functions was developed and codified for signal processing applications on RASSP. These functions define the mapping between 65 system engineering parameters and approximately the same number of PRICE parameters. The PRL import template is typically generated once for a company and should address the complete product line and application domains. The baseline PRL import file developed for RASSP is used in this example.

RDD-100 Output File **-** Attributes which characterize the system architecture, hardware, software and product life cycle must be populated in the RDD-100 database prior to exporting the file needed for cost estimating. The types of attributes that must be entered into the RDD-100 database for both a custom digital board (data i/o module) and software (signal processing firmware) are shown in Figure 4 **-** 9 . The data i/o module is a new, custom board development requiring 75 percent new design. This board has a VME size and expected to weigh 1.5 pounds. The maturity of the technology the board is built with is state-of-the-art technology with 70 percent of the board built with VLSI technology, 25 percent of LSI technology and 5 percent of SSIC technology. The signal processing software is new code to be developed which requires 50 percent new design. The technology maturity for the software development is leading edge. This software is written in the C programming language and is estimated to be 2400 lines of code. The software is characterized as real time software. Each system component must be characterized to this level of detail to generate a cost estimate. The complete set of attributes for each component in the first candidate architecture is given in the appendix.

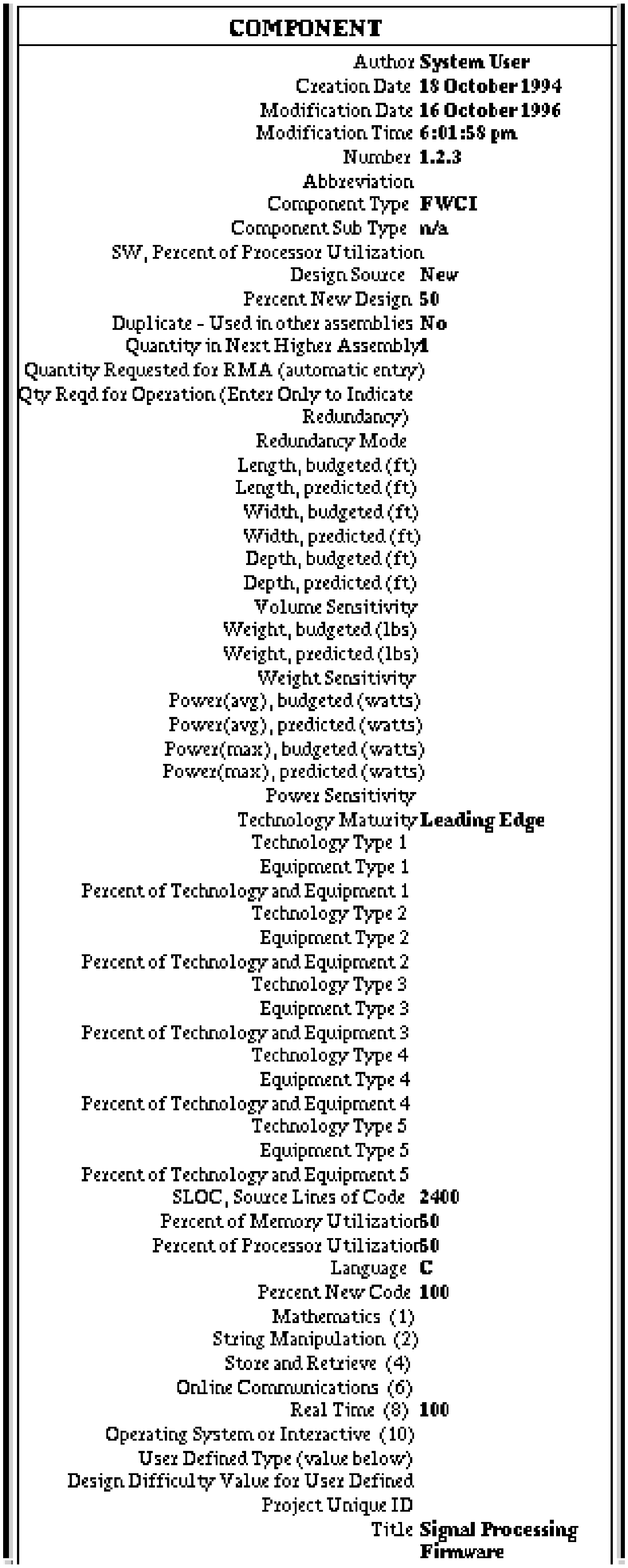
Attributes characterizing the system's life cycle must also be populated in the RDD-100 database. The life cycle attributes for this example are shown in Figure 4 **-** 10. The operational environment, deployment quantity, mission period and duration of the life cycle are shown in this figure. The sensitivity factors included at the bottom of Figure 4 **-** 10 are used in the RAM-ILS tool to optimize the physical configuration of the system to meet reliability requirements. The production and support costs have been emphasized in this example.

A consistency report is run in RDD-100 which identifies whether a sufficient set of attributes have been populated to get a valid cost estimate. Once the database is populated with the required parameters for costing purposes, an export report is run in RDD-100 that generates a file containing the system engineering parameters in the correct format for cost estimating.



COMPONENT

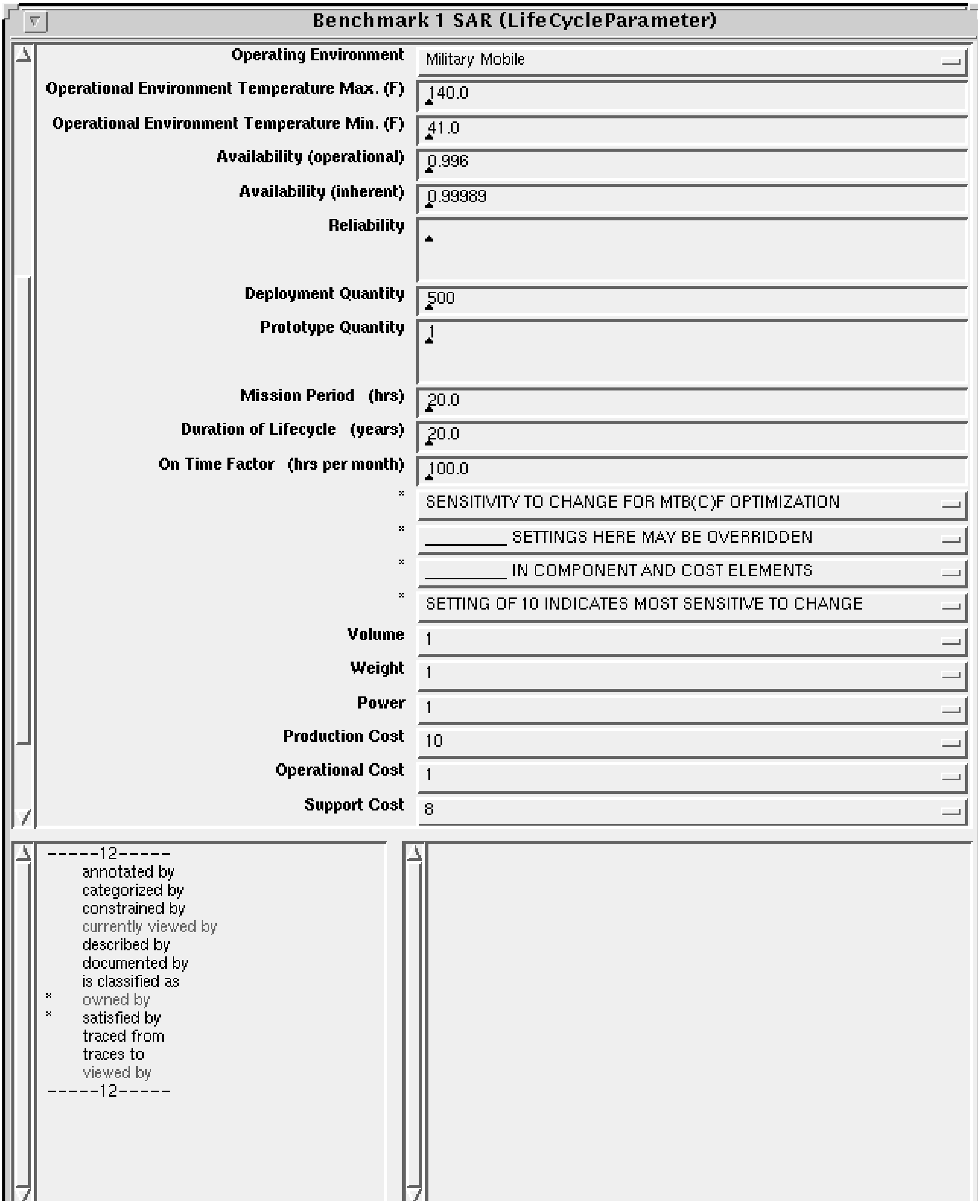
Author	**System User**
Creation Date	**1 July 1995**
Modification Date	**25 July 1997**
Modification Time	**9:37:42 am**
Number	**1.1.1**
Abbreviation	
Component Type	**HW Element**
Component Sub Type	**Board**
SW, Percent of Processor Utilization	
Design Source	**New**
Percent New Design	**75**
Duplicate - Used in other assemblies	**No**
Quantity in Next Higher Assembly	**1**
Quantity Requested for RMA (automatic entry)	
Qty Reqd for Operation (Enter Only to Indicate Redundancy)	
Redundancy Mode	
Length, budgeted (ft)	**0.525**
Length, predicted (ft)	
Width, budgeted (ft)	**0.0625**
Width, predicted (ft)	
Depth, budgeted (ft)	**0.767**
Depth, predicted (ft)	
Volume Sensitivity	
Weight, budgeted (lbs)	**1.5**
Weight, predicted (lbs)	
Weight Sensitivity	
Power(avg), budgeted (watts)	**15.0**
Power(avg), predicted (watts)	
Power(max), budgeted (watts)	
Power(max), predicted (watts)	
Power Sensitivity	
Technology Maturity	**State of the Art**
Technology Type 1	**VLSI**
Equipment Type 1	**Digital**
Percent of Technology and Equipment 1	**70**
Technology Type 2	**LSI**
Equipment Type 2	**Digital**
Percent of Technology and Equipment 2	**25**
Technology Type 3	**SSIC**
Equipment Type 3	**Digital**
Percent of Technology and Equipment 3	**5**
Technology Type 4	
Equipment Type 4	
Percent of Technology and Equipment 4	
Technology Type 5	
Equipment Type 5	
Percent of Technology and Equipment 5	
SLOC, Source Lines of Code	
Percent of Memory Utilization	
Percent of Processor Utilization	
Language	
Percent New Code	
Mathematics (1)	
String Manipulation (2)	
Store and Retrieve (4)	
Online Communications (6)	
Real Time (8)	
Operating System or Interactive (10)	
User Defined Type (value below)	
Design Difficulty Value for User Defined	
Project Unique ID	
Title	**Data I/O Module**



COMPONENT

Author	**System User**
Creation Date	**18 October 1994**
Modification Date	**16 October 1996**
Modification Time	**6:01:58 pm**
Number	**1.2.3**
Abbreviation	
Component Type	**FWCI**
Component Sub Type	**n/a**
SW, Percent of Processor Utilization	
Design Source	**New**
Percent New Design	**50**
Duplicate - Used in other assemblies	**No**
Quantity in Next Higher Assembly	**1**
Quantity Requested for RMA (automatic entry)	
Qty Reqd for Operation (Enter Only to Indicate Redundancy)	
Redundancy Mode	
Length, budgeted (ft)	
Length, predicted (ft)	
Width, budgeted (ft)	
Width, predicted (ft)	
Depth, budgeted (ft)	
Depth, predicted (ft)	
Volume Sensitivity	
Weight, budgeted (lbs)	
Weight, predicted (lbs)	
Weight Sensitivity	
Power(avg), budgeted (watts)	
Power(avg), predicted (watts)	
Power(max), budgeted (watts)	
Power(max), predicted (watts)	
Power Sensitivity	
Technology Maturity	**Leading Edge**
Technology Type 1	
Equipment Type 1	
Percent of Technology and Equipment 1	
Technology Type 2	
Equipment Type 2	
Percent of Technology and Equipment 2	
Technology Type 3	
Equipment Type 3	
Percent of Technology and Equipment 3	
Technology Type 4	
Equipment Type 4	
Percent of Technology and Equipment 4	
Technology Type 5	
Equipment Type 5	
Percent of Technology and Equipment 5	
SLOC, Source Lines of Code	**2400**
Percent of Memory Utilization	**50**
Percent of Processor Utilization	**50**
Language	**C**
Percent New Code	**100**
Mathematics (1)	
String Manipulation (2)	
Store and Retrieve (4)	
Online Communications (6)	
Real Time (8)	**100**
Operating System or Interactive (10)	
User Defined Type (value below)	
Design Difficulty Value for User Defined	
Project Unique ID	
Title	**Signal Processing Firmware**

Figure 4 - 9: RDD Links Between Functions and Components



Benchmark 1 SAR (LifeCycleParameter)

Operating Environment	Military Mobile
Operational Environment Temperature Max. (F)	140.0
Operational Environment Temperature Min. (F)	41.0
Availability (operational)	0.996
Availability (inherent)	0.99989
Reliability	
Deployment Quantity	500
Prototype Quantity	1
Mission Period (hrs)	20.0
Duration of Lifecycle (years)	20.0
On Time Factor (hrs per month)	100.0
*	SENSITIVITY TO CHANGE FOR MTB(C)F OPTIMIZATION
*	_________ SETTINGS HERE MAY BE OVERRIDDEN
*	_________ IN COMPONENT AND COST ELEMENTS
*	SETTING OF 10 INDICATES MOST SENSITIVE TO CHANGE
Volume	1
Weight	1
Power	1
Production Cost	10
Operational Cost	1
Support Cost	8

-----12-----
annotated by
categorized by
constrained by
currently viewed by
described by
documented by
is classified as
* owned by
* satisfied by
traced from
traces to
viewed by
-----12-----

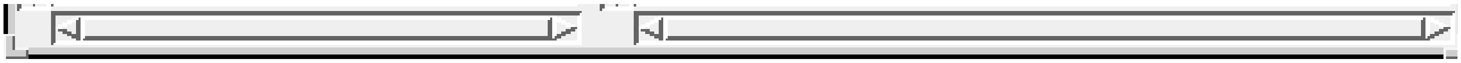
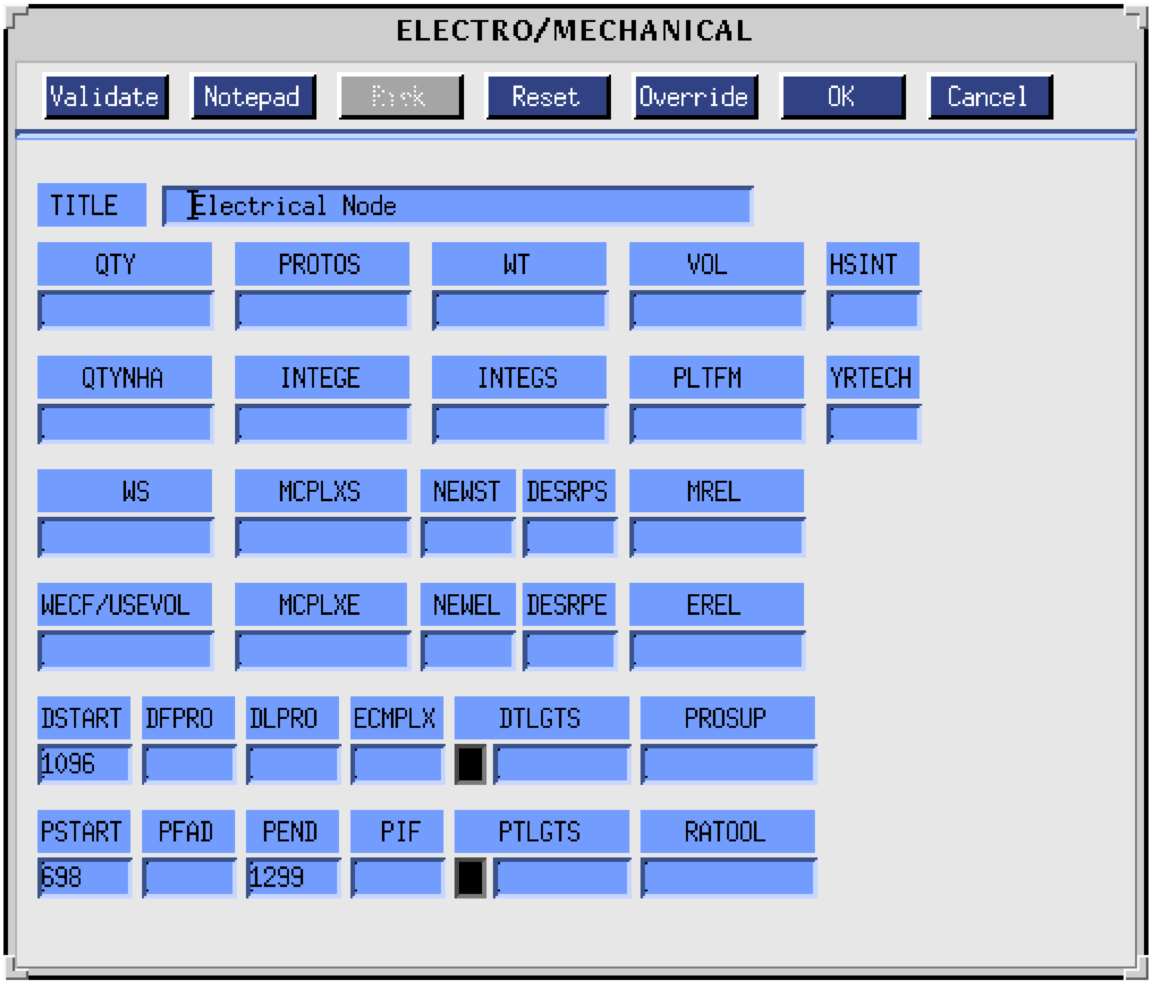


Figure 4 - 10: Life Cycle Parameters

Cost Analyst File - The cost analyst file contains default PRICE parameters for each component in the system which are missing from translation of the RDD-100 file. The parameters typically defined within the cost analyst file are prototype and production schedule, labor rates, escalation rates and other financial factors that the PRICE tool needs. The information is entered within the cost analyst file on an element type basis. The cost analyst is typically responsible for generating this file. The basic labor rates, escalation rates and financial factors included in the PRICE tool are used in this example. The only information specific for this example that is included within the cost analyst file is schedule information as shown in Figure 4 - 11. This figure shows the default parameters for the electro-mechanical element type. The only information included for this element type is a 10/96 development start date, a 6/98 production start date and a 12/99 production end date.



TITLE Electrical Node

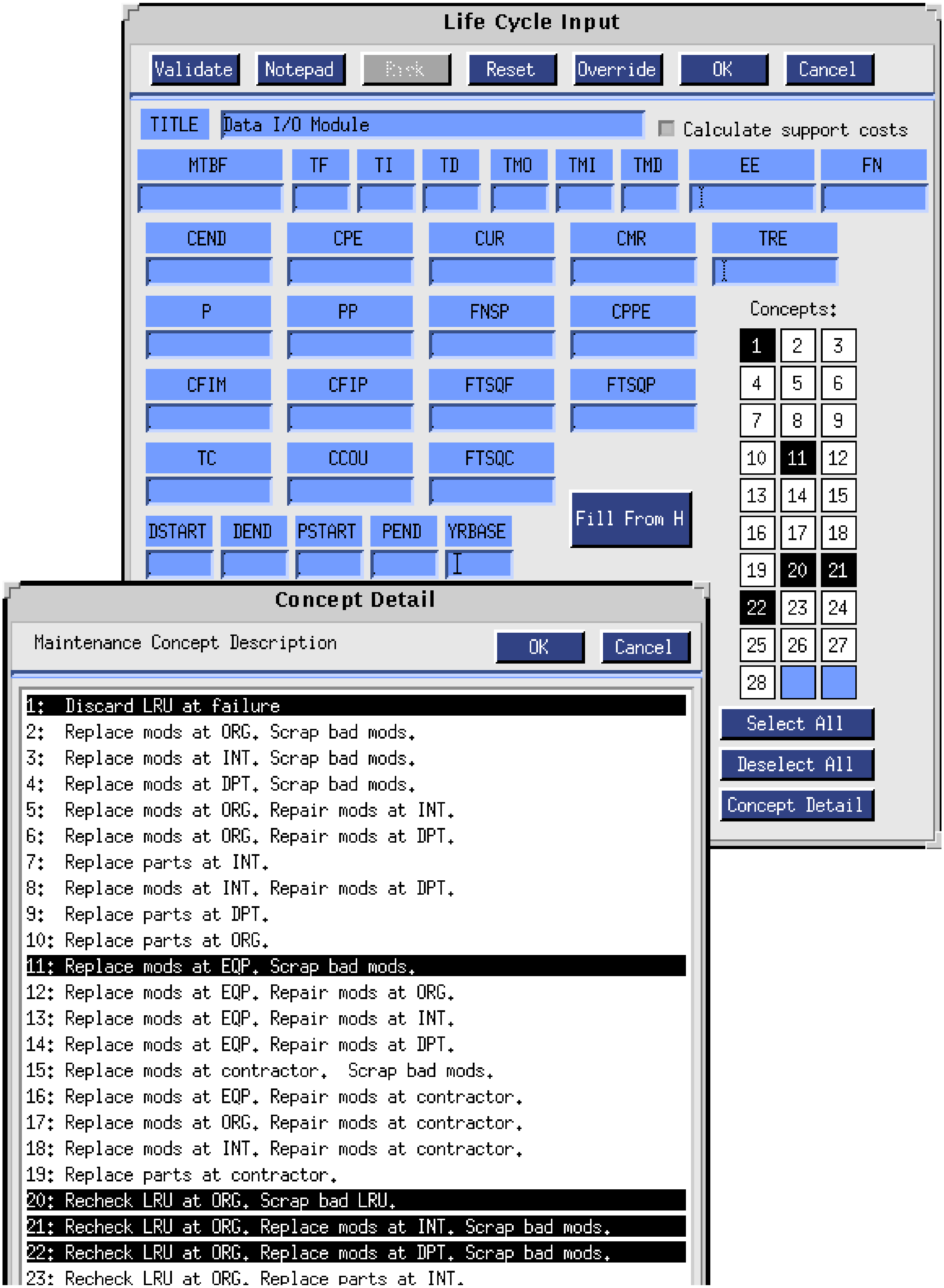
DSTART 1096

PSTART 698

PEND 1299

Figure 4 - 11: Default Parameters for Electro-Mechanical Element Type in Cost Analyst File

Synchronization File - The synchronization file contains PRICE parameters for each component which supersede any parameter set from either the translation of the RDD-100 file or cost analyst file. It is through the use of the synchronization file that the cost analyst can control the cost estimate since any parameter within this file will take precedence over values set from any other source. The cost analyst is typically responsible for generating this file. Information is entered in the synchronization file on a component name basis which enables each component to have its own parameters within the synchronization file. The PRICE life cycle tool has various maintenance concepts that the tool can choose from in determining the most cost effective concept . The maintenance concept is restricted to a subset of possible concepts in this example to illustrate the use of the synchronization file. As shown in Figure 4 - 12, the maintenance concept for the Data I/O Module is limited to concepts 1, 11, 20, 21 and 22 (the valid maintenance concepts are indicated by the black squares in this diagram). The definition for each of these concepts is included in the figure. The PRICE toolset will select the most cost effective concept among these five potential maintenance concepts for this module regardless of any data within either the RDD-100 file or cost analyst file.



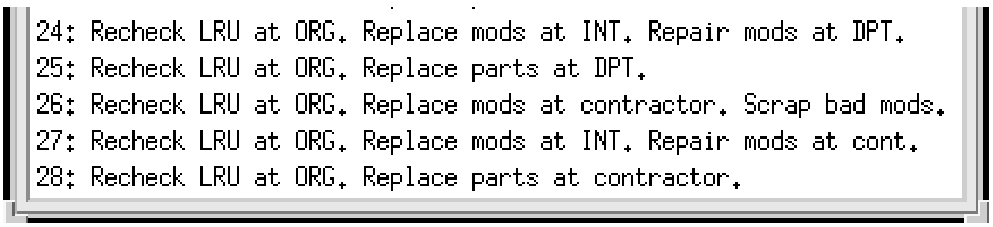
TITLE: Data I/O Module

□ Calculate support costs

Concepts: 1, 11, 20, 21 and 22 selected (of 1–28)

Maintenance Concept Description

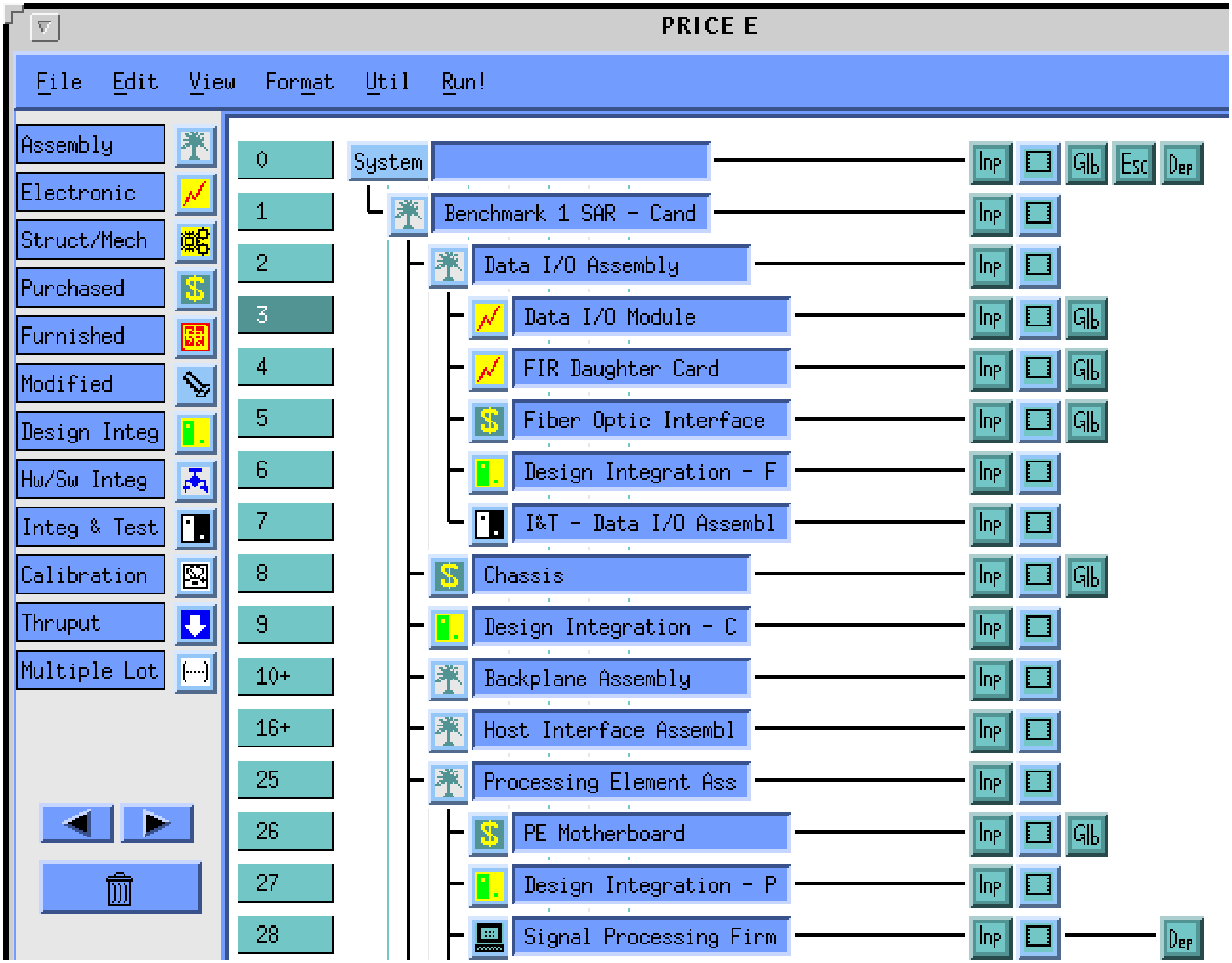
1: Discard LRU at failure
2: Replace mods at ORG. Scrap bad mods.
3: Replace mods at INT. Scrap bad mods.
4: Replace mods at DPT. Scrap bad mods.
5: Replace mods at ORG. Repair mods at INT.
6: Replace mods at ORG. Repair mods at DPT.
7: Replace parts at INT.
8: Replace mods at INT. Repair mods at DPT.
9: Replace parts at DPT.
10: Replace parts at ORG.
11: Replace mods at EQP. Scrap bad mods.
12: Replace mods at EQP. Repair mods at ORG.
13: Replace mods at EQP. Repair mods at INT.
14: Replace mods at EQP. Repair mods at DPT.
15: Replace mods at contractor. Scrap bad mods.
16: Replace mods at EQP. Repair mods at contractor.
17: Replace mods at ORG. Repair mods at contractor.
18: Replace mods at INT. Repair mods at contractor.
19: Replace parts at contractor.
20: Recheck LRU at ORG. Scrap bad LRU.
21: Recheck LRU at ORG. Replace mods at INT. Scrap bad mods.
22: Recheck LRU at ORG. Replace mods at DPT. Scrap bad mods.
23: Recheck LRU at ORG. Replace parts at INT.



```
24: Recheck LRU at ORG. Replace mods at INT. Repair mods at DPT.
25: Recheck LRU at ORG. Replace parts at DPT.
26: Recheck LRU at ORG. Replace mods at contractor. Scrap bad mods.
27: Recheck LRU at ORG. Replace mods at INT. Repair mods at cont.
28: Recheck LRU at ORG. Replace parts at contractor.
```

Figure 4 - 12: Maintenance Concept Selection for Data I/O Module in Synchronization File

Cost Estimate - The PRL import file, RDD-100 output file, cost analyst and synchronization files are used to populate the PRICE tool with all the parametric cost estimating attributes needed to perform a cost estimate. The process to translate the system engineering parameters from the RDD-100 file to PRICE attributes and to apply both the cost analyst and synchronization file takes several minutes for this example. The equipment breakdown structure for the first candidate architecture after the data has been imported into the PRICE toolset is shown in Figure 4 - 13. This equipment breakdown structure is identical to the physical equipment tree established in RDD-100. The tree icons in this figure represent assemblies which contain lower level components. Note that the backplane and host interface assemblies have been collapsed in this figure to reduce its overall size. The lightning bolt icon represent custom digital boards, while the dollar icons are COTS items. The workstation icon represents a software component. Note that elements of both hardware and software are contained within this single equipment tree. Design integration, hardware/software integration, and integration and test elements are added in the appropriate places to the PRICE equipment tree during the translation process even though these elements are not included within the RDD-100 database. The costs for these integration elements are included as a part of the assembly level costs when exporting the cost data back to RDD-100.



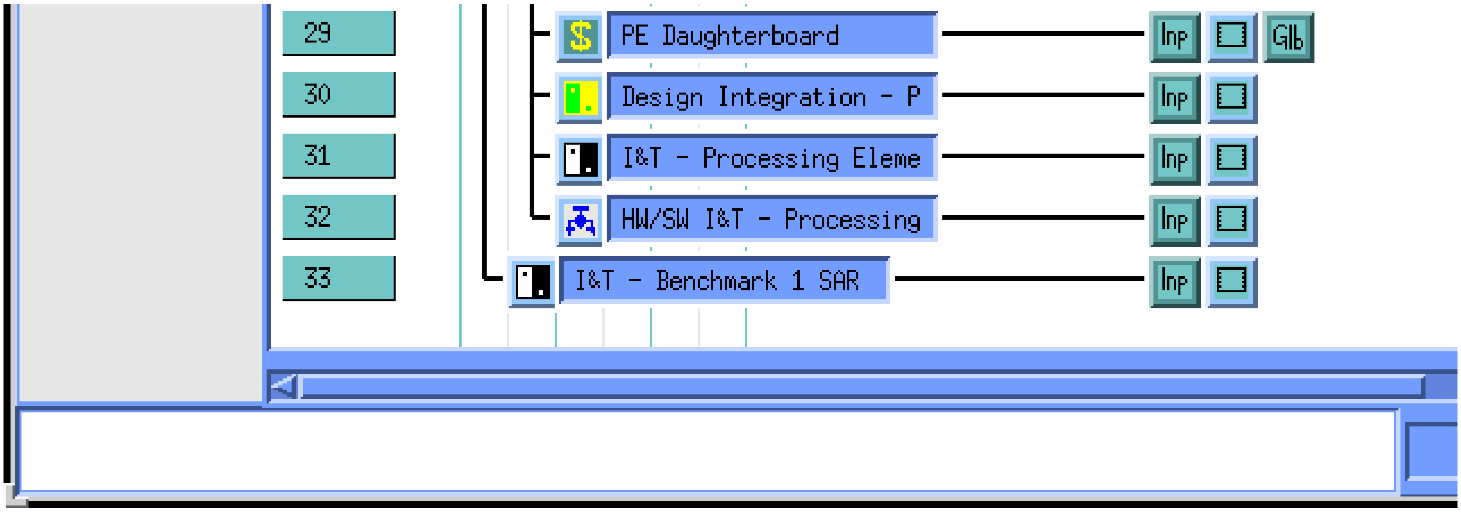
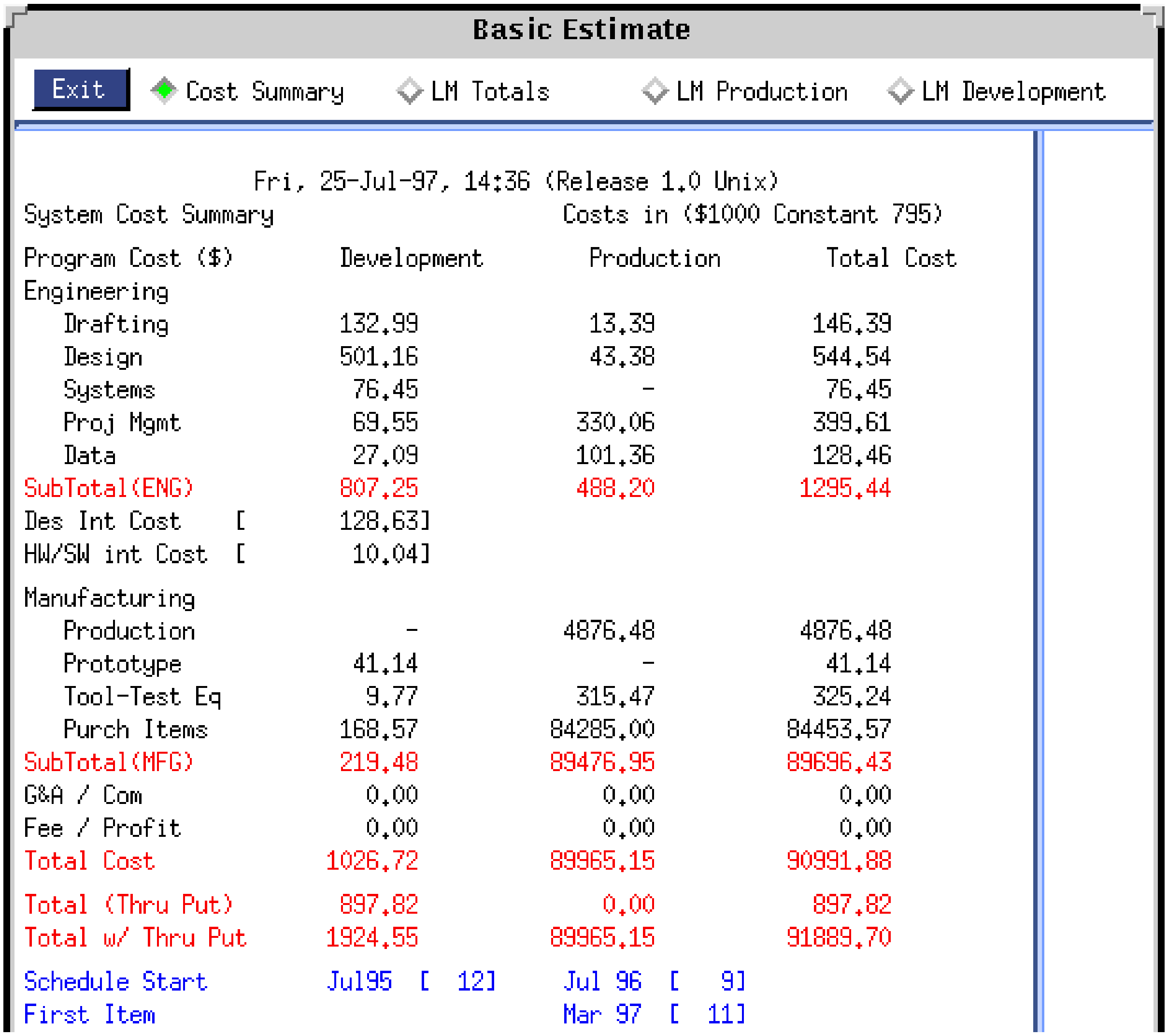
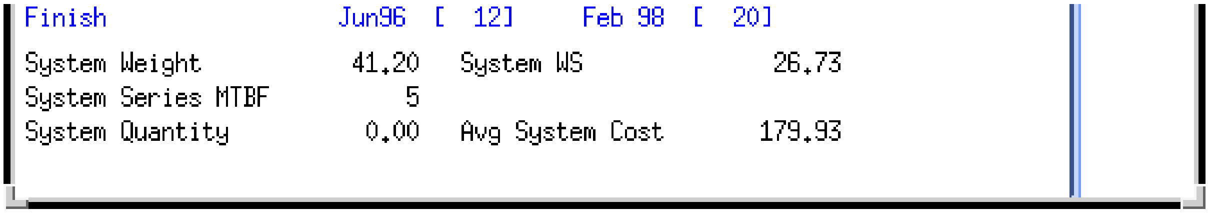


Figure 4 - 13: Equipment Breakdown Structure for the First Candidate Architecture

The development and production cost estimates from the PRICE tool for the first candidate architecture are shown in Figure 4 - 14. The first column in this figure shows the development costs and the second column depicts the production costs. The engineering costs are in the top half of the cost summary, while the manufacturing costs are in the bottom half. The development costs for the first candidate architecture is $1.9M and the production costs for 500 systems is $90M. The average system cost is $179.9K.



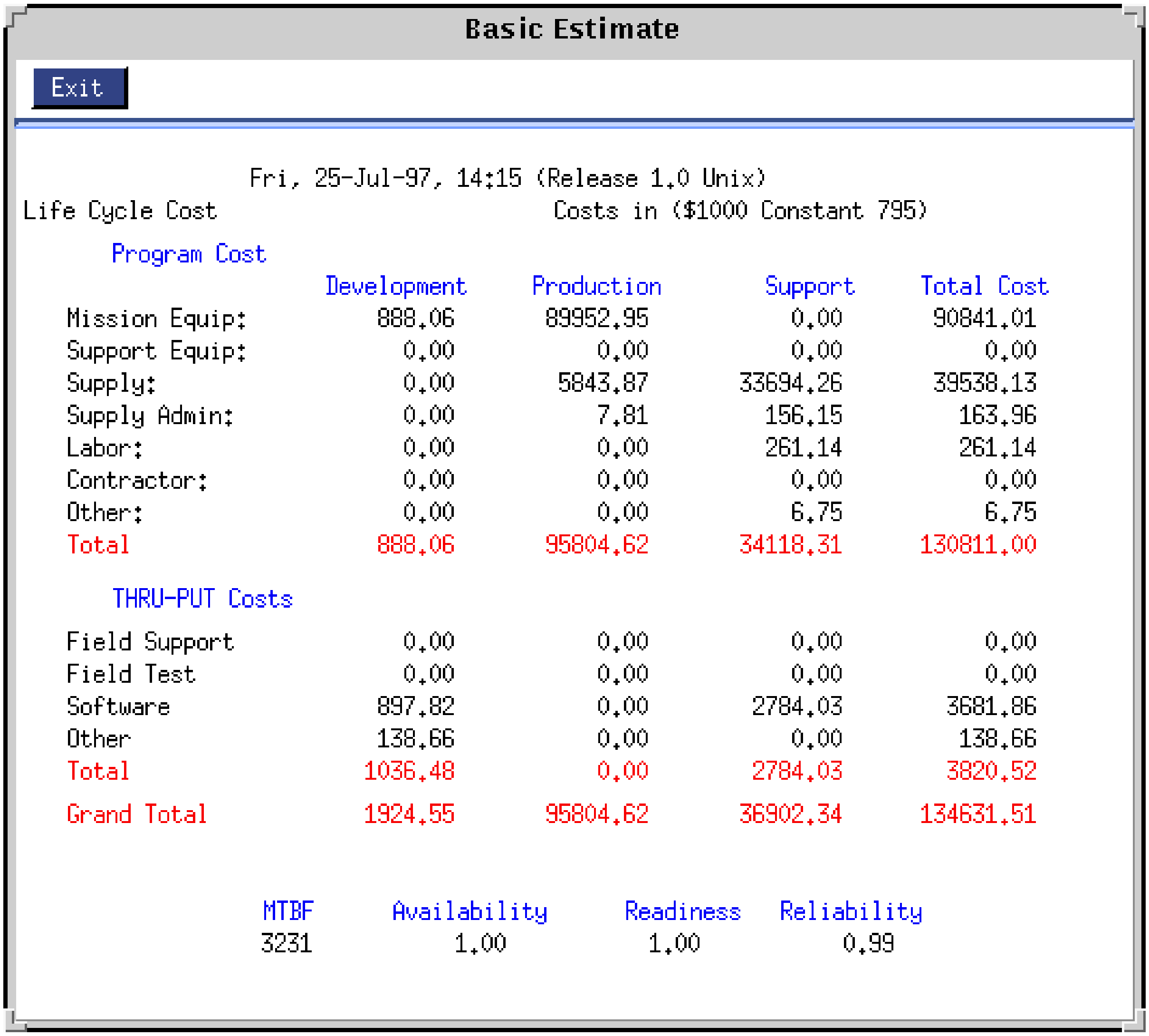
```
                    Fri, 25-Jul-97, 14:36 (Release 1.0 Unix)
System Cost Summary                     Costs in ($1000 Constant 795)

Program Cost ($)        Development         Production         Total Cost
Engineering
   Drafting             132.99              13.39              146.39
   Design               501.16              43.38              544.54
   Systems               76.45                  -               76.45
   Proj Mgmt             69.55             330.06              399.61
   Data                  27.09             101.36              128.46
SubTotal(ENG)           807.25             488.20             1295.44
Des Int Cost    [       128.63]
HW/SW int Cost  [        10.04]

Manufacturing
   Production               -             4876.48            4876.48
   Prototype             41.14                  -              41.14
   Tool-Test Eq           9.77             315.47             325.24
   Purch Items          168.57           84285.00           84453.57
SubTotal(MFG)           219.48           89476.95           89696.43
G&A / Com                 0.00               0.00               0.00
Fee / Profit              0.00               0.00               0.00
Total Cost             1026.72           89965.15           90991.88

Total (Thru Put)        897.82               0.00             897.82
Total w/ Thru Put      1924.55           89965.15           91889.70

Schedule Start         Jul95  [   12]    Jul 96  [    9]
First Item                               Mar 97  [   11]
```



```
Finish                  Jun96  [  12]     Feb 98  [  20]

System Weight            41.20   System WS           26.73
System Series MTBF           5
System Quantity           0.00   Avg System Cost    179.93
```

Figure 4 - 14: Development and Production Costs for the First Candidate Architecture

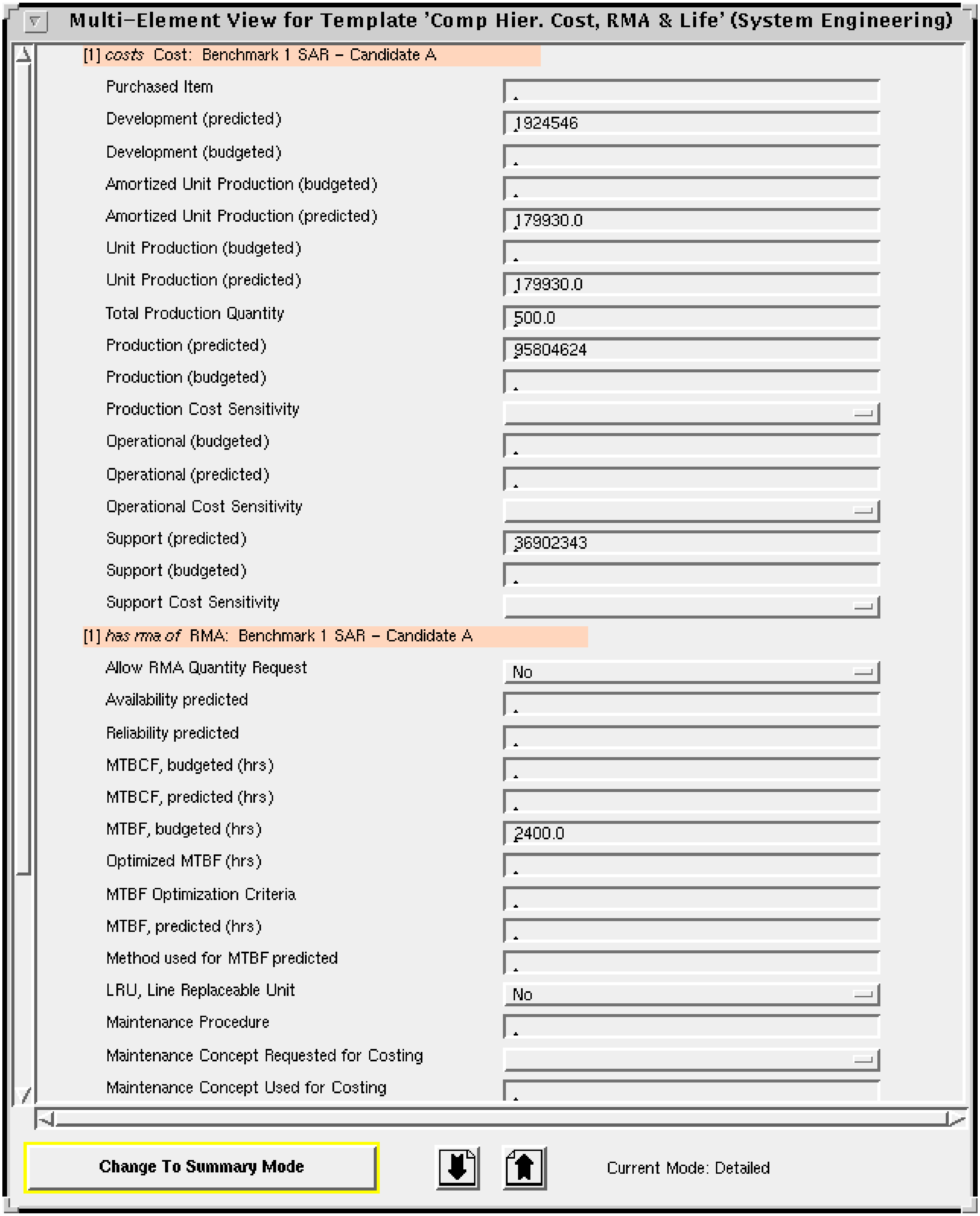
The life cycle cost estimates from the PRICE tool for the first candidate architecture are shown in Figure 4 - 15. The first column in this figure shows the development costs, the second column gives the production costs for the 500 systems and initial spares, and the third column shows the support costs over the twenty year life cycle. The total life cycle costs for this architecture is $134.6M.



```
                 Fri, 25-Jul-97, 14:15 (Release 1.0 Unix)
Life Cycle Cost                          Costs in ($1000 Constant 795)
       Program Cost
                      Development    Production      Support    Total Cost
   Mission Equip:          888.06      89952.95         0.00      90841.01
   Support Equip:            0.00          0.00         0.00          0.00
   Supply:                   0.00       5843.87     33694.26      39538.13
   Supply Admin:             0.00          7.81       156.15        163.96
   Labor:                    0.00          0.00       261.14        261.14
   Contractor:               0.00          0.00         0.00          0.00
   Other:                    0.00          0.00         6.75          6.75
   Total                   888.06      95804.62     34118.31     130811.00

       THRU-PUT Costs

   Field Support             0.00          0.00         0.00          0.00
   Field Test                0.00          0.00         0.00          0.00
   Software                897.82          0.00      2784.03       3681.86
   Other                   138.66          0.00         0.00        138.66
   Total                  1036.48          0.00      2784.03       3820.52

   Grand Total            1924.55      95804.62     36902.34     134631.51


                   MTBF      Availability     Readiness    Reliability
                   3231          1.00            1.00          0.99
```

Figure 4 - 15: Support Costs for the First Candidate Architecture

Export of Costs to RDD-100 - The development, production and support cost estimates calculated by the PRICE toolset are back annotated in the RDD-100 database. This back annotation is performed by exporting the cost data out of the PRICE tool into a file with the standard rdt format which RDD-100 uses. This file is generated by executing a PRL export script within the PRICE toolset. This file is then imported into RDD-100 using the standard import facility. An RDD-100 view of the cost and reliability data for the SAR system for the first candidate architecture is shown in Figure 4 - 16. The costs in this figure were calculated in the PRICE tool. The reliability data in this figure is empty, as this analysis will be performed in the next section.



Multi-Element View for Template 'Comp Hier. Cost, RMA & Life' (System Engineering)

[1] *costs* Cost: Benchmark 1 SAR – Candidate A

Purchased Item	
Development (predicted)	1924546
Development (budgeted)	
Amortized Unit Production (budgeted)	
Amortized Unit Production (predicted)	179930.0
Unit Production (budgeted)	
Unit Production (predicted)	179930.0
Total Production Quantity	500.0
Production (predicted)	95804624
Production (budgeted)	
Production Cost Sensitivity	
Operational (budgeted)	
Operational (predicted)	
Operational Cost Sensitivity	
Support (predicted)	36902343
Support (budgeted)	
Support Cost Sensitivity	

[1] *has rma of* RMA: Benchmark 1 SAR – Candidate A

Allow RMA Quantity Request	No
Availability predicted	
Reliability predicted	
MTBCF, budgeted (hrs)	
MTBCF, predicted (hrs)	
MTBF, budgeted (hrs)	2400.0
Optimized MTBF (hrs)	
MTBF Optimization Criteria	
MTBF, predicted (hrs)	
Method used for MTBF predicted	
LRU, Line Replaceable Unit	No
Maintenance Procedure	
Maintenance Concept Requested for Costing	
Maintenance Concept Used for Costing	

Current Mode: Detailed

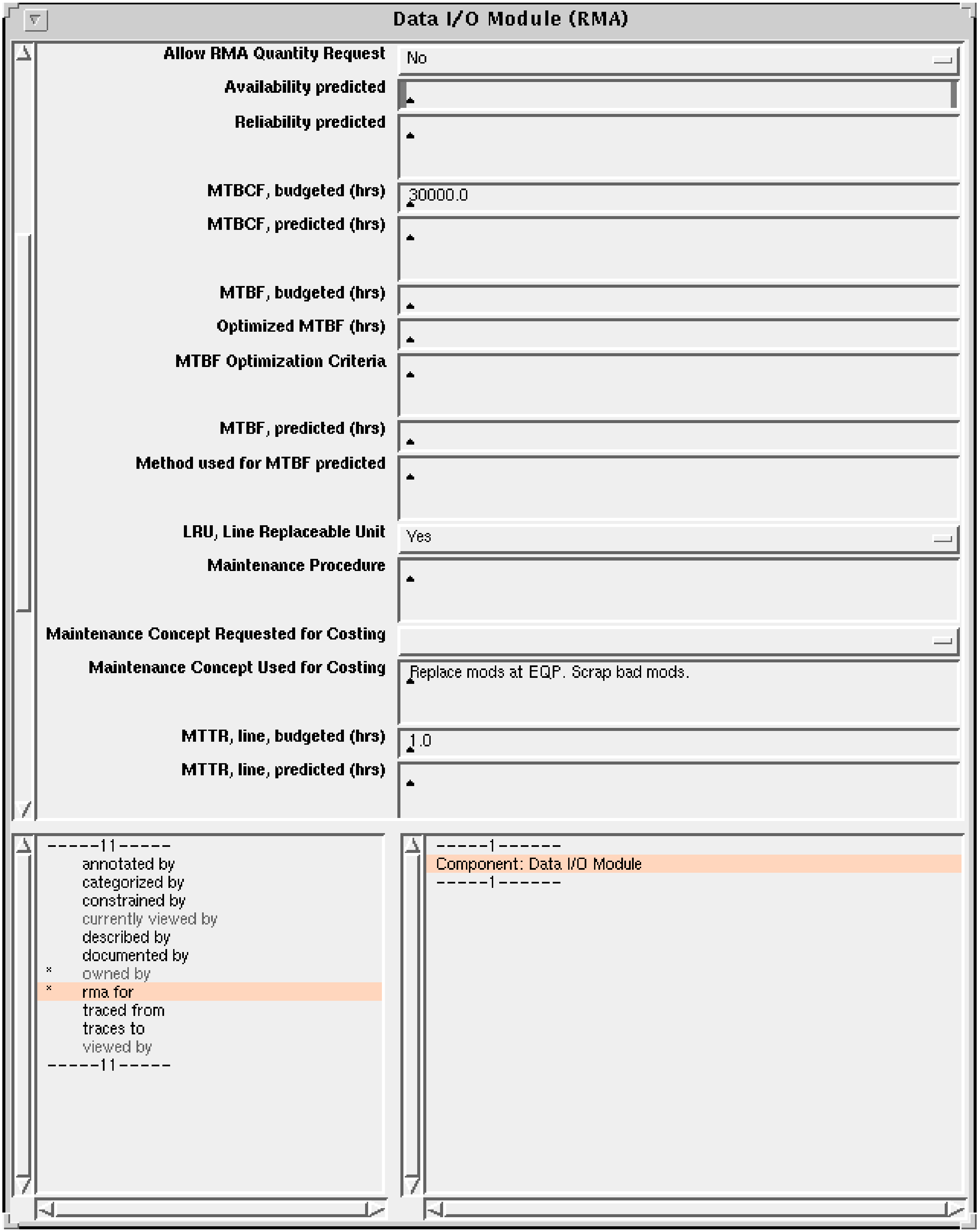
Figure 4 - 16: Cost and Reliability Data for SAR System

4.4.3 Reliability Assessment

A preliminary architecture has been defined and an initial cost estimate calculated for the first candidate system. A reliability assessment of this system is made next to ensure that all requirements are met. The RDD-100 schema has been extended on the RASSP program to support this reliability assessment. The only external data that the RAM-ILS toolset needs to perform a reliability assessment is contained within the file generated within RDD-100. This file contains the complete system architecture, physical attributes of the system components, allocated reliability budgets, parameters characterizing the operational environment, sensitivity parameters and cost data. Each of these elements is described in more detail below for the SAR example.

Attributes which characterize the system architecture and the hardware components must be populated in the RDD-100 database prior to exporting the file needed for reliability assessment. Most of these attributes are the same attributes needed to generate the cost estimate for the PRICE tool. The attributes used to characterize the data i/o module have been previously shown in Figure 4 - 9. Redundancy parameters are included in this set of parameters which are used within the RAM-ILS tool for the reliability assessment. For this example, there is no redundancy included in the initial system architecture. Tradeoffs are performed within the RAM-ILS tool in this example to determine how the system architecture can be changed in a cost effective way to meet system reliability requirements.

A reliability assessment can be performed using the RAM-ILS based upon previous designs, similar to designs or from the allocated failure rate budget. The reliability assessment in this example is based upon the allocated failure rate budget as the focus of this example is showing how the integrated tools can be used early in the design process when detailed design data is not available. The systems engineer allocates the system level mean time between critical failures (MTBCF) to the system components within the RDD-100 tool. The system engineer performs this allocation based upon available component data, interactions with reliability engineers and his judgment based on past experience. Parameters specific to reliability and maintainability are entered in the RMA entity type for each component. The RMA entity for the data i/o module is shown in Figure 4-17. A MTBCF of 30,000 hours has been allocated to the data i/o module. The system engineer also indicates in this entity type whether the component can be considered for redundancy when performing a system architecture optimization within the RAM-ILS tool. Redundancy is not allowed for the data i/o module in this example (attribute name is Allow RMA Quantity Request within the RDD-100 schema). Note that the maintenance concept which the PRICE toolset selected for its life cycle support optimization has been back annotated in the RDD-100 database. In addition, a mean time to repair (MTTR) and an indication whether this component is a line replaceable unit have been indicated in Figure 4 - 17. The complete set of attributes for each component in this example is given in the Appendix.



Data I/O Module (RMA)

Allow RMA Quantity Request	No
Availability predicted	
Reliability predicted	
MTBCF, budgeted (hrs)	30000.0
MTBCF, predicted (hrs)	
MTBF, budgeted (hrs)	
Optimized MTBF (hrs)	
MTBF Optimization Criteria	
MTBF, predicted (hrs)	
Method used for MTBF predicted	
LRU, Line Replaceable Unit	Yes
Maintenance Procedure	
Maintenance Concept Requested for Costing	
Maintenance Concept Used for Costing	Replace mods at EQP. Scrap bad mods.
MTTR, line, budgeted (hrs)	1.0
MTTR, line, predicted (hrs)	

```
-----11-----
      annotated by
      categorized by
      constrained by
      currently viewed by
      described by
      documented by
×     owned by
×     rma for
      traced from
      traces to
      viewed by
-----11-----
```

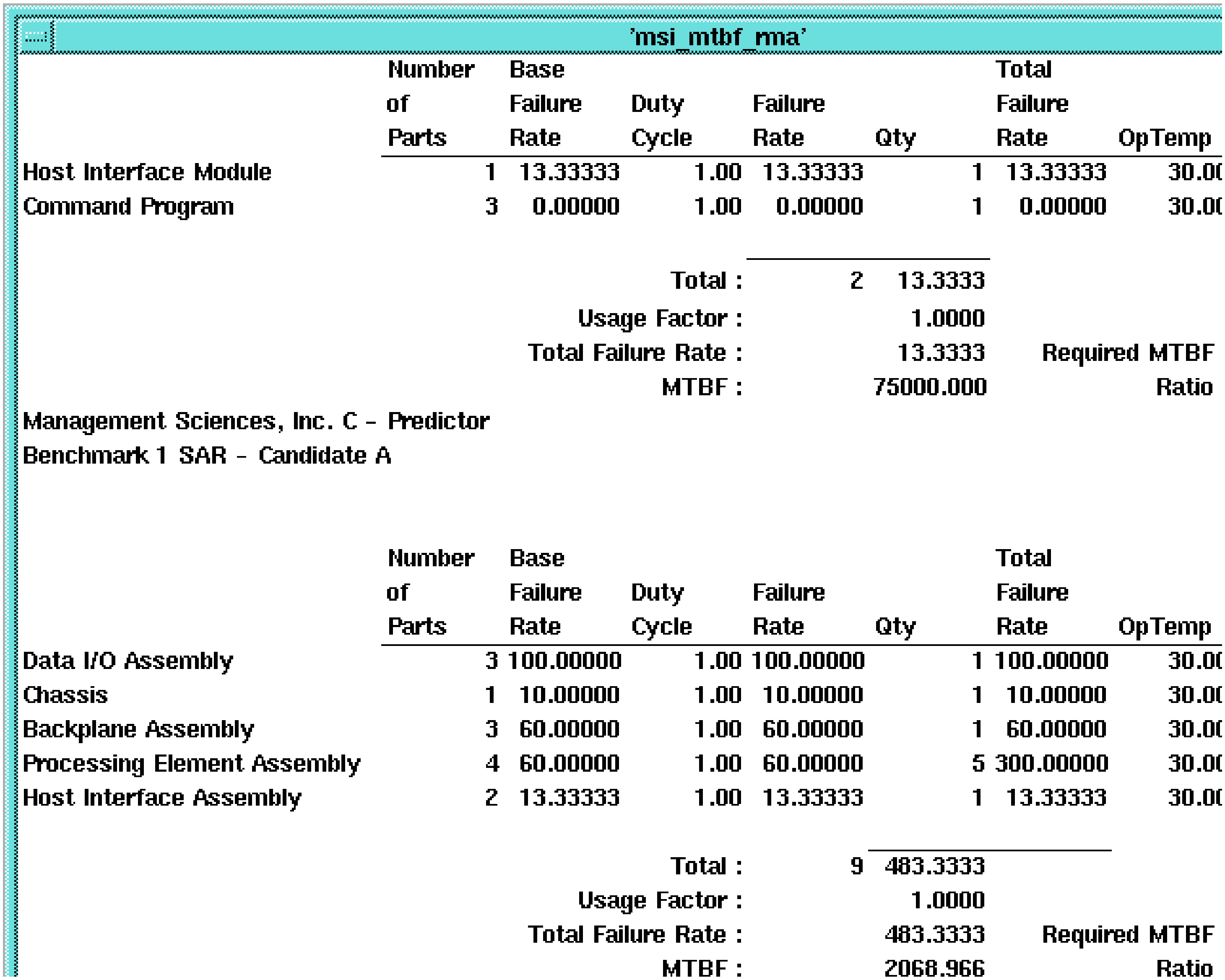
```
-----1------
Component: Data I/O Module
-----1------
```

Figure 4 - 17: RMA Entity for the Data I/O Module

The operational environment for the system must be defined prior to performing a reliability assessment. This environment is specified within the Life Cycle Parameter entity type in RDD-100 and the life cycle attributes for this example were previously shown in Figure 4 - 10 as many of these parameters are needed in the PRICE tool to calculate support costs.

Optimizations are performed within the RAM-ILS toolset to determine the most effective architecture which satisfies the system level reliability requirements. The system configuration can be optimized relative to size, weight, power, cost or a combination of these factors. The user must indicate the importance of these metrics on a relative basis either at the component or system level. The sensitivity factors are defined at the system level and emphasize production and support costs for this example as shown in Figure 4 - 10.

A consistency report is run in RDD-100 which identifies whether a sufficient set of attributes have been populated to get a valid reliability assessment. Once the database is populated with sufficient parameters, an export report is run in RDD-100 that generates a file containing the system engineering parameters in the correct format for reliability assessment. This file is then imported into the RAM-ILS toolset which establishes the system architecture and the pertinent parameters needed to perform a reliability estimate. A reliability assessment based upon the equipment configuration and allocated budgets is made within the predictor portion of the RAM-ILS toolset. An output report containing the reliability assessment at the system level from the RAM-ILS tool for this example is shown in Figure 4 - 18. The mean time to critical failure (MTBCF) for the first candidate architecture is calculated as 2068 hours. This system configuration did not meet the required 2400 hour system-level MTBCF. As a result, the system configuration must be changed to meet the reliability requirement.



'msi_mtbf_rma'

	Number of Parts	Base Failure Rate	Duty Cycle	Failure Rate	Qty	Total Failure Rate	OpTemp
Host Interface Module	1	13.33333	1.00	13.33333	1	13.33333	30.0
Command Program	3	0.00000	1.00	0.00000	1	0.00000	30.0

```
                 Total :        2   13.3333
          Usage Factor :             1.0000
     Total Failure Rate :           13.3333     Required MTBF
                  MTBF :         75000.000             Ratio
```

Management Sciences, Inc. C - Predictor
Benchmark 1 SAR - Candidate A

	Number of Parts	Base Failure Rate	Duty Cycle	Failure Rate	Qty	Total Failure Rate	OpTemp
Data I/O Assembly	3	100.00000	1.00	100.00000	1	100.00000	30.0
Chassis	1	10.00000	1.00	10.00000	1	10.00000	30.0
Backplane Assembly	3	60.00000	1.00	60.00000	1	60.00000	30.0
Processing Element Assembly	4	60.00000	1.00	60.00000	5	300.00000	30.0
Host Interface Assembly	2	13.33333	1.00	13.33333	1	13.33333	30.0

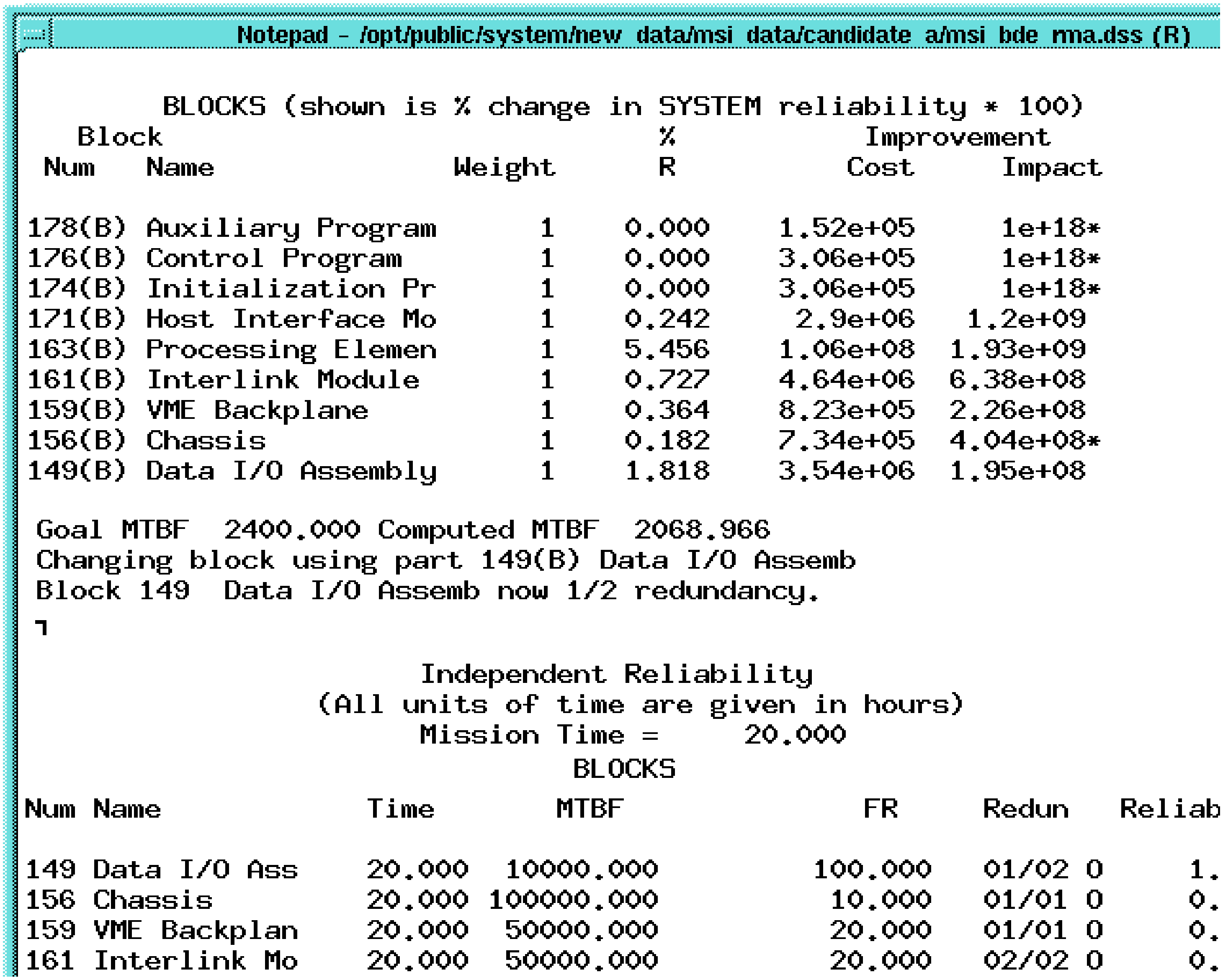
```
                 Total :        9  483.3333
          Usage Factor :             1.0000
     Total Failure Rate :          483.3333     Required MTBF
                  MTBF :          2068.966             Ratio
```

Management Sciences, Inc. C - Predictor

Equipment Breakdown

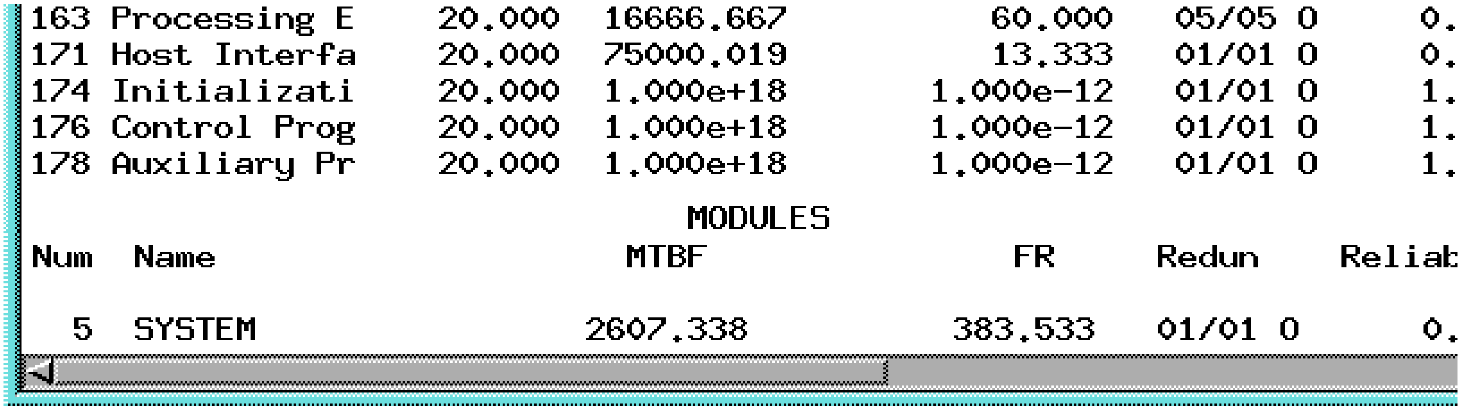
Figure 4 - 18: System Reliability Assessment

The block diagram evaluator (BDE) portion off the RAM-ILS toolset is used to optimize the physical architecture when requirements are not met. This optimization is performed by determining both the overall improvement in the system-level MTBF and the associated cost for adding redundancy for each hardware component in the system. The output of BDE for the first candidate architecture is shown in Figure 4 - 19. The first column in this output report shows the improvement in the system-level reliability when an additional redundant unit is added for a particular component. The second column shows the aggregate cost of this additional redundant unit which is calculated from the component's physical parameters and sensitivity factors. The third column in this report shows the impact of adding redundancy for this particular component which is calculated by dividing the cost of the redundancy by the overall improvement. The component with the smallest number in the third column is the most cost effective place to initially add redundancy to improve the system-level reliability. The data I/O assembly is the most cost effective component to add redundancy in this example. The RAM-ILS tool then determines the overall system-level MTBCF when the most cost effective redundant component is added to the system architecture. This procedure iterates until the system requirements are met. For this example, the MTBCF improves to 2607 hours and meets the overall requirements when a redundant data I/O assembly is added to the system architecture.



Notepad - /opt/public/system/new data/msi data/candidate a/msi bde rma.dss (R)

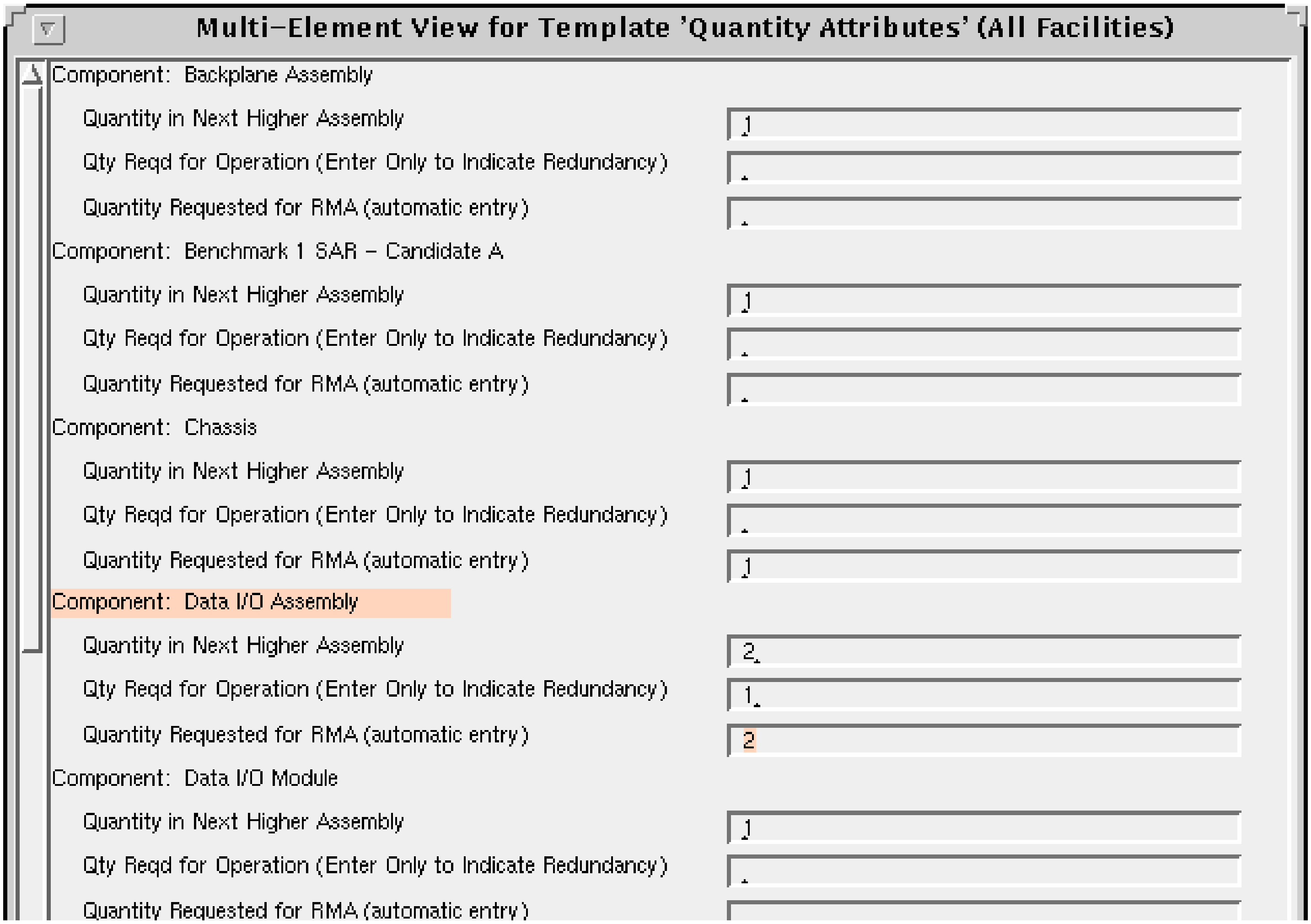
```
      BLOCKS (shown is % change in SYSTEM reliability * 100)
   Block                             %          Improvement
 Num   Name               Weight     R          Cost      Impact

178(B) Auxiliary Program       1     0.000    1.52e+05     1e+18*
176(B) Control Program         1     0.000    3.06e+05     1e+18*
174(B) Initialization Pr       1     0.000    3.06e+05     1e+18*
171(B) Host Interface Mo       1     0.242     2.9e+06   1.2e+09
163(B) Processing Elemen       1     5.456    1.06e+08  1.93e+09
161(B) Interlink Module        1     0.727    4.64e+06  6.38e+08
159(B) VME Backplane           1     0.364    8.23e+05  2.26e+08
156(B) Chassis                 1     0.182    7.34e+05  4.04e+08*
149(B) Data I/O Assembly       1     1.818    3.54e+06  1.95e+08

 Goal MTBF  2400.000 Computed MTBF  2068.966
 Changing block using part 149(B) Data I/O Assemb
 Block 149  Data I/O Assemb now 1/2 redundancy.
 ˥

                       Independent Reliability
                 (All units of time are given in hours)
                       Mission Time =     20.000
                                BLOCKS
Num Name              Time        MTBF               FR      Redun    Reliab

149 Data I/O Ass     20.000   10000.000         100.000   01/02 0      1.
156 Chassis          20.000  100000.000          10.000   01/01 0      0.
159 VME Backplan     20.000   50000.000          20.000   01/01 0      0.
161 Interlink Mo     20.000   50000.000          20.000   02/02 0      0.
```



```
163 Processing E    20.000  16666.667         60.000   05/05 0      0.
171 Host Interfa    20.000  75000.019         13.333   01/01 0      0.
174 Initializati    20.000  1.000e+18      1.000e-12   01/01 0      1.
176 Control Prog    20.000  1.000e+18      1.000e-12   01/01 0      1.
178 Auxiliary Pr    20.000  1.000e+18      1.000e-12   01/01 0      1.
                                MODULES
Num  Name                   MTBF                FR     Redun     Reliab

  5  SYSTEM               2607.338           383.533   01/01 0      0.
```

Figure 4 - 19: Redundancy Optimization for First Candidate Architecture

The results of the reliability assessment calculated by the RAM-ILS toolset are back annotated in the RDD-100 database. This back annotation is performed by exporting the reliability data out of the RAM-ILS tool into a file with the standard rdt format which RDD-100 uses. This file is generated automatically within the RAM-ILS toolset after the reliability calculations are made. This file is then imported into RDD-100 using the standard import facility. The reliability results that are back annotated into the RDD-100 database are for the baseline system configuration sent to the RAM-ILS tool. A critical issue is generated in the RDD-100 database when the allocated reliability budgets are not met. All optimizations results obtained using the RAM-ILS toolset are back annotated in the RDD-100 database as a suggestion for change, as shown in Figure 4 - 20 for this example. The system engineer must determine whether to accept the redundancy recommendation or make other changes to meet the requirements. For this example, redundancy at the data i/o assembly is acceptable and the system architecture must be changed by the systems engineer in the RDD-100 database to reflect this redundancy.



Multi-Element View for Template 'Quantity Attributes' (All Facilities)

Component: Backplane Assembly

Quantity in Next Higher Assembly: 1

Qty Reqd for Operation (Enter Only to Indicate Redundancy):

Quantity Requested for RMA (automatic entry):

Component: Benchmark 1 SAR - Candidate A

Quantity in Next Higher Assembly: 1

Qty Reqd for Operation (Enter Only to Indicate Redundancy):

Quantity Requested for RMA (automatic entry):

Component: Chassis

Quantity in Next Higher Assembly: 1

Qty Reqd for Operation (Enter Only to Indicate Redundancy):

Quantity Requested for RMA (automatic entry): 1

Component: Data I/O Assembly

Quantity in Next Higher Assembly: 2

Qty Reqd for Operation (Enter Only to Indicate Redundancy): 1

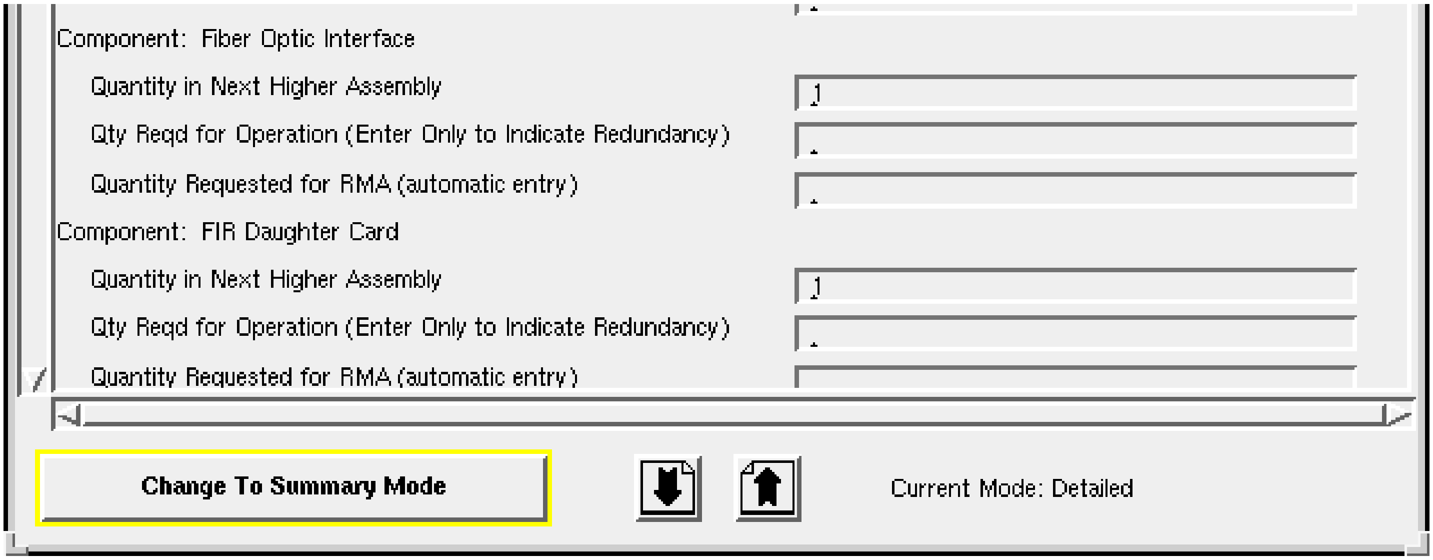
Quantity Requested for RMA (automatic entry): 2

Component: Data I/O Module

Quantity in Next Higher Assembly: 1

Qty Reqd for Operation (Enter Only to Indicate Redundancy):

Quantity Requested for RMA (automatic entry):



Component: Fiber Optic Interface

Quantity in Next Higher Assembly	1
Qty Reqd for Operation (Enter Only to Indicate Redundancy)	
Quantity Requested for RMA (automatic entry)	

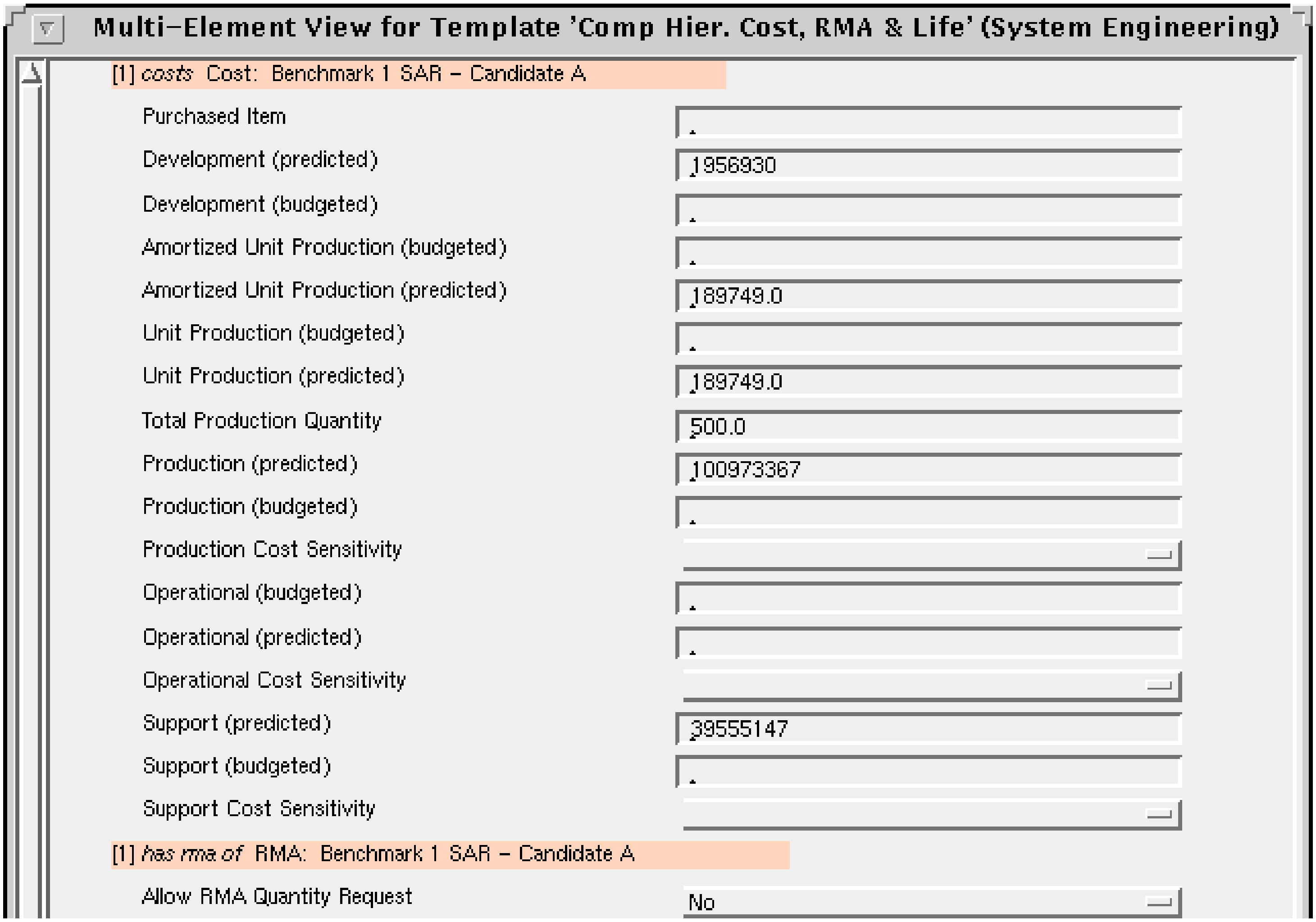
Component: FIR Daughter Card

Quantity in Next Higher Assembly	1
Qty Reqd for Operation (Enter Only to Indicate Redundancy)	
Quantity Requested for RMA (automatic entry)	

Current Mode: Detailed

Figure 4 - 20: Redundancy Recommendation Within RDD-100 Database

An updated reliability assessment and cost estimate is needed whenever the physical configuration of the system is changed. Thus, the processes used to generate both a cost estimate and reliability assessment are repeated when the data i/o assembly redundancy is added to the system architecture in this example. The resulting costs and reliability estimate for this system with redundancy are summarized in Figure 4 - 21. The development costs increased by $30K, the per unit production costs have increased by $10K and the support costs have increased by $2.6M when the redundant data i/o assembly is included in the system architecture for the first candidate.



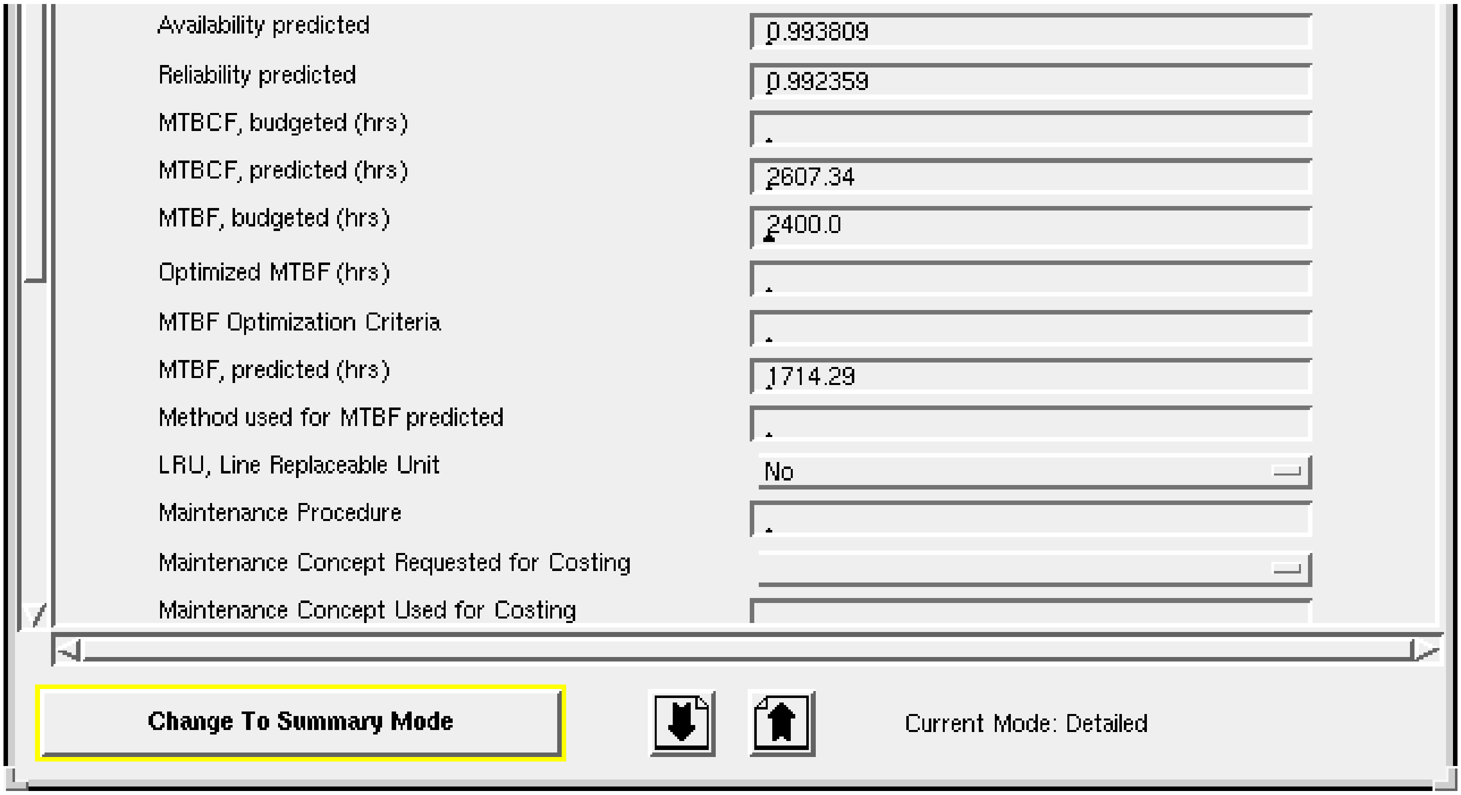
Multi-Element View for Template 'Comp Hier. Cost, RMA & Life' (System Engineering)

[1] *costs* Cost: Benchmark 1 SAR – Candidate A

Purchased Item	
Development (predicted)	1956930
Development (budgeted)	
Amortized Unit Production (budgeted)	
Amortized Unit Production (predicted)	189749.0
Unit Production (budgeted)	
Unit Production (predicted)	189749.0
Total Production Quantity	500.0
Production (predicted)	100973367
Production (budgeted)	
Production Cost Sensitivity	
Operational (budgeted)	
Operational (predicted)	
Operational Cost Sensitivity	
Support (predicted)	39555147
Support (budgeted)	
Support Cost Sensitivity	

[1] *has rma of* RMA: Benchmark 1 SAR – Candidate A

Allow RMA Quantity Request	No



Availability predicted	0.993809
Reliability predicted	0.992359
MTBCF, budgeted (hrs)	
MTBCF, predicted (hrs)	2607.34
MTBF, budgeted (hrs)	2400.0
Optimized MTBF (hrs)	
MTBF Optimization Criteria	
MTBF, predicted (hrs)	1714.29
Method used for MTBF predicted	
LRU, Line Replaceable Unit	No
Maintenance Procedure	
Maintenance Concept Requested for Costing	
Maintenance Concept Used for Costing	

Current Mode: Detailed

Figure 4 - 21: Cost Estimate and Reliability Assessment for First Candidate Architecture With Redundancy

4.4 System Cost And Reliability Assessment

In the previous section, a cost estimate and reliability assessment were made for the first candidate architecture. The same procedure is used to determine the cost and reliability for the second candidate architecture. The second architecture did not require redundancy as the baseline system met the system reliability requirements. The cost and reliability results for both architectures are summarized in Table 4-1. The development costs are $100K more expensive for the second architecture. However, the per unit production costs are $39K cheaper for the second architecture and the support costs are also approximately $10M cheaper. The total life cycle costs for the second candidate architecture is almost $30M cheaper than the first architecture. In addition, the reliability of the second candidate architecture is superior than the first system.

This example illustrates how important it is to perform cost and reliability trade-offs early in the design cycle. These tradeoffs must consider the entire life cycle as opposed to the costs for a particular phase of the program. As can be seen from this example, a slight increase in costs during the development phase results in significant life cycle cost savings.

	First Candidate Architecture	Second Candidate Architecture
Development Costs	$ 2.0M	$ 2.1M
Production Costs	$ 101.0M	$ 81.1M
Unit Production Costs	$ 190K	$ 151K
Support Costs	$ 39.6M	$ 29.8M
Total Life Cycle Costs	$ 142.5M	$ 113.0M
MTBCF	2607 hours	3297 hours
MTBF	1714 hours	3297 hours

Table 4 - 1: Trade-off Cost and Reliability Summary

4.5 Summary

The ATL RASSP team have developed a concurrent engineering environment consisting of three existing computer tools (RDD-100, PRICE cost estimating tools and RAM-ILS). This system design environment quickly provides more detailed and accurate information to the integrated product team and enables them to make better informed decisions early in the development process. Since these early decisions have the largest impact on the overall life cycle costs of a system, it is important that these decisions be based on all life cycle costs and not just the cost of initial development.

As shown in the SAR example, it is possible to select the wrong architecture if the decision is only based upon the development costs. The life cycle costs in this example were reduced by over 20 percent just by understanding these costs early in the development phase. This information is critical in achieving the RASSP goal of reducing life cycle costs by a factor of four. Although the RASSP concurrent system engineering environment has been developed to work well in the signal processing domain, many of these concepts can be extended for higher level systems.

Next Up Previous Contents

Next: 5 References **Up:** Appnotes Index **Previous:**3 Technical Description

Approved for Public Release; Distribution Unlimited *Dennis Basara*

Next Up Previous Contents

Next: 6 Appendix **Up:** Appnotes Index **Previous:**4 Application of Integrated System Tools to SAR Benchmark

RASSP Integrated System Tools Appnote

5.0 References

RDD-100 User's Manual for the Integrated System Engineering (ISE) (RASSP) Schema, Ascent Logic Corporation. [RDD_ISE_USERS_97]

ISE/RASSP Design Guide Reports Supplement, Version 1, Ascent Logic Corporation, December 1996. [RDD_DESIGN_GUIDE_96]

PRICE Enterprise Release 1.0 Client Reference, First Edition, PRICE Systems, June 1997. [ENTERPRISE_PRICE_97]

Specification for Ascent Logic Corporation, RDD-100 Schema Extensions to Support Lockheed Martin PRICE, Management Sciences Inc and JRS Research Laboratories Toolsets, Ascent Logic Corporation, December 1996. [SPEC_SCHEMA_EXT_96]

Barnett, G., and C. Fry, P. Blemel, J. Walter, R. Whitman, "The RASSP Integrated Systems Tool Set Provides a Concurrent Engineering Environment for Design Trade-Offs", 1997 Proceedings for the Annual Reliability and Maintainability Symposium, Philadelphia, PA, January 1997. [BARNETT_97]

5.1 Appnotes

Hardware/Software Codesign
Token Based Performance Modeling
Virtual Prototype

5.2 Case Studies

Synthetic Aperture Radar (SAR)

Next Up Previous Contents

Next: 6 Reference List **Up:** Appnotes Index **Previous:**4 Implementation of the RASSP Configuration and Authorization Management Models

Next Up Previous Contents

Next: **Up:** Appnotes Index **Previous:**5 References

RASSP Integrated System Tools Appnote

Appendix

RDD - 100 Component Attributes of First Candidate Architecture for SAR Example [RDD_ATTRIBUTES]

Next Up Previous Contents

Next: **Up:** Appnotes Index **Previous:**5 References

Dennis Basara