

AFRL-ML-WP-TR-2000-4140

JSF MANUFACTURING DEMONSTRATION PROGRAM



Raytheon Systems Company
PO Box 92426
Los Angeles, CA 90009-2426

October 2000

Final Report for Period 01 August 1995 – 01 January 1997

Approved for Public Release; Distribution is Unlimited.

Materials & Manufacturing Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, Ohio 45433-7750

REPORT DOCUMENTATION PAGE

1. REPORT DATE (DD-MM-YYYY) 01-10-2000	2. REPORT TYPE Final Report	3. DATES COVERED (FROM - TO) 01-08-1995 to 31-01-1997
4. TITLE AND SUBTITLE JSF Manufacturing Demonstration Program Unclassified	5a. CONTRACT NUMBER F33615-95-C-5529	
	5b. GRANT NUMBER	
	5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)	5d. PROJECT NUMBER	
	5e. TASK NUMBER	
	5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Raytheon Systems Company P. O. Box 92426 Los Angeles , CA 90009-2426	8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS Materials & Manufacturing Directorate Air Force Research Laboratory Air force Materiel Command Wright-Patterson AFB , OH 45433-7750	10. SPONSOR/MONITOR'S ACRONYM(S)	
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT A PUBLIC RELEASE Materials & Manufacturing Directorate Air Force Research Laboratory Air force Materiel Command Wright-Patterson AFB , OH 45433-7750		

13. SUPPLEMENTARY NOTES**14. ABSTRACT**

Acting through the Air force Research Laboratory at Wright Patterson AFB, the JSF Program Office selected a team led by Ratheon Systems company to refine and demonstrate a powerful set of lean practices and tools as part of the JSF Manufacturing Demonstration (JMD) program. the Raytheon JMD team of Raytheon systems Company (RSC, formerly Hughes aircraft Company), Boeing Aerospace (BA, formerly McDonnell Douglas Aerospace), Management Support Technology Corporation (MST), Computer Sciences Corporation (CSC), and Arizona State University (ASU) developed, refined and demonstrated lean practices and processes and an integrated design/cost database and tools that can aid the JSF community in significantly reducing JSF life cycle cost.

15. SUBJECT TERMS

Joint Advanced Strike Technology (JAST); Joint Strike Fighter (JSF); Manufacturing & Industrial Engineering

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Fenster, Lynn lfenster@dtic.mil
Unclassified	Unclassified	Unclassified	Public Release	190	19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number 703 767-9007 DSN 427-9007

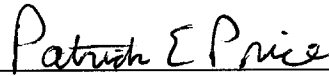
NOTICE

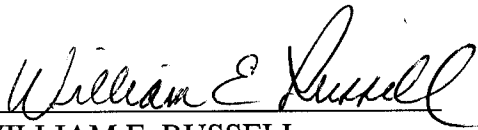
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASC/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


ALAN E. HERNER
Project Engineer
Information Integration Team
Adv. Manufacturing Enterprise Branch


PATRICK E. PRICE
Leader
Information Integration Team
Adv. Manufacturing Enterprise Branch


WILLIAM E. RUSSELL
Chief
Adv. Manufacturing Enterprise Branch
Manufacturing Technology Division

“If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify MLMS, Bldg. 653, 2977 P St., Suite 6, W-PAFB, OH 45433-7739 to help us maintain a current mailing list.”

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE			FORM APPROVED OMB NO. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE October 2000		3. REPORT TYPE AND DATES COVERED Final 08/01/95 - 01/31/97
4. TITLE AND SUBTITLE JSF Manufacturing Demonstration Program			5. FUNDING NUMBERS C F33615-95-C-5529 PE 63800F PR 2025 TA 50 WU 00	
6. AUTHOR(S)				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Raytheon Systems Company PO Box 92426 Los Angeles, CA 90009-2426			8. PERFORMING ORGANIZATION REPORT NUMBER PCD41272	
9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES) Materials & Manufacturing Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson AFB, OH 45433-7750 POC: Alan E. Herner, AFRL/MLMS, (937) 904-4354			10. SPONSORING/MONITORING AGENCY REP NUMBER AFRL-ML-WP-TR-2000-4140	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution is Unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT Acting through the Air Force Research Laboratory at Wright Patterson AFB, the JSF Program Office selected a team led by Raytheon Systems Company to refine and demonstrate a powerful set of lean practices and tools as part of the JSF Manufacturing Demonstration (JMD) program. The Raytheon JMD team of Raytheon Systems Company (RSC, formerly Hughes Aircraft Company), Boeing Aerospace (BA, formerly McDonnell Douglas Aerospace), Management Support Technology Corporation (MST), Computer Sciences Corporation (CSC), and Arizona State University (ASU) developed, refined and demonstrated <i>lean practices and processes</i> and an <i>integrated design/cost database and tools</i> that can aid the JSF community in significantly reducing JSF life cycle cost.				
14. SUBJECT TERMS Joint Advanced Strike Technology (JAST), Joint Strike Fighter (JSF), Manufacturing & Industrial Engineering.			15. NUMBER OF PAGES 190	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASS OF THIS PAGE. Unclassified	19. SECURITY CLASS OF ABSTRACT Unclassified	20. LIMITATION ABSTRACT SAR	

Contents

	Page
1.0 JOINT STRIKE FIGHTER (JSF) MANUFACTURING DEMONSTRATION OVERVIEW	1
1.1 JMD Program Executive Summary	1
1.2 Scope of This Final Report	3
2.0 OVERVIEW OF THE JMD METHODOLOGY	5
2.1 Enterprise Implementation of the JMD Methodology	6
3.0 RESULTS OF THE HDMP TILE SUBARRAY DEMONSTRATION	23
3.1 Introduction	23
3.2 Process Development	24
3.3 Cost Advantage Software	27
3.3.1 Process Models	30
3.3.2 Databases	48
3.4 Subarray Design Cost Trades	49
3.4.1 Application of the System to HDMP Design Trades	49
3.4.2 Subarray Design/Cost Trade Results	51
3.4.3 Subarray Cost Estimation Cycle Time Results	53
3.5 Lessons Learned	54
3.5.1 JMD Implementation Issues	54
3.5.2 Considerations for Implementation	56
3.5.3 Detailed Lessons Learned	56
3.6 Conclusion and Recommendations	58
4.0 ACTIVITY-BASED MANAGEMENT FOR THE JMD PROGRAM	61
4.1 Introduction	61
4.1.1 The Need for Activity-Based Management (ABM)	61
4.1.2 ABM Methodology Development	61
4.2 ABM Evaluation at HE Microwave	64
4.2.1 Planning Phase	64
4.2.2 Activity Analysis	65
4.2.3 Analysis of Output	68
4.3 ABM Integration with IT Structure	69
4.4 Lessons Learned	70

Table of Contents (Continued)

	Page
5.0 IMPLEMENTATION OF THE PROCESS CHARACTERIZATION TOOLSET (PCT).....	73
5.1 Introduction	73
5.1.1 PCT Methodology Development.....	73
5.1.2 Demonstration of PCT Methodology.....	75
5.2 Toolset Review	78
5.3 Implementation of PCT Methodology at HE Microwave	79
5.3.1 Programs.....	79
5.3.2 Application of PCT.....	81
5.3.3 PCT Influence.....	85
5.4 PCT/IT Integration.....	86
5.5 Lessons Learned	88
6.0 INFORMATION TECHNOLOGY ARCHITECTURE DESCRIPTION.....	92
6.1 Introduction	92
6.2 JMD IT Requirements	94
6.2.1 Top-Level Requirements for the Baseline IT Architecture	94
6.2.2 IPPD Team Requirements.....	94
6.2.3 Requirements for Integrating Architectural Elements	95
6.2.4 Specific Cost Tool Requirements	98
6.3 JMD Implementation Methodology.....	101
6.3.1 Overview	101
6.3.2 Design/Cost Tool and Data Implementation	103
6.3.3 Implementation of IPT Cost Modeling.....	110
6.3.4 Virtual Database Agent (VDA).....	114
6.3.5 Lessons Learned	134
APPENDIX A – DETAILED TECHNICAL SPECIFICATIONS	127
APPENDIX B – ACRONYMS, GLOSSARY AND REFERENCES	171

List of Figures

	Page
1-1 Planned and Calculated Monthly Cost Savings for a 192-Element Prototype Subarray.....	3
2-1 JMD Methodology Overview.....	6
2-2 Enterprise JMD Methodology Implementation Roadmap	7
2-3 IPPD Benefits Accrue to All	8
2-4 CAIV Encompasses DTC and Integrates Other Acquisition Methodologies .	11
2-5 The Cornerstones of Raytheon's Worldwide Purchasing	12
2-6 The PCT Is Aligned with the PDP, DTC and Six Sigma Strategies	14
2-7 Parametric Cost Models Are Used when Detail Information Is Unavailable .	16
2-8 The JMD Integrated Cost/Design Environment Provides Data to the IPT in Near-Real Time	18
2-9 Seven Steps to an Affordable Design.....	21
3-1 Generic Cross-Sectional View of the Prototype Tile Array Assembly Components	25
3-2 Tile Array Cost Model Utilization Matrix.....	26
3-3 CA Modeler Window.....	28
3-4 CA Modeler Context Window.....	28
3-5 CA Cost Summary Window	29
3-6 Multilayer Ceramic Substrate Fabrication Process Model Window.....	31
3-7 Multilayer Ceramic Substrate Fabrication Material Model Window.....	31
3-8 Link Between the Summary Table and the CA Explain Feature	33
3-9 Relationship Between Process Cost Listed in the Summary Table and How It is Defined in Editor Window.....	34
3-10 Relationship Between Process Cost Editor Window and Editor Windows Defining Those Factors Influencing Process Cost.....	35
3-11 Editor Window Defining Grind Time.....	36
3-12 Material Cost Editor Window	37
3-13 Tooling Cost Editor Window	38
3-14 A Table Within the Multilayer Ceramic Substrate Process Model.....	39
3-15 Substrate Assembly Process Model Summary Window	40
3-16 Module Assembly Process Model Summary Window.....	41
3-17 PWB Material Selection Window	42
3-18 PWB Process Selection Window.....	43
3-19 Printed Wiring Assembly Process Model Summary Window	44
3-20 TD6 Modeler Window	46
3-21 Material Window Associated with the Modeler Window of 3-20	46
3-22 Process Window Associated with the Modeler Window of 3-20	47
3-23 Array Assembly Modeler Window.....	47
3-24 Tile Array Window	49

List of Figures (Continued)

	Page
3-25 Total Costs Associated with the Assembly of a Tile Array Configuration.....	50
3-26 Planned and Calculated Monthly Cost Savings for a 192-Element Prototype Subarray.....	53
4-1 Variability Between Programs Makes it Difficult to Track Charge Accounts and the Activities That May Be Associated With Them	63
4-2 Each Process Engineer Works on a Variety of Activities.....	65
4-3 Comparison Between Program 1 Time Charges from the Manbase System and the Activities the Engineers Claimed They Worked.....	67
4-4 Program 1 Cost Comparison Between Charges Accumulated by Activity and Those from a Traditional Ledger System.....	68
4-5 ABM Data Can Be Used to Determine If a Process Is Proceeding as Expected	69
5-1 Flowchart for the Process Characterization Methodology and Toolset.....	74
5-2 The Process Characterization Toolset.....	75
5-3 Product Features of Programs Investigated	76
5-4 HEM Original Process Steps, Time Estimates, Yields, Etc., Were the Basis for the Cost Advantage Tile Assembly Model.....	78
5-5 PCT Methodology Adds Conditional Statements to Account for Product Variations	78
5-6 This Table Was Used to Demonstrate How to Take Data Out of the PCT and Put It in a Form That Could Be Used by Cognition's Cost Advantage Tool. .	80
5-7 Curve of Simulated Data Extracted From Database to Show Relationship of Machine Cycle Time to Die Area	82
5-8 The Die Size/Cycle Time Process Spectrum	83
6-1 Design/Cost Data Integration	93
6-2 Data Types	94
6-3 PIM Communications Infrastructure Map	97
6-4 CIS Communications Infrastructure Map.....	98
6-5 Flow of Integrated Design and Cost Data.....	103
6-6 Large Scale Point-to-Point Data Integration Complexity.....	109
6-7 Scope of the JMD VDA Data Integration	109
6-8 Scalability of the VDA Data Integration.....	110
6-9 JGUI's Main Screen.....	113
6-10 Object Schematic	115
6-11 Object with Interfaces	116
6-12 Prototype IT Architecture	119
6-13 VDA Module 1 Outputs to CA for the NC Milling Process Inputs (Data for Display Purposes Only)	121
6-14 Cross-Application Interfaces.....	124

List of Tables

	Page
1-1 JMD Design/Cost Tradeoff Achievements Summary.....	2
1-2 JMD Deliverable Documents (Distribution A Versions: Distribution Unlimited)	3
2-1 Application/Tool Summary.....	19
3-1 JMD Tool Selection	30
6-1 Data Types and Storage Repositories.....	95
6-2 PDM Application/Tool Summary.....	97
6-3 JMD IT Prototype Requirements vs. Raytheon's PDM Tools	102
6-4 CA Output File Types	122
6-5 Tool and Data Interfaces for JMD Demonstrations.....	124

1.0 JOINT STRIKE FIGHTER (JSF) MANUFACTURING DEMONSTRATION OVERVIEW

1.1 JMD Program Executive Summary

The cornerstone of the JSF program is affordability. As a result, the JSF program has been a leader in affordability activities such as the Lean Aerospace Initiative (LAI). Modeled after the Massachusetts Institute of Technology's International Motor Vehicle Program, the LAI is a joint U.S. Air Force – Industry effort to identify key principles and practices that will enable the implementation of lean manufacturing within the U.S. aerospace industrial base.

The LAI surveyed its members and found that those companies who had database commonality among design and cost information achieved significantly better schedule and cost performance than those who did not. As part of its research, the LAI identified the following attributes of an integrated cost/design database:

- Makes cost readily accessible to the design team
- Tailors application to fit Integrated Product Team (IPT) scope
- Maximizes use of actual cost data; minimizes dependence on cost models
- Cost data kept very current
- Cost impact of design changes can be determined at the micro (parts/assemblies) level
- Rolls up product costs frequently.

Acting through the Air Force Research Laboratory at Wright Patterson AFB, the JSF Program Office selected a team led by Raytheon Systems Company to refine and demonstrate a powerful set of lean practices and tools as part of the JSF Manufacturing Demonstration (JMD) program. The Raytheon JMD team of Raytheon Systems Company (RSC, formerly Hughes Aircraft Company), Boeing Aerospace (BA, formerly McDonnell Douglas Aerospace), Management Support Technology Corporation (MST), Computer Sciences Corporation (CSC), and Arizona State University (ASU) developed, refined and demonstrated *lean practices and processes* and *an integrated design/cost database and tools* that can aid the JSF community in significantly reducing JSF life cycle cost. The JMD program successfully completed three major thrusts:

1. Develop an Integrated Product and Process Development (IPPD) process that emphasizes cost as an independent variable; define and document key lean practices/processes developed for the IPPD environment; integrate design and cost data and tools, making cost and design data available to the IPT in near-real time.
2. Demonstrate the effectiveness of the JMD methodology and integrated toolset on the ARPA-funded High Density Microwave Packaging (HDMP) program.
3. Disseminate knowledge of the JMD methodology to the JSF community prior to

the E&MD phase of the JSF program, maximizing the JMD cost-reducing ability during the JSF engineering development phase.

To support best-value design choices, the JMD team developed a three-part IPPD methodology consisting of:

- An all-tier Product Development Process, with inclusive design to cost and design for six sigma, keenly focused on meeting all customer requirements
- A number of corporate strategies including design to cost, activity based management allowing for improved cost visibility, team motivation, and strategic sourcing and supplier development
- Software support tools that integrate design and cost information, enabling near-real time cost estimation and simplified knowledge base development.

The JMD team successfully demonstrated the effectiveness of the JMD methodology at a mini-demonstration held in January 1997 and during the initial phase of the full demonstration held throughout 1997. Advanced tile module active array technology was chosen as the demonstration vehicle for the JMD program because it promises to form the foundation for a new generation of low cost, lightweight active arrays. The tile module is being developed for the Advanced Research Projects Agency (ARPA) on the High Density Microwave Packaging (HDMP) program.

Using the integrated design/cost environment, the team was able to perform design trades to reduce the cost of the HDMP tile array. At the program's mini-demonstration held in January 1997, the cost of the tile transmit/receive electronic modules was reduced by 19.3% using the JMD methodology. The methodology also showed a 60% reduction in tradeoff cycle time and a 27% reduction in tradeoff labor hours. During the initial phase of the full demonstration held during 1997, the methodology was applied to a more complex subarray. A cost reduction of 10.5% was achieved on the subarray, and the tradeoff cycle time and labor hour reductions experienced during the mini-demonstration were also realized during the full demonstration. These achievements are summarized in Table 1-1. In December 1997, the JMD program was descoped at the convenience of the customer. The JMD team was well on the way to reaching its 20% cost reduction goal on the HDMP tile subarray for the full demonstration at the time of the descope. The monthly cost savings achieved on the tile subarray are summarized in Figure 1-1.

Table 1-1. JMD Design/Cost Tradeoff Achievements Summary

	Mini Demonstration (Jan. 1997) Tile Transmit/Receive Module	Full Demonstration (Jan. – Dec. 1997) Tile Transmit/Receive Subarray
Recurring Cost Reduction	19.3%	10.5%
Tradeoff Cycle Time Reduction	60%	~60%
Tradeoff Labor Hours Reduction	27%	~27%

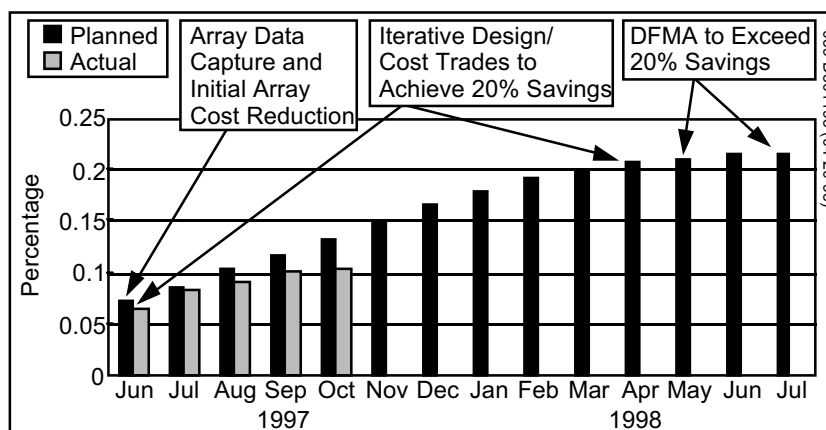


Figure 1-1. Planned and Calculated Monthly Cost Savings for a 192-Element Prototype Subarray. The cost savings analysis effort was descoped after completing only the first 5 months of the 14-month study.

As JMD proceeded, the team documented lean practices and processes and integrated tools that the program was developing. With the release of this final report, the JMD program has met all of the deliverable requirements specified by the contract. The JMD documents and videos are available to the JSF community through the JSF Program Office. The JMD documents and their delivery dates appear in Table 1-2.

Table 1-2. JMD Deliverable Documents (Distribution A Versions: Distribution Unlimited)

Document	Release Date
Lean Practices and Processes for the IPPD Environment	13 January 1999
Information Technology Architecture: Integrated Design and Cost Data and Tools (Version 1.0)	18 January 1999
Cost Estimation Tool Comparison Summary	15 January 1999
Design to Cost Guide	11 January 1999
Design to Cost Training Materials	11 January 1999
Process Characterization Toolset - User's Manual	11 January 1999
STEP Translator Evaluation Report	15 January 1999
JMD Mini-Demonstration Video	15 January 1997
JMD Final Report	13 January 1999

1.2 Scope of This Final Report

The JMD program began in August 1995 and was scheduled for completion in December 1998. However, because of federal budget constraints that were subsequently imposed, the program was descoped and funds were deobligated at the convenience of the customer in early December 1997. At that time, the customer specified that a final report documenting progress since the end of Phase I in January 1997 be generated.

As specified by the JMD customer, this report documents the work completed on the

JMD program up until its descope. This final report in conjunction with the CDRL deliverables listed in Table 1-2 constitute the full documentation of the JMD objectives. The specific subjects covered within this final report are:

1. JMD Methodology Overview and Enterprise Implementation
2. Results of the HDMP full demonstration
3. Implementation and evaluation of Activity-Based Management (ABM) at HE Microwave
4. Implementation and evaluation of Process Characterization Toolset (PCT) at HE Microwave
5. CORBA-compliant information technology (IT) architecture

2.0 OVERVIEW OF THE JMD METHODOLOGY

The Joint Strike Fighter program is developing a family of tactical aircraft to meet the next generation strike mission needs of the U.S. Navy, Marines, Air Force, and Allied Forces. The cornerstone of this program is affordability. With this in mind, the JSF program has been a leader in affordability activities such as the LAI.

Modeled after the Massachusetts Institute of Technology's International Motor Vehicle Program, the LAI is a joint U.S. Air Force–Industry effort to identify key principles and practices that will enable the implementation of lean manufacturing within the U.S. aerospace industrial base.

LAI surveyed its members and found that those companies who had database commonality among design and cost information achieved significantly better schedule and cost performance than those who did not. As part of its research, LAI identified the following attributes of an integrated cost/design database:

- Makes cost readily accessible to the design team
- Tailors application to fit IPT scope
- Maximizes use of actual cost data; minimizes dependence on cost models
- Cost data kept very current
- Cost impact of design changes can be determined at the micro (parts/assemblies) level
- Rolls up product costs frequently

To achieve these LAI goals, the JMD team developed a three-part methodology consisting of:

- An all-tier product development process keenly focused on meeting all customer requirements
- A number of corporate strategies, including Design to Cost and Activity-Based Management, allowing for improved cost visibility, team motivation, strategic sourcing, and supplier development
- Software support tools that integrate design and cost information, enabling near-real-time cost estimation and simplified knowledge base development

The JMD program developed a methodology that extended Integrated Product and Process Development (IPPD) to create more affordable, best-value products as the normal way of doing business. The overall JMD methodology is summarized in Figure 2-1.

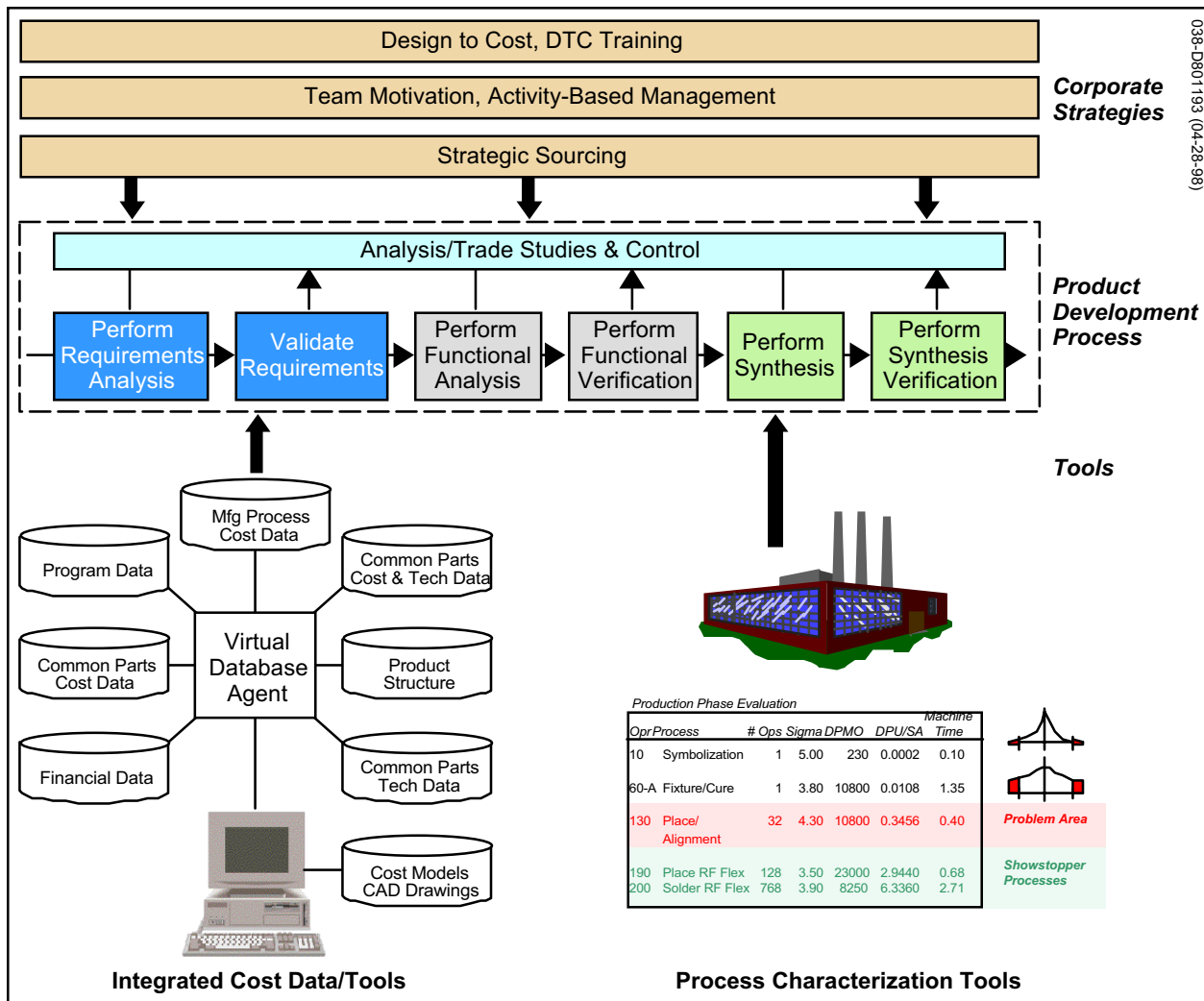


Figure 2-1. JMD Methodology Overview

2.1 Enterprise Implementation of the JMD Methodology

As a result of lessons learned on the work of integrating cost with product development, a 10-step methodology of JMD enterprise-wide implementation was developed.

10 Steps to an Affordable Design

1. Adopt a cost focus.
2. Institute a disciplined product development process (PDP).
3. Establish a process owner and management metrics (e.g. cost, Six Sigma, etc.)
4. Train IPTs in Design to Cost.
5. Establish procurement processes that ensure purchase of quality parts at the best possible price.
6. Implement processes and tools that allow manufacturing process costs to be fully

understood and use them to improve.

7. Perform a parametric assessment of early design space, identifying requirements and key characteristics.
8. Integrate part cost, process cost, and design data and make these detailed knowledge bases available to the IPT in near-real time.
9. Iterate detailed design and monitor progress towards goal.
10. Review, validate, and update integrated cost and design databases regularly.

The implementation process is summarized in Figure 2-2. Each of the 10 steps will be discussed below.

1. Adopt a cost focus

To excel in today's marketplace, companies can be competitive only by offering products that provide a robust balance among cost, performance, and supportability. The new defense procurement environment continues to demand premier-quality products with excellent technical performance. However, strong market forces favor the low-cost producers. To remain competitive, companies must create a culture that drives

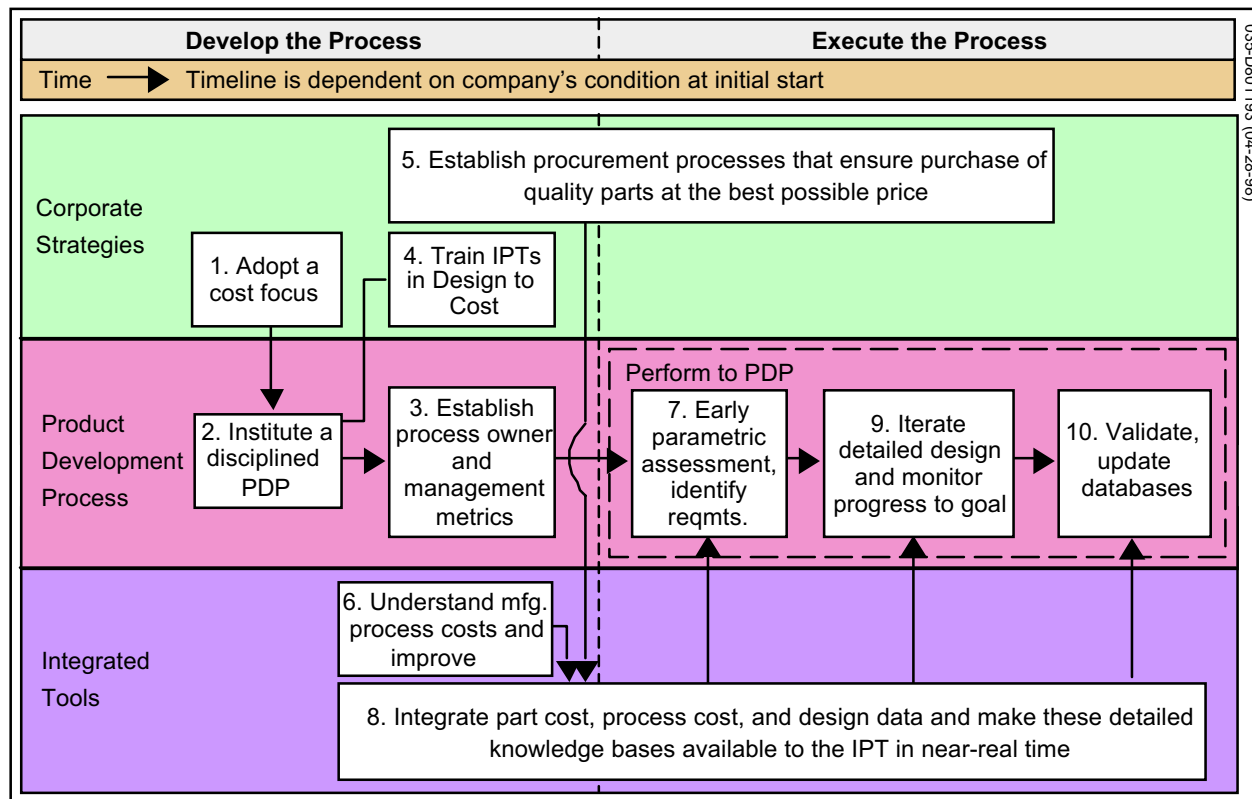


Figure 2-2. Enterprise JMD Methodology Implementation Roadmap

every development and manufacturing decision to eliminate non-value-added activity. This culture must be instilled with the commitment to meet or beat clearly defined

product cost targets.

Changing a corporate culture requires firm financial commitment. Lean corporate strategies and tools must be identified, evaluated, and implemented to fully realize cost reductions. The JMD program has done just that: identified, evaluated, and implemented the strategies that lead to lower-cost products. Any company within the JSF community that wishes to embrace the JMD methodology can evaluate and implement the lean practices and tools that are documented in these reports.

2. Institute a disciplined product development process

To fully embrace a cost focus, companies must implement a disciplined product development process (PDP) that treats cost as an independent variable, equal to performance and supportability requirements. The PDP incorporates IPPD philosophies. IPPD is a straightforward concept focused on bringing the right people with the right training together with proven tools and processes to satisfy customer needs. The IPPD philosophy is rooted in improved knowledge sharing among design team members and is facilitated by well-defined processes and ready access to the required design and cost data and tools. Because IPPD developments provide much closer ties among customers, contractors, and suppliers, there is a shared vision of economic, technical, and schedule priorities. The use of well-defined processes and tools provides order and predictability. Figure 2-3 highlights key benefits to customers, the enterprise, and employees. The creation and deployment of key lean practices and processes will leverage and magnify IPPD effectiveness, while improved access to cost data through an integrated toolset will reduce design trade effort and cycle time, allowing more and better cost assessments to be completed before irreversible design decisions are made.

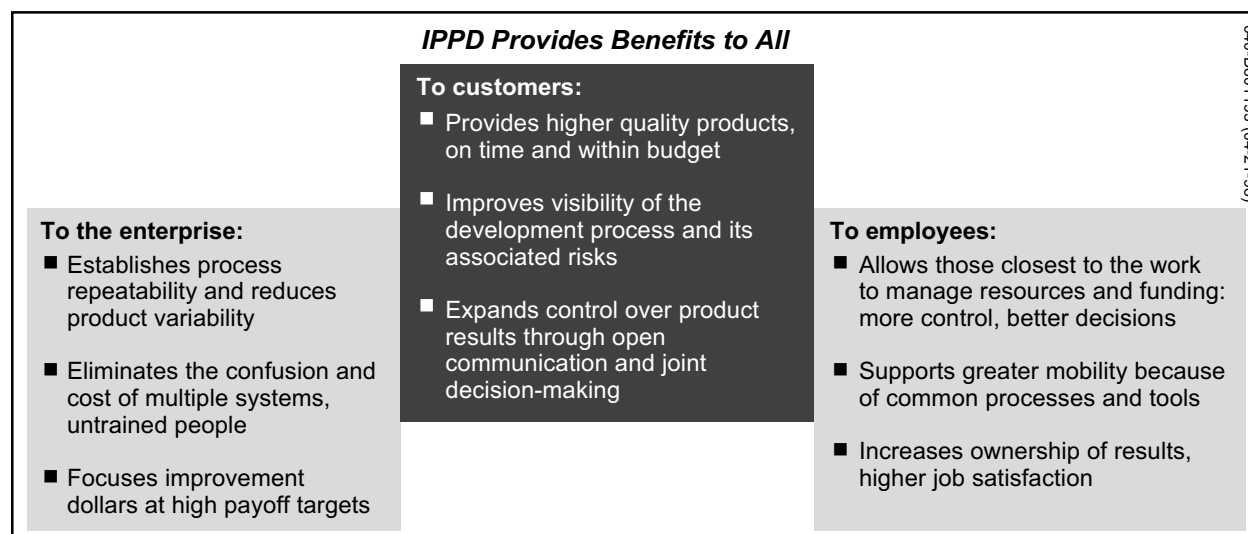


Figure 2-3. IPPD Benefits Accrue to All

The IPPD Methodology developed by the JMD program and Raytheon is documented in detail in the document “Lean Practices and Processes for the IPPD Environment” listed

in Table 1-2.

The JMD program has shown that the IPPD methodology approach to product development synchronizes the efforts of IPTs throughout the IPPD environment. Common processes and common tools are reciprocally supportive and greatly improve both technical and business communications, while adding substantially to work force transportability. The iterative and interactive application of PDP-derived product processes further improves communication and buy-in at all levels in the IPT hierarchy. Each of these benefits should help to improve the ability to control the design process, reducing oversights and mistakes grounded in task uncertainty and poor information exchange.

Raytheon's second generation IPPD system is in early deployment and is already showing measurable improvements in product cost control, as well as marked progress in effective IPT treatment of cost, manufacturability, yield and design cycle time. We believe that the JMD demonstrations showed that the IPPD approach to product design has the potential to reduce JSF cost significantly, while providing a product that balances cost, risk, and performance. Because most of the JSF community is pursuing process improvement, it is likely that other community members have defined development processes that offer a similar but unique set of benefits from our IPPD. JMD would be interested in sharing ideas with these community members so that we can improve the IPPD approach at an accelerated rate. Similarly, it is our hope that other members of the JSF community will consider piloting elements of the PDP design approach within their own operations so that they, too, can evaluate its efficacy in providing Best Customer Value.

An established commonality of design approach and tools among eventual JSF contractors will minimize cross-organizational boundaries and will ensure that the *integrated* aspect of IPPD will span the entire weapon system, reducing the non-value-added effort often expended at organizational boundaries.

3. *Establish PDP process owners and management metrics*

Having a lean product development process will result in improved product quality and lower cost only if managed correctly. Companies must ensure that the disciplines invoked by the lean JMD PDP are followed in every product line across the corporation. To do this, process owners must be identified who are responsible for implementing and improving the PDP as it applies to their particular product.

Process owners manage the changes to the IPPD methodology. To do this, they must identify and evaluate potential improvements, gain enterprise-wide consensus for the recommended changes, and revise the IPPD process descriptions to implement the changes.

The first responsibility of the process owners is to identify and evaluate potential improvements to the IPPD methodology. The main process improvement ideas will come from lessons learned by IPTs as they execute, measure and improve their tailored processes. Each program is, in essence, a laboratory that is testing process improvements. Process owners roll up metrics collected by individual programs, participate in process reviews, and conduct process assessments. They analyze the data to determine which of the tailored processes are candidates to replace the previously identified best practices captured in the IPPD processes.

Process owners also evaluate other sources of potential process improvements, such as benchtrending, and determine which ideas should be implemented. They also bear responsibility for getting enterprise-wide consensus on recommended changes and for updating the IPPD documentation to reflect approved changes.

Line management must ensure that the elements of IPPD are followed in every phase of a product's development. To accomplish this, Raytheon has implemented checkpoints, or gates, that every product must pass through as it progresses in its development cycle. At key points, programs are required to prove that every requirement within the PDP has been met before the product is allowed to continue in its development cycle. This ensures a disciplined approach across the corporation and contributes to a permanent change in corporate culture.

4. Train IPTs in Design to Cost (DTC) and Cost as an Independent Variable (CAIV)

CAIV is a new policy of the DoD, developed in 1995 under the leadership of Dr. Paul Kaminski, the Under Secretary of Defense for Acquisition and Technology. It is a more far-reaching policy than Design to Cost and in many ways supersedes the DTC approach. CAIV covers many topics of acquisition not addressed by DTC such as contract and subcontract management, connections with Integrated Master Planning and Scheduling (IMP/IMS) and integration with Risk Management. CAIV also puts greater emphasis on total ownership cost, which includes every aspect of the life cycle.

However, it should be noted that CAIV and DTC are mutually supportive. At Raytheon this means that DTC is the key analytical engine for designers and other members of the IPT to support this part of the CAIV methodology. Translated into the Raytheon standard engineering process, CAIV requires that the DTC "7 Step Method" as shown in Figure 2-4 and Figure 2-9 must analyze a broader range of design trade scenarios because CAIV requires that engineering (systems analysis) be used early on to evaluate the warfighter's needs from the mission perspective.

DTC is both a business culture and a way of conducting a development business. The JMD DTC approach goes far beyond the traditional DTC application because it is fully integrated into a comprehensive product development process that treats cost as an independent and critical requirement. Using this approach, each design choice is evaluated simultaneously for both cost and benefit. This process should begin prior to the Concept Exploration proposal and must remain vigorous throughout product development. DTC is focused on minimizing cost, identifying and eliminating non-value-added activity, reducing cost risk, and achieving well-defined product cost

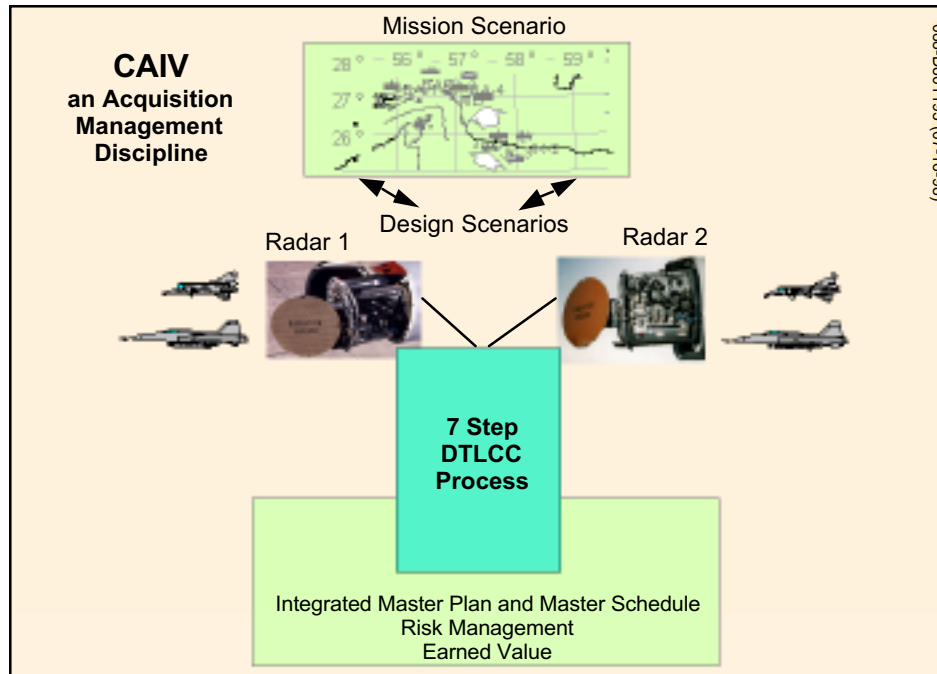


Figure 2-4. CAIV Encompasses DTC and Integrates Other Acquisition Methodologies

objectives by optimizing the entire product to provide Best Customer Value.

DTC is iterative, creating a multi-tier IPT that integrates Design Engineering, Process Engineering, Manufacturing, Materiel, Quality, and Business across all IPT levels to develop manufacturable products that meet well-defined cost targets while balancing cost, performance, supportability and risk.

Key DTC requirements/goals are as follows:

- Cost must be an Independent Design Requirement with importance equal to or greater than performance (i.e., the process must address CAIV as its primary focus).
- DTC focus must begin as early as possible in a program (pre-RFQ) for early cost driver identification.
- Lean practices and processes must be effectively leveraged.
- Cost estimation can be approximate in early program phases; progressively better during Engineering and Manufacturing Development (E&MD).
- Cost estimation cycle time must be near real-time by the detailed design phase.
- Design, manufacturing and cost data must be readily accessible.
- DTC tools must be user-friendly and accessible from the IPT's desktop.

- Manufacturing process costs must be understood.
- DTC training, deployment, and data collection must be given high priority.

The detailed JMD DTC process is described in the “Design to Cost Guide” and the “Design to Cost Training Materials” documents listed in Table 1-2.

5. Establish procurement processes that ensure purchase of quality parts at the best possible price

Historically at Raytheon, 50–80% of product cost has been in purchased material. In order to gain control over and be able to reduce product cost, we must create dependable, cost-effective suppliers. Strategic Sourcing, which started at General Motors (GM) and was instituted at GM Hughes Electronics (GMHE) 2 years ago, has rapidly grown into a dynamic toolset that is being deployed throughout our organization. Strategic Sourcing and supplier development allow the designer to select the best parts from the most cooperative, proven suppliers. In our world of rapid change, the process of integrating commercial practices from GM’s Worldwide Purchasing process into the Raytheon framework has been challenging and productive. Raytheon has experienced significant reductions in parts cost as a result of Strategic Sourcing. The cornerstones of Strategic Sourcing as realized in Raytheon’s Worldwide Purchasing approach are shown in Figure 2-5.

The details of Raytheon’s Worldwide Purchasing concept are outlined in the document “Lean Practices and Processes for the IPPD Environment” listed in Table 1-2.

Strategic Sourcing can be adapted and applied by members of the JSF community. Because many JSF participants are very large enterprises, they are particularly well positioned to leverage those strategic sourcing techniques that depend on being part of a large material-consumption base. JSF community members of any size can exploit

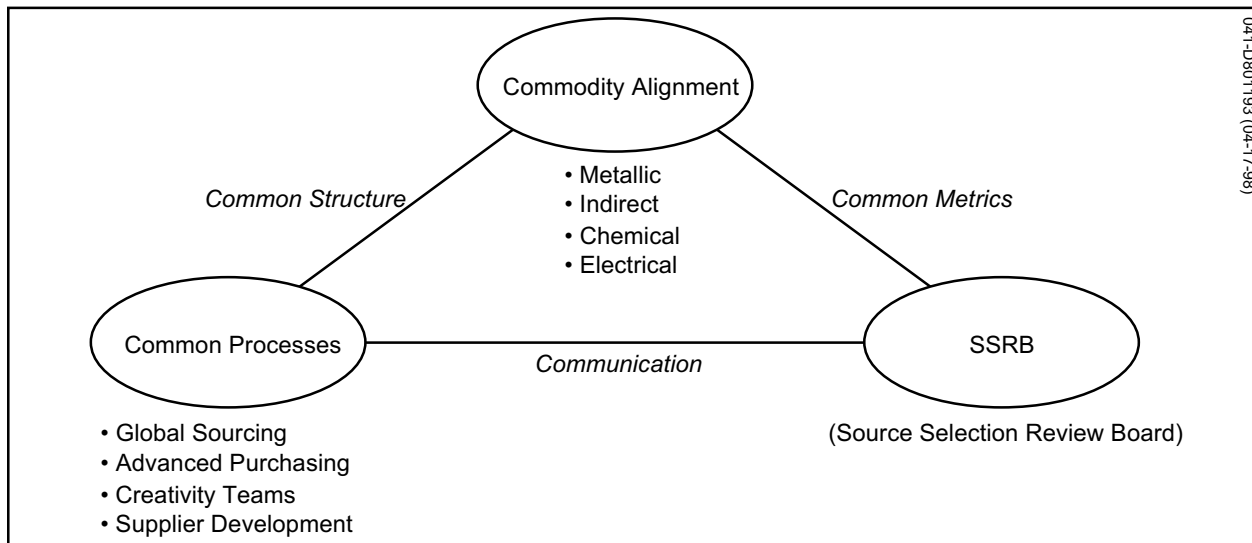


Figure 2-5. The Cornerstones of Raytheon’s Worldwide Purchasing
the techniques of Advanced Purchasing, focused Creativity Teams and the participatory approach to Supplier Development.

6. Implement processes and tools that allow manufacturing process costs to be fully understood and use them to improve

Understanding the manufacturing process—its flow, material, manpower, equipment requirements, time allotments, and possible unscheduled tasks (such as rework)—is the focus of process characterization. Process characterization provides the information needed for cost/yield estimation and for identification of process improvements that will lower costs and defects while increasing capability and efficiency.

To meet JSF recurring cost goals, manufacturing process costs need to be better understood, accurately modeled, and made available to the design IPT so that key manufacturing decisions can be made early in the development cycle. The JMD program developed and evaluated two approaches to better understanding manufacturing process costs.

- **Process Characterization Toolset**

To find and eliminate non-value-added activity, a major emphasis in the lean enterprise environment must be the clear definition and characterizations of critical processes. The importance of understanding manufacturing processes is further magnified by the JSF focus on modeling, simulation and the use of virtual reality technology.

As part of this process perspective, it is important to focus on two important facts. First, processes vary with time; second, changes in process will be responsible for changes in products. To illustrate, consider a machine, such as a production drill press. When it is new, it can hold specified tolerances. As it begins to wear, bearings and races may deteriorate and it may no longer hold these tighter tolerances, and the process is changing. As these changes occur, the holes drilled in products may become out-of-round or off-center, causing the product to change in an uncontrolled manner.

Process characterization, then, is important not only for process control as applied in SPC and exploited by Six Sigma, but also as an aid to understanding the relationship between product performance and process variability. This product/process relationship allows us to focus on key control characteristics (KCCs) in the process that will have the most impact on key product characteristics (KPCs) in the manufactured part.

Process characterization is also necessary if we are to make objective evaluations of new, alternative or evolving processes and to accurately model evolving process sequences in virtual manufacturing simulation efforts. For example, is there enough information to determine the most cost-effective way to build a part? Do we have enough information to decide if we can apply a new, “greener” process, and what is the cost impact? Do we have enough information to know when we need to replace rather than repair existing equipment? Is there enough information to determine if new production technologies are sufficiently mature to deploy? The above questions

can be addressed if detailed process characteristics are captured.

The JMD program has defined a methodology for process characterization that is summarized in the “Process Characterization Toolset (PCT) User’s Manual” listed in Table 1-1. The relationship between PCT methodology, PDP, Six Sigma and DTC is depicted in Figure 2-6. Because process characterization can be difficult, we have separated the stages of the methodology into discrete steps that occur within the stages; these steps are specifically supported by the COTS software tools that make up the Process Characterization Toolset. The toolset also provides a framework for making process characterization more consistent across different processes and organizations.

The JMD program evaluated the effect of PCT on the HDMP T/R module build at our HE Microwave facility in Tucson. The results are documented in Section 5 of this report.

- Activity-Based Management

The cost collection and distribution systems in the defense contracting environment are traditionally structured to collect and distribute cost around the

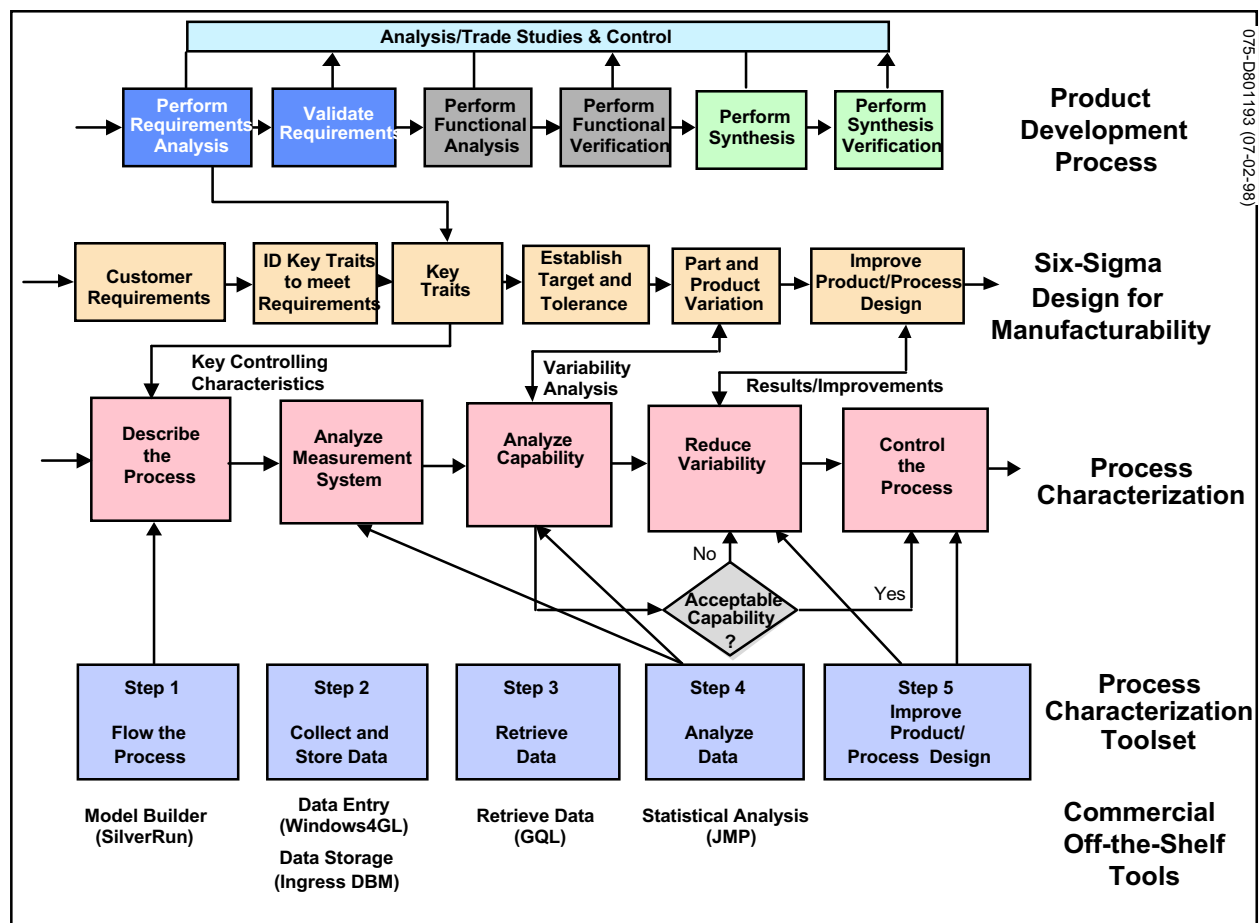


Figure 2-6. The PCT Is Aligned with the PDP, DTC and Six Sigma Strategies

elements of cost, functional organizations, and contracts. However, the traditional structure does not adequately handle the association of process cost to product cost. The traditional systems do collect assembly labor and product direct material at the product cost level, but the true cost of producing a product is masked by allocated support and applied overhead costs. Basically, the traditional systems approach to understanding resource consumption is:

- Organizations consume resources
- Programs use organizations' resources

To understand the true cost of products, a paradigm shift is required in the methods we use to collect and distribute cost. The cost structure of the future should serve as a tool that enables management to reduce costs, eliminate non-value-added activities, and determine the true costs of specific activities that contribute to product cost. Activity-Based Management is a management tool and Activity-Based Costing is an accounting method, both of which can enable a business to determine the true cost of the activities that are used to build their products.

Activity-Based Management (ABM) enables a business to better understand how and where its resources are consumed. In ABM, all major activities within an operation are identified and the costs of performing each activity are collected. The cost of performing each activity is clearly visible to management; however, the traditional cost distribution to cost objectives (i.e., the products or programs that benefit from the activities) does not change.

Activity-Based Costing (ABC) also enables a business to better understand how and where its resources are consumed. In ABC, all major activities within an operation are identified and the costs of performing each activity are collected. The resulting costs are then charged directly to the cost objective that consumed the activity.

Simply stated, the new paradigm in understanding resource consumption is:

- Activities consume resources
- Products consume activities

The ABM and ABC concepts are discussed in detail in the document, "Lean Practices and Processes for the IPPD Environment," listed in Table 1-1. As part of this program, a pilot study was performed at Raytheon's HE Microwave facility in Tucson, Arizona. The results of that study are included in Section 4 of this report.

7. Perform a parametric assessment of early design space identifying requirements and key characteristics

There is no single cost estimation technique that can serve DTC adequately for all system levels nor for all program phases. Parametric models are most often used when little detailed design information is available and approximate cost assessment is satisfactory. Parametric models relate physical and functional cost drivers to

product cost using cost estimating relationships derived from regression or other analysis of historical product/cost data. Their unique benefits are rooted in the ability to provide repeatable cost estimates before detailed design knowledge is available and to provide tests of reasonableness throughout the development. Parametric models are commonly used for concept tradeoff evaluations during high-leverage pre-proposal, proposal and Concept Exploration phases when little detail is known (see Figure 2-7). Previous product cost histories are used to develop cost relationships that are used to predict the cost of meeting new requirements. Common uses for parametric cost models include:

- Architectural cost tradeoff analysis
- Pre-proposal cost evaluations
- Rapid rough-order of magnitude (ROM) estimates
- Should-cost competitor/self proposal evaluation
- DTC cost target allocations
- Should-cost estimates for “buy” items (UPC)
- UPC estimation prior to E&MD
- Early RDT&E NRE cost estimation
- Early O&S cost estimates for DTLCC

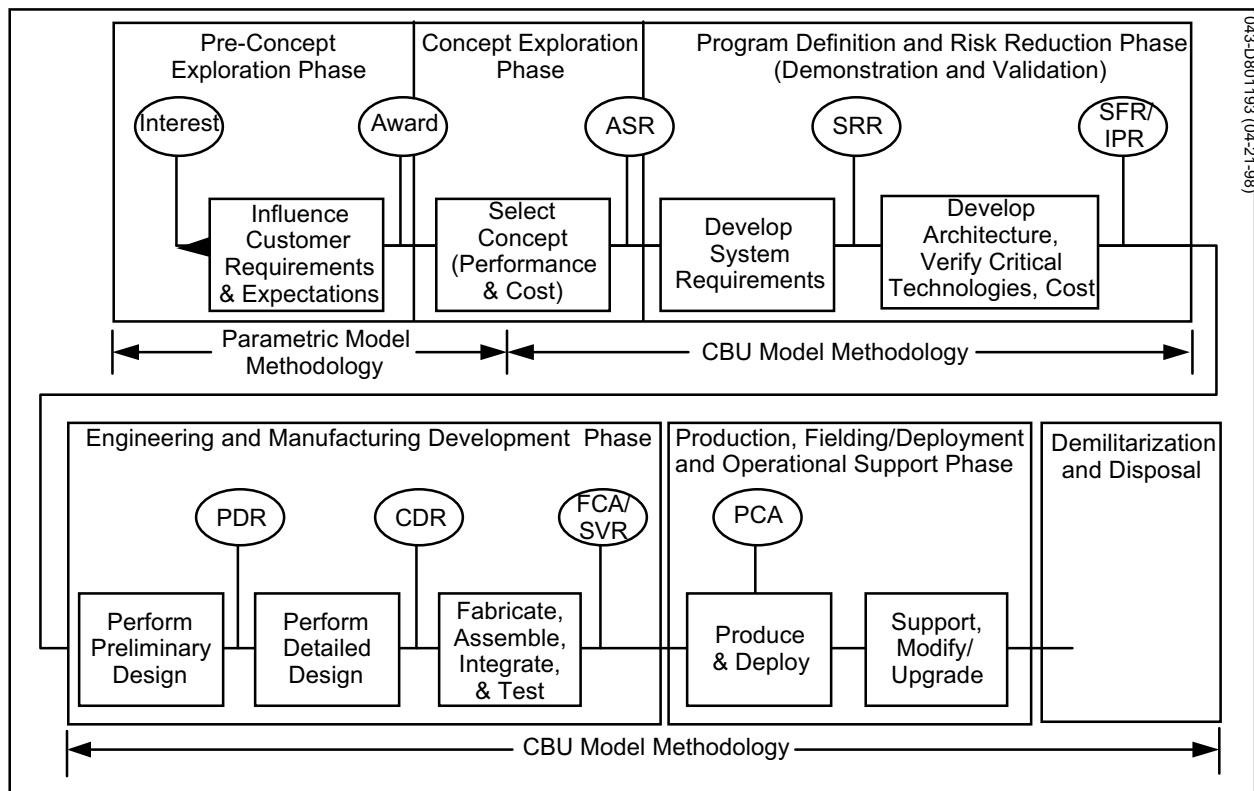


Figure 2-7. Parametric Cost Models Are Used when Detail Information Is Unavailable

- Cost estimation “sanity checks” throughout the development

The JMD program has identified a variety of parametric cost estimating tools that are available as COTS products. Some of the most commonly applied of these are described (including summaries of their respective inputs, outputs and key features) in the “Design to Cost Guide,” listed in Table 1-1.

As a product’s concept definition nears completion and design details become known, final Unit Production Cost (UPC) cost/performance trades and Operation and Support (O&S) sparing projections are best supported by detailed Cost Build-up (CBU) models. CBU models are based on the actual or quoted costs of a product’s constituent parts and processes. They are used to achieve the cost visibility needed to make specific design decisions, usually during the preliminary and detailed design phases of product development. These models aggregate costs from lower levels to higher levels of assembly. For UPC estimation, the same CBU tool can be populated with parts/processes from a previous product to provide a “by analogy” model or by supplier quotes and process cost/yield estimates to form a “detailed” model of the design at hand. Similarly, during the low rate initial production (LRIP) and production phases, the same tool can be populated with actual parts costs and measured process costs/yields to create “as built” models for production cost monitoring and cost estimation benchmarking/validation. The CBU tool can also be used to roll up subproduct estimates to estimate the cost of higher level products. CBU model outputs can be used to populate parametric models with the historical data that they require.

The JMD methodology describes the process to integrate parts cost, process cost, and design data, populate a CBU model, and make this information available to the IPT in near-real time.

8. Integrate part cost, process cost, and design data and make these detailed knowledge bases available to the IPT in near-real time

The desired capability in an integrated system is best characterized simply as “providing the right people with the right information at the right time.” The migration from a traditional serial workflow process to an IPPD environment, where all functional areas work together in parallel, magnified the deficiencies in the pre-JMD development toolset and data management environments. The traditional development tool sets at Raytheon consisted largely of legacy applications developed in-house that used a variety of partially integrated COTS applications. The once-adequate legacy systems were developed around the serial development processes of the past. They were designed to serve the needs of specific functional areas first and the needs of related functions only as an afterthought. COTS packages, while more robust, were generally intended to support a limited set of users. Engineering CAD tools and other primarily UNIX-based products have also been focused on narrowly defined user groups. The resulting environment was poorly positioned to serve the data and information processing needs of today’s interdisciplinary IPTs.

At Raytheon, the pre-JMD IPPD information environment was characterized by:

- Multiple sources of data in functionally separate application systems
- Multiple application systems resident on multiple computing platforms using multiple architectures, operating systems and databases
- Multiple formats for whatever common data that existed across functional areas

Such an environment of diverse data sources and sometimes redundant and overlapping tools led to unneeded complexity, marginal cost/benefit visibility and extended tradeoff cycle times. In addition, the propagation of changes in data in associated functional areas other than the originating functional area was rarely timely. Lags in information availability were often reflected in extended development time and increased product recurring cost. For example, many cost estimating tools used purchased part and material costs as components in their cost models, but the linkage between the source of the data (agreements in the materiel organization) and the user of the data (the cost model being used by the design estimator) rarely existed. As a result, an upward change in the market price of a raw material or commodity was not reflected in the cost estimating model in time to adjust the product design. The cost engineers discovered the change in market price only after analyzing a budget overrun for producing the part.

Inherent in the definition of an integrated design/cost environment was the idea that the timely exchange of information across traditional organizational boundaries had a significantly beneficial impact on cost and cycle time reduction. This concept, depicted

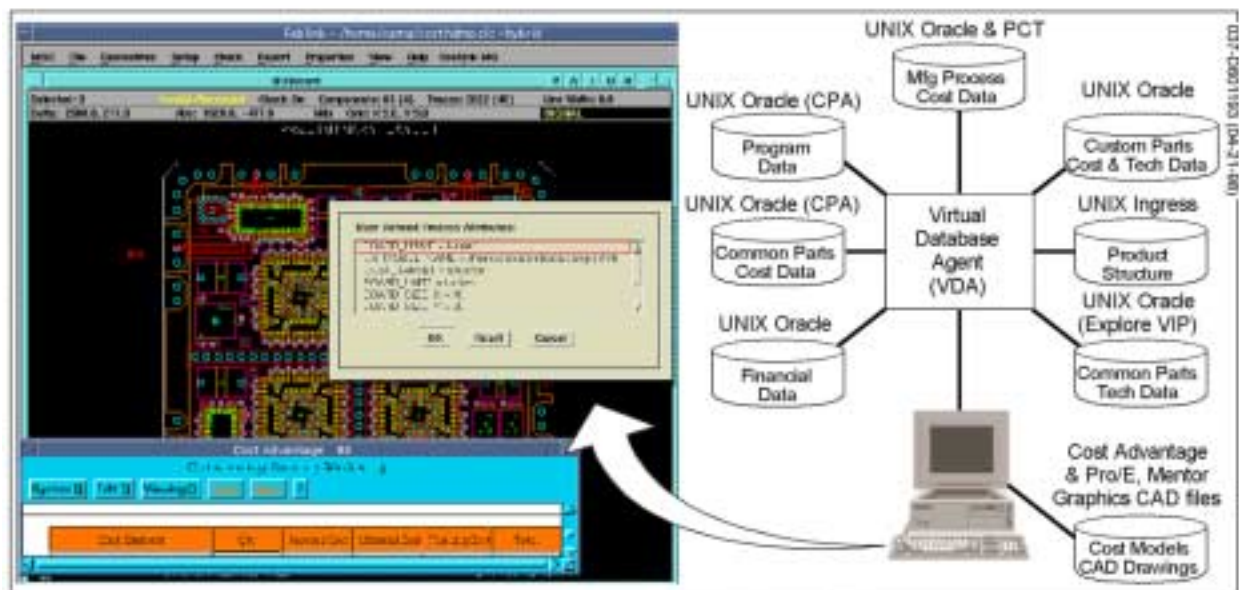


Figure 2-8. The JMD Integrated Cost/Design Environment Provides Data to the IPT in Near-Real Time

in Figure 2-8, was fundamental to the implementation of an integrated DTC environment.

Design and cost data needed to flow both across functional areas and within the IPT product hierarchy. The application systems and tool sets supporting the IPT in DTC had to respond to the IPT demand to provide the appropriate cross-functional data at the appropriate time to all IPT members. The functional applications and their respective data could no longer exist in isolation. The IPT environment demanded an application environment that was seamless, integrated and near-real-time in its responsiveness to enable the needed information exchange.

To convert these Lean Aerospace Initiative principles to a working implementation, a suite of models was assembled that leveraged the existing legacy systems at Raytheon and supplemented them with state-of-the-art software costing applications. A summary of the table defining these tools/applications is provided as Table 2-1. This table and the models are discussed in detail in Sections 3 and 6.

This seamless integration of design and cost information was the principal objective of the JMD program. The JMD program describes in detail the information technology architecture developed to achieve these goals in the document "IT Architecture: Integrated Design and Cost Data and Tools" listed in Table 1-1. The effort to upgrade this architecture to CORBA compliance is documented in Section 6 of this report.

9. Iterate detailed design and monitor progress towards goal

When taken in its most basic form, DTC is very simple and supports the CAIV

Table 2-1. Application/Tool Summary

Application/Tool	Supplier	Function
Product Information Manager (PIM)	Sherpa Corp.	Legacy Raytheon system for storage and configuration control on all engineering/product information
Explore (CIS)	Aspect Corp.	Legacy Raytheon system for database for component, material and process libraries
Pro/Engineer	Parametric Technology Corp.	Raytheon mechanical engineering CAD standard for 3-D Solid Model Design tool
Mentor Graphics	Mentor Graphics Corp.	Raytheon electrical engineering CAD standard for Schematic, Board & Hybrid Design tool
Cost Advantage	Cognition Corp.	Creates process models and Kbases, extracts features from Pro/E file, annotates/maps features from Pro/E solid models
HACOST	Raytheon/Galorath Associates	Standard design to cost tool for IPTs
Process Characterization Tool	Raytheon	Set of commercial software to gather, characterize and reduce process data for use by design IPTs
TCP/IP Network Protocol	SUN	Standard communications protocol of Raytheon's infrastructure
PDM Graphical User Interface (PGUI)	Black & White Software	Legacy Raytheon system for user interface
JMD Graphical User Interface (JGUI)	IONA	New user interface optimized for Lean Aerospace Initiative activities

methodology, concentrating only on meeting well-defined cost objectives. In the real world, many requirements must be satisfied simultaneously, requiring

thorough analysis and tradeoff among viable alternatives in order to achieve a balanced affordable design. Product development usually requires the concurrent development of key subproducts that are crucial to product affordability and/or functionality. Indeed, most complex products are made up of assemblies that are, in turn, composed of multiple subassemblies themselves made up of still smaller subassemblies, and so on.

The iterative design process begins with a product requirement that includes cost as a major priority. The ensuing requirements flowdown and associated design process are merged into a continuous multi-tiered interactive process. This process seeks to optimize the entire product by allocating all requirements at all levels in the product hierarchy to produce Best Customer Value. The generic seven steps shown in Figure 2-9 summarize a single tier of the iterative decomposition of a product into its subproducts. This representation of the iterative development process is referred to as the “Seven Steps to an Affordable Design.” A product-tailored derivative of these steps is applied to every subproduct at each level in the hierarchy, beginning in the Pre-Concept phase and continuing throughout Detailed Design. For multi-tier decompositions, the seven steps are applied at each subproduct level until no further decomposition is needed, so that initial design approaches and assessments can be used to steer each level of requirements flowdown toward a product-global optimum. Cost target allocations are a crucial part of this flowdown sequence.

Multi-tier interaction continues until we either establish that the requirements can be met with acceptable risk or we agree upon a best-value solution with the customer and define revised requirements that *can* be met; this agreement can be a critical part of design balancing. Iterative flowdown is performed or reviewed/revised in each development phase *before* we issue the formal requirements appropriate to the given phase.

The design is complete when the customer/contractor team has accomplished the following:

1. Performed detailed cost, performance, supportability, and risk assessments that indicate that all final requirements will be met with levels of cost, schedule and technical risk acceptable to both the customer and the Company
2. Allocated all requirements to NDI items or specific custom components
3. Completed the detailed design of all custom components
4. Successfully modeled/prototyped custom components and assemblies that can drive cost, performance or schedule
5. Completed a thorough manufacturing plan defining the approach to the fabrication or procurement of all components and the assembly, integration and test of the product and each significant subproduct
6. Complied with all customer and Company requirements for ILS, support, review, documentation, verification, scheduling, warranty, etc.

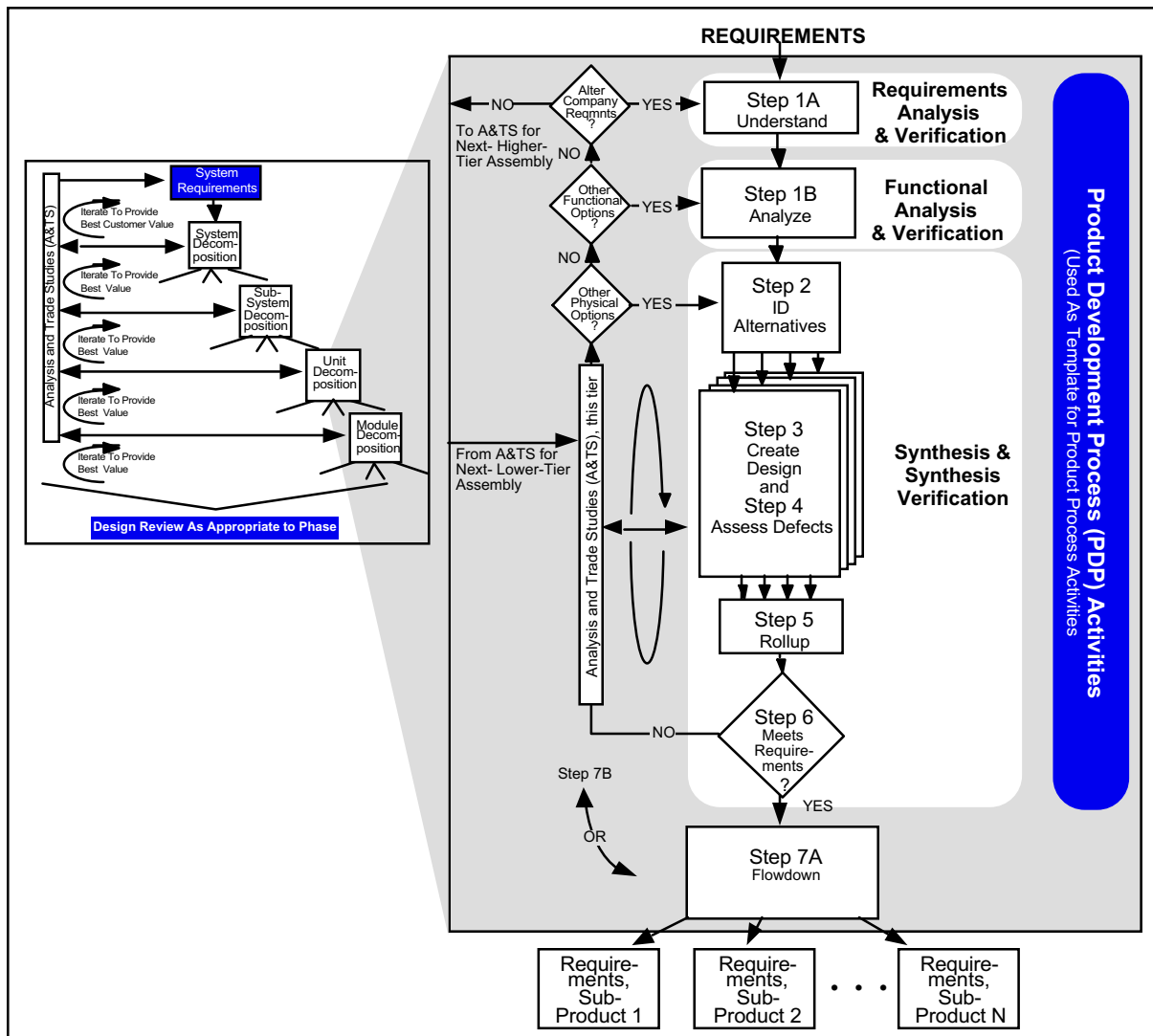


Figure 2-9. Seven Steps to an Affordable Design. This iterative process employs seven steps that are repeated at all levels in the product hierarchy.

10. Review, validate, and update integrated cost and design databases regularly

Associated with implementation of the JMD methodology, the enterprise must understand that the system requires some “maintenance” to ensure that the IPT is consistently making accurate cost projections. Databases that feed the system, such as purchased parts cost and technical parts data, must be updated with the most current pricing and technical information. Since the JMD process approaches real time, the maintenance of the support databases must also. The knowledge bases used in conjunction with Cost Advantage and the Process Characterization Toolset must be maintained with the latest available technologies and process costs so that design options can be maximized. Maintenance would also include the validation of cost projections versus actual costs experienced. The benefit of reduced product cost and cycle time afforded by the JMD system, increasing the overall competitiveness of the enterprise, outweighs the system maintenance cost.

3.0 RESULTS OF THE HDMP TILE SUBARRAY DEMONSTRATION

3.1 Introduction

This section of the JMD final report covers the full demonstration effort of Phase II. Phase I of the JMD program covered the development of cost estimation improvements, Activity-Based Management (ABM), and manufacturing process characterization improvements. That phase culminated with a mini-demonstration of the cost estimation models and supporting data. Phase II is a full demonstration utilizing the practices, processes, and data integration techniques developed in Phase I to cost out a prototype active array antenna. The goal was to show how cost estimation software could be used to document and quantify design/cost tradeoff reductions in cycle time and estimated production costs.

The active array antenna is being developed under the High Density Microwave Packaging (HDMP) Program. Raytheon is leading the development of active array radar that will revolutionize the state of the art in aircraft, space, missile and surface radar. Raytheon's HDMP program made a major step toward active array affordability by analyzing and improving the array assembly process that proceeds from the monolithic microwave integrated circuit (MMIC) chip level to the full array.

The purpose of the HDMP program was to develop a high density packaging approach suitable for modules containing a variety of device technologies operating at microwave frequencies. The approach selected fits diverse system needs with a focus on active array radar systems.

The Raytheon approach includes a truly innovative tile array packaging concept that we believe is the basis for the next generation active array. Our major effort focused on integrating and simplifying the module design and manufacturing process, facilitating data transfer and feedback among the various process steps and reducing overall cycle time and manufacturing costs.

The baseline design for the Raytheon HDMP Tile Module packages four T/R channels on three vertically stacked ceramic tile substrates. The seam-sealed package contains vertical interconnects captured in aluminum ring frames, which provide substrate-to-substrate interconnects. The tile packaging concept is referred to as "three-dimensional" because the substrates are stacked vertically on top of one another.

The tile packaging concept is a lower weight and lower cost alternative to the current "brick" architecture. The brick concept is referred to as a "two-dimensional" approach, since vertical stacking of substrates is not employed.

The JMD program began in August 1995 and was scheduled for completion in December 1998. However, because of federal budget constraints that were subsequently imposed, the program was descoped and funds were deobligated at the convenience of the customer in December 1997. Thus, the full demonstration results were never fully realized. Up to contract close, monthly subarray design/cost trades were performed to demonstrate the JMD Methodology and design/cost tradeoff

metric objectives were being realized and expected to meet program goals (See Figure 1-1).

This section provides an overview of Cognition Corporation's Cost Advantage™ (CA) software, used to map design features into manufacturing process steps and provide the resulting cost estimates. Each of the eight process models developed using the CA software and the Cognition-supplied Tool Design 6 (TD6) are described, along with a listing of the design trades the process model supports. (TD6 is a Cognition CA "off-the-shelf" process model used to cost analyze parts created using a single or combination of machining methods). The use of CA models to establish HDMP design cost trades is discussed, including what worked, what didn't work, and proposed changes in methodology.

Section 3.2 describes the array structure that was cost analyzed and the process utilization matrix. Section 3.3 provides an overview of the Cost Advantage process development software and describes the several process models that were developed. Section 3.4 covers the application of the JMD methodology to the design cost trades, monthly cost savings resulting from design/cost trades, and a discussion of the JMD methodology outcome. Section 3.5 contains the lessons learned and suggests areas of improvement. Section 3.6 is conclusions and recommendations.

3.2 Process Development

To estimate the cost of a prototype active array antenna, cost estimation methods and techniques were required that would include all aspects of the design, beginning with components and subassemblies and ending with the final assembly. Ideally, it should be possible to utilize a cost estimating system that would readily permit the determination of initial costs and the effect of modifications at the component or subassembly level. The decision had been made to use Cost Advantage as the basic cost estimation tool and it appeared that the Cost Advantage software could meet this criteria. However, it was necessary to make actual cost estimates and determine the effects of modifications to original inputs before the extent of program flexibility could be ascertained.

To accomplish this, a cost analysis of the active array developmental antenna design depicted in Figure 3-1 was completed. This design involved the fabrication and assembly of tile configuration Transmit/Receive (T/R) bottom surface of the cold plate. The T/R modules consisted of multilayer substrate assemblies separated by aluminum ring frames that were sealed in kovar housings; the cold plate, circulator assemblies, radiator assemblies, and shear plate were made from aluminum; the RF feeds and DC PWBs were made from copper-clad duroid and copper-clad polyimide respectively. This assembly provided an antenna test unit that could be used to obtain functioning parameter data. (The nomenclature "tile configuration" emphasized that the T/R modules were reminiscent of small pieces of tile that could be systematically arranged to form the array structure.) The design incorporated an array of 16 modules, with each module consisting of four radiating elements.

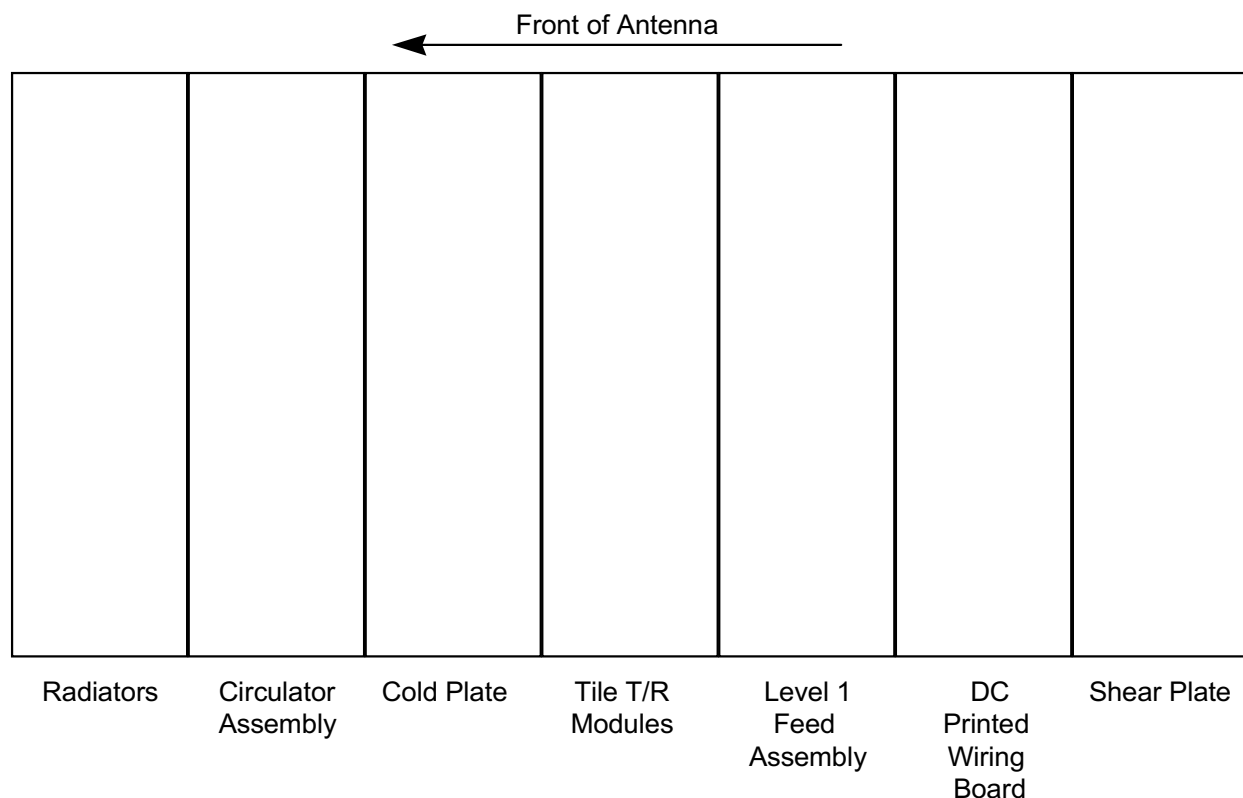


Figure 3-1. Generic Cross-Sectional View of the Prototype Tile Array Assembly Components.

The costs of fabricating this antenna were determined by developing models for each component based on Cognition's Cost Advantage software. (Generic drawing is not to scale.)

The assembly sequence, the process models, and where these process models were used are illustrated in Figure 3-2. To cost out the entire assembly, starting with the construction of the module assembly, nine different process models were needed. Of these only one, Tool Design 6 (TD6), was available from Cognition; the remaining eight were developed by Raytheon to handle unique component and assembly requirements. TD6 proved to be very useful when determining machining costs, such as those associated with the fabrication of the cold plate, radiator sections, and shear plate.

To fabricate and assemble the T/R modules, three process modules were written: one to cover fabrication of the multilayer substrates, another dealing with assembly concerns such as mounting devices on the substrates, and the third detailing assembly and test of these substrate subassemblies in the kovar housings to complete the final T/R units.

Two models, PWB fabrication and PWB assembly, were prepared that could be used to describe the fabrication and assembly of the RF feeds, radiator circuits and DC control boards.

One model, RF fabrication, was used to cover the details of mounting the radiator circuits onto the radiator base and placing a cover over the subassembly to complete the radiator strip assembly. This process model also covers RF feeds.

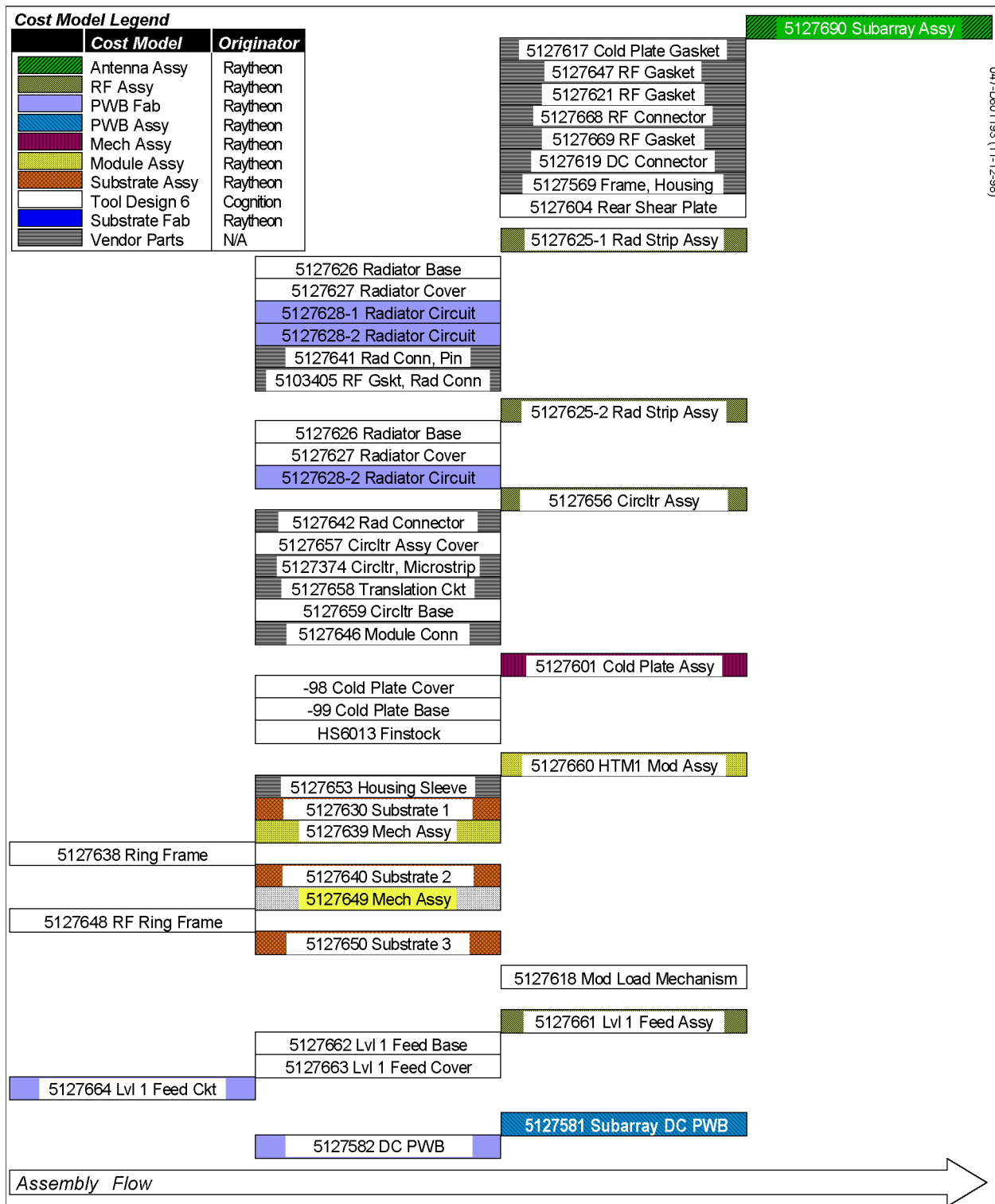


Figure 3-2. Tile Array Cost Model Utilization Matrix. This illustrates the component fabrication and antenna assembly sequence. Vendor costs were used in lieu of substrate fabrication process model cost estimates.

Costs of the antenna assembly were determined by accounting for the integration and test costs of the DC PWB assembly, Level 1 feed assembly, cold plate assembly,

circulator assembly and radiator strip assemblies using the subarray assembly model.

The assembly flow of Figure 3-2 begins on the left side of the page and ends on the right. The various process models and supplier parts are color-coded to identify where and when they were used.

3.3 Cost Advantage Software

Cost Advantage (CA) is Cognition Corporation's rules-based system for cost and producibility analysis. It captures manufacturing process knowledge and leverages that knowledge to provide design guidance and decision support information for cost identification and reduction. The model allows design and manufacturing engineers to prepare cost notes that can be used to analyze parts and assemblies and make optimal tradeoffs between individual component costs and the cost associated with assembling those parts. It helps engineers understand the cost impacts of their decisions.

The CA software system consists of four sequential components. These components are used as the framework for the development of process models and extend to an analysis of designs and the storage of these results.

1. The CA Modeler component is used to build or modify existing process models. The models are developed by the model builder in conjunction with a manufacturing specialist.
2. The process model stores the knowledge of the manufacturing specialists in the form of time and cost estimates for assembly and fabrication operations.
3. Cost Advantage itself analyzes designs using the data input into the process model and generates estimates from the component, fabrication, and assembly information input into the process model. This is the component that the design engineers use to perform cost estimates and design/cost trades.
4. The cost note stores the cost and part information and is used to generate cost reports.

The CA Modeler component is used to develop process models. These are initiated starting with the Modeler window (Figure 3-3), which is used to define the desired Process, Material, and Feature aspects. For example, when detailing a hole drilling operation in a part, the Process callout would include a component that is machined, the alloy being drilled would be designated by Material, and the drilling operation noted by Feature. The Context window (which is opened from the Modeler window) shown in Figure 3-4 is used to open editor windows in which the costs, characteristics, restrictions, rules, tables and help pages are separately defined. The model can contain "hard" restrictions that limit the choices a designer may have or "soft" restrictions that advise the user of cost penalties associated with certain design choices. Commonly used information and data can be accessed by the model from external databases or from tables stored in the model itself.

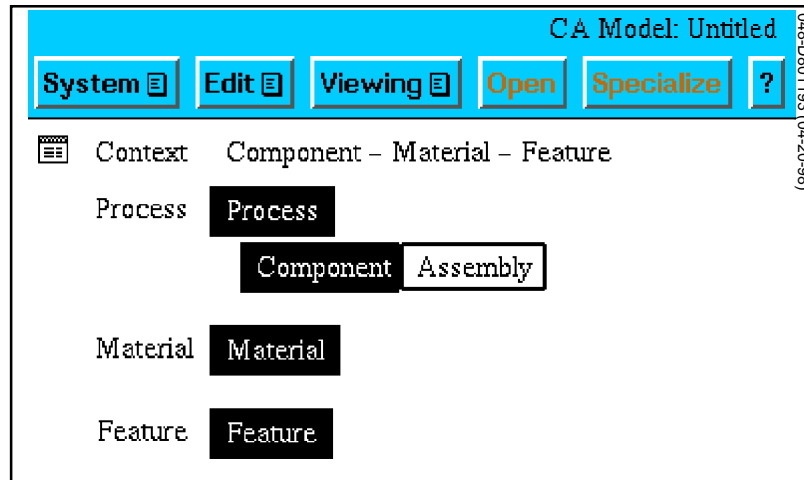


Figure 3-3. CA Modeler Window is used to define the desired process, material, and feature aspects within the process model.

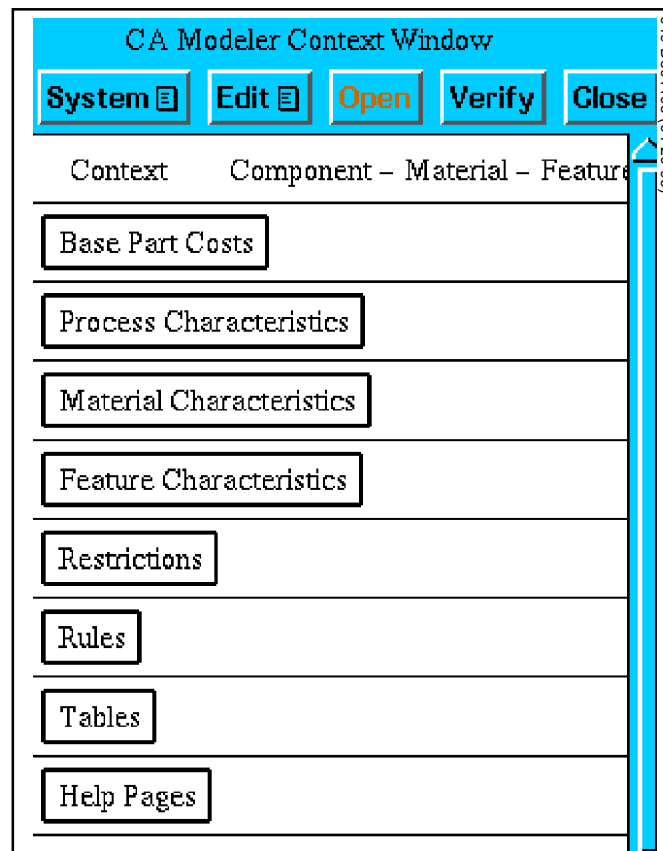


Figure 3-4. CA Modeler Context Window defines the formulas, values, and storage of data in the model that is used to provide cost estimates.

By repeating this approach, i.e., defining the particular Process, Material and Feature aspects in the Modeler window (which defines the Context level) and then detailing the process variables in the appropriate editor windows, the knowledge

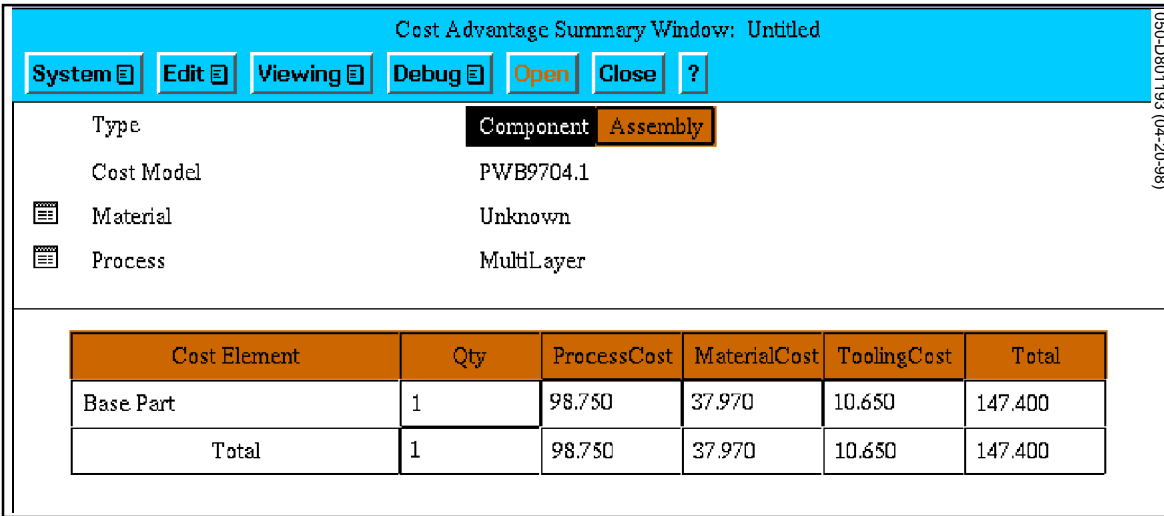
details of numerous common processes can be added to complete a model.

Once manufacturing and cost knowledge is captured in the process models, the models can be used to determine costs associated with the use of various components and assemblies. The user then executes a CA cost session that permits costs to be estimated using the same common consistent methods and data. This can be used to provide preliminary cost estimates early in the design cycle, identify key cost drivers before they impact manufacturing, investigate alternative materials or production methods, produce make-or-buy decisions, and leverage supplier negotiations with comprehensive cost analysis reports.

The process models provide time and cost estimates for assembly and fabrication operations such as handling, orientation, insertion, fastening, inspection, interconnection, drilling, machining, plating, and so on. Time estimates are based on the results of standard time and motion studies previously made to determine the labor required to perform assembly and fabrication tasks. The model uses these time estimates to calculate process, material, and tooling costs, which are then displayed in summary windows (see Figure 3-5) and can be printed out in cost reports.

The cost and part information for a component or assembly that has been tabulated using CA can be stored as notes for future reference and modification. These Cost Notes can be accessed and used in subsequent assemblies, thereby eliminating the need to recalculate previous component or assembly cost analyses.

During the course of the JMD contract, five releases of the CA software were used, starting with release 1.6 and concluding with 1.9.2. Each release was an enhancement



The screenshot shows the 'Cost Advantage Summary Window: Untitled'. It has a menu bar with 'System', 'Edit', 'Viewing', 'Debug', 'Open', 'Close', and '?'. Below the menu bar, there are fields for 'Type' (Component/Assembly), 'Cost Model' (PWB9704.1), 'Material' (Unknown), and 'Process' (MultiLayer). At the bottom, there is a table with 6 columns: Cost Element, Qty, ProcessCost, MaterialCost, ToolingCost, and Total. The table has two rows: 'Base Part' and 'Total'.

Cost Element	Qty	ProcessCost	MaterialCost	ToolingCost	Total
Base Part	1	98.750	37.970	10.650	147.400
Total	1	98.750	37.970	10.650	147.400

Figure 3-5. CA Cost Summary Window provides a breakdown of the process, material, and tooling costs associated with the fabrication of a component or the assembly of a unit.

to a previous version. At this writing, revision 2.0 is pending release. The Cognition annotator software used is release 1.1 and the Costlink-PE software is release 2.2. These latter programs were used when transferring Pro/E design features to a CA

model. Table 3-1 lists the JMD tools and their functions.

3.3.1 Process Models

It was described previously how process models, either off-the-shelf or user-defined, can be used to develop costs of assembled and fabricated items. For this study, user-defined process models were developed for multi-layer ceramic substrate fabrication, substrate assembly, module assembly, PWB fabrication, PWB assembly, mechanical assembly, RF assembly, and array assembly. Off-the-shelf process model TD6 was utilized to estimate machining process costs.

A description of each of the process models used in the program follows. As an example, a detailed overview of the approach used when preparing a process model is provided by the multilayer substrate fabrication description in Section 3.3.1.1. The same method was followed when preparing the eight user-defined process models. An extensive step-by-step explanation of the model development format is provided in reference [1].

3.3.1.1 Multi-layer Ceramic Substrate Fabrication. The design/cost trade options of the multilayer ceramic substrate fabrication process model are made by utilizing the CA process and material windows revealed when the model is opened (Figures 3-6 and 3-7, respectively). These, along with the Summary window (Figure 3-5), are the only windows the operator sees when analyzing a design.

Table 3-1. JMD Tool Selection

Application	Tool	Supplier	Version	Function
Cost Advantage	Cost Modeler	Cognition Corp.	1.6–1.9.2	Create process models and Kbases
	CA Process Models/Notes Annotator	“	---	Kbase for cost estimating
		“	1.1	Annotates/maps features from Pro/E solid models
	Costlink	“	2.2	Extracts features from Pro/E file
Pro/Engineer	Detail, ADV, and others	Parametric Technology Corp.	17–18	3-D solid model design tool
DSS		Management Support Technology	1.0	Extracts/annotates features from Pro/E solid models
Mentor Graphics	Design Manager	Mentor Graphics Corp.	8.5_2.2	Idea, Board & Hybrid Station design tools
Cost Advantage	Costlink MG	Cognition Corp.	1.1	Extracts features/data from Mentor Graphics file

Process	MLC			051-D801193 (04-20-98)
	BrazeSubstrate PlainSubstrate			
LotSize		25.0	[...]	
FabLaborRate	Σ	0.02	\$/sec [...]	
AssyLabRate	Σ	0.02	\$/sec [...]	
XYTolerance		.002 .010	in	
ThickTolerance		.001 .005 .010		
LaserDrill	Σ	Yes No		
PostFireMetal	Σ	Thick Thin		

Figure 3-6. Multilayer Ceramic Substrate Fabrication Process Model Window. These data values are specified by the user and are used to determine the processing costs listed in the model summary window.

Material	Ceramic			1052-D801193 (1-1-98)
xdim		1.2		
ydim		1.2		
LayerThick	Σ	0.01		
AveViasPerLayer	Σ	150.0		
NoOfLayers	Σ	10.0		
LidCost	Σ	15.0	\$	
NumberOfPins	Σ	25.0		

Figure 3-7. Multilayer Ceramic Substrate Fabrication Material Model Window. These data values are specified by the user and are used to determine the material costs listed in the model summary window.

The CA process window is shown in Figure 3-6. Lot size can be itemized and selections made from a list of options defining XY tolerance, thickness tolerance, if laser drilling will be done, and whether the post fire metallization is thick or thin film. The fabrication labor rate and assembly labor rate used in the cost calculation are displayed for the operators reference.

The CA material window is shown in Figure 3-7. The operator enters the x- and y-direction dimensions, the individual layer thicknesses, average number of vias per layer,

the number of layers that are to be laminated, the lid cost, and number of pins.

The Summary table at the top of Figure 3-8 itemizes the process, material, and tooling costs that occur when the cost analysis is run. In this instance, a process cost of \$168 has been incurred. The user can access additional detail cost information (which is shown in the Explain windows of Figure 3-8) by using the CA Explain feature. This feature provides an explanation of an object you choose, such as a command, a symbol, or a calculated value. All standard windows in CA have the “?” button, the rightmost of the menus and buttons. To use the Explain feature, choose the “?” button (the cursor becomes a ?), click on any object in a window or menu and CA displays an explain window for the chosen object. Figure 3-8 shows the explain window for the calculated value of the process cost.

The windows shown in Figures 3-6 through 3-8 can be accessed by the user when making a cost estimate. A brief explanation of how a cost model is developed by the model builder and how editor windows are used to input detailed process information is described below. This information can be input only during model development and can be accessed or modified only by the model builder. The information contained within the process model cannot be modified by anyone using a model to do a cost analysis.

The CA Modeler component used to develop this model followed the approach mentioned previously (and shown in Figures 3-3, 3-4, and 3-5): the Modeler window was used to define a particular context, the Context window was opened to expose the list of Editor windows (Figure 3-4), and selected windows were accessed so that detailed descriptions and formulas dealing with processing variables could be added.

An example of how the process model calculates costs using the parameter settings mentioned in Figure 3-6 and 3-7 is shown in Figures 3-9 through 3-11. The ceramic brazed substrate summary table showing the total process costs is shown at the top of Figure 3-9, with the process cost editor window (of the process model) at the bottom of the figure illustrating those processing factors—layer fabrication time, lamination time, saw cut time, grind time, laser drill time, and post fire metallization time—entering into process cost calculations. The total time associated with these factors is then simply multiplied by the fabrication labor rate to arrive at the process cost of \$168.

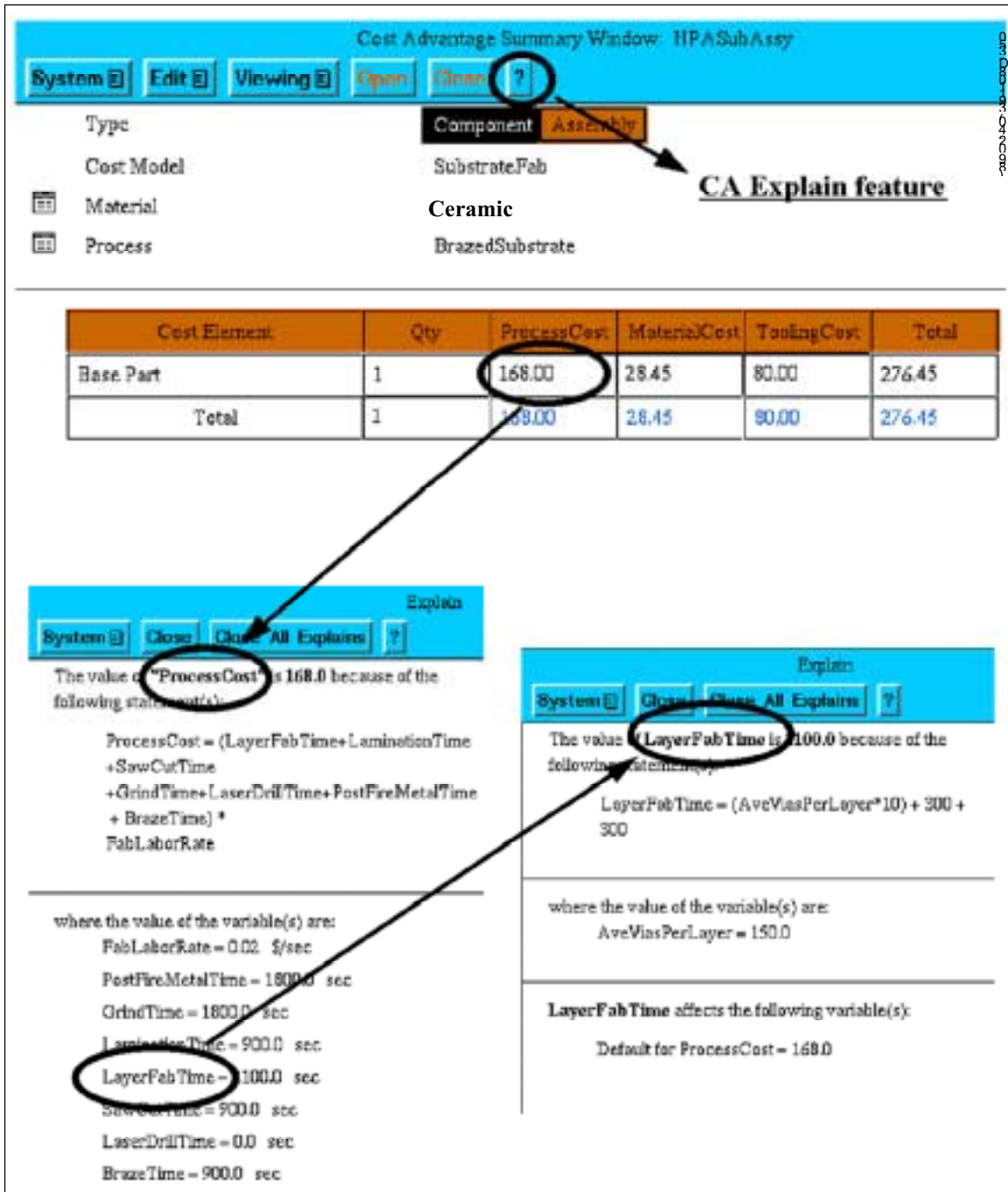


Figure 3-8. Link Between the Summary Table and the CA Explain Feature. Using this feature, the assumptions that were used to determine process, material, or tooling costs can be accessed by the user. The user selects the "?" button (the cursor becomes a ?) and places the "?" over the area where more information is required. In the example above, this area is identified by the circles.

C:\Program Files\Oracle\OPN1000\

Type
 Cost Model
 Material
 Process

Component Assembly
 SubstrateFab
 Ceramic
 BrazeSubstrate

Cost Element	Qty	ProcessCost	MaterialCost	ToolingCost	Total
Base Part	1	168.00	28.45	80.00	276.45
Total	1	168.00	28.45	80.00	276.45

Type numeric
Entry Type type in
Units
Type of default value

none constant equation table

Equation

1) ProcessCost (LayerFabTime+ LaminationTime+ SawCutTime + GrindTime+ LaserDrillTime+ PostFireMetalTime) * FabLaborRate

Explanation text

Explanation graphics

Associated Help Pages

Notes

Figure 3-9. Relationship Between Process Cost Listed in the Summary Table and How It is Defined in Editor Window. The times for the various factors are added together and multiplied by the labor rate to establish process cost.

The individual factors entering into the process cost calculations and the labor rate are completely specified in other editor windows. The links in this relationship is demonstrated in Figure 3-10. Within the specified context (MLC-Ceramic-Feature), each of the factors mentioned earlier were defined. In the examples of Figure 3-10, there is a window for SawCutTime that defines the equation for that parameter, and another window for FabLaborRate in which the rate is specified. Grind time, which also entered into the process cost calculation mentioned in Figure 3-9, is defined as shown in Figure 3-11; this would also be linked to process cost in the same fashion, as previously described.

Type	numeric
Entry Type	Type in
Units	
Type of default value	none constant equation table
Equation	1) $\text{Process Cost} = (\text{LayerFabTime} + \text{LaminationTime} + \text{SawCutTime} + \text{GrindTime} + \text{LaserDrillTime} + \text{PostFireMetalTime}) \times \text{FabLaborRate}$
Explanation text	
Explanation graphics	
Associated Help Pages	
Notes	

Context	MLC - Ceramic - Feature
Name	FabLaborRate
Type	Numeric text logical
Entry Type	Type in one of n many of n table
Lower bound	
Upper bound	
Units	\$/sec
Type of default value	none constant equation table
Value	.02
Displayed	yes no conditionally
Explanation text	This labor rate equates to 72.00 \$/hr.
Explanation graphics	
Associated Help Pages	
Notes	

Context	MLC - Ceramic - Feature
Name	SawCutTime
Type	numeric text logical
Entry Type	type in one of n many of n table
Lower bound	
Upper bound	
Units	sec
Type of default value	none constant equation table
Equation	1 $\text{SawCutTime} = (900 \text{ when } XY\text{Tolerance} == .002; 300 \text{ otherwise})$
Displayed	yes no conditionally
Explanation text	When tolerance is .002 part must be fired prior to trimming to size. Since the fired part is more difficult to cut than the green part, the cost is greater.
Explanation graphics	
Associated Help Pages	
Notes	

Figure 3-10. Relationship Between Process Cost Editor Window and Editor Windows Defining Those Factors Influencing Process Cost. This illustration shows the equation defining saw cut time in one window and the fixed labor rate contained in the other.

Using the same approach used when determining the process costs shown in the Summary table of Figure 3-9, the material costs were calculated using data that was input to the Material window of Figure 3-7. As shown in the editor window of Figure 3-12, material cost is determined by multiplying ceramic cost, x-dimension, y-dimension, layer thickness, and number of layers. Each of these factors was defined in the appropriate editor windows that were accessed by the material cost formula. These editor windows defining x-dimension, y-dimension, layer thickness, and number of layers used the designated values that were manually input into the Material window to calculate the data subsequently fed to the material cost editor window.

056-D801193 (11-12-98)

Context	MLC – Ceramic – Feature		
Name	GrindTime		
Type	☒ numeric	☐ text	☐ logical
Entry Type	☒ type in	☐ one of n	☐ many of n
Lower bound			
Upper bound			
Units	sec		
Type of default value	☐ none	☐ constant	☒ equation
Equation			
1)	GrindTime = {1800 when ThickTolerance == .001 ; 900 when ThickTolerance == .005 ; 450 otherwise}		
Displayed	☒ yes	☐ no	☐ conditionally
Explanation text			
Explanation graphics			
Associated Help Pages			
Notes			

Figure 3-11. Editor Window Defining Grind Time. This factor was one of those taken into consideration when calculating Process Cost, which is defined in Figure 3-8.

The tooling costs shown in the Summary window of Figure 3-9 were calculated in the same fashion. The tooling cost editor window is shown in Figure 3-13. In this instance each of the various cost items—layer fabrication tooling, grind tooling, lamination tooling and laser tooling—were defined in separate editor windows. Lot size was as defined by the user in the Material window when the CA model was opened.

An example of tabulated data that is stored in a process model that could be accessed using a call function is shown in Figure 3-14. In this case, assembly rates and factors are listed, and these factors can be used in an equation described in another editor window by calling up the table by name and then selecting the column and row cell containing the particular value. Tables are particularly useful, since a large quantity of data can be located in one spot and the equations using that information scattered about various editor windows. This limits the number of additional editor windows that otherwise need to be developed, and greatly simplifies the modification of data, since only one table needs to be accessed to implement changes to many data points.

057-D801193 (11-12-98)

Context	MLC – Ceramic – Feature		
Name	MaterialCost		
Status	☐ inherited	☒ overridden	
Overridden context	MLC – Ceramic – Feature		
Type	numeric		
Entry Type	type in		
Units			
Type of default value	☐ none	☐ constant	☒ equation
Equation			
	1)	MaterialCost=CeramicCost*xdim*ydim*LayerThick*NoOfLayers	
Explanation text			
Explanation graphics			
Associated Help Pages			
Notes			

Figure 3-12. Material Cost Editor Window. This is the material cost shown in the summary table of Figure 3-8. The equation used the ceramic cost defined in a subsequent editor window and layer parameters that were input to the Material window of Figure 3-7.

There are three other Editor windows that were not utilized when developing the Substrate Fabrication model: Restrictions, Rules, and Help Pages. These would have been implemented as the process model became more complex and it was necessary to avoid situations that were not practical or detailed notes were needed to maintain historical continuity. For example, the maximum number of laminated layers permitted could be stated.

Restrictions define limitations. An example could be restrictions on the thickness of substrate material that could be via punched. Rules cover tests that are made which affect cost characteristics. In this case, the cost impact of producing tight tolerance vias might be addressed. The Help Pages windows are used to provide information that promotes the use of good design practices by the model users.

058-D801193 (11-12-98)

Context	MLC – Ceramic – Feature		
Name	ToolingCost		
Status	☐ inherited ☒ overridden		
Overridden context	MLC – Ceramic – Feature		
Type	numeric		
Entry Type	type in		
Units			
Type of default value	☐ none ☐ constant ☒ equation ☐ table		
Equation			
1)	$\text{ToolingCost} = (\text{LayerFabToolingCost} + \text{GrindToolCost} + \text{LaminationToolCost} + \text{LaserToolCost}) / \text{LotSize}$		
Explanation text			
Explanation graphics			
Associated Help Pages			
Notes			

Figure 3-13. Tooling Cost Editor Window. This tooling cost is shown in the summary table of Figure 3-8. The cost factors in the equation were defined in other editor windows; the lot size was defined in the Process window of Figure 3-6.

3.3.1.2 Substrate Assembly. The Substrate Assembly process model is used to price out part and assembly costs when mounting various electrical components on a substrate. The model features the assembly of passive parts, active devices (face-up or as flip chip), and wire bond interconnections. The user selects the feature part and inputs the component part number and quantity of wire bonds for face-up chips. The process cost, part cost, and the associated tooling costs for each part are shown in the summary window. If the quantity of a specific part is more than one, that quantity is changed in the summary window and the corresponding costs are updated (Figure 3-15).

Context MLC – Material – Feature

Name RatesFactorsData

Number of columns 2

Number of rows 6

05
9-
D
80
11
93
(1
1-
12

"Item"	"Rate"
"AssemblyRate"	.02
"TestRate"	.02
"AssyEffFactor"	1
"RFIndex"	1
"Ceramic"	100

Explanation text

Explanation graphics

Associated Help Pages

The assembly and test rate are in \$/second. The assembly efficiency factor is 1 (100%), as is the RF Index (Realization factor index).

Figure 3-14. A Table Within the Multilayer Ceramic Substrate Process Model contains process model common data, which is accessed using the lookup function call in the model formulas.

3.3.1.3 Module Assembly. The Module Assembly process model is used to price out the assembly of various substrate assemblies, ring frame assemblies, and housings to complete the module build. The user selects substrate assembly, ring frame, and housing parts from the cost note library. The build lot size is input and the appropriate assembly feature is selected for each of the library parts. The model returns the process cost, part cost, and the associated tooling costs in the summary window for the completed module, as well as library part costs and the cost to assemble the library parts. If more than one part is required, the quantity change is made in the summary window and the corresponding costs are automatically updated (Figure 3-16).

Cost Model		SubstrateAssy				
 Process	Substrate					
	Cost Element	Qty	ProcessCost	MaterialCost	ToolingCost	Total
	Assembly Costs	1	18.10		9.50	27.60
	Assembly of SubAssy 3	1	.19			.19
	Assembly of RF Cap	32	5.99	10.40		16.39
	Assembly of MMIC A	8	11.00	336.50		347.50
	Assembly of MMIC B	4	5.45	22.00		27.45
	Assembly of Alignment Structure	3	.56	.19		.75
	Parts					
>>	SubAssy 3	1	42.00	7.12	20.00	69.12
	RF Cap	32				0.00
	MMIC A	8				0.00
	MMIC B	4				0.00
	Alignment Structure	3				0.00
	Total	1	83.30	376.21	29.50	489.00

Figure 3-15. Substrate Assembly Process Model Summary Window. The cost to assemble a substrate is itemized for both component and assembly levels. This Tile Assembly cost is subsequently included in the Module Summary Window of Figure 3-16. (The ? at the left of Assy of Alignment Structure indicates that this needs further definition.) Data shown is for explanatory purposes only.

3.3.1.4 Printed Wiring Board Fabrication. The Printed Wiring Board Fabrication process model provides cost estimates for single-sided, two-sided and multilayer PWBs. Information is input into the model regarding the quantity, physical size, material type, and number of layers the PWB contains. Based on the input criteria, the optimal quantity of PWBs that would be run on a multi-up panel and the number of panels processed to complete the order quantity are calculated. The associated unit and tooling costs are displayed in the process window.

A variety of material and process cost trade options are available to the user. These are shown in Figures 3-17 and 3-18 and permit selecting between board materials; specifying the number of layers, copper thickness, number of controlled impedance layers, and line size; solder masking, silk-screening, electrical test costs, backside sealing, gold plating; heatsink requirements and bonding; and the types and quantities of terminals or pins.

	Cost Element	Qty	ProcessCost	MaterialCost	ToolingCost	Total
	Assembly Costs	1	103.20		8.00	111.20
	Assembly of SubAssy 1	1	.08			.08
	Assembly of Housing sleeve	1	.08			.08
	Assembly of LRFAssy	1	.08			.08
	Assembly of SubAssy 2	1	.08			.08
	Assembly of URFAssy	1	.08			.08
	Assembly of SubAssy 3	1	.08			.08
	Parts					
>>	SubAssy 1	1	100.03	242.28	51.50	393.81
	Housing sleeve	1	37.50	.25	4.31	42.06
>>	LRFAssy	1	35.70	18.57	1.25	55.52
>>	SubAssy 2	1	72.18	160.95	27.00	260.13
>>	URFAssy	1	36.63	32.55	1.25	70.43
>>	SubAssy 3	1	83.20	376.25	29.50	488.95
	Total	1	468.92	830.85	122.81	1422.58

Figure 3-16. Module Assembly Process Model Summary Window. Library parts and other feature part costs are shown in the bottom half of the table. (In the parts cost section, library parts are denoted with the » symbol and new parts by the small “table” symbol). The cost to assemble each corresponding part is shown in the top half of the table. Note Sub Assembly 3 which was totaled in Figure 3-15. Data shown is for explanatory purposes only.

Costs are shown in the CA Cost Summary Window for each of the combinations selected. This allows the designer to identify major cost drivers and make appropriate design choices prior to manufacturing.

3.3.1.5 PWB Assembly. The PWB Assembly process model provides cost estimates for surface mounted assemblies of the sort typically found in a tile subarray. Model information is input regarding the type and quantity of components, assembly production quantity, conformal coating, and electrical testing. Part numbers are entered for each component type and the model then searches data bases to find part costs. The associated processing cost is displayed for each type of component and quantity selected. Design trades are made by changes to the type of component and part number, quantity of parts, conformal coat needs, and final electrical test. Generally, PWB assembly involves the mounting of passive components on a surface. An example of the costs incurred for a typical tile subarray populated PWB assembly is shown in Figure 3-19.

BoardType	Σ	Half-VME	Full-VME	UserDefined
Length	Σ	8.0	Inches	[...]
Width	Σ	6.0	Inches	[...]
NoOfLayers		10.0	Number	[...]
NoOfContrldImpedenceLayers	Σ	0.0	Number	[...]
BoardMaterial		FR4_GF	PolyImide_GI	G10 Duroid
HalfOzLayers		10.0	Number	[...]
OneOzLayers		0.0	Number	[...]
TwoOzLayers	Σ	0.0	Number	[...]
ThreeOzLayers	Σ	0.0	Number	[...]
GoldPlateRequired?	Σ	No	Yes	
ContactsOrSurface?	Σ	Contacts	Surface	
GoldThickness	Σ	50	100 150	MicroInches
HeatSinkRequired?	Σ	No	Yes	
EtchedOrMachined?		Etched	Machined	
BondMethod		Adhesive	Prepreg	
TerminalsOrPins?	Σ	No	Yes	

Figure 3-17. PWB Material Selection Window. Different design/cost trades are possible using various material options.

Process	TwoSided	MultiLayer	SAM
TotalProduction			100.0 Units [...]
LaborKey	Σ		12X7C95
LaborRate	Σ		1.008 [...]
Vendor	Σ		Hughes Vendor2 Vendor3
MinLineWidth	Σ		0.01 Inches [.004...]
SilkScreen?	☒		No Yes
SilkscreenSides	Σ		OneSide BothSides
SolderMask?	Σ		No Yes SMOC
ElectricalTest?	Σ		No Yes
BacksideSeal	Σ		No Yes

063-D801193 (04-20-98)

Figure 3-18. PWB Process Selection Window. Different process requirement selections are possible, thereby permitting various design/cost trades between several process options.

3.3.1.6 Mechanical Assembly. The Mechanical Assembly process model provides cost estimates for typical mechanical operations not covered in the CA Tool Design 6 “off-the-shelf” process model. Operations included are brazing, mechanical bonding, hardware selection and installation, and various platings and coatings. The model prices out three braze processes (vacuum, inert, and dip), mechanical bonding, typical mechanical hardware (keensert, helicoil, dowel pin, coolant fitting, pin insert), and five plating and coating processes (passivate, chem film, anodizing, cadmium plating, and gold plating). Cost trades can be performed between braze choices, including the complexity of the part, part size, braze material thickness/cladding, and if the assembly is to be pressure tested. Designers can compare the costs of various hardware installations and the different plating and coating processes. The resultant costs for each feature are shown in the CA Cost Summary Window.

Type

Component Assembly

Cost Model

PWA9705.2

Process

Printed Wiring Assembly

Cost Element	Qty	Process Cost	Material Cost	Tooling Cost	Total
Assembly Costs	1	15.69		25.00	40.70
Assembly of DC-5127562	1	1.26			1.26
Connector_92Pin	1	3.47	6.25	10.42	20.13
Connector_104Pin	1	3.47	6.25	10.42	20.13
Caps_TantalumA	224	19.58	140.00		159.58
Caps_TantalumB	24	2.10	15.00		17.10
Caps_CeramicA	32	2.80	50.00		52.80
Caps_CeramicB	32	2.80	50.00		52.80
Fuses_10A	128	11.19	48.00		59.18
Parts					
» DC-5127562	1	110.40	3.47	109.83	223.73
Total	1	172.75	319.00	155.68	647.50

06

4-

D

80

11

93

(1

1

16

Figure 3-19. Printed Wiring Assembly Process Model Summary Window. This illustration shows the cost of assembling connectors, capacitors and fuses to a PWB. Data shown is for explanatory purposes only.

3.3.1.7 RF Assembly. The RF Module Assembly Process model provides cost estimates for three different subassembly packaging configurations: brick (see Section 3.1), intermediate tile, and advanced tile. In each instance, information concerning supplier components, in-house designed parts, and assembly processes is added.

Various cost trades are possible:

1. Four structural attachment processes
 - Screws
 - Rivet screws
 - Bonding
 - Dry bonding
2. Torque or sealing of screws
3. Electrical test costs
4. Choice of interconnection style and quantity
 - Solder

- Wire bonding
- 5. Bonding material selections and how the bonding material is prepared
 - Laser trimming
 - Automated or manual cutting
- 6. Varying supplier item quantities

3.3.1.8 Machining – Tool Design 6 (TD6). TD6 is a CA process model used to provide cost and producibility analysis for the fabrication of parts using a variety of machining methods. The model was developed by Cognition in conjunction with an American jet engine manufacturer.

TD6 provides cost estimates when using different tools on various materials. The model selects the optimal available stock size based on the specified material, part type and dimensional requirements. Time estimates are generated based on standard milling, drilling, EDM and lathe machining times, as well as material characteristics. Time estimates are used to calculate process and material costs that are tabulated in the CA Summary window.

This is a very useful basic model, and modifications to the default time and cost values can be made to incorporate the characteristics of a particular shop, rather than those of the originator.

The model generates estimates from the component description provided by the user.

A sample view of the Modeler page of TD6 is shown in Figure 3-20. In this example, the process is component machining, the material is aluminum, and the hole drilling Feature option has been selected. The different hole drilling selections are shown, with one of these to be chosen. In this instance, a standard hole was desired, and the costs could be estimated after entering the information requested in the associated Material and Process windows of Figures 3-21 and 3-22.

Should the costs of other machining operations be desired, such as milling, the proper Modeler window would be accessed, from which the associated Material and Process windows would be presented.

CA Model: TD6costlink

System
Edit
Viewing
Open
Specialize
?

☰

Context

Machining – Aluminum – Std

Process

Process

Component

Assembly

Machining

Material

Material

Ferrous

Non_Ferrous

Polymers

Composites

Ceramics

Aluminum

Copper

Nickel

Feature

Feature

Hole

Mill_Operations

Lathe_Operations

Surface_Profile

Edge_Finish

Std

Cbore

Csunk

Threaded

CboreThreaded

CsunkThreaded

Figure 3-20. TD6 Modeler Window. The selected parameters in this instance are machining-aluminum-standard hole.

Cost Advantage Material Window

System
Edit
Close
?

Material

Ferrous

Non_Ferrous

Polymers

Composites

Ceramics

Aluminum

Copper

Nickel

Machinability	Σ	250.0	
CostPerUnit	Σ	3.0	\$/lb
X_Extents			
Y_Extents			
Z_Extents			
Material	Σ	"Aluminum"	

Figure 3-21. Material Window Associated with the Modeler Window of Figure 3-20. The X, Y and Z dimensions of the part to be machined are defined at this point. Since aluminum was added from the top material selection list, the machinability and cost per unit were automatically added from a table of values within the TD6 model.

Cost Advantage Process Window

System Edit Close ?

Process Machining

Part_Type	Σ	Bar Hex Square Sheet Tubing Bulk L_Angle C_Channel I_Beam	
Envelope_Length	Σ		inches [...]
Envelope_Width	Σ		Inches [...]
Envelope_Height	Σ		Inches [...]
Stock_Height	Σ		Inches [...]
Stock_Width	Σ		Inches [...]
Stock_Length	Σ		Inches [...]
Vendor_Name	Σ	Vendor1 Vendor2 Vendor3 Vendor4 Vendor5	
Machine_Type	Σ	Mill Lathe Drill Wire_EDM	
Part_Tolerance	Σ	0.02	Inches [.0001....1]
Surface_Finish	Σ	Under_16AA 16AA_to_32AA Over_32AA	
Face_Mill_Size	Σ	3.0	Inches [.125...3]
SetUp_Mill	Σ	Simple Average Complex	

Figure 3-22. Process Window Associated with the Modeler Window of Figure 3-20. The envelope dimensions are those from the X, Y and Z inputs to Figure 3-21; the corresponding initial stock dimensions are then automatically determined.

3.3.1.9 Array Assembly. The Array Assembly model was tailored for tile array assembly and developed at Raytheon so that the assembly costs incurred when putting the array together could be easily estimated. The features covered are shown in the Modeler window of Figure 3-23. These are the major components or subassemblies—Circulator Assembly, Cold Plate, T/R Module, Level 1 Feed, PWB Assembly, Shear Plate and Radiator—that are put together to form the array. Unique costs incurred when installing selected subassemblies are accounted for when these are selected during the assembly sequence. In this example, the radiator assembly is to be installed and the options of using either conventional screws or an alternate (rivscrews) is shown. The rivscrew option was introduced because a large number of small screws were being used to attach sections of the circulator. This was a labor intensive operation, and the time required could be considerably reduced by using rivscrews, which could be quickly installed using a hand-held automatic feeder. The screw/rivscrew option was also available when mounting the circulator assembly.

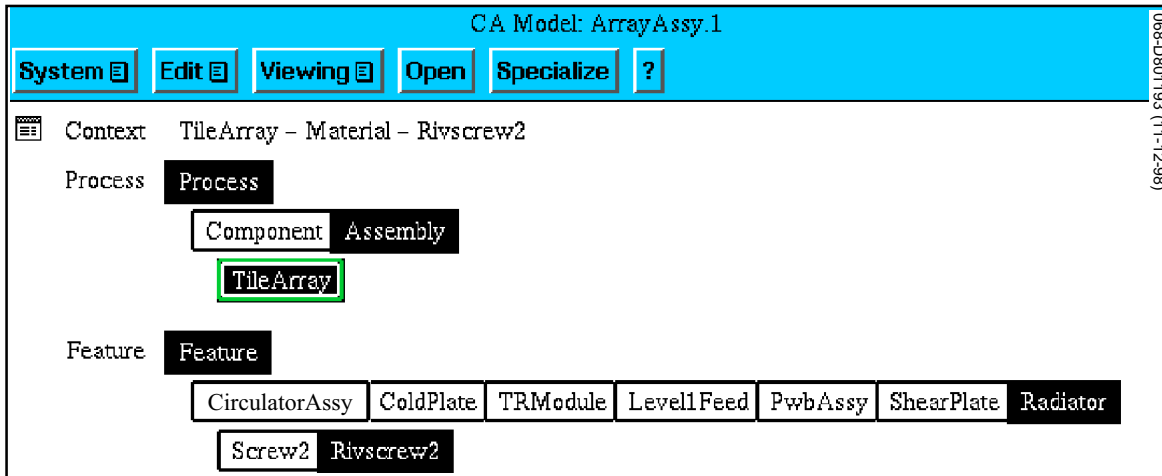


Figure 3-23. Array Assembly Modeler Window illustrates the selection of the radiator assembly installation.

This model includes items associated with producing an electrically functioning array and factors in the number of systems to be produced per month along with any additional equipment costs. This Process window is shown in Figure 3-24, in which concerns such as electrical test, fault isolation, burn-in, and fixture costs along with the labor costs incurred are identified.

The cost of an array assembly was determined by first selecting the appropriate Tile Array factors from those listed in Figure 3-24, then denoting the part to be included (such as the radiator assembly) and the mounting option (in this instance, either screw or rivscrew). It was then a simple matter to subsequently access the library for components or subassemblies and include their costs. In this example, the radiator assembly costs would be imported. If more than one radiator assembly were to be incorporated in the array, a change in the total parts was made to the resultant cost summary list, where the additional costs would be automatically calculated. The complete cost window for an array that was assembled using rivscrews for circulator and radiator attachment is provided in Figure 3-25. This provides a breakdown that separately lists the components or subassemblies and the installation costs of each.

3.3.2 Databases

Various databases can be accessed by the process models. Some of the databases are simple flat files developed for common and specific process model applications that are maintained in the UNIX model global directory. Other databases are in tables within a given process model. Typical information contained in the databases include part number and associated part cost, Process Characterization Tool Set (PCT) data, and labor rates and factors.

TileArray
069-D801193 (04-20-98)

System
Edit
Close
?

Process

TileArray

ElectricalTest?		No	Yes	
LaborRate	Σ	60.5		[...]
LaborKey	Σ	12X7C95		
FaultIsolation?		No	Yes	
BurnIn?		Yes	No	
SystemsPerMonth	Σ	1.0		numeric [0...]
AdditionalTestFixtureCost	Σ	0.0		\$ [...]

Figure 3-24. Tile Array Window. This incorporates electrical test considerations along with the number of systems to be produced and test fixture costs. The cost of test fixtures becomes a major item if more than one system per month is produced.

Databases are accessed using lookup functions contained within the process models. The information requested by the lookup function is selected from the appropriate database and returned to the process model. This data is then used to develop cost or process information. For a more detailed discussion of databases, their workings, and methods of accessing, see the Information Technology IT Architecture final report in Section 6.

3.4 Subarray Design Cost Trades

To determine the effect on array costs of various design modifications, monthly design/cost trades were completed utilizing the Cost Advantage models and using the initial HDMP tile array concept as the baseline. How the JMD methodology was applied is discussed below, along with a comparison of the calculated cost savings to the monthly cost savings goal. This is followed by citing what worked and what did not, along with suggested improvements.

3.4.1 Application of the System to HDMP Design Trades

Design information taken from HDMP drawings was used to determine process, material and tooling costs.

Cost Advantage Summary Window: 5127620SAassy192aRscrow					
System  Edit  Viewing  Debug  Open  Close  ? 					
Cost Element	Qty	ProcessCost	MaterialCost	ToolingCost	Total
Assembly Costs	1	1683.00		0.00	1683.00
 Assembly of 5127601ColdPlate	1	0.01			0.01
 Assembly of 5127618ModuleLoad	3	1.48			1.48
 Assembly of 5127661LevelFeed	3	12.02	1.80		13.82
 Assembly of DC_PWA_5127581	3	21.04	3.15		24.19
 Assembly of DC Connector	48	192.38	28.80		221.18
 Assembly of 5127604ShearPlate	1	7.01	1.05		8.06
 Assembly of 5127625RadiatorStrip	24	96.20	33.60		129.80
 Assembly of Radiator_assy_dummy	4	16.03	5.60		21.63
 Assembly of TileModuleAssy	48	23.76			23.76
 Assembly of 5127656CirculatorAssy	3	12.02	4.20		16.22
Parts					
>> 5127601ColdPlate	1	454.00	47.75	125.00	626.75
>> 5127618ModuleLoadMechanism	24	245.91	2.09	0.00	248.00
>> 5127661LevelFeedAssy	3	1555.85	27.15	150.00	1733.00
>> DC_PWA_5127581	3	518.25	957.00	467.00	1942.25
 DC Connector	48		180.00		180.00
>> 5127604ShearPlate	1	150.97	20.33	0.00	171.30
>> 5127625RadiatorStrip(real)	24	7348.75	84.00	857.25	8290.00
>> Radiator_assy_dummy	4	1287.20	11.72	116.08	1415.00
>> TileModuleAssy	48	27200.00	39950.00	7350.00	74500.00
>> 5127656CirculatorAssy	3	3285.00	15922.50	225.00	19432.50
Total	1	44110.88	47990.74	18580.33	110681.95

Figure 3-25. Total Costs Associated with the Assembly of a Tile Array Configuration. The top portion of the table lists the assembly costs, while the bottom section contains the costs of the parts making up the array. Data shown is for explanatory purposes only.

In preparing the cost estimates, the costs of lower-tier items were determined first and the appropriate cost notes developed. These were placed in a notes library and accessed when higher-tier item costs were determined. The process of calling up library parts as feature characteristics in the higher level drawing estimates worked well, with cost notes providing detail part cost data.

Options offered by the assembly process model included being able to select a feature, library part, or a new part. If the feature option was selected, then the cost for that feature was contained within the process model itself and the cost added to the total. If a library part was required, then a parts number list was presented and the costs for that part added to the new cost estimate. By double clicking on the sub-tier cost note, the Process Model for that note could be loaded and the note details made available for review. This process provided considerable cost design analysis flexibility.

It is also possible to modify a library cost note by revising the quantity, part number, or part cost. Configuration management is critical at this point. If the change results in a new cost note, then the “save as” feature works well. If the change simply updates an existing design, then the corresponding cost reports and higher level estimates must be updated accordingly in order for the new costs to roll up to the higher levels. The effectiveness of this is somewhat dependent on the tenacity of the user to carry through and update the other estimates.

When calling up a cost note from the note library, CA will notify the designer that a cost estimate or process model is revised from the original. The designer must still determine what the change is and the effect it has on the design.

To use the system effectively, supplier item part numbers and items that were not being priced using the CA system had to be listed in a database along with the corresponding costs. If a part cost was not known, then an estimated cost could be added or revised in the summary window. This approach satisfied the immediate needs, but there was a loss in traceability. Information on how the cost was determined was not captured, so the cost information was not available for subsequent estimates.

Once a cost note had been developed, it was saved and added to the parts library. Cost reports for each cost note could be generated and printed or saved separately. However, if a cost note was updated, then a new cost report had to be generated. To resolve configuration control problems, copies of the cost notes, the corresponding process models, and cost reports were saved in separate monthly directories.

3.4.2 Subarray Design/Cost Trade Results

Baseline costs were established for a 192-element prototype subarray using the tools and techniques established in Phase I. Using this baseline, design/cost tradeoffs were to be made on a monthly basis, with an objective of reducing the estimated subarray cost by 15–25% over a 14-month period (June 1997–August 1998). It was expected that iterative design/cost trades would provide a 20% cost savings, with further cost reductions realized when design producibility was subsequently reviewed at a Design for Manufacturing/Assembly (DFMA) workshop.

The design/cost savings for a 192-element subarray in June 1997 were 7.5%. The accumulated cost savings through August 1997 was 9.6% compared to a goal of 10.4%. During the August cost reduction exercise, some errors were noted in the baseline cost estimate. For example, the feed was added to the module connectors at both the level 1 feed assembly level and the array assembly level. In addition, design modifications were such that the baseline configuration no longer correlated to the original 192-element subarray integrated parts list (IPL).

It was necessary to rebaseline the subarray to a new IPL and reprice the monthly cost/performance savings. Rebaselining showed the June design/cost trade savings to be 6.9%. In July this increased to 8.4% and in August the total subarray savings was 9.2%. The accumulated cost savings through October 1997 was 10.5% compared to a 13.4% goal. The monthly factors that provided these savings were:

1. June '97: A reduced cost tile module
2. July '97: Replacing the bellowed housing sleeve of the module housing with a solid sleeve
3. August '97: Applying the Strategic Sourcing concept to the internal ring frames in the module
4. September '97: Using rivscrews to assemble the radiators, attaching the radiators to the circulator assembly, assembling circulator assembly and feeds, and eliminating keenserts from the circulator assembly for radiator attachment.
5. October '97: Removing all features in the cold plate relating to a discontinuous lid on the module (there would be a reduction in cold plate machining costs).

An example of the savings possible using an alternate assembly approach is provided by considering the September 1997 tradeoff, where conventional screws were replaced by rivscrews. Large numbers of small screws were used to attach the radiator covers to the bases, to attach the radiators to the cold plate, and when assembling the circulator assemblies and feeds. When determining the initial costs, it was assumed that all screws would be manually inserted. This was labor intensive, especially since the small screws (sizes 0-80, 2-56 and 4-40) had to be individually handled and care had to be taken to ensure that the screws were vertically inserted and the threaded holes were not stripped. It was obvious that an automated screw insertion approach would provide significant labor savings and limit the likelihood of screw misalignment.

The use of rivscrews as an alternate to conventional screws was revealed during a survey of possible suppliers offering rapid fastening methods. This technique (available from Textron, 704-888-3583) combines the speed placement of a rivet with the removability of a screw. Cost savings are possible since there is no need to thread the rivscrew holes (or, in the case of radiator attachment, there is the elimination of keenserts) and the threaded rivscrews can be automatically inserted into unthreaded holes using a power tool. Manual handling of individual rivscrews is eliminated, since initially these are provided in the form of long rods (each containing multiple rivscrews) and can be quickly loaded into the power tool prior to installation. Individual

rivscrews can be replaced once using the automated insertion equipment. Subsequent rivscrews would have to be replaced manually, just as would be required when using conventional screws. The use of rivscrews rather than screws provided an estimated 0.9% reduction in costs.

It was expected that the 20% savings goal would have been satisfied over the succeeding months through the use of enhanced circulator carriers, improved DC connector design, and integrating and compacting some of the active monolithic microwave integrated circuits (MMIC) chips in the module. The cost saving calculations for the month of November 1997 had been initiated just when program stop work instructions were received. The monthly cost saving results and cost reduction goals are shown in Figure 3-26.

3.4.3 Subarray Cost Estimation Cycle Time Results

It was mentioned previously that the primary objective of the Phase II full demonstration efforts was to show significant design/cost tradeoff reductions in tile subarray production costs. A secondary objective was to show that a 50–75% reduction in tile subarray design/cost tradeoff cycle time was possible. Although this latter effort was not finalized because all subarray design drawings were not complete at the time the program was descope, it was apparent that the targeted savings could be easily realized.

The conventional method of estimating production costs involves obtaining supplier quotes, and quotes were requested using some completed drawings. In these cases, the quote cycle ranged from 2 to 4 weeks. For this evaluation, it was estimated that the average cycle time for each quote would have been 3 weeks. It is estimated that 30+ drawings would have been necessary to complete the array design. Assuming that it requires a minimum of 4 hours to process each quote (this would include interfacing of the designer with the company purchasing agent and the interfacing of the purchasing

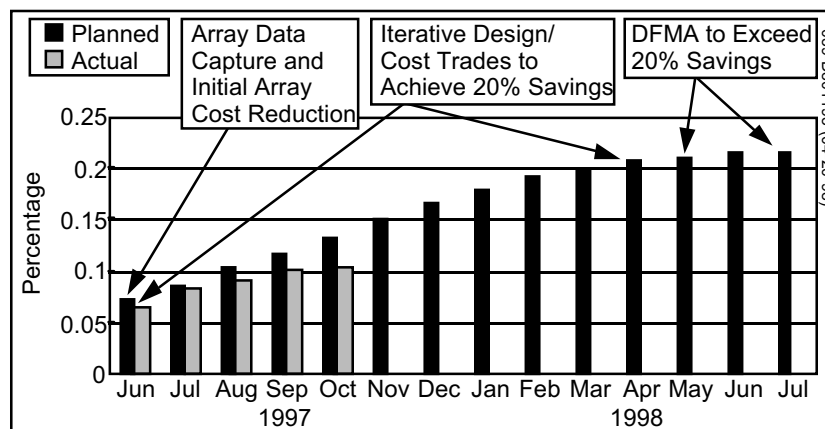


Figure 3-26. Planned and Calculated Monthly Cost Savings for a 192-Element Prototype Subarray. The cost savings analysis effort was descope after completing only the first 5 months of the 14-month study.

agent with the supplier), it would require at least 120 hours of labor at Raytheon (the supplier cost is not estimated).

In contrast, only 64 hours were required to establish baseline costs using the JMD software. And subsequently, both the cycle time and labor time required when determining the cost reduction trades was only a few hours each month. The June 1997 cost savings, involving a simple change, were determined in less than 3 hours. For the following months, July through October, the average cycle time and labor required when making the cost savings estimates was approximately 6 hours per month. This equates to less than 90 hours total using the JMD methodology. In addition, the 3-week turnaround cycle was avoided, along with the need to interface with purchasing agents or supplier representatives.

These figures illustrate that it is possible to enjoy considerable savings, both in time and expense, when initial comparative cost estimates are readily available without obtaining supplier quotes. This is particularly advantageous when making relative comparisons between a number of different approaches or when varying a design concept. In these instances, cost data could be obtained within a few hours and decisions buttressed by cost estimates made using consistent analyses. Whereas, if supplier quotes were necessary, it would be very difficult to obtain a cost breakdown regarding a number of alterations, especially if there were many and they were made sequentially.

Using this approach, by providing consistent cost analyses using a software estimating system such as Cost Advantage, it is possible to economically determine the most economic direction (or the least expensive choices) from a menu of options. Then, once this is completed, obtaining competitive estimates from a variety of sources can be developed. It was considerably less expensive to compare various design directions using cost analysis software than to obtain supplier quotations, especially when calendar time is considered, and when it is realized that many cost estimates can be completed in the time it takes to send a drawing out and receive a single evaluation.

3.5 Lessons Learned

3.5.1 JMD Implementation Issues

The mini and full demonstrations showed the utility of the JMD methodology in assisting design IPTs to make rapid, well-informed cost decisions. We believe the demonstrations proved that the JMD methodology is sound, and can ultimately result in more robust products at significantly reduced cost. However, as anticipated, implementing the toolset that supports the methodology within a large corporation proved to be challenging for several reasons.

When selecting the JMD toolset, we knew that we would have to interface with Raytheon's legacy databases, and the format of the legacy process/component cost databases would not be directly useable by whatever toolset JMD selected. For the mini-demonstration, we were able to populate our databases with the

necessary information required to support design choices. We were also able to demonstrate how, using Virtual Database Agent (VDA) technology, the data could be pulled from legacy databases into Cost Advantage and used by the IPT in near-real time. The system worked well when pricing out standard manufacturing processes using linkages that employed lookup function calls.

As we proceeded with the full demonstration, applying the methodology to a higher-level, more complex assembly, we found that process model effectiveness was greatly impacted by the detailed manufacturing process data and options captured within the databases. We found that the information contained within the databases was not broad enough to support a wide variety of design choices. In some cases, the databases did not contain new processing techniques and associated costs for new state-of-the-art technologies that would enable more robust, lower cost designs. We experienced some lack of part number cost data and process characterization data associated with new parts and processes.

During the full demonstration we also learned that the system was affected by variability in model building techniques. Because we had four engineers develop nine separate process models, there were some inconsistencies between models in the way data was input and exported. It became clear that standardization was important.

We also ran into configuration control issues as the full demonstration progressed. As we began updating our process models to add more details and design choices, we found that keeping close configuration control is critical. Cost Advantage software did not incorporate clean methods for maintaining configuration control. We found that trying to maintain tight control on model configurations caused updates to higher level cost notes to be much slower and time consuming.

During the mini-demonstration, shown to be linking of a simple Pro/E design to Cognition's TD6 fabrication process model was possible. Demonstration parts consisting of plates with various holes, cuts and slots were successfully cost analyzed by automatically transferring the design to the model. The flexibility of the transition process was also demonstrated by removing and adding features to the drawing and easily importing the data to the TD6 model. However, when applying the link to a more complex design such as those developed for the HDMP concept, it proved difficult and time consuming. Problems with annotating the Pro/E features immediately became apparent. These features were either difficult to find (the screen magnification had to be continually adjusted) or the features were too complex and additions to the interface software were required to facilitate the data transition. It was apparent that, further considerable development was necessary to overcome these issues.

As the program progressed, we developed plans to solve all of these issues. We began adding "breadth" to our process model databases by capturing new technologies and components that were being developed by other active array programs. We nearly completed updating our CORBA-compliant electronic linkage capability to legacy databases and the process characterization toolset. We strategized on how to maintain the system with the latest component and process cost information at the corporate

level. We began generating documentation that would standardize how our process models were developed across the corporation. Working with Cognition Corporation, we held several meetings and weekly conference calls to address the configuration control and Pro/E link issues. We began strategizing on how to standardize our approach to 3-D modeling so that the Pro/E link could be made more user friendly. This included establishing standard feature libraries and the developing software that would aid the transfer of feature data to the CA Process Model. We were making good progress at defining, solving and documenting all of these important large-scale implementation issues when the program was descoped. We are confident that the program could have successfully addressed all implementation issues by the originally scheduled completion date. However, for now, these issues still remain to be resolved.

3.5.2 Considerations for Implementation

Suggestions to improve the usability of the system are noted below.

1. Allow sufficient time for planning and organizing how the models are to be introduced and what they will cover prior to beginning the writing of the models themselves. This could require considerable “brainstorming” and concept development so that a detailed roadmap could be initially prepared. At the corporate level, a comprehensive plan should be well thought out and funded so that all material, process, and component options can be included within the process models. The maintenance of the process models is something that a company must sign up for if the JMD system is to remain effective.
2. Accessing legacy data banks outside the normal computer environment is not a simple task, and sufficient time, money and focus are needed before this can be completed. Also, there needs to be an accounting of the maintenance costs associated with an expansion of this degree, since factors such as outside equipment and software purchases have influence.
3. A plan for configuration control has to be established at the beginning of a program. If CA were to be extensively used, a protocol would have to be mandated, otherwise model control would quickly become nonexistent and early cost notes would lose flexibility as models were modified without recording.
4. Linking of complex Pro/E models to any CA model will require commitment at the design management level. To take advantage of such a capability, it is necessary that designers be trained to use only those features that can be converted to a CA model. Extensive feature software feature description and updating of models might be necessary.
5. Adequate training of those responsible for the writing and maintaining process models must be assured. This also includes familiarity with the UNIX computer system.

3.5.3 Detailed Lessons Learned

The following are the lessons learned that occurred during development of the

knowledge bases and when making design/cost trades using JMD methodology.

1. Training in model building techniques is critical. Model building candidates need to be trained in the functionality of the CA system, UNIX, and be thoroughly familiar with the user manuals. The manuals cover the basics and assume that the model builders have had on-line training with some model building experience. The manuals can be difficult to follow for a beginner.
2. To prepare adequate models, good communication between authors is necessary; i.e., it requires the joint efforts of personnel familiar with the CA programming approach working with those familiar with the processes of interest. Experience with the UNIX system is necessary in the former instance, along with considerable practice in module development, while those responsible with providing process information must have a clear idea of what is to be covered and how it is to be addressed.
3. Detailed planning covering the scope of each model and the detailed process options that are to be included is needed prior to the preparation of cost models. The orientation or focus of the process models developed to cost out the phased array antenna was selected (because of program time constraints) after considering only the subassemblies and components making up the tile array. When developing the tile array process models, some of the models were too product-specific and could not be used in other applications. As an example, an assembly model was generated that dealt only with particular aspects of putting the antenna array together. From a generic viewpoint, it would have been better to develop an assembly model that encompassed a wider variety of assembly techniques.
4. Presentation of design/trade options needs to be pre-planned. During model development, there were difficulties encountered when selecting design options or features from within the CA system. These could be chosen either by selecting "add feature" from the pulldown menu or by placing the design options in the process window for selection. It seemed that the pulldown menu method worked well for the process models when the feature characteristics were sequentially assembled and known. If the trade options were not known, then the process window listing the various selections seemed to work best.
5. Keep detailed notes in the appropriate editor windows when developing a cost model. This is very important because it often is the only documentation describing why a particular approach or formula was adopted.
6. There is no way of differentiating between an estimate and a quote. This is needed so that when costing exercises are completed using quotes for some components and estimating for others, this difference can be identified.
7. Learning curves should be included in a cost model so that volume production estimates can be provided.
8. TD6, the basic machining model provided by Cognition, provides good

relative estimates when using a variety of machining techniques and can be readily modified to incorporate aspects (such as different labor costs, various assumptions concerning particular equipment) unique to a particular machining site.

9. Being able to import or export information to other programs such as Word or Excel would greatly expand CA's flexibility.
10. To maintain a cost estimate history when making process changes it is necessary, each time a record is desired, to open a designated directory and save all the files that were involved. Configuration control can become difficult. Configuration control must be consistent so that tracking changes to a model or cost note is possible. Often a model would be updated after it had been used to complete a cost note, which would negate use of the original model when attempting to make a subsequent cost analysis using the original note.
11. When developing very simple designs, it is possible to link Pro/E to the CA TD6 model and automatically arrive at an estimated cost.
12. The linking of Pro/E to a CA model is not sufficiently developed for use in automatically determining costs of designs that incorporate many different features (complex or unusual shapes, combinations of different structures). To expand the linked cost estimating capabilities to the degree necessary for routine use, it is necessary that methods be developed that will convert Pro/E features into identities that can be read by the model. This might be accomplished by maintaining a Pro/E User Defined Feature library that would be used exclusively to prepare drawings. The cost of establishing such a library and restricting designers to its use needs to be evaluated from a return-on-investment viewpoint.
13. When a cost note is completed by using the link between Pro/E and CA, any identifiers (such as changing the nomenclature "cut d4" automatically provided by CA to the identifier "Tile Module mounting slot" by the user) are lost when the note is upgraded. This then requires that the descriptive identifiers be retyped by the user, which can be very time-consuming and often leads to errors in spelling and naming inconsistency.
14. Attention should be paid to any continued supplier development in linking the CA system to Pro/E and evaluated in the future when the suppliers have resolved current linking issues.

3.6 Conclusion and Recommendations

The major objectives of this investigation, to document and quantify design/cost tradeoff reductions within a demonstration concept, were met. The Cognition CA software was used to estimate the costs associated with the fabrication and assembly of a complex design. The example selected consisted of a variety of structures that had to be machined, fabricated or assembled to complete the build of a prototype array.

The flexibility associated with CA was used to capture existing Raytheon processes and it was shown that these could be readily modified or updated as needed to incorporate new processing steps. This provided a consistent, easily used method of determining relative costs. The effects of design changes or the use of alternative assembly steps on costs were readily documented and identified; this was shown by illustrating the percentage reductions in overall costs that could be realized when introducing design modifications, such as changing the module's bellows sleeve configuration to an extruded concept. The possibility of directly linking a Pro/E computer generated design with a CA model was demonstrated by determining the costs associated with various features machined into a plate. The limitations of the Pro/E to CA linkage were also explored and it was shown that considerable development would be required before this capability could be used on a routine basis to analyze complex structures. It was shown that data from external databases could be accessed and used in cost determinations. The importance of adequately planning the scope of the process models and training of those that are to prepare the models was emphasized.

The advantage of having a computerized system that could be used to rapidly determine the costs associated with a particular design are obvious. There are a number of cost analysis programs available, but few that deal with the details associated with machining, fabrication and assembly of parts. The Cognition CA system addresses this niche when dealing with simple structures or when data can be manually introduced into the model. An example of this is the development of Cognition ToolDesign6, which can be used to determine the effects of various machining parameters. This model could be adopted for use by any organization by modifying the internal tables dealing with labor rates, machine speeds, etc., so that they reflect the particular capabilities. Linking CA to outside databases and using this data to determine cost estimates is possible. This program did not complete the verification of the system on complex assemblies using CORBA-compliant information technology. We did learn that linking, either from a computer-aided design base or to tabulated data, requires considerable support to implement and maintain. To expand the use of CA within the corporate environment (i.e., widen the implementation range), it is necessary that it be fully accepted, embraced and maintained and that the parameters of use be carefully defined.

This page has been intentionally left blank

4.0 ACTIVITY-BASED MANAGEMENT FOR THE JMD PROGRAM

4.1 Introduction

4.1.1 The Need for Activity-Based Management (ABM)

Traditional accounting systems show what costs have been accumulated in a manufacturing process weeks or months after the costs have occurred. By then it is usually too late to call attention to unplanned activity of events which could result in eventual cost overruns before variations from the budgeted amount have occurred. In addition, the cost collection accounts are not consistently defined from program to program and may reflect variations in the system that are not related to either the accounting process or the manufacturing process. For example, on one product the “design” account may be used only by designers working on the design stage of a product; on another product, it may include design, prototype manufacture, redesign, and preproduction tooling.

Detailed information about cost elements for electronic assembly processes was collected according to the methodology set forth in the Activity-Based Management (ABM) manuals produced earlier in this program. The methodology describes how to collect detailed information about time charges, purchased parts, capital equipment and other costs associated with manufacturing in an activity-oriented accounting system. This information is immediately made available to the program manager, the business personnel, or the manufacturing supervisor for the purpose of tracking cost variations as a product is being designed and manufactured.

Within the JMD program, this task was leveraged by coupling it closely with the Process Characterization Toolset (PCT) task. ABM or ABC (activity-based costing) systems collect information, by activity, on what is actually happening in near-real time, while the PCT provides the necessary detail and history, by process, for what should be happening. As we have defined it within the program, the activity and process describe the same manufacturing event. While ABM defines what is occurring, PCT defines what is in control. Both methodologies require working with personnel and documentation on the manufacturing floor and in the accounting organizations to obtain the necessary information, which can be used to drive cost accountability based on activities. In the JMD program, the information has been gathered from the HE Microwave (HEM) factory in Tucson, Arizona. During the actual build of a product in a representative program, ABM data was to be collected and compared with data from the traditional ledger system. This comparison would result in both a refinement of the original activity description and an accurate assessment of the origin of costs associated with the product. The original demonstration vehicle for this program was the tile transmit/receive module being developed under the ARPA-funded High Density Microwave Packaging (HDMP) program.

4.1.2 ABM Methodology Development

The model that was constructed of the HE Microwave factory in the previous phase of ABM activities in the JMD program was used to distribute manufacturing floor and process engineering labor according to the activities that would be associated with the HDMP product. Because the HDMP program began suffering production schedule slips due to material issues, we decided to extend our ABM methodology by comparing and contrasting additional programs at different stages of production to HDMP data, with the approval of HEM and the additional program offices. Even though we didn't have the complete data for any one design-to-production process, we had pieces of information that could be patched together to form a "big picture" for manufacturing evolution from single, small scale prototypes to production volumes that could be seen typically for defense manufacturing or low-volume commercial manufacturing. This unique opportunity to examine cost drivers in the HDMP program and compare them with other similar programs became the demonstration vehicle that we would use to illustrate the usefulness of the ABM methodology. Because the program descope cut short our data taking, we were not able to verify all the trends that we thought might exist. However, in the following sections, we will show how costs in traditional ledger systems do not clearly map into activities and that the variations between programs are not uniform. Figure 4-1 illustrates this point by showing the activities and resources from the ABM model in the first column compared to the actual charge accounts used for three example programs (including HDMP) in the next three columns. The blank rows indicate which activities and tasks are not being tracked at all within the traditional systems.

All of the activities evaluated by the JMD program contained three major cost variables that the designer could control based on design trades. Direct labor for supervisory, engineering and production personnel as well as labor for administrative, finance and purchasing support was nominally the largest category for data collection. The second category was the costs attributed to the purchased parts used in production. Equipment utilization was defined as being the third and last category that would have an impact on cost. Areas such as building rent, taxes, and other overhead related costs, although they contributed to cost evaluations in ABM, did not appear to be variables that the designer had any influence over in the design process. They also were not variables that would be controlled by manufacturing.

We made one very important addition to the original program by coordinating ABM information with the PCT, allowing us to define and track activities down to the task level. By working with production, we defined detailed activity drivers, or tasks. Even though this was more detail than production personnel were accustomed to, they could see the benefits and were willing to help us track the data. Since it was not clear from this attempt at ABM how to accurately define the level of detail required, this appeared to be the proper approach to collecting information without incurring unnecessary burden to either the JMD or the HDMP programs. This approach would allow us to collect cost information after design, but before the low rate initial production (LRIP) phase. For the JMD program, this would be sufficient to show how the tools worked. For the ABM task, this additional information allowed us to track

	PROGRAM #1	PROGRAM #2	PROGRAM #3
	Demo Program	HDMP	Low Power Module
ENGINEERING			
PROJ MGT		B21	C1
CONCUR ENG			C2
NRE TEST STATION			
PRODUCTION READINESS			
PRODUCTION CONTROL			
SYST INTEB. MFG			
FACTORY SUPV			
BALL INTERCONNECTS		B22	
FLIP CHIP R&R		B23	
HERMETIC SEALING		B24	
TILE MODULE REDESIGN		B25	
PROCESS DEVELOPMENT			C3
TEST DEVELOPMENT			C4
QUALITY ENG			
INDUSTRIAL ENG			
DESIGN/DRAFTING/DOCUMENTATION			
PROCESS AND TOOLING			
CONCEPT UNITS			
BUILD			C5
TEST			C6
PARTS PROCUREMENT			
EDU BUILD			
"BETA BUILD"			
PROCUREMENT SUPPORT		B26	
MATERIAL		B27	
PRODUCTION LABOR		B28	
PRODUCTION SUPPORT		B29	
PROJ MGT			
PROCESS/MFG ENG/TOOLING			
QUALITY			
TEST ENG & TOOLING			
INDUSTRIAL ENG			
DESIGN/DRAFTING/DOCUMENTATION			
PRODUCTION CONTROL			
SYST. INTEG MFG			
FACTORY SUPV			
TEST FIXTURE DESIGN AND BUILD			
LRIP			
MATERIALS	A442		C8
ALL LRIP NRE; PROG MGT & ENG	A411		
ANDY CONNORS/STE	A412		
MCC CARDS	A413		
LRIP SHOP FLOOR BUILD SUPPORT	A431		
LRIP SHOP FLOOR REWORK	A432		
LRIP & PROD PURCHASING SUPT	A441		
LRIP TOOLING	A451		
LRIP PERISHABLE TOOLS	A452		
LRIP PACKAGING	A453		
LRIP RECALL REWORK	A433		
ASSEMBLY			C91
TEST			C92
SUPPORT			C93
PROBAM MANAGEMENT			
PROC & MFG ENG./TOOLING			
QUALITY			
INDUSTRIAL ENG			
DESIGN DRAFTING & DOCUMENTATION			
PRODUCTION CONTROL			
SYST INTG. MFG			
FACTORY SUPV			
PRODUCTION			
PILOT MTLs	A443		
PROD BLD LABOR/MTLS			
PROD BLD SUPPORT LABOR			
PROD BLD ENG. BUILDS			
PROD BLD SPECIAL ACTIVITIES			
PROD BLD OTHER			
LINFINITY MATERIALS	A444		
SPARES	A445		

Figure 4-1. Variability Between Programs Makes it Difficult to Track Charge Accounts and the Activities That May Be Associated With Them. Each program defines charge accounts based on the program's direction and the program manager's experience. Note that not all programs are at equivalent manufacturing levels.

manufacturing evolution, since the change in distribution of time charges would change from process development (at the beginning) to assembly for an individual activity.

The activity-based model developed to map the HEM organization into activities, drivers, resources, etc., in turn mapped very well into the processes that were defined in the process characterization methodology. In principle, the process characterization methodology would be collecting information and data on variations in the process based on design changes. It seems natural that since the ABM activities map well to the processes, then ABM should be collecting the costs about the actual process variations. In short, ABM confirms what is happening in the processes, while the process characterization details what should be happening in the processes.

In this final report, we will show the data that we did accumulate, with the caveat that “piecing the puzzle” together may not be as objective as we would like nor should it be considered final, since some of our assumptions might have changed if we had additional data.

4.2 ABM Evaluation at HE Microwave

In this section we discuss in detail the lessons we learned from trying to implement ABM cost tracking in a facility that produces both commercial and military hardware. HE Microwave assembles T/R modules designed by Raytheon. T/R module designs are very complex by nature. Difficulties in the designs have caused increases in costs and scheduling problems in the various programs on which the two firms have collaborated. HE Microwave process engineers have a great deal of experience in identifying and correcting these problems, bringing production programs in under schedule and under budget. Because Raytheon is always pushing the state of the art in T/R module design, each new program brings a new set of challenges. This environment provided the opportunity for cost studies with the PCT that were more global, focusing on the generality of the theory that better knowledge would produce better designs and thus lead to lower costs for government programs.

4.2.1 Planning Phase

One reason the HDMP program was originally chosen as an ABM demonstration vehicle is that it was to be a small build (originally fewer than 200 modules), giving it a short build schedule of less than 6 months. As mentioned, although designs had already been done, it was felt that design-related data could be captured by using the interview process with HE Microwave engineers. By the time this project was under way, the volumes on HDMP had been reduced first to fewer than 100 modules, then to only 25 modules. It was felt that this might not yield sufficient data or provide a long enough timeframe over which to study the relationship between activities and costs.

Two additional programs were added to the project, both of which were products with similar designs and activities to the HDMP product. The low power module product was also past the design stage and already in prototype (or EDU) development and build. Its design is similar to HDMP but simpler (see Figure 3 in the PCT report for a

comparison of technical details). The Program 1 product was in Low Rate Initial Production (LRIP) and required more parts, but was a more “traditional” design. The addition of these two programs would provide tracking of a product through the various stages of production, allowing us to piece together the total picture for this particular module design. Because of the limitations in our data collection that were already noted, we will focus primarily on data from Program 1.

4.2.2 Activity Analysis

HEM already had a complete activity-based model of its facility built using the OROS software. OROS is an ABC modeling, PC software program that is commercially available from ABC Technologies in Beaverton, Oregon. The addition of the other two programs gave us the data necessary to populate that model with information related to the various phases of production. Utilization of the individual activities varied and is explained in more detail in the next section.

HEM’s labor and shop floor control system is a software program called Manbase. Manbase is commercially available from Insopower, Inc., in Plano, Texas, but HEM has customized the program to better meet its shop floor control needs. This traditional system can supply labor hours charged in a program by person and by task. No accounting exists for activities. For this reason, one of the first steps necessary for the

NAME: DATE: LABOR COLLECTION WORKSHEET	ADHESIVE EPOXY/ SOLDER	SURFACE MOUNT ASSEMBLY	MECHANICAL ASSEMBLY	SERIALIZE	DIE ATTACH	WIRE BOND	OPEN FACE MODULE TEST	HERMETIC SEAL	MODULE TEST	ENVIRONMENTAL TEST	OTHER
PROCESS DEVELOPMENT/REFINEMENT											
MACHINE PROGRAMMING											
INSPECTION											
TOOLING (design/acquisition)											
TROUBLESHOOT											
SETUP (machine/materials)											
MATERIAL STORES											
ASSEMBLY											
QUALITY/MRB											
INDUSTRIAL ENGINEERING											
PRODUCTION CONTROL											
TRAVEL BETWEEN BLDGS 809 & 848											
OTHER:											
TOTAL HOURS											

084-D801193 (04-23-98)

Figure 4-2. Each Process Engineer Works on a Variety of Activities. Within the activity there are tasks that consume time at different stages of the program and the process maturity.

ABM modeling was to create the table shown in Figure 4-2 so that process engineers could define their hours based on activities and activity drivers, at the task level (this additional information would also help correlate observations with the PCT). The process engineers also helped us to correlate the names of the production personnel in Manbase with the activities that they would be performing according to the ABM model. This way we could partition the time charges for the production personnel approximately the same way the work would be carried out on the factory floor. These two simple additions to the Manbase timekeeping system allowed us to achieve accountability for manufacturing by activity.

Purchasing was able to supply us with the cost for each of the parts in the bill of materials (BOM). This information was to be gathered by program but did not include any Raytheon-supplied parts. An equipment utilization model used by the process engineers was also incorporated in the ABM. With the labor distribution, parts costs and equipment utilization, all the necessary information for ABM was available.

We collected data on the three programs and put them into two different formats. In the first format, the normal accounting system, time charges are accrued under one of the several charge numbers shown in Figure 4-1. In the second format, our ABM approach, each person defines his or her time charges based on activities (these are the same activities that are defined in the PCT) and tasks within the activities (these are the same as the COST parameters in the PCT).

This initial data shows a number of interesting effects open to multiple interpretations since we were working with data for only a few months and for two different programs (HDMP and Program 1). The summary table shown in Figure 4-3 was compiled from the Labor collection worksheet shown in Figure 4-2 and an analysis of the Manbase system for the same time period and the same process engineers that were represented on the worksheets. At this stage it is worth listing the effects:

- Figure 4-3 represents time charged on Program 1, which is in the LRIP phase. As shown in the figure, there are distinct differences between time charges accrued in the normal accounting system and the ABM system. ABM actually shows that each of the major activities are being worked by the process engineers. All the time charges occur in the first activity (the first process step). ABM also shows which tasks within the activity are being worked. The traditional accounting system shows only that work is being charged to “Shop Flr Bld/Supt” without detailing which activity is being worked.
- The added detail in the ABM methodology helps highlight the tasks being worked within each activity. Also note that the discrepancy in the total hours can probably be attributed to that particular engineer working on tasks that are not related to the particular process; e.g., design reviews, etc.
- Although not shown in the figure, we collected data that shows that there are distinctly different distributions of charge numbers between the HDMP program (just starting) and Program 1 (entering LRIP). This would also be expected, since the programs are at different activity and maturity levels. The

distribution of charges helps highlight program maturity.

NAME	ACTIVITY CENTER CHARGED	ABM METHOD		TRADITIONAL METHOD	
		TASK CHARGED	HOURS	MANBASE ACCOUNT CHARGED	HOURS
Engineer #1	OTHER	TRAVEL	2	A431-SHOP FLR BLD/SUPT	11
	OTHER	OTHER	1		
			3		11
Engineer #2	WIREBOND	PROC DEVEL	7.5	A411-PROG MGT/ENG	7
		MACH PRGMG	11.5	A431-SHOP FLR BLD/SUPT	56
		INSPECTION	7.5		
	OTHER	TRAVEL	3.5		
			30		63
Engineer #3	MECH. ASSBLY	MACH. PRGMG	7.5	A411-PROG MGT/ENG	24.5
	SURF MT ASSBLY	MACH PRGMG	2.5	A431-SHOP FLR BLD/SUPT	49.25
	MECH ASSBLY	TOOLING	21.5	A432-REWORK	3
	ADH SOLDER	TROUBLESHOOT	3		
	OTHER	OTHER	1.5		
	OTHER	TRAVEL	2		
	MECH ASSBLY	INSPECTION	8.5		
	MECH ASSBLY	SETUP	2		
	MECH ASSBLY	PROC DEVEL	5.25		
			53.75		76.75
Engineer #4	HERMETIC SEAL	PROC DEVEL	1.5	A431-SHOP FLR BLD/SUPT	52.6
		ASSEMBLY	23		
		TOOLING	0.5		
	REWORK	ASSEMBLY	4.7		
		OTHER	0.5		
	OTHER	TRAVEL	1.5		
			31.7		52.6
Engineer #5	OTHER	QUALITY	35	A431-SHOP FLR BLD/SUPT	186
	ADH SOLDER	QUALITY	5		
	DIE ATTACH	QUALITY	19		
	WIRE BOND	QUALITY	20		
	OPEN F MOD TEST	QUALITY	14		
	HERMETIC SEAL	QUALITY	2.4		
	OTHER	TRAVEL	0.9		
	MECH ASSBLY	QUALITY	6		
	MODULE TEST	QUALITY	5		
			107.3		186
Engineer #6	MECH ASSBLY	QUALITY	13.5	A431-SHOP FLR BLD/SUPT	74
	OTHER	TRAVEL	4.5	A432-REWORK	2
	MECH ASSBLY	PROC DEVEL	2		
		TOOLING	3		
		SETUP	1		
	SERIALIZE	SETUP	3		
	WIRE BOND	QUALITY	3		
	SERIALIZE	QUALITY	1		
		INSPECTION	1		
		ASSEMBLY	2		
			34		76
	DIE ATTACH	MACH PRGMG	14	A431-SHOP FLR BLD SUPT	88
		TROUBLESHOOT	2		
Engineer #7		INSPECTION	3		
		PROC DEVEL	5		
		SETUP	4		
			28		88

Figure 4-3. Comparison Between Program 1 Time Charges from the Manbase System and the Activities the Engineers Claimed They Worked

Figure 4-3 helps to demonstrate how ABM is useful for tracking not only costs, but also program status. For example, during the startup of LRIP, it would be expected that there would be charges to tooling and process development. Figure 4-3 shows

that this is indeed the case with Program 1. As LRIP matures, there should be virtually no tooling or process development charges. Charges to these tasks are readily apparent in the ABM system, and could possibly go totally undetected in the traditional system. A charge to tooling under LRIP would be a red flag to a program manager that there was a problem with an activity center. Under the traditional system, the tooling charges would get buried in one of the “generic” accounts. Cost escalation due to a process problem might go undetected for some time.

4.2.3 Analysis of Output

For Program 1, one month's worth of production data is shown in Figure 4-4. This figure allows us to compare data for the cost of equipment, parts and labor for this product from the ABM model. In addition, the figure has the cost for each of these categories broken down by activity (Figure 4-3, which incorporates the task spreadsheet from Figure 4-2, takes the detail one step further: we now know by task what work was being accomplished). From this figure, it is easy to see that more than 33% of the total cost of the product is associated with buying and placing the electronic components (sum of the dollar value for the items in the row marked "die attach" divided by the sum of the total cost). Another 30% of the product cost can be accounted for in the testing of the module after it is built.

The second set of columns contrasts this information with expenses from a traditional ledger system. The first noteworthy item is that there is no accountability for equipment costs. Second, parts are only accounted for as a single lot value. Finally, by referencing the account numbers to those shown in Figure 4-1, there is absolutely no way of identifying which process costs the most and, consequently, which process should be targeted for improvement and cost reduction.

VISIBILITY WITH ABM					TYPICAL LEDGER ACCOUNTING				
	ABM				TRADITIONAL LEDGERS				
	PARTS	EQUIP- MENT	LABOR	TOTAL COST		PARTS	EQUIP- MENT	LABOR	TOTAL COST
Adhesive Solder	6.30%	0.87%	4.89%						
Surface Mt Assy		0.53%							
Mechanical Assy	4.79%	0.12%	1.96%						
Serialize		0.10%	1.60%						
Die Attach	22.79%	1.35%	9.22%						
Wire Bond		0.70%	8.57%						
Open Face Mod Test		5.32%	9.49%						
Hermetic Seal	0.41%	0.41%	6.27%						
Module Test		2.65%	5.99%						
Environmental Test		0.04%							
Other			5.68%						
Rework									
					A411			7.21%	
					A412			3.88%	
					A413			0.00%	
					A431	0.03%		34.52%	
					A432			3.81%	
					A433			0%	
					A441			1.66%	
					A442	5.29%			
					A451				
					A452	0.15%			
					A453	1.49%			
Total Cost (Qty. 208)	34%	12%	54%	100%		7%	0%	51%	58%

Figure 4-4. Program 1 Cost Comparison Between Charges Accumulated by Activity and Those from a Traditional Ledger System. (Numbers are percentages of total module cost.)

	ABM LABOR ESTIMATES	CALCULATED COST (TO BUILD 208 MODULES @ 10/SHIFT)	I/E WORKSHEETS (TO BUILD 208 MODULES @ 60/SHIFT)	I/E WORKSHEETS (TO BUILD 208 MODULES @1/SHIFT)
ADHESIVE SOLDER	4.89%	10.01%	0.31%	12.78%
SURFACE MT ASSY				
MECHANICAL ASSEMBLY	1.96%	1.60%	0.06%	2.13%
SERIALIZE	1.60%	6.01%	0.17%	7.30%
DIE ATTACH	9.22%	4.25%	0.28%	5.32%
WIRE BOND	8.57%	3.40%	0.28%	4.26%
OPEN FACE MOD TEST	9.49%	7.01%	0.14%	8.76%
HERMETIC SEAL	6.27%	1.46%	0.06%	1.83%
MODULE TEST	5.99%	7.20%	0.15%	9.00%
ENVIRONMENTAL TEST				
OTHER	5.68%			

Figure 4-5. ABM Data Can Be Used to Determine If a Process Is Proceeding as Expected. For this figure, we have made some assumptions about number of hours, hourly rates, etc., which may not be valid. However, they illustrate the capability of ABM data to indicate the presence, or absence, of problems. (Numbers are percentages of total module cost.)

By making the connection between the above information and the PCT, we can also see whether the results match with those that were expected. Figure 4-5 shows the same labor

charges as Figure 4-4. However, now they are compared with the values that would have been expected if we used only the Industrial Engineers standard labor times and estimates. HE Microwave program managers may modify these standards if they know a job has an element of risk or difficulty that might outweigh the simple standard. Most of this information, along with expected program requirements (such as total number of modules built, time frame to build them, etc.), is compiled when the program manager estimates the cost of the job.

We have taken some of that information and applied it in Figure 4-5, columns 4 and 5. In column 4, we have calculated the time that would be necessary to build the product based on the cycle time, setup time and yield from these sheets. For this column, we assume that we process 60 parts per shift (we need this assumption to be able to allocate machine setup times on a per module basis). Column 5 takes the same type of information but assumes building a single part at a time. For both of these cases, we divided the cycle or setup times by the yield in order to attach a "penalty." This simple calculation does not consider rework. Instead, the assumption is made that if the setup or cycle times were equivalently longer, then the yield would be 100%.

The information in columns 4 and 5 was used to generate column 3, which has the additional information that HEM built 208 modules in one month or approximately 10 modules/shift. (This estimate was assumed to be acceptable since, in any

manufacturing environment, one needs to know the work in progress and its status before an accurate estimate of throughput for any one month can be determined.) By fitting column 4 and 5 data to a quadratic curve and interpolating to the estimated 10 modules/shift, the cost values in column 3 were determined.

These three cases can be used to illustrate an important but very subtle application of ABM. The data shows which processes cost the most. But, by comparing the ABM data with the industrial engineering standards, it is easy to see which processes are operating most “efficiently.” As an illustration, the ABM data shows that die attach accounts for 9.22% of the total module cost. To build 10 parts/shift, it will account for 4.25% of the labor costs. The ratio of these two cost figures is 46%. If we go through the same approach for the “serialize” process, we see that the ratio is 375%. This shows that “serialize” is already operating much more efficiently than the learning curves would have predicted, while die attach is still behind the expected behavior. As we follow this procedure for all processes and compare that information for all cases, we can begin to identify which processes are “on target” and which processes may actually be having problems. If we had been able to collect more information from the process engineers as to which tasks they are working on, we could understand more completely where the problems were and what was being done to correct them.

Using the ABM data and correlating it with industrial engineering standards, or PCT information about the impact of design choices, it is possible to identify which processes and which designs are driving costs. This information can be presented to production managers, program managers and/or the customer to get a resolution based on the actual facts.

4.3 ABM Integration with IT Structure

A decision support system such as the combined ABM-PCT model is a form of expert system, assisting in the decision-making process. The knowledge for such a system is human knowledge modeled in such a way that the computer can deal with it. The models used for this system included both off-the-shelf software and customized applications. Excel was used to set up the spreadsheets to capture cycle time, yield data, and the ABM interviews, timekeeping, and other cost driver data. The cost data was then implemented into OROS, a powerful ABC modeling software package. Process cost data from the ABM model would have then been fed into the Cost Advantage model. This model would have been a truly customized system for a specific set of users (design engineers). It requires great participation not only by design engineers, but also process engineers, manufacturing engineers, industrial engineers and accountants to make it work. The Cost Advantage software is flexible and can be updated as needed for changes in the rules developed. The ultimate goal for this system was to use it for routine analytical tasks (what materials to use for a “good” design, etc.) and for problem solving (why did this design fail, why were costs higher on the program as compared to another, etc.). Because much of the design of the system is being done by the team itself, there will be little or no user-designer communication

gaps. The collective expertise of many professionals has been coordinated through the use of various personnel and different types of models all culminating in the database in the Cost Advantage software.

4.4 Lessons Learned

The ABM work that was started on this program was aimed at producing quantitative cost comparisons. Initially, these comparisons were to highlight the inadequacies of the traditional cost systems and possible advantages of having an ABM system. The expansion to include other programs and to track program maturity using ABM evolved later in the program. The following list of observations is presented to highlight the observations made from trying to get ABM implemented.

- *ABM is a much more powerful cost tracking tool than traditional cost ledgers.* The discrepancies between time charges in traditional ledgers and the ABM approach illustrate the fact that across the industry time charging is not always fully understood or accurate. Tracking and controlling charges, in any system, will be difficult until this can be rectified.
- *ABM evaluations provide rough approximations of how time is spent.* However, those approximations change for individuals and for job classifications, continuously. Therefore, even ABM leads to some cost distortions, although not at the magnitude of the traditional system. ABC captures the daily reality of how time is spent, which would give even more accuracy to product costs.
- *Initially acquiring the necessary ABM information is difficult because it requires a change in culture.* The traditional systems cause the data to be stored within departments and not by activity. Also, personnel are not used to being accountable for detailed movement and their participation is often difficult to obtain for this reason. Successful participation is dependent upon providing them with user-friendly tracking systems, as well as their buy-in of the proposed benefits. Although there are a number of philosophies on why to track costs and a number of techniques on how to obtain the required information, ABC and ABM require the most sophisticated and detailed accounting procedures. For this reason, although the detailed information that is required to perform the analysis that is possible with ABM can be made available, it is very difficult to obtain—both philosophically and actually.
- *ABM provides immediate insight into the overhead costs for the period, and only those overhead costs directly applicable to the program are immediately applied.* The more traditional system currently in use applies overhead based on a yearly estimate to all programs based on the direct labor base. At the end of the year a reconciliation is done between the estimate and the actual rate for the year, often resulting in additional billings or credits to the customer. The activity approach applies overhead based on appropriate drivers to activities, which are then applied to programs, also based on appropriate drivers.

- *Using ABM to provide analysis/correlation between task charges and ROI could lead to some valuable cost saving measures.* We are focusing primarily on “direct charges”: those that come from building the part and those that come from activities, such as purchasing, that are related to the build. This information will help us properly assess where cost control efforts should be placed; e.g., if 70% of our costs are in material, what can be done during purchasing to reduce these costs? As an example, maybe all we need to do is to allocate 15% more time to material reduction activities to get the job done properly in areas such as supplier development.

ABM combined with PCT addresses producibility issues. Do you immediately plan for automation, or when a project is in startup do you start out with manual processes, move to semi-automation for EDU or prototype phases, and then move to full automation with the LRIP or full production phase? If we knew what the plan was to transition from one phase to another, designers could pick processes and materials to make the transition easier. Under much of today’s environment, products often undergo major changes between phases, causing costs for such things as tooling, equipment, and even materials to escalate. The use of the combined model would greatly enhance the design to manufacture process.

HE Microwave is a dual-use facility manufacturing products for both military and commercial applications on common production systems, using common accounting systems as well. Currently the military reporting and testing requirements are far more stringent than those used on the commercial programs. The capital and maintenance costs associated with meeting these stringent requirements get spread over the entire organization under the dual-use concept. The ABM/PCT tools could show the government how their costs could be cut by showing areas where their reporting and testing requirements are far more substantial than their commercial counterparts. For HE Microwave, as well as all defense contractors looking to commercialize many of their products, these cost savings would benefit all their customers under the dual-use concept.

This page has been intentionally left blank.

5.0 IMPLEMENTATION OF THE PROCESS CHARACTERIZATION TOOLSET (PCT)

5.1 Introduction

5.1.1 PCT Methodology Development

Detailed information about electronic and mechanical assembly processes was collected as set forth in the Process Characterization Toolset (PCT) manuals produced earlier in this program. The flowchart shown in Figure 5-1 identifies the steps of the process characterization methodology. Although a tremendous amount of information exists in the form of process instructions, process flow sheets, routing sheets and other forms of process documentation, this information is not always available to the designer when a product is being designed. The purpose of this task was to collect that information and make it available to the designer using Cognition's Cost Advantage

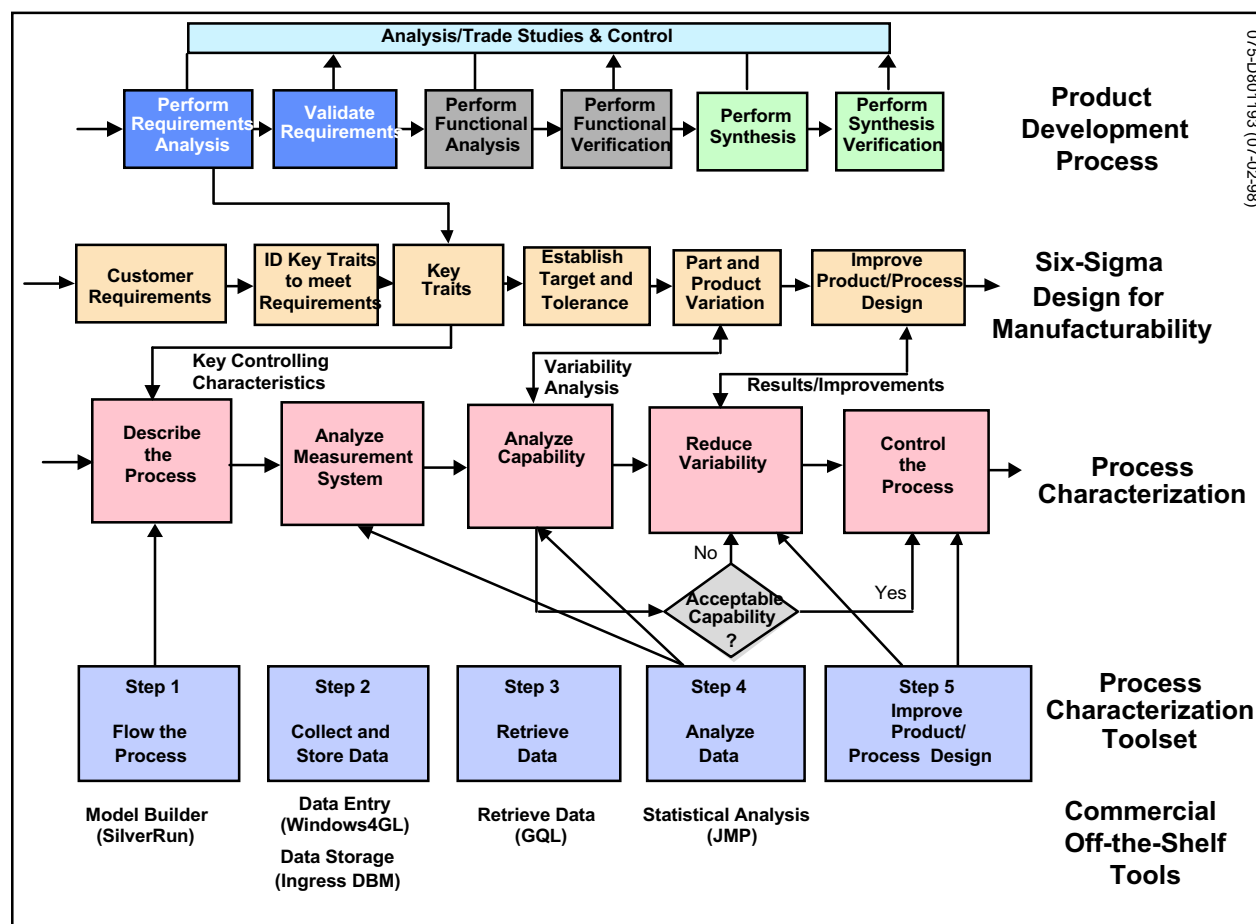


Figure 5-1. Flowchart for the Process Characterization Methodology and Toolset

software.

The flowchart shown in Figure 5-2 is a modification of the original methodology for

getting the information into the designer's hands. It shows the additional step necessary to make the information available to the designer using Cognition's Cost Advantage software. For the JMD program, the process characterization information was gathered from the HE Microwave factory in Tucson. During the actual build of a product in a development program, additional process characterization data would have been collected and compared with the initial input. This comparison would result in both a refinement of the original process description and an accurate assessment of the origin of costs associated with the product. The demonstration vehicle for this program was the tile transmit/receive module being developed under the ARPA-funded HDMP program. Because all of the PCT information collected on this program is proprietary to HE Microwave, we cannot include it in this final report. However, we will show examples of the databases and how the data was used.

This approach would have allowed us to characterize a "point" design of the HDMP tile T/R module. Delays in production of the HDMP tile, however, allowed us to expand the program. First, HE Microwave was willing to allow us to access other programs that were in different stages of production. Even though we didn't have the complete data for any one design-to-production process, we had pieces of information that could be patched together to form a "big picture" for manufacturing evolution from single, small

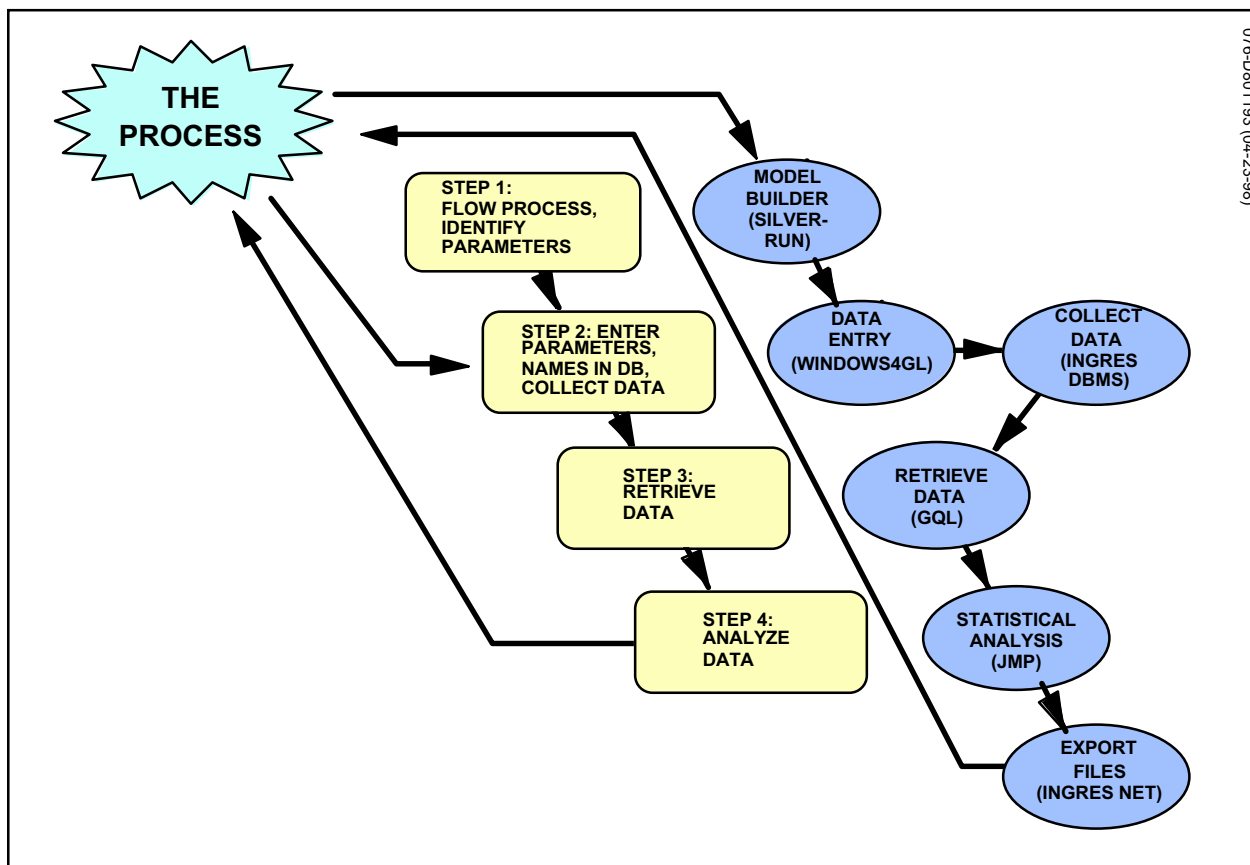


Figure 5-2. The Process Characterization Toolset. As we prepared to import information into Cost Advantage, we realized the need for adding an additional "export files" step.

scale prototypes to production volumes that could be seen typically for defense manufacturing, or low volume commercial manufacturing. Figure 5-3 shows the different programs investigated and some salient product features that we thought might make the products “equivalent” in their manufacturability (e.g., size, number and size of components, etc.).

5.1.2 Demonstration of PCT Methodology

To demonstrate the utility of the process characterization methodology and toolset, the end goal would be to have designers, using a software cost tool such as Cost Advantage, show how they could influence design trades using information gained from the process characterization methodology. Ideally, we would be able to make this comparison by having cost models built both with and without information from the process characterization methodology. To accomplish this goal, we planned to use the following three step approach. First, we would follow the methodology as illustrated in Figure 5-1. Second, once the appropriate information had been collected and organized, we would reorganize and reformat, as appropriate, to facilitate electronic transfer (this new step is indicated in Figure 5-2). Finally, we would aid the designers in building a cost model with sufficient detail to utilize the process characterization information.

From the outset, an issue has been to define, with some clarity, the information that is supplied via process characterization versus any and all other kinds of information that designers or IPTs may have at their disposal. From the JMD program, we have verified that the following information is generally made available to both production personnel and designers:

- **Process flows, route sheets and travelers** – these show the steps, in varying degrees of detail, that one goes through to assemble a particular product or family of products.

- **Detailed process specifications/instructions** – these will tell how to execute a process. Generally, these instructions include how to set up machines, perform inspections and tests etc. They may also include information about a family of products. Each of the detailed process specifications refers to a step in the process flow, route sheets or travelers.
- **Industrial engineering standards** – there are usually time studies to establish how long it takes to do a certain function that is described in the detailed process specifications above.
- **Realization factors (RFs)** – these are adjustments to the standards, above, that account for inefficiency in the execution of process steps. Although they are usually meant to apply to human issues such as sickness, training etc., they are frequently expanded to cover additional process steps (not shown on the route sheets) or process inefficiencies such as rework, additional inspections, etc.
- **Design guidelines** – these identify the preferred, or optimal, process variables that, if adhered to, will minimize the time it takes to perform an operation. Essentially, optimum use of the guidelines will result in minimum process costs for each individual process step. Unfortunately, guidelines do not usually incorporate state-of-the-art technology in products.
- **Interviews, IPTs, etc.** – these techniques allow us to gather informal information from subject matter experts. This information is extremely useful but is not always quantitative and is based on historical behaviors. It is frequently the basis for brainstorming for new and innovative ideas.
- **Accounting information** – this includes labor grades, overhead rates,

	HDMP Tile Array	Program 1	Low Power Module
Manufacturing Level	EDU	LRIP	EDU
Substrate Area (in ²)	1.41	1.57	1.41
Number of Substrates	3	1	2
Number of Components	152	59	40
Number of Part Numbers	20	28	11
Smallest Component (in ²)	0.0007	0.0003	0.0027
Largest Component (in ²)	0.048	0.053	0.053
Number of Attachment Types	4	4	3

077-D801193 (04-28-98)

Figure 5-3. Product Features of Programs Investigated

equipment utilizations, etc. This is covered more thoroughly in the ABM report (Section 4), but it does bear an important impact on process characterization

since cost usually drives some process behaviors (e.g., increasing throughput or decreasing inspection times), at the expense of yield.

Process characterization better supports design trades by tying together all of the above information and making it available to IPTs. We noticed this important distinction as we began populating the Cost Advantage models. The original cost models basically followed the process flows, or route sheets; options were not allowed. Also, as we prepared the information to be exported, it became apparent that although we had some information for each process, we didn't necessarily have the information that links one process to another or that contains the conditional statements that result from the process characterization methodology.

To illustrate this point, Figure 5-4 shows part of the table used to construct the original input into Cost Advantage for the HDMP tile module assembly model. The example data in the table applies to the face-up chip placement process. By comparison, Figure 5-5 describes the same process step, but contains the conditional statements for variations in process inputs/outputs and for variations in volume, learning curve, etc. The initial attempts at the "addition rules," which are used to describe how one step in the process may relate to other steps in the sequence, are contained in a separate spreadsheet (not shown for proprietary reasons). The two figures and the spreadsheet represent the main difference between all other collections of manufacturing information and the process characterization methodology verified within the JMD program.

Independent process data													
Oper Num	Operation Description	Machine	Sub. per Fixture	Fix. per Batch	Total Mach Cycle Time	Mach. Cycle Time per sub	Mach. Rate	Mach. cost per sub	Total Oper. Cycle Time	Oper. Cycle Time per sub	Oper. Rate	Oper. cost per sub	Tooling cost per oper.
10	StencilFlip												
20	StencilDiscrete												
50	ReflowSolder												
60	FluxRemoval												
90	EpoxyCure												
100	PlasmaClean												
120	Test												
120c	ReworkTest												
Dependent process data													
Oper Num	Operation Description	Machine	Item of dependency			Mach. CT per item	Mach. Rate	Mach. cost per item	Oper. CT per item	Oper. rate	Oper. cost per item		Tooling cost per oper.
30	PlaceDiscrete												
40	PlaceFlip												
45	PreReflowInsp												
55	PostReflowInsp												
70	EpoxyDisp												
80	PlaceFaceUp	Hughes 3500	# of faceup			180.0	X	0.05X	7.5	Y	0.17Y		Z
110	WireBond												
120a	Troubleshoot												
120b	Rework												

Figure 5-4 HEM Original Process Steps, Time Estimates, Yields, Etc., Were the Basis for the Cost Advantage Tile Assembly Model. The spreadsheet shown here is an excerpt of HEM original estimates and was used as an actual table in the cost model.

Independent Process Data															
Op Num	Operation Description	Machine	Item of Dependency	Item of dependency units	Item of dependency value	Sub/ Fix	Fix/ Batch	Mach CT Total	Mach CT units	Mach CT/ Sub	Machine Rate	Machine cost/ Sub	Total Oper CT	Oper CT units	Oper CT/ sub
80	PlaceFaceUp	Hughes 3500	die area	mm2	16	1	1	15	sec	180	X	0.05X	30	sec	30
					25			12	sec	144			20	sec	
					36			10	sec	120			17	sec	
			die thickness	mm	0.5	1	1	15	sec	180	X	0.05X	30	sec	30
					1			10	sec	120			20	sec	
					2			10	sec	120			17	sec	
			die pitch	mils	16	1	1	50	sec	600	X	.17X	30	sec	30
					20			25	sec	300			20	sec	
					25			15	sec	180			17	sec	
Independent Process Data (Continued)															
Op Num	Operation Description	Machine	Item of Dependency	Item of dependency units	Item of dependency value	Oper Rate	Oper cost/ sub	Tooling cost/ oper	Tooling cost units	Machine setup	Machine setup units	yield	yield units	process Eng Support	Process Eng Support units
80	PlaceFaceUp	Hughes 3500	die area	mm2	16	Y	.008Y	Z	\$	180	min	A	%	A	% of MST
					25			0	\$	100	min	B	%	B	% of MST
					36			0	\$	80	min	C	%	C	% of MST
			die thickness	mm	0.5	Y	.008Y	Z	\$	180	min	D	%	D	% of MST
					1			0	\$	100	min	E	%	E	% of MST
					2			0	\$	80	min	F	%	F	% of MST
			die pitch	mils	16	Y	.008Y	Z	\$	180	min	G	%	G	% of MST
					20			0.5Z	\$	100	min	H	%	H	% of MST
					25			0	\$	80	min	I	%	I	% of MST

Figure 5-5. PCT Methodology Adds Conditional Statements to Account for Product Variations

The demonstration medium, then, is the distinction between normal process bookkeeping and the models that one could extract from process characterization information.

5.2 Toolset Review

The process characterization toolset shown in Figure 5-1 will allow the user to collect and refine information about manufacturing processes as they evolve

and develop. On the JMD program, we discovered the need to perform at least two other additional tasks (described below) and a need to consider the time element or revision version. These tasks do not expose any inadequacies in the original methodology but, rather, highlight the additional sophistication necessary in order to be able to take advantage of the information in an IPT environment.

The first task required is to communicate the information electronically by exporting into it into a CORBA-compliant environment (this will be discussed in the PCT/IT integration section). This requirement results in the modification seen in Figure 5-2 and will probably have to be tailored for each user and each application of the system, since satisfying the requirements depends on the entire system and its interfaces. In order to have data bases talk to each other electronically, interfaces have to be established for either the automatic transfer of data (e.g., on a daily or weekly basis) or the event-driven transfer of data (e.g., change in process, personnel, etc.). As the figure shows, this requires additional software in the toolset.

Second, this same information needs to be imported into a cost tool such as the Cognition Cost Advantage Modeler. The need develops in response to the difference in sophistication between the process characterization output and the ability of the model builder to handle very sophisticated information. Figure 5-6 shows a sample table that has been derived from Figure 5-5. This table has been set up to allow the information from process characterization to be imported into the model builder by taking discrete values for specific instances of a manufacturing process. In a sophisticated environment, a single equation, typical of what would be seen in “electronic prototyping,” would describe all of these instances (and others). The task of importing this information into a model builder remains to be completed.

The time element relates to updating information in the model builder and warning the user that the version number of the information has changed. The act of updating the process characterization database needs to trigger all of the successive software downstream that depends on this information. Simple changes, such as a refinement in derived values, can be transmitted as a result of the revision number. More sophisticated changes, such as a new or changed process, may need to impact the entire model building process.

5.3 Implementation of PCT Methodology at HE Microwave

5.3.1 Programs

Originally, we were going to collect the process information described above only on the tile T/R module for the HDMP program. As discussed, we wanted to maximize our opportunity to perform design tradeoffs based on possible alternatives in the processes used to construct the HDMP tile or other similar products. HEM provided us with the opportunity to collect additional

Process Name	Process Variable	Cost Variable	Value 1	Value 2	Value 3
place die	die area	machine CT	1	2	3
	mm2	seconds	180	144	120
place die	die area	Operator CT	1	2	3
	mm2	seconds	30	20	17
place die	die area	tooling cost	1	2	3
	mm2	\$	X	0	0
place die	die area	machine setup time	1	2	3
	mm2	minutes	180	100	80
place die	die area	yield	1	2	3
	mm2	%	A	B	C
place die	die area	process Eng Support	1	2	3
	mm2	%	A	B	C
place die	die thickness	machine CT	1	2	3
	mm	seconds	180	120	120
place die	die thickness	Operator CT	1	2	3
	mm	seconds	30	20	17
place die	die thickness	tooling cost	1	2	3
	mm	\$	X	0	0
place die	die thickness	machine setup time	1	2	3
	mm	minutes	180	100	80
place die	die thickness	yield	1	2	3
	mm	%	A	B	C
place die	die thickness	process Eng Support	1	2	3
	mm	%	A	B	C
place die	die pitch	machine CT	1	2	3
	mils	seconds	180	120	120
place die	die pitch	Operator CT	1	2	3
	mils	seconds	30	20	17
place die	die pitch	tooling cost	1	2	3
	mils	\$	X	50	0
place die	die pitch	machine setup time	1	2	3
	mils	minutes	180	100	80
place die	die pitch	yield	1	2	3
	mils	%	A	B	C
place die	die pitch	process Eng Support	1	2	3
	mils	%	A	B	C

Figure 5-6. This Table Was Used to Demonstrate How to Take Data Out of the PCT and Put It in a Form That Could Be Used by Cognition's Cost Advantage Tool. This table contains the same data as in Figure 5-5. However, this table (Figure 5-6) was generated from the Ingres database using a simple scripting language.

information on other programs that their manufacturing personnel were producing. These programs and salient features about the products have been laid out in Figure 5-3. The variations in these products allowed us to define the “normal” production in the HEM factory and we had the opportunity to perform a more complete process characterization.

In addition to variations in features, each of these products were in different stages of producibility. Although the product features appeared to have a number of similarities and the parts lists for these products contained a number of similar parts, the cycle time, touch labor time, yield and process development had noticeable variations. Presumably, this could be attributed to variations in the product maturity. The short description that follows is included to provide background on what we intended to accomplish.

- **HDMP** – original program for process characterization. This product represented a radical new approach to T/R module design and promised to be the best candidate for the JSF radar. The product maturity most closely matched the JMD program schedule.
- **Program 1** – this product was just entering LRIP as this JMD task started and it represented more of a “classical approach” to T/R module design, which would be a good benchmark to determine how much the HDMP design saved in assembly.
- **Low Power Module** – a design similar to HDMP but made for a lower power product that had different specifications, cost targets and producibility requirements.

Because each of these programs had different producibility requirements, it was hoped that the additional information would have helped refine the process characterization methodology. The personnel at HEM each shared responsibility for more than one product. As a result, it was easy to get them to compare different learning curves and producibility issues for each product. This additional, unique information would have helped us identify how process information gets developed and communicated for innovative manufacturing processes as well as mature processes.

5.3.2 Application of PCT

The process flows, time estimates, yields etc. from the original HEM bid on the HDMP project were used to construct a process flow diagram in the SilverRun tool. Using the information seen in Figure 5-4, it was possible to construct a process flow diagram with the necessary detail as a starting point for the process characterization and as a guide to obtain additional information from the process engineers.

The additional information was obtained by conducting interviews with a representation of process engineers from each major process used in the construction of the tile. This information was collected during conversations where the process engineers were encouraged to discuss features that contributed to a “good” or “bad” design, examples of process variables that were important to their particular process, and where they saw additional work that might contribute to improvements in the

processes they managed.

The process engineers then edited a written version of the interview and made additional comments, as necessary. The information exchange was directed at the HDMP product in particular and/or examples of other products that were similar to the HDMP tile. Emphasis was placed on process variations that were robust as well as those that could lead to problems. This same information became the basis for an updated set of design guidelines that were made available to the design engineer. As stated earlier, the design guidelines offered “preferred” values of features in the processes. This information was incorporated in the PCT as the baseline for process attributes such as cycle time, yield, setup time etc.

At this stage we had enough information to simulate process runs in the Ingres database. Since no tiles were being built at this time, these simulations allowed us to enter process data in the data base, as we would expect to take it during the HDMP build and to extract it from the database for analysis. In an environment where parts were being built, we would have used the data to refine the estimates made by the industrial and process engineers. We followed up by extracting the data with GQL and importing it into JMP for statistical analysis. From our mock data set, we were able to produce a series of analyses. One example is shown in Figure 5-7.

The entire series was compiled into the single surface shown in Figure 5-8, which we would ultimately export as an equation into the Cost Advantage Model Builder. The

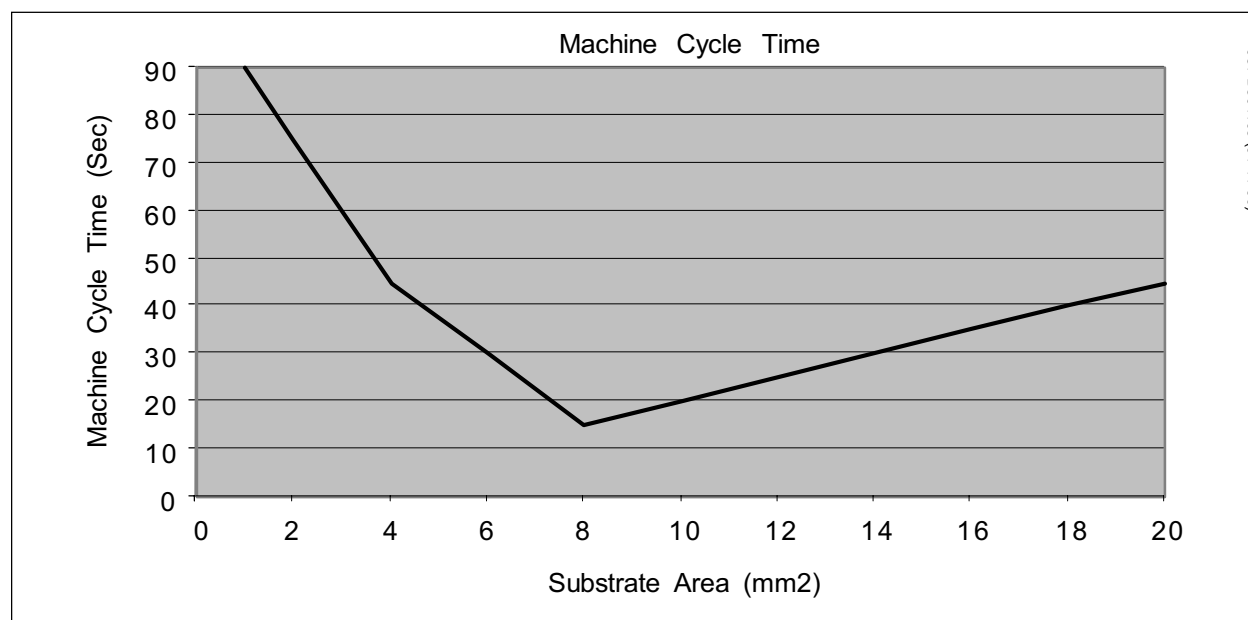


Figure 5-7. Curve of Simulated Data Extracted From Database to Show Relationship of Machine Cycle Time to Die Area. This simulation was for a low volume of parts that would be placed manually. Data shown is for explanatory purposes only.

machine cycle time vs. die size simulation was expanded to include variations in the operation that could go from small experimental runs to large volume runs.

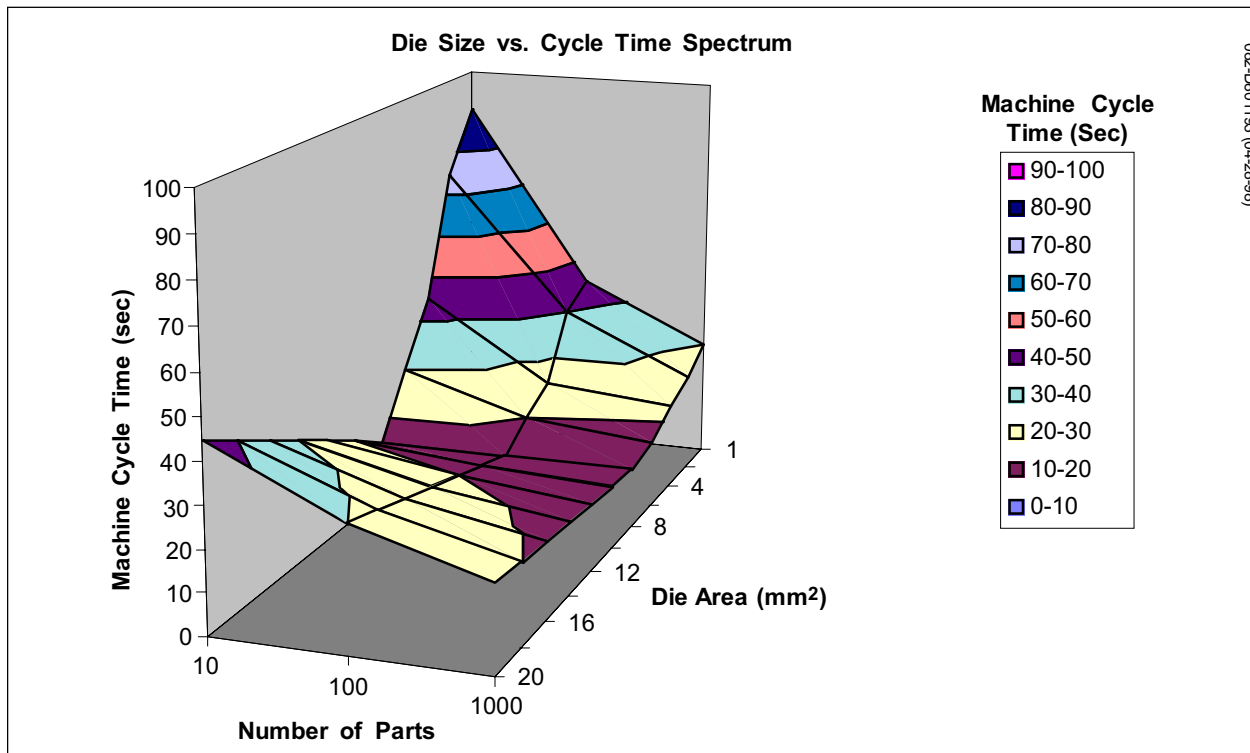


Figure 5-8. The Die Size/Cycle Time Process Spectrum. Data shown is for explanatory purposes only.

This representation shows that there may be tradeoffs between part size and production volume. It is suggested that other representations would exist for labor grade and touch labor time, part cost, machine cycle time, machine setup time, yield, process development time, (unscheduled) inspection/rework and tooling.

We began to explore and add other programs (that had already been compared and contrasted to HDMP during the interviewing process) to our process flows. This additional information allowed us to change our processes and process flows so that we could include more opportunities for design trades. For the JMD program, we switched from using SilverRun to using Excel spreadsheets to store the information. This was done because HEM did not have a copy of SilverRun that they could run on a PC. The flexibility that we gained from using Excel was traded off against the ability to link subordinate processes together. This tradeoff will be discussed in the lessons learned.

The detailed process information began to open up areas of investigation that involved cost. For example, when does it become more useful to use an expensive machine versus a cheaper, though less efficient machine? Answering these types of questions required the assistance of the accountants and finance personnel that helped produce the ABM report (Section 4) in order to address labor costs, machine utilization, and yield issues more fully. In Figure 4-4 of the ABM report, one can readily see that for most processes, labor costs far outweigh machine utilization costs (especially in low volume builds), and even parts costs. When we combine the detailed information that can be supplied by process characterization with the elements of the process costs supplied by ABM, it is easier to make data-driven decisions.

We collected actual production data from the factory floor. The process characterization methodology creates a structured environment for collecting and analyzing data. The strength of the methodology lies in its ability to identify correlations between production information and design requirements. Once the qualitative information is used to make these correlations, the data base can then be used to continuously collect and update data as the production process evolves and matures.

The information from all of the sources was compiled in table form. The resulting table consisted of rows that mapped directly into the rows of processes shown in Figure 5-4. (The same major processes, along with the associated costs, are shown in Figure 4-4 of the ABM report.) However, each process in Figure 5-4 contained up to 15 additional rows of process information and associated cost variables (labor, machine set-up, tooling etc.) based on how variables in features might cause/impact process changes. Due to its proprietary nature, we cannot include the table in this report. However, the columns in the table were set up to define the following attributes:

- **Feature** – This term was used to link the processes that process engineers were familiar with to hardware that a design engineer would generate. The sole function of the term is to help organize all the information into a common language.
- **Feature variable** – This describes an attribute of a feature that might be of concern to either the design or process engineer. Feature variables usually refer to a measurable attribute of the part, such as size, or a measurable attribute of the process such as receiving packaging.
- **(Resultant) process** – The process used to assemble the feature that has just been described is named in this column.
- **Process variables** – Important attributes of a process, such as temperature, are included in this column. Sometimes the process attributes, such as part size, are also feature variables. The inability to characterize feature variables in terms of process variables causes most of the problems while characterizing processes.
- **“Engineering” cost rules** – In an attempt to relate to design guidelines, this column contains information to aid the designer in minimizing feature costs by explaining what must be done.
- **Engineering variables** – Tradeoffs that the designer could make, such as size vs. packing density information, are contained in this column. The information is not meant to be all-inclusive, but rather to give the designer a hint from the process engineers of what directions might be most cost-effective.
- **Estimated cost for process (from baseline)** – Process engineers can aid the designer in minimizing costs by optimizing the feature, process, and engineering variables. These last three columns contain input from the process engineers that can help maintain cost targets.

- **Process cost** – The next three columns contain the information for estimating costs based on number of parts. These three columns contain the crucial information that is exported out of the PCT into a cost estimating tool such as Cognition’s Cost Advantage.
- **Process rationale** – Information included here simply defines what process attributes are driving the cost estimates established in the previous columns. The information contained in this column contains suggestions for process improvements, design changes or capital expenditures.
- **Rule addition rules** – If more than one feature variable impacts a process cost, it is necessary to identify how the variables add together. For the JMD program, this information was collected as part of the process characterization, but not linked to all the other processes. Not linking with other processes allowed us to use Excel, a simpler tool than SilverRun.

The next step was to get the information from the process cost columns and the rule addition column into Cognition’s model builder. The additional information that was collected would find a place in the model builder as notes that would be used to aid the designer in finding a more cost-competitive solution during the design. This will be addressed in Section 5.4.

5.3.3 PCT Influence

As we collected the process information for HDMP, a number of “opportunities” became immediately apparent, especially since we contrasted the HDMP information with similar processes on other programs. Because a number of observations from the PCT were already being considered by the HDMP designers, it was apparent that we were on the verge of being able to impact the overall cost of the product. At the very least, the information contained in the PCT could be contrasted with the available information that the designer used to make the original design tradeoffs.

Collating all the information in the PCT has allowed us to produce an extensive table of information that HEM will use to augment its own design guidelines. The design guidelines which used to simply state the “preferred” approach can now be embellished to identify cost penalties for deviation from the preferred approach. Further, the relationship between the variation in costs and a specific variable, such as die size, can now be quantified. The extreme detail that is captured in the PCT goes beyond even detailed process instructions. In its current state, the PCT product contains enough information to compromise HEM’s competitive position if the information were to be released in this report. However, it is important to note that the information will be used on the JSF active array program.

For this reason, the following example of PCT’s thoroughness in establishing the cost impact of design trades is presented in lieu of the actual data collected during this program. This example also refers to information in the ABM report to help emphasize the impact of the combination of these two tools.

Die that are placed on a substrate come in a variety of sizes. For the

programs we investigated, some of this information is contained in Table 5-1. At the extreme small size, the die tax the capabilities at the lower limit of the die placement machines. In Figure 4-4 of the ABM report, we can see the allocation of costs based on labor, parts, and equipment utilization. From that figure, it is easy to see that die placement is one of the more expensive operations and it is much more expensive than was originally estimated.

From the PCT, we can analyze the reasons for the costs and actually suggest a number of alternatives to reduce cost. First, “programming” the die placement machine is a much more sensitive task and requires either a lot of time for a beginning process engineer or may require the services of a more senior person. This can create a large cost for small volume runs. Second, while trying to recognize and identify the part to be placed, the machine will have a very high false reject rate for parts that tax the size limit of the machine. This will result in having a production operator maintain a constant watch on the machine to manually override the machine’s errors. Also, an additional operator will be necessary to inspect and possibly rework each and every substrate before it is passed on to the next cell. Third, especially for small and or thin substrates that may be placed too close to their neighbors or in epoxy that is too thick, another set of problems can present itself, which will result in additional delays in manufacture, additional inspection steps and additional rework. For at least one set of these conditions, the rework necessity may not be noticed for an additional two steps in production. Each of these steps is value added for the product and increases the cost if the product is scrapped.

In the above example, Program 1 is using the smallest part of all three programs. If the above conditions existed in Program 1, the solution might be to go to a larger part with equivalent operability for that die. HDMP uses the largest number of parts on its substrates. For that program, dense packing of the die might be the cost driver. The low power module design seems to have the fewest problems (as far as this example is concerned). In all cases, it appears that the best choice is to use the very best machines that can handle the smallest die and place them in the shortest cycle time, since the machine cost is small compared to the labor costs. As a result of this study, the design guidelines now capture this quantitative PCT information and ABM data that shows the tradeoff between part, equipment and labor costs. Originally, the guidelines were limited to preferred part sizes.

The PCT data that was collected helped to underscore where additional impacts could be observed. In the previous example, the distance between two die can be a critical parameter. Some of that can be addressed by the designer or in the manufacturing process. However, some of the problem may exist because the die are not accurately cut to their specified dimension. The problem of how the output of one process can effect a “downstream” process still needs to be addressed. The “rule addition rules” in the PCT begin to make some of the necessary correlations. However, for the most part, individual process owners did not have enough quantitative information to be helpful. Continuing to collect process details as discussed above will

help future processes not only as they approach “six-sigma,” but how the entire process flow can minimize variation and the attendant costs associated with that variation.

5.4 PCT/IT Integration

Integration of the PCT information with the IT aspects of the JMD program focused on the need to export the information from PCT’s Ingres data base in a fashion that would be specific enough to capture the very detailed information in the data base, but general enough so that a structure could be specified that would allow the information to be routed through the CORBA-compliant system. A second aspect of this task was to be able to make the information readily available to the users of Cognition’s Cost Advantage software. The communication between these software packages was demonstrated in the program. In the following text, the explanation of the approach and the results are highlighted by specific tasks that needed to be accomplished.

- **Create an exportable table in the Ingres data base** – To accomplish this, a sampling of rules and equations was taken from the HEM process characterization along with the relevant variables associated with those rules and equations. This information was used to construct a table, as shown in Figure 5-6. The rows of the table consisted of the various processes and the related information used in the HDMP build. The columns consisted of the variables, the cost elements (this will link the PCT database to the ABM cost elements), and the cost factors (such as number of parts, tooling, etc.). The current table mimics the table used in the cost tradeoff during the mini-demo. except that it has considerably more detail and many more options/alternatives for cost tradeoffs. The table design is generic so that more data elements can be added as they are identified and the table can be used to support the additional processes defined for the entire tile array.

Two forms of the table were considered. The simple table just described uses discrete values from the data base and was used to demonstrate the utility of the entire PCT, VDA, and the Cognition Cost Advantage system in the JMD program.

A more sophisticated version of the information would include calculations and, ultimately, equations that modeled the process. For these more sophisticated analyses, such as the relationships between number of parts and cycle time per part, as shown in Figure 5-8, an additional step must be added. This step requires issuing a query to the data base in order to retrieve data for a statistical analysis (this will be accomplished using the JMP software). The results of the analysis can then be entered into a specific table in the Ingres data base.

- **Establish the mechanism for how this table is going to be moved through the system** – When finished, the table was then exportable to Cognition through a single call from the VDA. The VDA queried the PCT database for all

the required fields in the specific tables and created a file to store the data. In order to perform this querying task, in the easiest case, both of the computers involved must have Ingres DBMS and Ingres Net. More advanced versions of this task would allow different databases to communicate by using additional software, commonly called “middleware” and referred to as “gateways” that go (unidirectionally) from one specific database to another specific database. This will allow the querying machine to send SQL commands in a form that the receiving machine will understand. The data file is created on the receiving machine, where it is manipulated to comply with the format that Cognition requires. The format of this file was determined by the method necessary for importing it into Cognition. Each record extracted from Ingres needs to be transformed into two rows, one to carry the value of the cost variable and the other row to carry the value of the process variable. This can be done by any scripting language.

The transfer from the PCT backend server to the client was successfully tested using the new CORBA VDA before the program ended. This simple test allowed us to verify that the necessary data from the PCT could be transferred electronically and prepared for incorporation in the Cognition software.

- **Verify that the table could be imported into the Cognition Cost Advantage software for the cost analysis** – Simple updates to an existing cost model, such as refinement of the error bars on existing values, can be imported into Cost Advantage by simply replacing an existing table with a new revision. The table that was used to demonstrate the system was modeled specifically so that it could be used as a replacement for the existing table in Cognition. However, even this simple example highlights the problems involved in handling more sophisticated updates. This simple table caused modifications to the existing model because it allowed for more alternatives than the current model embraced. As a further example, during this program the entire HE Microwave factory changed location. This change, which HE Microwave used to refine their process flows, resulted in improvements in their process and flow and new rationale for some cost variations. Flagging this change and modifying the cost models to reflect the new rationale is a major concern (especially if no one is notified of a significant process change). This gets back to the serious question of process ownership vs. model ownership, discussed in the Lessons Learned section (Section 5.5).
- **Establish a criteria for when this table was going to be made available to the system and whether a “push” or “pull” system would be used** – This issue was about to be addressed by the program teams. How to update data, make distinctions between revisions, and keep the models current is a task that needs to be considered in light of continuously evolving processing technologies and product technologies.

5.5 Lessons Learned

Our experiences on the JMD program have shown that the process characterization methodology, as an information/data gathering tool, works as we had described in the original drafts of the PCT manuals. We have applied the methodology to electronic assembly products at HEM and, to a limited extent, at one of the mechanical fabrication facilities at Raytheon. Although the detailed data gathering necessary to support the PCT concept can be a daunting task, the JMD program accomplished its original goals by collecting and storing more process information in one location than had previously been done at HEM or at the mechanical fabrication facility. The success of this task underscores the need for additional information to help “fill in the blanks” as well as a new type of information to help link “upstream outputs” to “downstream inputs.” This information, which can be obtained only from a diminishing number of experts, will be crucial to cost reductions in high technology systems.

Experience gained on this program has produced the following suggestions for improvements that need to be incorporated into future versions of the methodology:

- The process characterization methodology can be of tremendous benefit to the designer and the manufacturing personnel. Unfortunately, the good designers/manufacturing engineers are usually too busy to document their design trades and process improvements. These people need to better understand the potential savings and/or improvements PCT methodology can make in their jobs so that they become motivated to document this information. Tools must be made available to them to capture the information as it is produced.
- In the designer’s environment, work starts at a concept level that does not require a lot of detail and/or a lot of accuracy in the information. This means that process characterization can start at a high level. An important addition to the process characterization methodology in conjunction with the Cognition Model Builder is that worthwhile models for concept, preliminary, final, and critical design levels might be constructed from the “rolled up” details of each successively more detailed model. The flow diagrams and the data base in process characterization can already support this approach, since the data is collected at the most detailed level. Establishing a hierarchical relationship would be relatively simple.
- Designers, manufacturing engineers, industrial engineers and accountants all have different pieces of information that may not be simply appended to each other; e.g., the “designed” part may be different from the part used in manufacturing due to unavailability at the time of manufacture. The complete trail of cost, technical description and part number need to be pieced together in order to collect accurate cost information. The process characterization methodology provides what can amount to a single repository for all of this data. However, this immediately gets into questions of model ownership. It seems obvious that the manufacturing personnel need to control,

update and modify the models to reflect changes in the manufacturing process. However, the designers are the ones that need to use these models in the design environment. It would make sense for manufacturing to own and configuration control the PCT models used in the design/cost environment. These models should not be changeable by anybody outside of the controlling manufacturing organization.

- We found that flow charts, route sheets, IE standards, process instructions, etc., do not contain all the necessary information. Too many assumptions are buried in the data they contain. For example, IE standards assume a well-behaved process that can be repeated reliably. RFs do not contain the necessary detail to identify why the RFs vary. And, for the designer, none of the information is pieced together to help identify which standards need to be linked to define a single process.
 - One problem we found was that designers, production personnel and accounting personnel all speak very different languages and have different IOs that need to be translated before the information can be assembled. To avoid confusion, we need to standardize on terminology between disciplines.
 - Cost is a complex variable usually composed of dozens of elements. However, when characterizing manufacturing processes and assigning cost attributes, we were able to reduce the number of variables to the six described below:
1. **Process development time/efforts** – Startup and new processes need to be “debugged.” The more exacting the design specifications, the more time that is spent at this stage of the process. Usually Process Engineering is an activity seen at the beginning of production. However, changes in machinery, personnel, material, etc., may force this activity to show up at any time during production.
 2. **Unscheduled inspection** – Like process development, this cost usually occurs at the beginning of a process/program. However, changes in suppliers, material etc. may cause an additional employee to be assigned to a given process in order to inspect and/or repair any defects before the product gets to the next process.
 3. **Tooling** – Tooling costs are generally accepted as being a process cost. The PCT has identified potential tradeoffs between tooling (for automated and large volume production) and manual labor.
 4. **Machine setup/programming time and yield** – The costs associated with setting up a machine or station depend on the demands at that station. If the station is operating well within the spec limits, the costs are minimal. If it is operating near an extreme, the costs can become unacceptably high. This usually manifests itself as process yield. However, the distinction should be made between yield (the results) and setup time (the investment to get improved results).

5. **Machine cycle time** – Costs associated with machine cycle time are usually accepted as standards; e.g., 15 seconds/part. However, by counting the total cycle time, it has been noted that other functions, such as manual intervention, may drive the average cycle time to higher numbers. This may be a function of either the design, or problems with the process and the equipment.
6. **Operator cycle time** – Same as the previous costs item except that it applies to the operator.
- In some areas, we have learned that manufacturing personnel are still reluctant to provide any of their information to the design community. The information has been used as a weapon against manufacturing in the past and has caused the current culture. The need for major cultural changes in this area remains.
 - PCT information is far more detailed and useful than what is usually supplied in simulations because it contains all the variations that can be quantitatively described in a process. The methodology is a crucial first step for electronic prototyping, but needs to be linked with more advanced software and methodologies. Since electronic prototyping is becoming more common, the expert system could also serve as the basis of an entire electronic model of the process. Process modeling would be more transferable between programs and factories. By establishing minimal baseline data for individual factories, it would be possible to adjust the model for a number of operational scenarios. The development of an expert system of an electronic model based on the information gathered in process characterization studies would be a tremendous aid to providing the designer with a full range of design choices.
 - Focusing the PCT on design applications raises the need to translate from the production environment to the design environment by adding artificial intelligence to the tool. For the most part, designers do not even have enough information to submit a query to the tool, using GQL as we had originally described. During the JMD program, engineering judgment was applied to the manufacturing information to generate the cost rules that could have been the basis for Cognition's Model Builder. Using the information gathered by the tool as a basis for an expert system seems to offer the most attractive alternatives to providing the raw data alone.
 - An expert system, even a rule-based system, would offer the ability to interpolate in the areas where no data exists. Data may not exist because production has never had to work in that area or because the data was simply not collected. Because technologies migrate from research to production, it would be extremely useful to start collecting information in an expert system in the research stages of a technology development. In at least one case, we were able to link a model that had been developed under a research project with processes that were being used on the HEM production floor. Use of the

PCT will help identify the overall usefulness of this model to both manufacturing and design.

- Although we recognize that some changes in the mechanics of how data is moved within the PCT would be helpful, we have learned that while collecting the information takes place as originally described, output of the PCT needs to be tailored to the input of the next tool down the line (Cognition's Cost Advantage, in this case) and communication within an IT structure will put some additional tailoring constraints on the tools.

6.0 INFORMATION TECHNOLOGY ARCHITECTURE DESCRIPTION

6.1 Introduction

This section describes the information technology (IT) architecture for Raytheon's JMD Program.

The JMD IT architecture integrates the design, cost data and tools for demonstrating the JMD methodology. It has a tiered approach compatible with Raytheon's Product Data Management (PDM) system. While the structure and functionality of the JMD IT architecture were implemented within the framework of Raytheon's PDM system, the requirements and the generic architecture are suitable for application in a range of modern PDM environments and suitable for all JSF team members. The intent of the JMD methodology demonstrations was to show a solid foundation for the continued development and productization of some validated concepts and requirements, rather than to develop a total solution to the integration of the design, cost data and tools.

This section is a companion to the final version of the JMD IT Architecture document updated to reflect two significant program events at the time of publication:

- The conversion of the initial implementation of the Virtual Database Agent (VDA) to the COTS industry Common Object Request Broker Architecture (CORBA) standards*
- The operational status of the JMD IT prototype as of the date of this document

The desired capability in an integrated system is best characterized simply as "providing the right people with the right information at the right time." The migration from a traditional serial workflow process to an IPPD environment, where all functional areas work together in parallel, magnified the deficiencies in the pre-JMD development tool set and data management environments. The traditional development tool sets at Raytheon consisted largely of legacy applications developed in-house that used a variety of partially integrated COTS applications. The once-adequate legacy systems were created around the serial development processes of the past. They were designed to serve the needs of specific functional areas first and the needs of related functions only as an afterthought. COTS packages, while more robust, were generally intended to support a limited set of users. Engineering CAD tools and other primarily UNIX-based products have also been focused on narrowly defined user groups. The resulting environment was poorly positioned to serve the data and information processing needs of today's interdisciplinary integrated product teams (IPTs).

At Raytheon, the pre-JMD IPPD information environment was characterized by:

- Multiple sources of data in functionally separate application systems
- Multiple application systems resident on multiple computing platforms using multiple architectures, operating systems and databases
- Multiple formats for whatever common data that existed across functional areas

* Raytheon selected I-Kinetics Inc. of Burlington, MA as its supplier of object computing for the COBRA standard.

Such an environment of diverse data sources and sometimes redundant and overlapping tools led to unneeded complexity, marginal cost/benefit visibility and extended tradeoff cycle times. In addition, the propagation of changes in data in associated functional areas other than the originating functional area was rarely timely. Lags in information availability were often reflected in extended development time and increased product recurring cost. For example, many cost estimating tools used purchased part and material costs as components in their cost models, but the linkage between the source of the data (agreements in the materiel organization) and the user of the data (the cost model being used by the design estimator) rarely existed. As a result, an upward change in the market price of a raw material or commodity was not reflected in the cost estimating model in time to adjust the product design. The cost engineers discovered the change in market price only after an analysis of the budget overrun.

Inherent in the definition of an integrated design/cost environment was the idea that the timely exchange of information across traditional organizational boundaries had a significantly beneficial impact on cost and cycle time reduction. This concept, depicted in Figure 6-1, was fundamental to the implementation of an integrated Design to Cost (DTC) environment.

Design and cost data needed to flow both across functional areas and within the IPT product hierarchy. The application systems and tool sets supporting the IPT in DTC had to respond to the IPT demand to provide the appropriate cross-functional data at the appropriate time to all IPT members. The functional applications and their respective data could no longer exist in isolation. The IPT environment demanded an application environment that was seamless, integrated and near-real-time in its responsiveness to enable the needed information exchange.

Support for this seamless integration of design and cost information was the principal objective of the IT architecture task.

Section 6.0 is divided into three parts. Section 6.1 introduces the topic. Section 6.2 describes the tiered flowdown of requirements from the top (a productized system) to the bottom (the JMD methodology demonstrations). In it, the top-level requirements are discussed in relation to Raytheon's product development process, within which the integration of design, cost data and tools was required to take place. Section 6.3 defines the tool set and describes the data needed by the IPTs to effectively work the problem in real time. (This is a fundamental principle of the Lean Aerospace Initiative.) Also covered are the topics of Mini and Full Demonstrations,

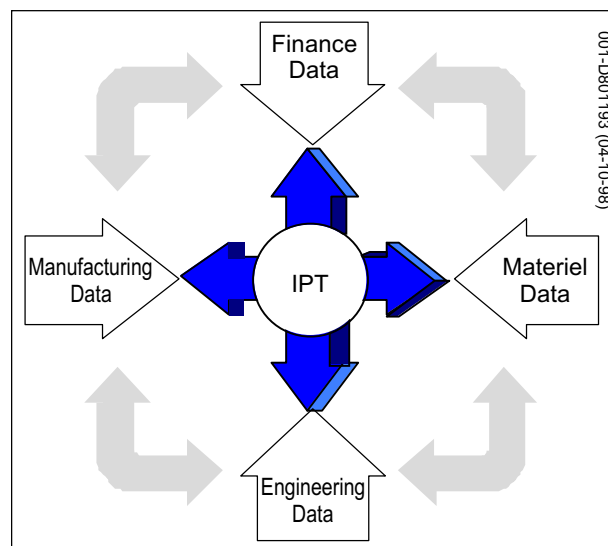


Figure 6-1. Design/Cost Data Integration

Virtual Database Agents (VDAs) and lessons learned. The detailed specifications for components of the IT Architecture are provided as Appendix A.

6.2 JMD IT Requirements

6.2.1 Top-Level Requirements for the Baseline IT Architecture

To assist JMD development and selection of architectural elements, the following four top-level guiding baseline IT architectural features were developed to support the migration of the methodologies into other JSF communities and to ensure a longer useful application life:

1. The integrated system must minimize single points of failure and be fairly fault tolerant.
2. Growth/add-on features for the integrated system should anticipate:
 - Scalable client deployment (hundreds of users) beyond UNIX workstations (i.e., PCs and Macs)
 - Scalable and diverse distribution of servers for both processing and data
 - Platform (server and client) independencies (rehostability)
 - Background system readiness and fault isolation software
 - Identification/control of system state (item versions and distribution of updates to application S/W, databases, models, compatibility/integrity of versions, etc., as well as access privileges, security, etc.)
3. COTS implementation of middleware
4. COTS implementation of infrastructure software (communications, database access/management)

6.2.2 IPPD Team Requirements

From Raytheon's HDMP Tile Array program, some operational criteria were identified for both the acceptability and success of the JMD IT architecture. A criteria checklist focused on features for (1) sustaining a design trades and cost estimating environment on a typical program, and (2) a roadmap for building/refining any productization methodologies demonstrated or verified on the JMD program. These criteria were developed in the early phases of the JMD program with key members of the module design IPT and the JMD IPT and are summarized in the following text.

Figure 6-2 depicts several of the data types requiring integration via the JMD IT architecture. This integration included:

- The linkage and translation of common data shared between the Pro/Engineer (Pro/E) and Cost Advantage (CA) tools. A discussion of the reasons for choosing these tools was provided in the document “Information Technology Architecture: Integrated Design and Cost Data and Tools (see Table 1-1).

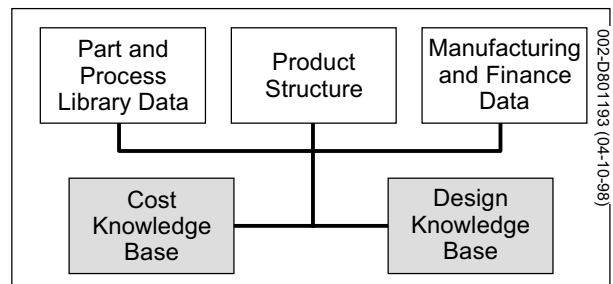


Figure 6-2. Data Types

- The means of automating the population of the CA-tool cost models with cost data from sources other than Pro/E, such as Raytheon's Cost Information System (CIS), Raytheon's tailored Product Information Manager (PIM), and other data repositories as were deemed necessary in the development of cost models that were continuously being refined to enhance the design/cost trade space. (PIM is a component of the Raytheon Product Data Management (PDM) system.)

Table 6-1 shows the data types addressed within the scope of the JMD methodology demonstrations.

The implementation baseline for the IT architecture was COTS software and hardware whenever possible. Due to the highly dynamic state of development for both intra- and Internet communications, as well as the proliferation of "tools of choice" for both hardware design and costing, the JMD IT architecture was required to be modular and reconfigurable. The architecture outlined herein and the methodologies that were subsequently demonstrated, especially the upgrade of the communications infrastructure from Raytheon's VDA to the industry standards (CORBA) for VDAs, were clearly indicative that a "plug-and-play" environment was indeed achieved for several of the design tools and databases.

The success of Raytheon's JMD IT architecture was its ability to use a CAD tool for design features linked to a cost tool that used ancillary hardware data extracted from distributed databases in Raytheon's PDM environment. Specifically, the Pro/E mechanical CAD tool and the CA design advisor and cost estimating tool were integrated to enable the sharing of data with cost models for building (fabricating and assembling) the system hardware that was under detailed design scrutiny by the IPTs. This was demonstrated using the HDMP T/R module and cost models developed to estimate production costs (refer to Section 3.0) based on actual process data derived from manufacturing process characterizations of Raytheon factories where the hardware was actually being built.

Table 6-1. Data Types and Storage Repositories

Data Type	Storage Repository
Product (design) data	PIM
Manufacturing process data	Ingres database
Existing cost models	UNIX directories/PIM/Excel spreadsheets
Component costs	CIS/Oracle database
Design models/drawings	Pro/E UNIX directories/PIM
Cost estimates	CA UNIX directories/PIM

6.2.3 Requirements for Integrating Architectural Elements

The IPPD environment at Raytheon uses a common PDM tool set that consists of basic categories of hardware and software for the applications/tools that support the five following architectural features, each of which will be further discussed in detail:

1. Product Structure Information
2. Component, Material, Process and other libraries
3. IPT/Engineering Tools

- Computer-Aided Design
 - Cost Estimating (models)
 - Process Characterization
4. Communications Infrastructure for the PDM Distributed Databases
 5. User Interface

Product Structure Information. Product information at the Hughes Aircraft legacy portion of Raytheon is managed by Sherpa Corporation's Product Information Management (PIM) software, which is functionally representative of commercially available product information management systems common throughout the industry. PIM provides the following primary functions:

- Manages product structure
- Provides work flow management capabilities
- Provides configuration management and vaulting capabilities

Component, Material, and Process Libraries. Component, Material and Process data as well as other data libraries are maintained within Raytheon's CIS software. This CIS software is a commercially available product produced by Aspect and named "Explore," which provides the following primary functions:

- Component, materials, and process library management
- Library search capability against user-specified criteria
- Linking of related information across libraries

IPT/Engineering Tools

CAD – Product design at Raytheon is supported by tools such as Pro/E (mechanical design, assembly) and Mentor Graphics (electronic design). These tools reside and execute on UNIX workstations.

Cost Estimating Tools – DTC product cost estimating uses both UNIX workstations for Cognition's Cost Advantage product and PC/Mac-based applications for software such as Excel.

Table 6-2 summarizes Raytheon's application environment as well as the supporting hardware platforms and system/data management software. As can be seen from the table, many of the COTS software products contain proprietary database management systems that have made the interface/methods for database access unique to the JMD IT communications infrastructure. However, the functionality of the JMD IT architecture achieved by this methodology is applicable to and reusable with other systems and databases.

Communications Infrastructure for the PDM Distributed Databases. The UNIX servers and workstations that support the PDM tool set communicate over a 10 Mb/sec Ethernet network using the TCP/IP network protocol. Figures 6-3 and 6-4 show the PIM and CIS communications infrastructures, respectively.

Table 6-2. PDM Application/Tool Summary

Application/ Tool	Function	Vendor	Platform	Operating System
PIM V3.2.1	Product structure information	Sherpa	Sparc 1000	Solaris 2.4
Explore (CIS)	Component, material and process libraries	Aspect	UNIX workstation	Solaris 2.4
Pro/E	IPT tool: mechanical CAD	Parametric Technology Corporation	UNIX workstation	Solaris 2.4
Mentor Graphics	IPT tool: electrical CAD	Mentor Graphics	UNIX workstation	Solaris 2.4
Cost Advantage	IPT tool: cost estimating (knowledge base model)	Cognition	UNIX workstation	Solaris 2.4
Excel V4.0	IPT tool: cost estimating (spreadsheet model)	Microsoft	PC/Mac	Windows 3.1/Mac S7.X
Process Characterization Tool	IPT tool: process characterization	Raytheon	IBM PC	Windows 3.1
TCP/IP Network Protocol	Communications infrastructure	SUN	UNIX workstations and servers	Solaris 2.4
PDM Graphical User Interface	User interface	Black & White Software	UNIX workstations and servers	Solaris 2.4
JMD Graphical User Interface (JGUI)	User interface	IONA	All platforms	All
CORBA Object Broker	IDL	I-Kinetics, Inc.	UNIX Workstation	Solaris 2.4

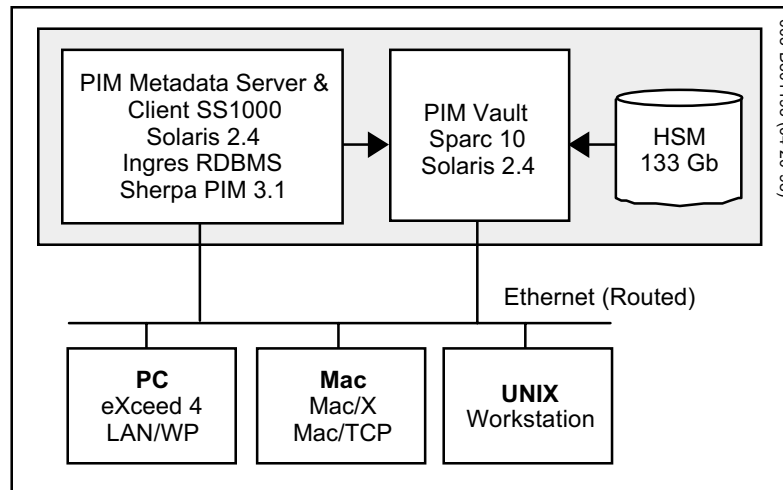


Figure 6-3. PIM Communications Infrastructure Map

User Interface. The PDM Graphical User Interface (PGUI) was the pre-existing and the common access method to the PDM applications. This software provides a more user-friendly means than the standard UNIX command line method of starting the PDM applications and performing common tasks such as file transfers and user logon administration. The PGUI interface exists as a floating

toolbar in an X-windows session. It contains buttons that will execute scripts that, in turn, can perform virtually any preassigned task. Additional applications, tools and functionality can be added as necessary to provide enhanced integrated capabilities from a single user interface.

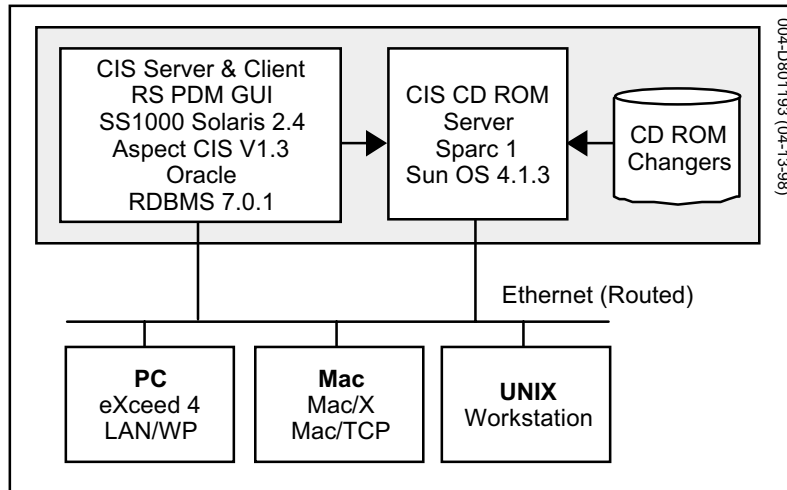


Figure 6-4. CIS Communications Infrastructure Map

6.2.4 Specific Cost Tool Requirements

The foregoing end-user IPT requirements were translated into specific key requirements for the JMD prototype development of the cost estimating methodology. This next level of requirements determined two scope of work items:

1. Promote/host related or missing capabilities within an in-place suite of applications/tools, such as those depicted in Table 6-2
2. Establish the communications and data highways between the various applications/tools

These were used to create a systems integration roadmap for demonstrating the cost-estimating methodology and providing lessons learned.

6.2.4.1 Cost Estimating – Prototype Requirements. Three candidate cost estimating requirements were identified as critical in the migration from legacy estimating systems into an integrated womb-to-tomb cost estimating system:

1. An integrated design/cost estimation capability consisting of a design model tool, such as Pro/E, directly linked to a cost estimating tool, such as CA, to produce a detailed cost estimate
2. A cost rollup capability (spreadsheet or equivalent) to develop cost estimates for export into a placeholder in some next higher level of an IPL cost rollup, or to originate directly a cost estimate for those product elements for which an integrated design/cost estimation capability is not suitable, justified, or the detailed design and cost models are simply not available (i.e., for items bought

from or made by other vendors)

3. A “point estimate” for a particular component produced by a wide spectrum of sources from a tool such as PRICE-M to that acquired from a supplier quotation.

By building a system using the above capabilities, cost estimates for complex assemblies consisting of one, two, three, or even more levels of parts indenture will be able to have a top-level cost estimate generated, while the cost estimates at the lower levels are either maturing or new estimating models are in the process of being developed. The mix of methods above allows pieces of legacy cost estimating systems to be integrated with more powerful integrated systems, such as the initial integration of the Pro/E and CA tool suite.

Integrated Design/Cost Estimation Requirements. The electrical or mechanical design model tool must be feature-based and must have a feature definition library that can be customized/tailored to reflect company/program standards that will be universal and common to all users (engineering, manufacturing, etc.) of that design tool and that will support a single universal interface. In this way, the subelements of the design (features such as holes, edges, surfaces, cuts, fills, vias, trace widths, flatness, locations, etc.) are consistently described and captured in a database that can be accessed by some other external application such as the CA cost estimating tool.

There are similar requirements for the cost estimating tool. The detailed cost model must account for the manufacturing processes used to produce the component or part, which are ultimately a function of the component or part design features captured in the electrical or mechanical design tool. Therefore the cost modeling tool must incorporate all elements of the manufacturing process around the design features used to describe the design.

With design features and rules for these design features as the common ground for communicating between the design and cost estimating tools, the design and cost models are more effectively linked for a real-time or non-real time exchange of the design information that is required for the cost estimating model.

In addition, the amount of manual I/O in this tool linkage can be minimized and even eliminated depending on several important conditions: (1) the prearranged collaboration on the design features and rules that are generated by one tool and consumed by the other, and (2) the capability that each tool supplier (e.g., Pro/E and CA) provides for the export/import of database components for the sharing between applications.

Because the cycle time for cost estimate updating was to be minimized, the following requirements were deemed necessary:

1. A direct and automated link between the two applications
2. Collaborative mapping of design features from the one tool into the manufacturing features of the cost model tool
3. Manufacturing process characterizations in the cost models that are themselves functions of these design features

It can be seen from this flowthrough that the entire cost estimating process appears to be very feature dependent and that the success of an integrated system is very dependent on there being a common set of coordinates.

A typical usage scenario for such an integrated system would be:

1. A design change in the design model
2. An export of the new features database with changes to an external database
3. An import of this new features database into the cost model by the cost estimating tool
4. A recalculation/update of the cost estimate by the cost model
5. Based on the impact the changes have had on the cost, a reiterative loop back to step 1 until the design matures to the satisfaction of the IPT members

The preceding usage scenario was achievable on a variety of platforms, with the biggest performance discriminator being the speed of computations and the cycle time for a cost estimate update through recursion on steps 1 through 5. Only a UNIX-based Sparc workstation was evaluated in the JMD methodology development.

Cost Rollup Tool Requirements. A cost rollup tool must allow the user to create a “cost snapshot” of a product (or group of products) in accordance with an indentured parts and processes breakdown. Each cost snapshot contains various subelement configurations and alternatives, as well as variations of programmatic ground rules and assumptions. The cost rollup tool should produce a cost estimate and a range of uncertainty if possible. The cost estimates should be organized by cost element. The unit production cost of parts and assemblies should be rolled up to the total product level. Process characterization to the level needed to account for rework and scrap cost should also be included.

Cost Throughput Requirements. This requirement is open-ended, and its inclusion allows any external estimate to be inserted into an overall cost structure as a placeholder so that a cost rollup is not held hostage for some less critical data that may not require a more detailed or sophisticated cost model. These inputs might be produced through analogy (to an earlier product), a Parametric model, an informal supplier quotation, a design memorandum estimate, or a cost budget/bogey that may be the goal of some other IPT.

6.2.4.2 Configuration Management of Cost Data Requirements. The fundamental Configuration Management IT requirement is for a repository that stores five key items:

1. A fully indentured parts and processes list (IPPL) for each configured version or release of a project
2. Cost estimate values for parts at all levels of the indenture
3. Databases consumed or generated by all models
4. The models themselves

5. Design and cost snapshots of the models once they have been made design-specific by user selections and tailoring of the cost models within the design trade space that has been modeled

The repository will allow the user to:

- *Load the top-level cost rollup tool* – The cost rollup tool must be loaded at the top (product) level from the repository with a specific version of the IPPL and its associated design and cost estimate models. This step is the starting point for a new project that is created as a modification to an existing project, or for a new set of designs and cost estimates associated with an existing project version.
- *File new or revised product structures* – When a new drawing is created or an existing drawing modified, the repository must accept the new or revised data for storage and subsequent retrieval.
- *File revised cost estimates* – If a product IPPL is unchanged, but a revised cost estimate is created, the repository must store the revised estimate with its associated cost model details.
- *Retrieve cost estimates* – An existing cost estimate, with its associated design and cost models, may be retrieved from the repository for review or re-estimation.
- *Annotate configurations* – An existing product configuration may be annotated. For example, a particular configuration or element of a configuration may be marked as invalid. This may be necessary when it is discovered that the design is not feasible to produce or the cost estimates were based on erroneous assumptions.

6.3 JMD Implementation Methodology

6.3.1 Overview

The primary purpose of demonstrating the methodology was to prove merit and feasibility, and then to refine and validate the final requirements for an integrated Raytheon production system. Given this goal, it was determined that a complete implementation of the above JMD prototype IT requirements into a totally new system architecture was neither practical nor advisable at this phase of the program. The JMD implementation methodology was guided by some fundamental precepts.

Two key facts determined that it was mandatory to remain synchronized with the existing Raytheon environment to the maximum extent possible, for management buy-in and acceptance:

1. An architecture/infrastructure (PDM) was already in place at the Hughes Aircraft legacy portion of Raytheon with its many applications/tools of choice already well established within that IT architecture as Raytheon's company standards
2. Some of these applications/tools were also under evaluation for potential upgrade or replacement by committees and groups reviewing company-wide

requirements and utilizations

As was previously pointed out, the methodology is transferable to the entire JSF community, although the interfaces to the proprietary databases made some of the system “wiring” unique to the database interfaces. However, the functionality of the system IT layers above these interfaces was made as tool independent as practical, or at least anticipated the need for the additional work at some future time.

The other key implementation issue worthy of note was that the Raytheon committees responsible for tool selection and infrastructure changes at the corporate level became aware and sensitized to the new requirements from the JSF community.

Table 6-3 depicts the JMD prototype IT requirements of Section 6.2 alongside the in-place applications/tools of Raytheon’s PDM environment from Table 6-2. Table 6-3 includes several tools/applications, such as CA and Explore, that were relatively new additions to Raytheon for programs such as the HDMP and JSF.

With the adoption of the tool set of Table 6-3, the JMD IT Architecture had the following implementation attributes:

- Changes to the in-place (Raytheon’s) standard network architecture were not required
- Changes to the existing computing platform infrastructure architecture were not required
- Modifications to the Sherpa PIM data structure were not required

- All tools can run on the existing PC, UNIX workstations, or file servers, as indicated earlier in Table 6-2.

To summarize the effort presented in earlier sections and to prepare for the information to be presented on the VDA, Figure 6-5 is provided as a map to Raytheon's integration of people, processes, tools, and data. Due to the wide ranging and overlapping nature of the JMD subject matter, topics have been covered in many sections of this final report. Section references are provided in the upper right-hand corner of each box to help the organization of the flow.

Table 6-3. JMD IT Prototype Requirements vs. PDM Tools

JMD Prototype IT Requirement	PDM Application/Tool	Rationale/Comments
Integrated Design/Cost Estimation System	• Pro/E – mechanical design • Mentor Graphics – electrical design • Cost Advantage (CA)	• Company standard • Company standard • Knowledge base/rules capability
Cost Rollup Tool	• HACOST • Cost Advantage (CA)	• Company standard • Excel-like capabilities with a significant knowledge base capability
Point Estimate Tool	• PRICE	• Company legacy system
Configuration Management	• Product Data Management (PDM)	• Company standard
Cost Estimating Databases – Mfg Process Library	• PCT – Ingres	• Company legacy system
Cost Estimating Databases – Machine & Assembly Cost Factors	• PCT – Ingres	• Company legacy system
Cost Estimating Databases – Parts Costs	• Product Data Management (PDM) Explore	• Company standard • Candidate as a Raytheon standard
Cost Estimating Databases – Programmatics	• Raytheon Proprietary – Oracle	• Company standard

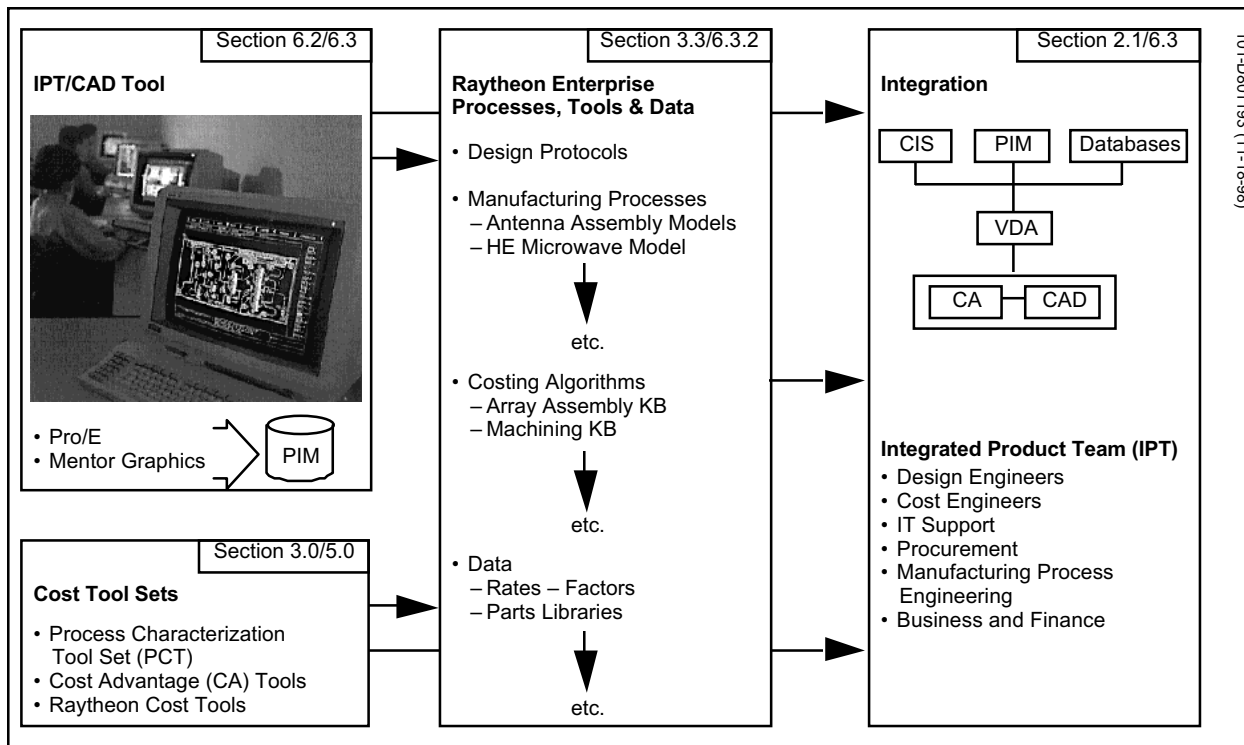


Figure 6-5. Flow of Integrated Design and Cost Data

6.3.2 Design/Cost Tool and Data Implementation

The JMD implementation approach can be subdivided into the tool-to-tool integration part for the two JMD prototype IT tools, Pro/E and CA, and a database integration part for the cost estimating databases as outlined in Table 6-2. Both of the tools being integrated can be executing concurrently under the control of the same workstation because the UNIX environment is the foundation for the tool set, and the operating system allows multiple windows to be open concurrently. The display screens provide a “cut and paste” capability as a manual means to “pick and place” data between applications. It is the least desirable, however, because it is somewhat time-consuming, requires specific user intervention, and is prone to the usual “man-in-the-loop” repeatability/reliability issues.

Where the application tools supported it, the JMD implementation first took advantage of the vendor-provided “encapsulation,” the wrapping of interface software around application software. Encapsulation provided the ability to exchange data with other applications by keeping the formatting, mapping, and communications functionality isolated from and outside of the actual application itself. Applications communicated and sent/received data in accordance with their internal protocols/interfaces and the encapsulation software then provided the necessary translations for the incoming and outgoing data. Vendor-provided encapsulations supported relatively easier tool integration as opposed to either modifying or specially

customizing the COTS S/W.

Where vendor-provided encapsulation did not exist and the manual cut and paste methods proved too tedious, customized encapsulation software was defined and developed to support the point-to-point data exchange between tools. This encapsulation software used varying combinations of operating system capabilities, custom program code and third-party vendor solutions to provide the necessary data coupling between the two tools. Functionally the software provided the same level of integration as did the vendor-provided tool encapsulation, except that the software was not vendor-owned with the consequences that whenever a new version of the vendor S/W was released, the custom software was at risk of being incompatible. Since it was highly desirable to keep the vendors working towards vendor-provided encapsulation, all vendors were involved with these custom encapsulation activities with the intent that the software would one day become part of the Pro/E and CA S/W itself. Further, this involvement helped define requirements for data-exchange standards between tools as a goal toward establishing some reusable industry standards.

Development of a Cost Link to Mentor Graphics (MG) CAD Software. This effort was added scope to the JMD SOW, which was jointly funded by Raytheon, Northrop Grumman Corporation, Cognition Corp. (as uncompensated labor), and the JMD Program Office through Wright Laboratories Material Directorate. The purpose of the task was to develop an automated link between Cognition's Cost Advantage (CA) tool and MG's Idea Station, Board Station and Hybrid Station tool set.

A specification for the Costlink-MG was provided to Raytheon and Northrop Grumman in December 1996 for review and comment and an alpha test copy was delivered in January 1997. The final version (CL-MG version 1.1) was delivered in April 1997 along with an Installation Guide and User's Guide. This final delivery has been accepted by both companies.

In support of this effort, Raytheon created a prototype CA model for substrate fabrication, which was delivered to Cognition for demonstrating the usefulness of Costlink-MG. For the JMD program, Costlink-MG was used with a substrate knowledge base on the tile T/R module.

Several documents have been generated by Cognition that provide the details of this customized encapsulation. These are:

- Costlink-MG Installation Guide – CL-MG version 1.1, April 1997
- Costlink-MG User's Guide – CL-MG version 1.1, April 1997
- Product Specification – Costlink-MG, April 1997
- Costlink-PE User's Guide – CL-PE version 2.0, October 1996

6.3.2.1 Design and Estimating Tool Integration. To keep cost-estimating cycle times at a minimum so that design changes and resulting cost impacts were available almost in real time, the Pro/E-CA tool and data integration required a mix of encapsulation and point-to-point (direct) interfaces, because both the Pro/E CAD tool and the CA cost

modeling tool had complex, proprietary methods to access and manage their internal design and knowledge databases. At this point in the methodology demonstration, changes to the vendors' application software were strictly avoided and maximum use of encapsulation and unique interface development were utilized instead.

The purpose of the integration was to make the design features from the Pro/E CAD tool available to the CA cost model so that it could determine the cost of fabricating/assembling an item with those design features. These features became requirements/inputs to the manufacturing processes modeled in the cost tool.

While the Pro/E CAD tool provided the design features using definitions and formats native to the Pro/E application source code, the CA cost tool (specifically the cost model) had its own definitions and formats for design features and formats for model variables. A fully automated linkage between the two required either or both vendors to make significant changes to their software. Without these changes, a significant amount of artificial intelligence was required to enable the automated connection of the design features presented by the CAD tool to the closest or most appropriate design feature modeled in the cost model. Needless to say, it became intuitively obvious that a set of rules/standards needed to be established between the CAD designer and the cost model designer regarding the requirements for the generation/consumption of design features, along with standards for nomenclature, format, and design-detail breakdown/descriptions. These standards would enable the design and the cost model to remain "data-synchronized" without any artificial intelligence to resolve the discrepancies between design features on the one side that had no corresponding cost model feature on the other.

Libraries/standards were required for the descriptive breakdown and capture of design features, specifically those features that were in the design-trade space, which were of primary interest for the first cut of a cost model. Since time was of the essence, and while pure encapsulation by the vendors was being promoted, and while the Raytheon enterprise established the necessary standards/rules, an interface "bridge" was developed to map the design features from the one tool to the other. Once the mapping had been established, the data exchange was automatic so that as the design changed, the cost model automatically tracked it, unless some new features were added that required additional mapping.

Instead of replicating the design disclosure and system details of the interface "bridges" between the Pro/E-to-CA and the Mentor-Graphics-to-CA tool combinations, reference is made to the documents that have been published for that purpose. For the Pro/E-CA interface, the document "Costlink-PE User's Guide" from Cost Advantage provides information on the implementation. For the Mentor Graphics/CA interface, the "Costlink-MG Installation Guide" and the "Product Specification: Costlink-MG" provide information on that tool-to-tool interface. A brief top-level description follows simply to provide the necessary insight into the JMD prototype approach.

6.3.2.2 Pro/Engineer Cost Linking

Initial Feature Linking. Design feature types and dimensions from Pro/E were manually mapped to labels in the CA knowledge base. The mapping process used the CA Annotator utility. These characteristics were then mapped to specific processes/cost estimates using previously defined cost estimating relationship (CER) algorithms that resided in the CA knowledge base. This sequence can be summarized as follows:

- Each design feature was mapped to a specific CA manufacturing feature, which had been previously defined as being the outcome of applying a specific manufacturing process. The end user (or model developer) must be able to specify that a specific design feature (e.g., stepped hole) will be created by a specific manufacturing process (e.g., multiple drilling operation).
- Each characteristic of a design feature (e.g., diameter, depth and tolerance for each element of a stepped hole) was mapped into cost estimates. An appropriately constructed knowledge base then estimated the cost for performing the process based on the feature attributes and predefined CERs. These CERs can be developed in a number of ways, with the highest resolution and accuracy being achieved when they are based on accurately measured manufacturing process characteristics. Key CERs for the JMD full demonstration were developed in this way.

Feature Updating. When feature dimensions or tolerances were changed, the updated information was automatically linked directly into the CA model. The linkage used the map created in the previous step via the Costlink utility that operated between Pro/E and CA.

Initial feature mapping can be tedious for complex products, and this in itself provided the drive for a more user-friendly approach. To this end, Raytheon and other users had approached the tool vendors, Pro/E and CA, to undertake the required modifications to the application S/W. In the interim, Raytheon's team member, Management Support Technology (MST), had designed and prototyped a Decision Support System (DSS) that greatly reduced the burden of feature and dimension annotation. It is described in the next section.

6.3.2.3 MST Decision Support System for Pro/Engineer. Even while the vendor upgrades to the application software were under way concurrently in an indeterminate time frame, and without relying on any aspect of such vendor upgrades, the MST feature-labeling DSS provided a linkage from Pro/E features-dimensions-tolerance to the CA cost model. The DSS approach was generic in that it could be extended to other feature-based CAD tools with some modest effort.

The DSS mechanized and presented a consistent graphical user interface (GUI) to the user, such that the initial man-in-the-loop Pro/E-CA linking effort was more acceptable. The functional description of the DSS follows.

Feature Characteristic Description. Each design feature in Pro/E included a name and values for its characteristics; i.e., dimensions and tolerances, and these name/values

were what was required by the CA cost model. Unfortunately, the names/format used on the CAD side are different than those used on the CA side. The CA cost model required that there be a label on the Pro/E feature that corresponds to the label in the cost model. Since it was easier to do this mapping in middleware software for reasons mentioned earlier, the DSS was developed to facilitate this functionality.

For a model with a moderately large number of features (50–100) and characteristics (100–500) this process tended to be tedious and time-consuming, and it increased the initial cost estimation cycle time significantly, but it was a non-recurring event. If CAD design changes were such that new features had been added, then only these changes needed to be reviewed and labeled. To assist in the manual labeling effort, the DSS utilized the CA feature extraction capability of the CA Annotator utility, which provided a list of the features that had been incorporated into the cost model along with the labels that had been assigned to them on the CA side of the interface. The man-in-the-loop used this information to complete the “bridge” to the Pro/E side of the interface by identifying the Pro/E counterpart features and dimensions/tolerances. Although still man-in-the-loop oriented, the linkage was mechanized and was at least repeatable/consistent in its implementation.

Feature Characteristic Labeling. The DSS greatly reduced the time required for the user to label the required feature characteristics. With the Pro/E (MAP.PE) feature database as an input, the DSS prompted the user by highlighting the feature and its dimensions in a wire-frame view of the design model. The user was then required to select only those characteristic labels that applied to the cost model for a particular feature. In cases where the Pro/E feature had fixed characteristics, the labeling was done automatically. The DSS also allowed the user to selectively label individual features that had changed in a design iteration, rather than relabeling the entire design model as was required on the initial (non-recurring) linkage.

Mapping Design to Manufacturing Features. The DSS allowed the user to create an additional file (MAP.DSS) that identified those single Pro/E design features that may have been associated with multiple CA manufacturing features. For these ambiguous design (CAD) features, the user was prompted to decide which of the cost model manufacturing processes was to be used. The user's choice was then stored as an “alternate type” characteristic in the Pro/E model, and this was what was used to link the design model to the CA cost model. This DSS capability functions without the MAP.DSS file, although the list of choices for an ambiguous Pro/E feature must then include all the available manufacturing processes in the cost model. Using the DSS in this manner required an additional step to save the Pro/E model in an external form. The external form was then input to CA. The DSS produced an external description of the Pro/E model that was then imported into a spreadsheet model. Because it functions externally to Pro/E, it does not allow real-time linking but is used periodically throughout a CAD session, requiring a few minutes per update. When the Costlink upgrade becomes available, the DSS will accelerate annotation and then iterative updates can occur

automatically via Costlink with no further intervention, unless a new feature is added that requires additional annotation.

Mapping Design Features to Manufacturing Processes. Each design feature in Pro/E has a “feature type” that is integral to the process of building the Pro/E model. Unfortunately, the feature types assigned by Pro/E, and subsequently mapped to the CA manufacturing process model, have inadequate resolution for distinguishing important process/cost characteristics. For example, a straight hole and a counter-bored hole both have a feature type of “HOLE.” However, the manufacturing processes necessary to create each differ greatly. While the Pro/E user can modify a feature name or add another characteristic called “alternate feature type,” the existing CA model linkage uses only the “feature type” to link the design and cost models. Since this limitation drastically limited the precision of the model, Cognition released an enhancement to Costlink that used the “alternate type” to map the design model to the cost model.

6.3.2.4 Cognition CA Performance Enhancements. Raytheon’s exploratory modeling approach used “cost coefficients” as CERs for each element of an NC milling process. These coefficients were, in general, applied to a measure of the process complexity based on the dimensions and tolerances extracted from the Pro/E model. For the exploratory work, the coefficients were based on handbook values or other external sources, rather than on actual cost data. Initial attempts at estimating the product cost associated with NC-milling from a Pro/E model using this technique took 20 minutes to execute for a relatively complex part. When this concern was discussed with Cognition, they responded with a solution in a very timely manner, releasing a performance enhancement kit that allowed cost equation functions to be “compiled” in LISP. Implementation of this kit substantially reduced the time that it took to execute the cost model to 30 seconds or so.

6.3.2.5 Design/Cost Data Integration Approach. Because the data formats of most applications are often strongly focused on the internal requirements of that application, Raytheon had chosen to address data integration as an important and separate task from the tool integration, even for applications adhering to open systems standards. This was done to assist in the development of data requirements that were not unduly influenced by any particular application or vendor, and that were at a higher level of systems implementation so as to remain relatively tool independent. This provided a means to develop and then “push back” data requirements to the many vendors whose applications were part of Raytheon’s enterprise. For the JMD demonstration, all methods of data integration were characterized by data format translations and data mapping translations for both the sending and receiving of data between the tools being integrated.

Data integration was accomplished first with database consolidation, where economically advantageous, and second and more importantly through a more directed form of data integration.

Database Consolidation – Database consolidation involved the physical collocation of data in a common repository. This was accomplished by the expanded use of existing functionality in the latest release of the CIS product. This product supported a soft-modeling capability that allowed the inclusion of an almost infinite number of data types of data in a common repository. Data types in the repository could be linked without changing the underlying database structure. This capability in a COTS package added dramatically to the capability to house heterogeneous data in a common repository.

Point-to-Point Forms of Data Integration – While the scope of direct integration for the JMD demonstration was manageable in size, extension of a direct integration approach to the full IPT environment could quickly create an unmanageable number of point-to-point integration solutions. This complexity is portrayed in Figure 6-6, where there are N databases for M users requiring up to $M \times N$ integrations/connections. This was not viable on a large scale, especially given the limited number of vendor-provided encapsulations that were available. Further, given the variety of data types and formats to be integrated, there was greater risk that complex and incompatible solutions would be developed for many of the interfaces. The administrative aspects of such an implementation (security, permissions, configuration control, etc.) made a solution tending more towards $M+N$ integrations/connections more highly favored.

A “virtual” data integration scheme using a virtual database agent (VDA) was the solution used for the JMD IT prototype. The VDA provided a data access resource for delivering the right data in the right format to an application at the right time. The VDA presented the image of a coherent data source to an application. In this way, the application was isolated from the actual data interface and did not need to have the database format or protocol. The VDA itself presented a “virtual” data interface on the one side to one or more applications, while on the other side provided the one-shot actual interface to the sources of data. Of the $M+N$ connections, M are encapsulations that ultimately would be vendor-provided, and N are identical Web-like connections to a UNIX server. The solution scales nicely, and the VDA provided a centralized piece of middleware software for housing the M interfaces to the backend data servers (as illustrated in Figure 6-12, Prototype IT Architecture), which made configuration control and design changes much more manageable.

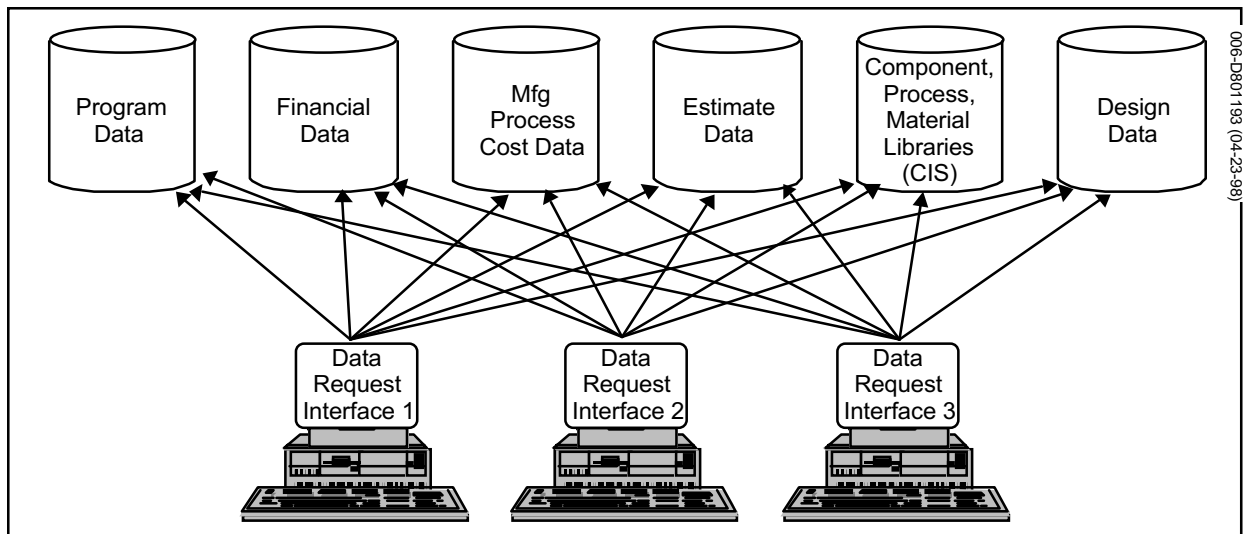


Figure 6-6. Large Scale Point-to-Point Data Integration Complexity

In summary, the VDA was characterized as:

- A coherent source of data to the client-side application
- A homogeneous platform for the acquisition and transformation of all types of data from heterogeneous sources
- An intelligent agent for efficiently gathering data based on the business logic that can be made part of the VDA

By providing a transparent mechanism for managing data interfaces, the VDA has shown the potential to reduce the cost and complexity of managing data integration across a given business unit, and ultimately across enterprises. For the JMD prototype, the VDA enabled the exchange of data necessary to the Pro/E and Cost Advantage tools, as depicted in Figure 6-7.

This capability was fully extensible to the broader scope of integrating all the data necessary to support a full IPT, as shown in Figure 6-8.

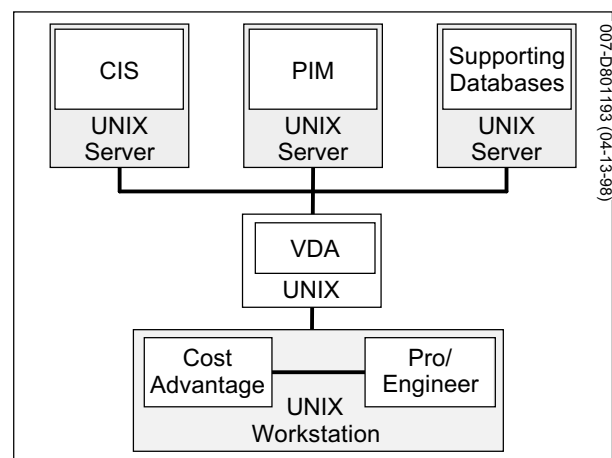


Figure 6-7. Scope of the JMD VDA Data Integration

6.3.3 Implementation of IPT Cost Modeling

6.3.3.1 Cost Estimation Using Cost

Models. Design, cost and manufacturing engineers on the module IPT identified the need to produce usable, documented, and traceable cost estimates in some automated

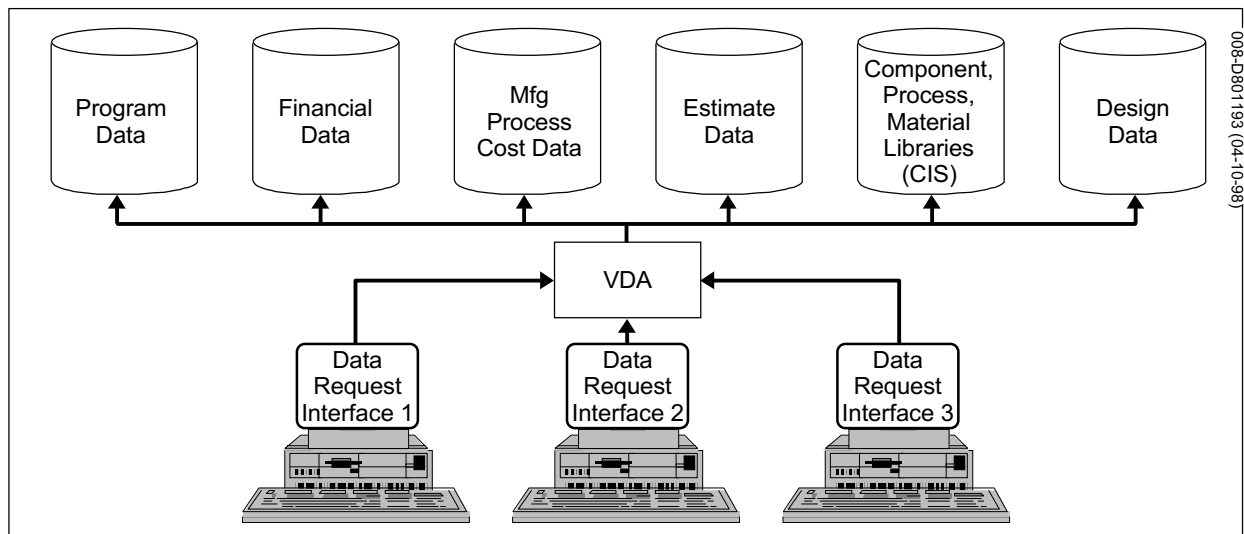


Figure 6-8. Scalability of the VDA Data Integration

manner using controlled and validated cost models.

The IPT felt that the basics for such a need for a cost model included at least:

- Being database-driven as a means to control and configure deployment and to facilitate changes without re-engineering the cost models for:
 - Descriptions of the manufacturing processes used to model the production of a class of parts or components
 - Current (or historical) values that characterize the descriptions of manufacturing process used in the cost model
 - Current (or historical) costs of purchased parts included in the indentured assembly drawing depicted by the cost model
 - Tailoring information such as rates, factors, build quantities and time frame for a particular program
- The automatic capture of the design characteristics and parts list of a part or component that affect the fabrication or assembly cost
- Producing a usable estimate of unit manufacturing cost in a relatively short period of time (minutes or less for a simple design alteration)
- Producing repeatable results, and tailorable reports
- Validation of subassemblies for integration into next-level design assemblies, in accordance with some designated product structure information
- Support for a wide spectrum of the design life cycle using models that:
 - Enabled “make from similar articles” with simple additions/deletions during the early stages of a design when drawings or design details were simply not available
 - Enabled design details to be turned on/added as the knowledge about the

design solidified

Based on the above, and for the purposes of assessing the applicability of the various vendor supplied (COTS) tools that were available for generating cost estimates, several additional features were identified and are given below.

1. Have feature “bandwidth” (selectable features or characteristics) that include the critical design trades that a design or cost engineer might invoke to test the sensitivity of the cost of a design. Design features/characteristics include but are not limited to materials, part types and part characteristics (such as form, reliability, testability, availability, preferability, size, weight, etc.), manufacturing characteristics and processes required as a function of part types and part characteristics, etc. For example, to enable the substitution in a milled part of a simple drilled hole for a stepped hole, the cost model must include and be able to select between the manufacturing implementation that contains formulas/equations for these two features, track the design configuration of this and other selections as they are being made, all while dynamically computing a total cost for the entire design/cost model.
2. Present both a screen and hard copy detailed cost breakdown of a design in such a manner that the costs of features are not only visible but accessible for changing. The presentation of information must be detailed and appropriate for the determination of cost “hot spots,” so that design analysis and trades will naturally focus on features that have the greatest impact not just on reducing cost but on those features that might increase costs if the manufacturing process behind them is not properly monitored/controlled. The depth of detail must correspond to the degrees of control that are potentially available to the various IPT members to influence and drive down costs. For example, if inspecting a complex assembly requires special equipment, special handling, special environments, etc., then these have to be modeled so that they are part of the cost breakdown and cost presentation.
3. Be credible and reasonably accurate for IPT team member buy-in (i.e., acceptance and utilization). This means that the cost model must be validatable, preferably by being able to accurately replicate the actual cost of previously built articles, and the cost model must provide sufficient visibility to the detailed elements of cost that are generated by the cost model to enable IPT team members to correlate cost data with the actual factory or manufacturing processes that have been captured. For example, if a test engineer does not see testing as an element of the cost model and detailed cost breakdown, then the model will understandably lack credibility and will have appeared to have excluded testing from the consequences of the design trade space.
4. Be repeatable, especially in light of the run-time operator judgments and run-time selections of options that the model exposes to the design/cost engineer as a means to configure the cost model for optimizing the hardware design. As important to the actual run-time configuration of the cost model are the

configuration of the authored equations/knowledge in the cost model, and the other external databases that can be consumed in those equations to calculate costs. All of these can be under a constant state of change to reflect the most accurate present state of external conditions (i.e., manufacturing improvements, rates and factors, component prices, yields, reliability data, etc.). There are two aspects to repeatability required for the different stages of the design/manufacturing life-cycle: short-term and long-term.

5. Produce revised estimates with turnaround times of minutes for simple changes to not more than a day for more complex changes that may include a roll-up from the lowest level to the highest level of an indented assembly.
6. Be capable of interface with electrical and mechanical design (CAD) tools that generate/capture design features born in this part of the design process and that are design trade variables that need to be globally visible to the overall cost estimating enterprise so that they can be inputs for consumption in a cost model; inputs which might otherwise have to be manually entered into a cost model. This interface can be either a real-time (direct) link for a more interactive rapid flowdown of design changes into a cost model or a non-real-time (indirect) link, in which case some level of manual intervention may be necessary. In either case both interfaces require the design tool to expose the features database for export to other applications such as the cost modeling tool. This subsequently requires that the cost modeling tool be receptive to importing the data for consumption by the cost model equations. As concerns repeatability and validation, the features database and the design tool/model that generates it add one more dimension to the management of databases and the configuration of an integrated two-tool system. A direct link is optimal and most desirable; however, the indirect link would suffice for the early stages of methodology exploitation.

The PDM Graphical User Interface (PGUI) was initially chosen in order to provide the IPTs with a user friendly method for launching cost application software. Unfortunately, the PGUI was very platform specific (i.e., UNIX based), the result of a proprietary development that produced a reusable element for Raytheon's tailored PIM Sherpa product development. In the early stages of the JMD methodology development, it was an expedient choice for a GUI because other architectural and design issues were more important to resolve. However, subsequent COTS developments in both the intra- and Internet technologies and the GUI technologies offered such significant benefits for an architectural upgrade that the industry CORBA standard was subsequently adopted for the JMD program. Figure 6-9 illustrates the JGUI, which is further described in Appendix B – Detailed Technical Specifications, under A.5.1, JMD GUI (JGUI) and in Exhibit I – JMD Operational Test Procedures. (Note the name change from PGUI to JGUI for JMD Graphical User Interface.)

6.3.3.2 Cost Modeling Configuration Management.

The utilization of a cost estimating model produces the need to configuration manage each variant of the hardware design in some way. If there are N non-orthogonal (i.e., intra-dependent) design-trade variables in a trade-space, then there are many combinations of

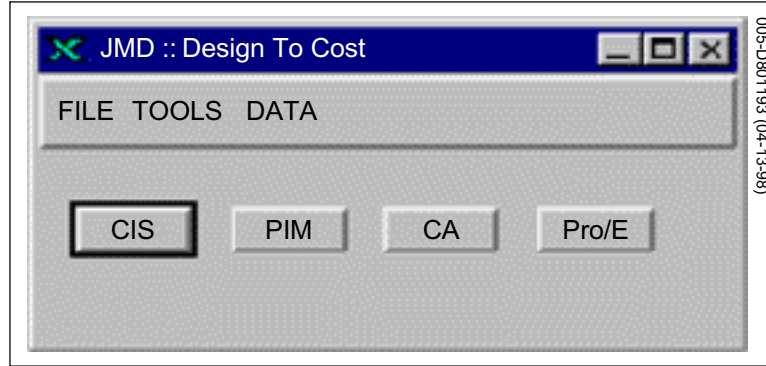


Figure 6-9. JGUI's Main Screen

1, 2, up to N-at-a-time permutations that might be tested by the design/cost engineer. At some point in time, some of these designs may be discarded in favor of those design approaches that have cost merit and cost credibility for a continuing evolution of additional trades that will ultimately lead to the final design. Even those designs that have been discarded may need to be archived and retrievable for some period of time until such time that a final design has ultimately been declared.

Eventually drawings would be generated and the most favored design cost model would be updated, and perhaps even the cost model itself revised to reflect the new wealth and depth of the detailed knowledge now available about the design.

In the foregoing utilization scenario of a design-trades life cycle, some special configuration management requirements, over and above the normal requirements for a product development, were identified:

1. Develop and track multiple alternative designs and their corresponding cost estimates without releasing a new product version
2. Release a new product version incorporating the then-current “approved” design and cost model results
3. Retrieve previous versions of a product at any level of an indentured parts list (IPL) for review or reuse in a subsequent program
4. Incorporate and associate the full detail of the design (Pro/E or other model) and cost (CA or other model) with each configured version

6.3.3.3 Cost Database Accessibility. Just as a features database was identified in the export/import link to the cost model with a CAD tool four broad classes of cost data were organized into databases to support DTC cost estimation requirements. Each class of data had multiple sources or databases within the existing IT framework. Where practical, a common format or “data warehouse” was identified for the integration of multiple sources. Where such integration was not practical, a VDA was used to create the appearance of an integrated database.

The following four classes of cost data were addressed in the JMD IT Prototype architecture:

- *Manufacturing Process Library* – identifies and describes the manufacturing processes available at a specific facility.
- *Machine and Assembly Cost Factors* – each entry in the manufacturing process library must contain the applicable cost factors in a form usable by the cost estimation tools.
- *Parts Cost* – contains part cost information and the relationship of cost to lead time, quantity and other considerations. CP/CPK figures by tolerance should also be supported.
- *Programmatic/Business Factors* – contains factors for labor rates, overhead factors, build rates, learning curve position, etc., applicable to a given program and product.

Manufacturing or Assembly Process Characterization. This data provides the cost model framework for a particular manufacturing or assembly process with the current (or historical) values that are needed to produce an estimate of the unit manufacturing cost of a specific part or assembly. For a complete DTC capability, manufacturing processes are required to include fabrication, assembly, integration and test. The IPT must be able to view the available processes as a process library, from which team members can select and locate the particular machine, assembly or other cost element. Exhibit D of Appendix B provides more detail on the implementation of the manufacturing process characterization.

Parts Cost. The cost of each purchased part (from an outside vendor or Raytheon cost transfer) to be included in an assembly must be available. These costs may vary by lead time, quantity and other considerations. Sigma and DPU data should also be accessible.

Programmatics and Business Factors. Program and business factors—rates, overhead factors, build quantity and build rate—that affect product cost must be available. In addition, the position on and slope of the “learning curve” may also affect the cost estimate.

6.3.4 Virtual Database Agent (VDA)

6.3.4.1 VDA Overview

Object Oriented Technology Concepts. Object technology was fundamental to the implementation of a VDA. The notion was that data and process (code) were linked into a single entity called an *object*. An application program did not directly access data from storage. Instead, the application program got data by calling a method associated with the required piece of data. A method was simply a procedure or subroutine that was *bound* to the object, and had the job of delivering the value of the requested data item. The calling application program, sometimes referred to as a client, had no need to know — and should not care — how a method got the desired data value. It was the job

of the object to deliver the desired data to the application any way it could. Figure 6-10 is a simple abstraction of an object.

The characteristic most important for a VDA is called *encapsulation*. The only way for an application outside the object to obtain the value of a data item was to call a method associated with that particular data item.

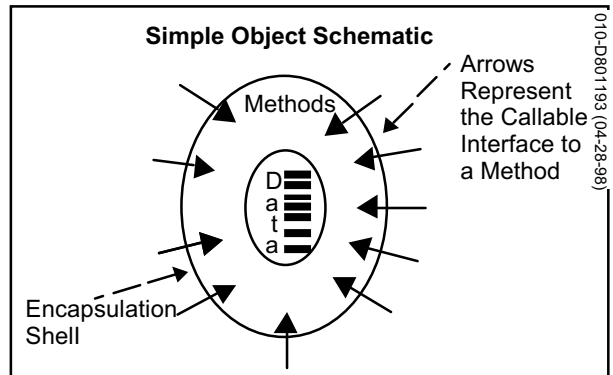


Figure 6-10. Object Schematic

Leveraging Encapsulation. Encapsulation offered significant advantages over direct access to the data. Given a request for data, it was up to the object itself to figure out how to get and deliver that data. That delivery method actually executed some complicated code to produce the required data. For example:

- The required data was locally or internally stored within the object and simply delivered by the method.
- The required data was known to reside at a remote location and the object executed a remote process to retrieve the data and deliver it.
- The data was available in a format compatible with the calling application; e.g., a CAD program that was expecting a key-value parameter string.
- The data had to be transformed by the object to conform to the format required by an application. For example, data stored using a relational database management system (RDBMS) was accessed by a CAD program that expected a key-value parameter string.

Providing Different Interfaces to the Same Data. Because encapsulation restricted data access to the use of methods, it was possible to group methods into coherent sets that presented the same data in different formats. This grouping of data was called an *interface*. Thus, one application may request data using a traditional structured query language (SQL) access method such as Microsoft's ODBC. Another application, such as a CAD program, might want data delivered as a drawing object. The software designer can create interfaces that provide these very different windows on the same underlying data contained in an object. The schematic for an object shown in Figure 6-11 illustrates the grouping of methods into interfaces.

Combining all these features created an object that could gather data from different sources, store some data locally if necessary, and retrieve other data on demand. An object also appeared to have the characteristics of different data sources depending on the needs of the application requesting data. It was this object that was being defined as a VDA.

The single layer “encapsulation shell” depicted in Figure 6-11 appears simplistic. In reality there are several layers. Those layers closest to the data implement methods to access/format data, whereas those layers on the outside implement methods more related to the communications/networking of objects. In between, there can be layers that provide the generic functionality, such as that described in the following sections.

When JMD prototype development first started, a COTS software implementation of a VDA with the required functionality simply did not exist. Extensive industry activity under way at the beginning of the JMD program has since resulted in applicable COTS software becoming available. The following functionality descriptions for a VDA are relevant for both the Raytheon VDA developed in the first phase of the JMD prototype and the subsequent upgrade that incorporated the COTS CORBA software that became available in 1997.

6.3.4.2 Desired VDA/CORBA Functionality

Administrative Methods and Intelligent Agents. If a VDA were simply to provide a coherent but passive interface to an application, it would not be much more powerful than a well-defined standard Application Programming Interface (API). However, the VDA contained code as well as data. It acted as an application program or client, thereby becoming an *intelligent agent*. The VDA might then call other VDAs, applications or data sources to perform a wide range of access, scheduling or error processing functions. To act as an intelligent agent, the VDA had to be able to receive instructions on what to do in various situations. This was accomplished by adding to the data interface methods a body of administrative methods and the associated self-contained storage that let an outside program or programmer provide the appropriate instructions. For example, an administrative method could schedule the polling of a data source on a regular basis or select an action in the event a data transmission fails. Finally, the VDA contained a set of methods that allowed an outside program or programmer to discover everything about the VDA’s state.

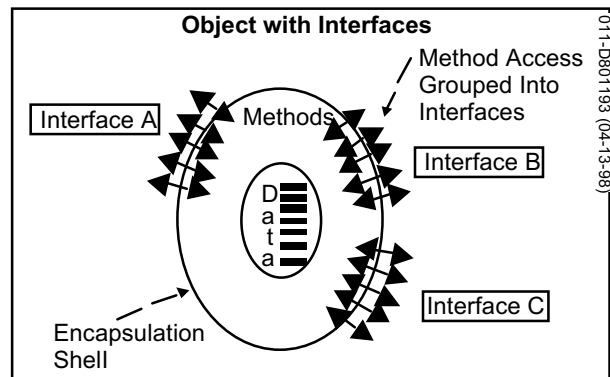


Figure 6-11. Object with Interfaces

Events and Event Brokers. A special kind of method, called an *event*, is very useful in an environment where a number of VDAs or other objects exist and must work together.

An event is basically a reverse method where the VDA or other object publishes an occurrence of a particular event. This notification of the event is then available to any other object that may be interested in the event as a trigger for some other processing upon its occurrence. For events to work there must be an *event transaction broker* that listens for events and knows what objects, programs or VDAs may be interested in them. The broker invokes a method in the interested object to signify that the particular event has occurred.

The use of events and an event transaction broker was not required to implement a VDA, but their capabilities greatly simplify the design and architecture of systems that incorporate VDAs. Event handling is similar to encapsulation, because it makes the linking of components in such a system connectionless. For example, a VDA can send an event message saying it needs some data from a particular data source. The broker takes the message and stores it until the processor with that data source is on-line and available. When that data source is ready it can send data back to the broker, which will forward it to the requesting VDA.

6.3.4.3 Important VDA Features

Interfaces

The VDA was required to support a number of different interfaces. Its primary interface was to the target application itself. Requests from the target application were interactive and focused on the application's processing requirements. For example, a VDA supporting manufacturing cost models delivered cost, manufacturing and design data in a "key-value" parameter format. Since this data was accumulated from many sources, it was "buffered" in the VDA. To accomplish this, the VDA gathered larger sets of data covering a best guess of less specific data required for a particular cost modeling session (i.e., a domain-limited subset of the global cost database). The VDA retrieved remote data on a near real-time (adequate for interactive processing) basis if the necessary data had not already been successfully buffered in such a domain limited best-guess set. Specific data availability and cycle time requirements were used to determine the best approach for each data type. In addition to the direct target application (in this case a cost model), future VDA capabilities should support a related interface to allow larger sets of data to be accessed by a standard reporting program. This might be done by supporting a general RDBMS API such as ODBC.

The idea of an interface was quite general. By simultaneously supporting more than one interface, the VDA acted as a heterogeneous data supplier. It fit the needs of any application program requiring data in a format supported by the VDA. The VDA allowed new interfaces to be defined and methods to be provided for those new interfaces. The ultimate scope and number of interfaces that may be supported cannot be predicted until the prototype VDA has been beta-tested in a controlled environment focused on benchmarking user requirements on real design developments and production.

Access Routines

On the "back end" (i.e., the data source/inner side of the interface), the VDA had access routines to the various data sources that it needed to support the requirements of the target application. The ability to add a set of access routines is modular so that new sources can be added as needed. As with interfaces, industry standard access routines such as ODBC were supported whenever possible. The largest challenge was the creation of routines to access data embedded in programs or archived in non-standard formats.

Scripting

The VDA required a flexible scripting language to connect interface methods to the access routines, to deal with scheduling and to handle events. Some characteristics of the scripting language were:

- Was the glue between the interface methods and the access routines, producing a coherent set of actions when an interface method was invoked.
- Handled a set of asynchronous events and executed scheduled procedures. In

order for the VDA to be efficient, it was unrealistic and potentially impossible to synchronously access all the necessary data sources required for a data request. Instead, some data was locally buffered within the VDA.

- Scheduled procedures to investigate data source changes and retrieve updated data when changes had occurred. In addition, some changes in data sources may be asynchronously reported to the VDA. The scripting language allowed procedures to be written to handle these events.
- Was available as an off-the-shelf technology.

Persistence

Theoretically, the VDA should have no persistence; that is, there should be no need to store any data or state information once the client applications it served are no longer active. However, because of the requirement to find, open and access remote data sources the lack of persistence results in inefficiencies. The inclusion of an internal data-store with persistence significantly improves the efficiency of the VDA by providing data integrity and synchronization capabilities. For example, when the local persistent store recognized that the remote state information or data had changed, it could refresh itself appropriately. However, this local persistent store now had to be included in the configuration management scheme, and although this added complexity, it may be determined to be required in any practicable cost-estimation implementation of the VDA.

Push/Pull Data Buffering

In its simplest form, the VDA operates in a purely “pull” mode. Data is gathered by the VDA on demand from a client application. Unfortunately, some of the data sources that the VDA accesses are not formal or well organized. For example, data can be embedded in manufacturing process software or resident on individual workstation spreadsheets. In such cases it is easier to use the event capability to “push” data from the remote data source to the VDA when certain events occur. The data is then held in a local store until it is accessed on a “pull” basis by a client application. Handling the conversion from push to pull is one of the greatest strengths of the VDA concept.

6.3.4.4 VDA Prototype Architecture. Figure 6-12 shows the prototype IT architecture, which includes two VDA modules.

Both of these modules have three significant components:

- The client-side and backside interfaces
- The business logic that determines how and where to route requests

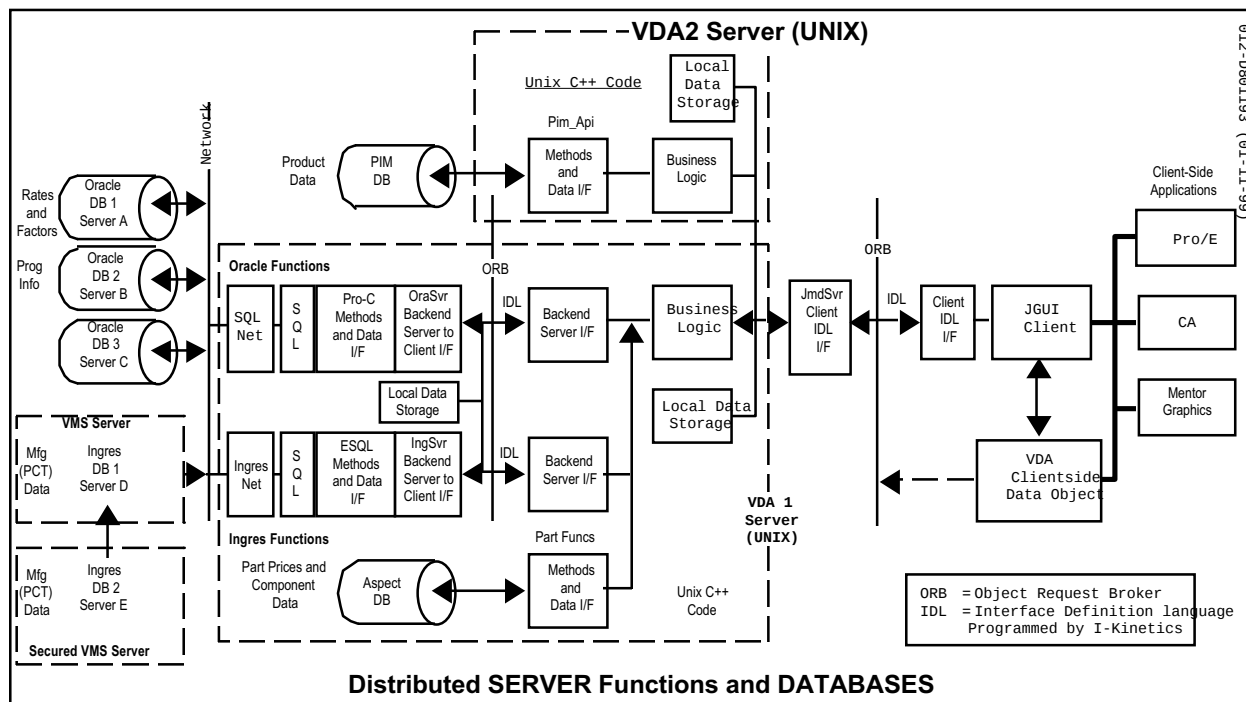


Figure 6-12. Prototype IT Architecture

- The methods that translate and format the requests to extract/deposit data from/to the target databases

It is very important to note that within the VDA modules are large segments of functionality (methods and data I/Fs) that are accessible on an ORB, which means that any computer on the network using the IDL in the ORB depository can access and use the same functionality from anywhere on the network. In addition, the VDA modules can be replicated many times, or can be tailored for specific business logic, and also reside on the same or different servers to act as the middleware between the client applications and the backend database servers. The JMD IT architecture is robust, modular and scaleable, to match requirements for many simultaneous users and for a graceful productization of the JMD capability.

As shown in Figure 6-12, VDA Module 1 “translates” cost and configuration data from the four classes of cost related data (Rates and Factors, Program Information, Manufacturing (PCT) Data, and Parts, Prices, and Component Data) into a form usable by the cost estimating tools. VDA Module 2 translates data produced by the integrated design/cost estimation tools into a form usable by the CM repository and cost rollup tools.

Additional detail on the IT architecture of the two VDA modules is contained in the following subsections.

VDA Module 1 – Cost and Parts Data to Estimating Tools. Figure 6-12 illustrates the multiple database connections established by VDA Module 1. VDA Module 1 operates on a “pull” model. The client-side applications request the needed data through VDA Module 1, which in turn extracts the data in the form needed by the client-side application.

Several data objects are required by the client-side applications Pro/E, CA, and Mentor Graphics. To avoid the need to modify these application tools, the data objects are delivered as sets of files into the UNIX environment on the client-side, shown as a “client-side data object” in Figure 6-12. These files are located in a particular subdirectory of the UNIX file system accessible to the client-side applications. Separate subdirectories are created for each “class” of part or assembly to be designed or estimated, which as an entity become a “data object library” on the client-side. For example, there would be separate subdirectories for NC milled parts, electronic assemblies and mechanical assemblies. Once this data is on the client-side, it represents a small snapshot of the gigabytes of data stored on the backend servers, and until other requests for snapshots are made, this client-side data is in a controlled frozen state for the duration of the cost-estimating exercise.

At this point in time, the construct of the client-side data object has been kept simple and is in the form of “flat files.” Nothing in the JMD prototype IT architecture precludes these files from becoming a true hierarchical data object that resides on the ORB (as shown by the dotted lines in Figure 6-12). This would enable a client-side data object to be visible and sharable from anywhere on the ORB network, and would significantly ease configuration management and control of a single data object for a program or at the program management level. However, this feature was left as an enhancement for the next phase of the IT architecture development.

One of the more significant data connections established by VDA Module 1 is that of the Ingres connection to the backend Process Characterization Tool (PCT) database. The database construct, for which the ESQL methods shown in Figure 6-12 were developed, is described in Appendix A, Detailed Technical Specifications, Section A.4 – Data Model.

VDA Module 2 – Estimate Result to Rollup Tool and PIM. Figure 6-12 shows a very simplistic VDA Module 2, mainly to illustrate that the VDA itself is modularly expandable to meet the specific needs and uniqueness of such legacy systems as Raytheon’s PIM (COTS) application. Rather than having a large monolithic VDA, the segmenting of it into modules helps to isolate changes and enhances the reusability of those elements that stand alone.

VDA Module 2 is in many respects similar to VDA Module 1 in that it has the three components: the client-side and backside interfaces, the business logic, and the methods that translate and format the requests to extract (deposit in this case)

data from the target databases. However, there are some differences that are worth noting: (1) the VDA Module 2 operates both on a “push” model for archiving the cost-estimate resulting from a cost modeling session on the client-side, and then on a “pull” model for those client-side applications that wish to retrieve that cost data from PIM; and (2) the backend I/F does not have an IDL and is not on the ORB.

Raytheon believes that developing the IDL for the ORB for COTS software, such as the PIM and Aspect in Figure 6-12, is truly a vendor activity and responsibility, and given the tremendous industry shift towards the CORBA standards being developed by the industry itself, it is simply a matter of time for all vendors to supply an IDL for their products, not unlike the APIs they currently provide, except that the IDLs will be in a standard protocol that is ORB compliant.

6.3.4.5 Mini-VDA Demo. The following describes the VDA mini-demo completed in August 1996. The mini-demo illustrated the feasibility and use of the VDA architecture as a robust solution for the integration of applications and databases for the Design-to-Cost (DTC) enterprise, and the full demo is targeted for a major radar assembly using multiple process and cost models to roll up the cost of the indentured subassemblies.

VDA Module 1. The scope of VDA Module 1 was limited to accessing the backend server databases for the data required by the pilot program (HDMP T/R module design/development) cost models and the Active Array Radar Antenna assembly, which uses the T/R modules.

Output

Figure 6-13 is a sample VDA Module 1 output to CA for the NC Milling process.

Inputs

Inputs for VDA Module 1 varied based on the data that was being requested from the VDA and the calling application or tool. A message protocol consisted of a set of formats, rules and standards that defined the contents of messages between two communicating processes. These contents included general information such as user, message type, and length, as well as transaction-specific information. All requests for data from the VDA followed a standard message protocol. The protocols for the JMD IT architecture are described in Appendix B of this document.

VDA Module 2. The scope of VDA Module 2 for the mini-demo was to archive the cost estimates generated for the various part numbers associated with the pilot programs, which were generated from the cost modeling exercises on the CA application on the client-side. For the full demo, the file sets produced by the Pro/E and CA applications are to be archived on PIM. The Pro/E outputs will be delivered as a pair of .PRT and .DRW files containing the design model and its display representations.

The CA outputs will be delivered as the set of files shown in Table 6-4. Point estimates are delivered by a direct procedure call that references the part number.

Table 6-4. CA Output File Types

File Types	File Contents
.est	Cost Estimate Instance
map.*	Mapping to Pro/E
.prc	Process Characterization
.mat	Material Cost
.ppr	Part Price
.rat	Rates and Factors
.pgm	Program Information

6.3.4.6 Planned Full VDA Demo. The example below illustrates a typical use of the tool and data integration as had been planned for the JMD full demonstration. The

<u>Process Characterization DATA</u>					
Manufacturing Process: NC_Milling			Fabrication Labor	D	E
Machine Type: Matsuura 800			Assembly Store Room	F	G
Material Cost: \$2.00/lb			Inspection	H	I
Material Type AL			Production Support	J	K
			Implementation	L	M
Coeff.	Units	Feature			
A1=1.0	Minutes/Bits	Hole(s) Location	Quality Pool	N	O
A2=.02	Minutes/(Depth*Bits)	Hole(s) Material Removal	Secretarial Pool	P	Q
A3=1.0	Minutes/(Depth*Bits)	Contour (Material Removal)	Secretarial Pool	R	S
A4=1.0	Minutes/(Depth*Bits)	Chamfer (Material Removal)	Management Pool	T	U
A5=1.0	Minutes/(Depth*Bits)	Round (Material Removal)	Management Pool	V	W
A5=1.0	Minutes/Bits	Cosmetic Location			
A7=.05	Minutes/(thrds/in.*Th read #*Thrd Dia.)	Thread Cutting	Fringe Dollars		X
			Overhead Dollars		Y
			Fabrication Raw and Bulk		Z
<u>Rates and Factor DATA</u>			Material		
	%	\$/Hour	Material Burden		AA
		%	Supplier Material Burden		BB
Touch Labor		A	G&A		CC
Labor Attrition	B	C			

Figure 6-13. VDA Module 1 Outputs to CA for the NC Milling Process Inputs (Data for Display Purposes Only)

scenario is the design of a mechanical part for the tile array module. The two primary tools used in the example are the mechanical CAD tool Pro/E and the cost estimating/design advisor tool, Cost Advantage. Three additional applications/databases provide supporting data. Included with each user action is a description of the events occurring behind the scenes. This scenario assumes that a prior modeling session (the mini-demo) has been completed, in which the IPT cost engineer has:

- Created the needed CA knowledge base and cost models using the CA application tool
- Used the VDA to retrieve part and process data appropriate to the design being analyzed (rates and factors, process characterization, programmatics, part prices, etc.).

Step 1. The user clicks on the JGUI toolbar on his desktop to retrieve the Pro/E solid model or drawing and the corresponding CA cost model framework with which he wishes to work.

- A JGUI script is executed that initiates a request to a VDA for the model or drawing.
- Upon receipt of the request, the VDA retrieves the models from the appropriate sources and copies them to the appropriate user directory.

Step 2. The user clicks on the JGUI toolbar on his desktop to initiate a session with the Pro/E and the CA in a linked mode.

- A JGUI script is executed that opens and links both applications.

Step 3. From within Pro/E, the designer opens the model representing the part and activates the “costlink” from a pulldown menu.

- The associated CA cost model framework is automatically opened.
- CA retrieves the information associated with the current design model and displays the cost rollup.

Step 4. The user begins design modifications to the Pro/E model (see steps 7 and 8 for conditional requirements/alternate procedures).

- Costlink updates the CA features with the changes and CA then computes costs and gives design advisories (such as “outside desired limits”) if applicable. Design advisories, and the actions permitted as a result of an activity or parameter exceeding the knowledge base thresholds, are strictly a function of the intelligence authored into the cost model by the cost model developer.
- If design advisories were issued and specific user directions given, CA records these actions for subsequent reviews.

Step 5. The user reviews the estimate and determines if the design meets cost and performance goals. If it does not meet cost goals, step 4 is repeated until the cost goals are either met or the lowest cost has been achieved within the limitations or

capabilities of the cost model. Whenever a design and cost are to be saved for continued refinements and approvals, a configuration designator is assigned to both the design and cost models so that they can be archived without destroying the predecessor configurations.

Step 6. The user requests design advisory override reports and reviews them with the manufacturing/process engineers for possible impact or suggested alternatives.

Step 7. If any design feature, process data, or part selected is determined to be an unknown to the cost model because: (1) the existing client-side “snapshot” of the databases does not have the required data, or (2) the cost model itself does not have the necessary processes wired into its manufacturing/assembly schematic, then one of two things must happen next. For (1), the user must initiate the update of the client-side databases using the VDA; for (2), the cost modeler must update the cost model to include the additional intelligence required for the process/features required. This type of updating is done outside of the design session as follows:

- For (1):
 - A JGUI script initiates a request to the VDA to retrieve the data type selected on the JGUI menus (i.e., part cost data for a particular part from CIS)
 - The VDA’s business logic determines the backend database server and flows down the request
 - The VDA receives the information and formats it for the return to the JGUI, which then puts it into the client-side data object so that it is available for the next design session and will no longer be an unknown
- For (2):
 - The cost model is revised, its configuration designator bumped up by one, and the latest release is archived into PIM so as to be available for step 3.

Step 8. Designer returns to step 3 now that the unknowns have been resolved in the databases and the cost model.

Figure 6-14 updates the initial interfaces of Figure 6-8 to illustrate the cross-application interfaces necessary to support the JMD demonstrations. Table 6-5 describes the tool/data interfaces planned to support the JMD demonstrations.

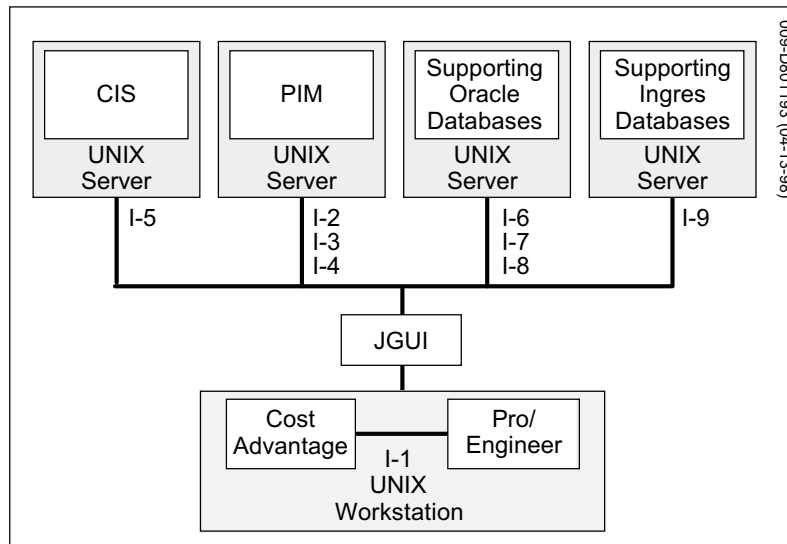


Figure 6-14. Cross-Application Interfaces

Table 6-5. Tool and Data Interfaces for JMD Demonstrations

Path ID	Requester/ Initiator	Provider	User	Data Description
I-1	Cost Advantage	Pro/E	Cost Advantage	Pro/E model information
I-2	JGUI	PIM	Cost Advantage	Indentured parts lists
I-3	JGUI	PIM	Pro/E	Drawing/model files
I-4	JGUI	CA/rollup tool	PIM	Estimating tool outputs
I-5	JGUI	CIS	Cost Advantage	Part data including cost and sigma data
I-6	JGUI	Oracle database	Cost Advantage	Process data including cost and sigma data
I-7	JGUI	Oracle database	Cost Advantage	Labor, overhead and other rates
I-8	JGUI	Oracle database	Cost Advantage	Programmatic data such as build quantity
I-9	JGUI	Ingres database	Cost Advantage	Manufacturing data captured by Raytheon's Process Characterization Tool (PCT)
All	JGUI	I-Kinetics VDA	Cost Advantage	Virtual database agent's object request broker

6.3.5 Lessons Learned

The JMD IT prototype development effort and the methodologies demonstrated clearly indicate that the principal objectives of the program were met and worthy of being the foundations for a future productized version of the concepts and capabilities. This conclusion is summarized as follows:

1. *A reduced number of sources for data through the use of common libraries and common repositories.* The tools of choice that had been selected for global use within Raytheon, provided an adequate and sufficient set of information sources and destinations for the development of common design and cost models. The common libraries and repositories established a manageable and effective framework for data connections and infrastructure development.

2. *Improved access to commonly used data, first through a direct integration and second through a “virtual” integration of data.* The VDA, with its capabilities for determining requested data access and formatting, provided an effective single-point interface between the client-side data consumers and the back-end data suppliers.
3. *Enhanced information exchange between applications and tools used by the functional organizations.* With the common tools of choice forming a solid basis for use within Raytheon, and the exchange and interface of data among them provided by the VDA, the functional organizations were able to proceed with a full demonstration on the JMD program. The benefits of standardization as a result of these enhancements were very important to promoting confidence in the data being generated and shared, as well as acceptability of the results that were being cooperatively generated.
4. *Enterprise wide support will take dedication and continual maintenance.* Experience on maintaining many UNIX based CAD and cost programs had shown that version changes and operating environment changes necessitated the need for continual updating of software connectivity by system administrators.

The value of COTS and of standardizations toward the successes of the JMD program cannot be over-emphasized. For example, the CORBA industry standards that were utilized for the IT infrastructure have shown that the VDA can be made very independent of the nature and characteristics of the client-side and back-end servers, and thus can achieve much greater reusability among the JSF community. The experience on this program has shown that CORBA implementation can be effectively created by outside suppliers such as I-Kinetics. Use of CORBA experts minimized the cost and schedule which is a key reason for selecting this standard. The details of Raytheon's JMD implementation of the methodologies have been provided in the Appendix as a means to assist in the establishment of similar capabilities by other users. Although the Ingres and Oracle back-end CORBA interfaces were not completed to the extent of being debugged or demonstrated, the client-side CORBA interfaces clearly demonstrated the benefits and simplicity of utilizing the CORBA industry standards for this application. It was not our intention to fully explain the details of the CORBA standard within this report. References are provided in the reference section of this report.

This page has been intentionally left blank.

JAST/JMD Program:

1. Integration of engineering design data with cost data. This included, where necessary, creating new repositories for design or cost data to facilitate their integration. This covered the following data types:
 - Part cost data
 - Manufacturing process coefficients
 - Rates and factors
 - Programmatic data
 - Cost estimates
2. Provision for a common, user-friendly graphical user interface that allowed common and simplified access to the engineering design to cost tool set

The JMD application integration effort consisted of three basic components:

- Graphical user interface (GUI)
- Virtual database agent (two modules)
- Database and communication servers

These will be described in detail later in individual sections.

A.4 Data Model

The data model consisted of two major elements.

The first was the classification and description of all parts used by the JMD tile array. This was represented by the class structure defined in the CIS Explore product. Figure A-2 illustrates the relationships between the major classes and associative entities. For a more detailed listing of the class structure see A.8.1, Exhibit A (CIS Explore Class Structure) of this Appendix. This model was altered in two ways:

1. The addition of the following class properties that were attached to the internal part number, manufacturing part number and parts lists classes and inherited through all their subclasses. The properties were:
 - Part price: cost of part
 - Part price type code: type of price (estimate, quote, etc.)
 - Part price type qty: quantity at which price is determined
 - Part price date: date price was given
 - Part yield: first time yield

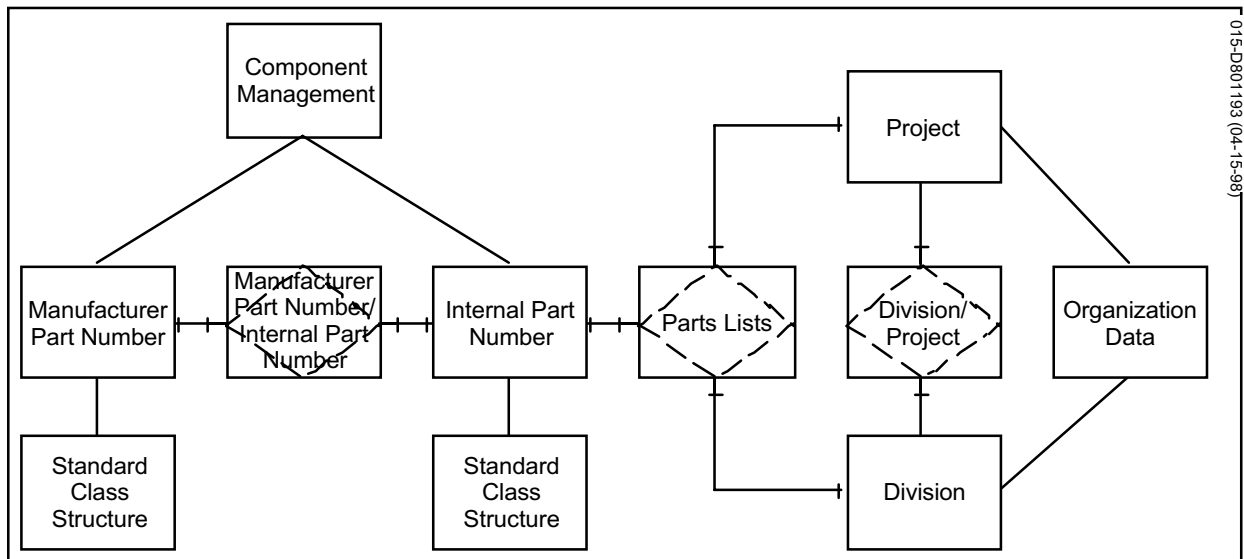


Figure A-2. CIS Explore Classes and Associations

2. The addition of Raytheon specific classes and properties that were necessary to accurately describe the tile array parts and their associated properties

The second element of the data model was the data necessary to support the estimating process. This included the following Oracle types of data. (See A.8.2, Exhibit B – Data Dictionary and A.8.3, Exhibit C – Oracle Data Definition Language)

1. Corporate Purchase Agreements
2. Rates and Factors
3. Programmatic

A description of the backend server Ingres database that was created by Raytheon's proprietary Process Characterization Tool (PCT) and the client-side table functions required to access that data that the VDA had retrieved across the networks is detailed in A.8.4, Exhibit D – PCT Interface Specifications.

A.5 User Interface

A.5.1 JMD GUI (JGUI)

The JGUI was a Java Windows-based application designed to provide a common integrated user friendly interface to the tool set used by a design engineer. This application was developed using the ORBIX WEB Development Software and replaced the PDM I (PGUI) used in the early part of the JMD program development phase.

Table A-2 details the GUI functions and associated programming methodologies.

Figure A-3 illustrates the interfaces between the JGUI and the tools it supported, and the databases and files accessed by those tools.

Table A-2. GUI Functions

Function	Technique
PIM Access	Remote shell script to start the PIM application from remote production server
CIS Access	Remote shell script to start the CIS Explore 2.5.3 from remote server
Pro/E Access	Remote shell script to start the Pro/Engineer application from remote server
CA Access	Remote shell script to start the Cost Advantage application from remote server
Data Requests	CORBA calls to the VDA
UNIX File Editing	Shell scripts to start X-Windows application

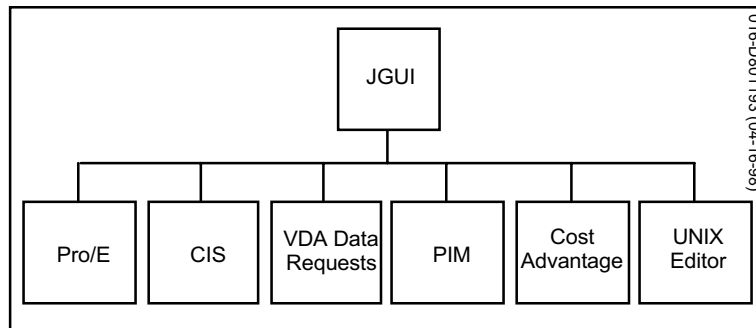
**Figure A-3. JGUI Interfaces**

Figure A-4 shows the hierarchical class structure for the JGUI (CORBA) implementation, which is operationally sequenced in the procedures outlined in A.8.9, Exhibit I – JMD Operational Test Procedures. These procedures were developed to help validate the transition from the old PGUI to the new JGUI CORBA software. The transition to CORBA software affected only the interfaces between client-servers and not the functionality of the VDA or the GUI.

A.6 Virtual Database Agent

Two VDA modules were developed. VDA Module 1 handles all requests for data to be supplied to support the estimating process. VDA Module 2 handles all requests for updating PIM and CIS with new cost estimates generated by the estimating tools and the cost models used to generate the estimates. The general architecture related to the interfaces between the components is illustrated in Figure A-5.

The interfaces to the VDA were broken down into two categories:

- JGUI to VDA
- VDA to database servers

Class Hierarchy	
class java.lang.Object	
class IE.Iona.Orbix2.CORBA.BaseObject (implements IE.Iona.Orbix2.CORBA._ObjectRef)	
class JMD.PIM (implements JMD._PIMRef)	
class JMD._boaimpl_PIM (implements JMD._PIMOperations)	
class JMD._tie_PIM	
class JMD.VDA (implements JMD._VDARef)	
class JMD._boaimpl_VDA (implements JMD._VDAOperations)	
class JMD._tie_VDA	
class JMD.ClientMsg (implements IE.Iona.Orbix2.CORBA.IDLCloneable,	IE.Iona.Orbix2.CORBA.Marshalable)

```

class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
class java.awt.Canvas
class JMD.MainPane1
class java.awt.Container
class java.awt.Window
class java.awt.Dialog
class JMD.utils.FileMesg
class JMD.utils.Mesg
class java.awt.Frame (implements java.awt.MenuContainer)
class JMD.EcuGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)
class JMD.utils.FileFrame
class JMD.IplGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)
class JMD.LoginFrame (implements java.awt.event.ActionListener)
class JMD.MainGUI (implements JMD.LoginEventListener, java.awt.event.ActionListener)
class JMD.PartPriceGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)
class JMD.ProcessCoefficientsGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)
class JMD.ProgrammaticsGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)
class JMD.RatesFactorsGUI (implements java.awt.event.ActionListener, JMD.GUIcallback)

class JMD.toolkit.Configuration
interface JMD.DataFilter
class java.util.EventObject (implements java.io.Serializable)
class JMD.LoginEvent
interface JMD.GUIcallback
class JMD.IPLKey (implements IE.Iona.Orbix2.CORBA.IDLCloneable, IE.Iona.Orbix2.CORBA.Marshalable)
interface JMD.LoginEventListener (extends java.util.EventListener)
class JMD.LoginInfo
class JMD.MainApp
class JMD.SecurityInfo (implements IE.Iona.Orbix2.CORBA.IDLCloneable, IE.Iona.Orbix2.CORBA.Marshalable)
class IE.Iona.Orbix2.CORBA.ServerDispatcher
class JMD._dispatcher_PIM
class JMD._dispatcher_VDA
class JMD.ServerResponse (implements IE.Iona.Orbix2.CORBA.IDLCloneable, IE.Iona.Orbix2.CORBA.Marshalable)
class java.lang.Thread (implements java.lang.Runnable)
class JMD.toolkit.CISObject
class JMD.toolkit.CostAdvObject
class JMD.Ecu (implements JMD.DataFilter)
class JMD.Ipl (implements JMD.DataFilter)
class JMD.toolkit.PIMObject
class JMD.PartPrice (implements JMD.DataFilter)
class JMD.toolkit.ProEObject
class JMD.ProcessCoefficients (implements JMD.DataFilter)
class JMD.Programmatics (implements JMD.DataFilter)
class JMD.RatesFactors (implements JMD.DataFilter)
class JMD.toolkit.TextEditor
class java.lang.Throwable (implements java.io.Serializable)
class java.lang.Exception
class JMD.FilterException
class JMD._PIMHolder
interface JMD._PIMOperations
interface JMD._PIMRef (extends IE.Iona.Orbix2.CORBA._ObjectRef)
class JMD._VDAHolder
interface JMD._VDAOOperations
interface JMD._VDARef (extends IE.Iona.Orbix2.CORBA._ObjectRef)
class IE.Iona.Orbix2.CORBA._sequence_Char (implements IE.Iona.Orbix2.CORBA.IDLCloneable, IE.Iona.Orbix2.CORBA.Marshalable)
class JMD.CharStream
class IE.Iona.Orbix2.CORBA._sequence_Octet (implements IE.Iona.Orbix2.CORBA.IDLCloneable, IE.Iona.Orbix2.CORBA.Marshalable)
class JMD.OctetStream

```

Figure A-4. Class Structure for the JGUI (COBRA) Implementation

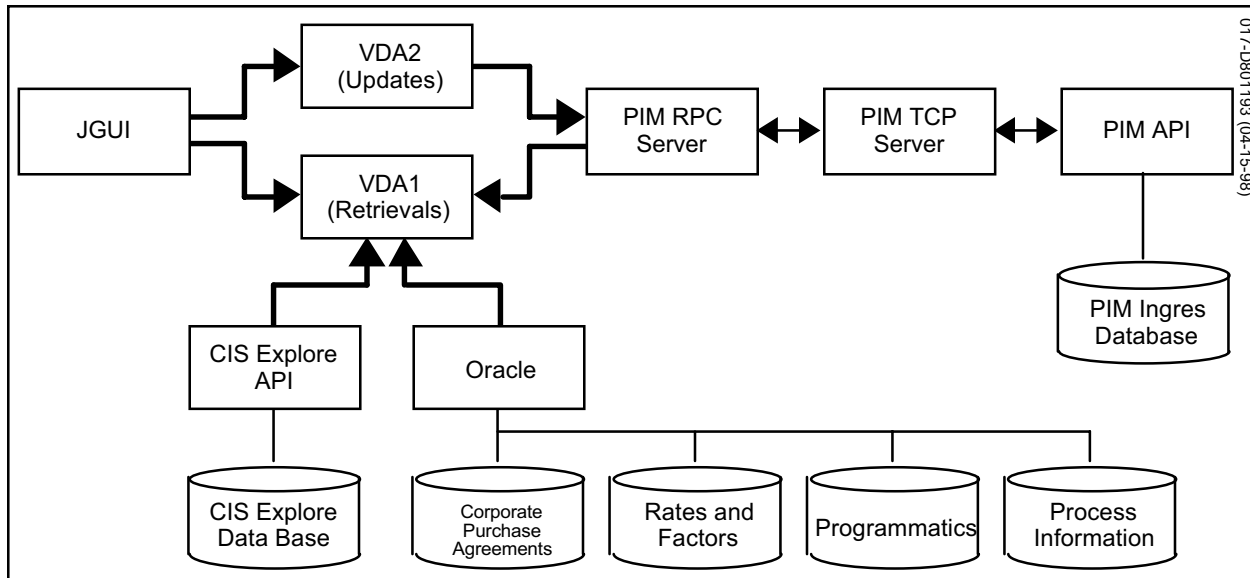


Figure A-5. JGUI, VDA and Database Interfaces and Data Flow

A.6.1 Virtual Database Agent Module 1 (VDA1)

VDA1 provides the interface between the JGUI and all data sources supporting the estimating tool. These data source types are illustrated in Figure A-6. Each type required a different method of interface. The VDA initially used the remote procedure call (RPC) as its basic call structure. This was subsequently “encapsulated” with the CORBA IDL to enable an ORB implementation while salvaging all of the previously developed software. VDA1 was designed to be running all the time waiting for incoming messages. Upon receipt of a message, one of the services in the following sections is performed.

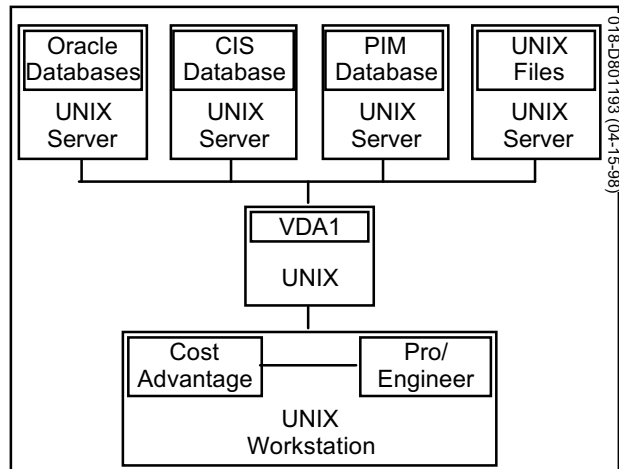


Figure A-6. VDA1 Data Interfaces

A.6.1.1 Internal Class Structure. An internal class will be defined for each of the data sources to be accesses. In the case of PIM two classes will be established, one describing PIM as a data source for inquiries and the other describing PIM as data source for update. Table A-3 is a list of the sources and the corresponding class describing it. Each class structure is currently a single level. This can be expanded to include multiple levels in the future as the

Data Source	Class Name
CIS	partfuncs
Oracle	oraclefuncs
PIM (extract)	pim1
PIM (update)	pim2

Table A-3. Data Sources and Class Names

Data Source	Class Name
CIS	partfuncs
Oracle	oraclefuncs
PIM (extract)	pim1
PIM (update)	pim2

number of interfaces to a particular data source expands.

A.6.1.2 JGUI Requests for Data from VDA1. The following defines the call interfaces between the JGUI and the VDA1 and between the VDA1 and its data sources. See A.8.5, Exhibit E – VDA Call Format Parameter Definitions, for call parameter definitions.

Interface #1

Call Name	product structure
Description	call to VDA1 to be used by any request for data pertaining to product structure
Format	product_structure (IPL_key, file_path, qualifiers, security_info, return_code)
Comments	IPL key is a structure containing the information necessary to uniquely identify a specific product structure and the number of levels of the structure. The structure contains the following fields: - Part - Revision - Version - Cage Code - Promotion Status - No. of Levels
Qualifiers	indicates type of product structure processing to be done. One or more from the following list may be requested: - proe = Pro/E model files - ca = Cost Advantage files - ipl = indentured parts list - proc = indentured process list
This is a pass through interface. All processing will be handled by the existing PIM interface routine.	
Class Name	pim1
Method	RPC call to the PIM RPC Server (pass through) to request PIM IPL extract. This is a synchronized linkage where the VDA1 will wait until a response is sent back from the PIM Server. Similarly, the linkage to the PGUI is also synchronized.
Method format:	pim_data (IPL_key, file_path, qualifiers, security_info, return_code)

Interface #2

Call Name	part price
Description	call to VDA1 to be used by any request for data pertaining to part or component cost
Data Source(s)	Oracle CPA database CIS Explore database
Format	part_price (file_path, over_write, qualifiers, security_info, return_code)
Comments	The file path may specify a part price file (.PPR) or an indentured parts list file (.IPL). If a PPR file is specified, it contains the list of parts to be priced and the quantity of each part in one unit of the finished product. If an IPL is specified, the VDA first creates a PPR file by analyzing the indenture levels and quantities in the IPL and summarizing them to the level of each unique part in the finished product Part price optionally accepts the following qualifier to identify a specific parts list: - organization - project The underlying database allows the user to name and save a list of “eligible” parts for use in a specific design. The list may include a specific price that the user wants to incorporate for evaluation, and may also designate a make versus buy decision. If the optional qualifier for parts list identification is specified, the VDA will first search the parts list for a designated price. If no price is found in the parts list, the VDA will next examine the make versus buy indicator and look for a price in the corresponding location, i.e., Raytheon parts list or corporate purchase agreement (CPA), respectively. If no price is found by this method, the VDA will use the standard retrieval algorithm described below. A standard rule is used to return part prices when a parts list is not specified, or when no price is available from the parts list. The priority order for determining the price is: - CPA - Supplier price - Raytheon make price - Parts list price See A.8.7, Exhibit G – Pricing Logic Pseudo-Code, for part pricing pseudo-code Build quantity is included as part of the .ppr file and designates the number of units of finished product to assume for part pricing. If volume agreements apply to the part price, the program quantity will be multiplied by the part count per unit finished product to determine the purchase volume.
Class Name	partfuncs
Method	Embedded CIS Explore C++ API call to extract Part Price data. CPA interface will embed SQL code as part of the method.
SQL pseudo code	: Select price where part = input_part and quantity GE qty n and LE qty n+1. Note: this will logic will be repeated comparing the current bucket against the next volume bucket

Method Format	Standard SQL and CIS API commands
---------------	-----------------------------------

Interface #3

Call Name rates and factors

Description call to VDA1 to be used by any request for data pertaining to rates and factors

Data Source(s) Oracle rate and factors database

Format rates_factors (file_path, over_write, qualifiers, security_info, return_code)

Comments Labor and overhead rates require the following qualifiers:
source code
date

Class Name oraclefuncs

Method Rates factors interface will embed SQL code as part of the method

SQL pseudo code: Select rate_category and rate where
source code = input_source_code and (input_date GE valid_from_date and
LE valid_through_date)

Method Format Standard SQL

Interface #4

Call Name programmatics

Description call to VDA1 to be used by any request for data pertaining to the program or project specifically such as build quantities, build year and other misc. data.

Data Source(s) Oracle Programmatics database

Format programmatics (file_path, over_write, qualifiers, security_info, return_code)

Comments Program information requires the following qualifier:

Class Name oraclefuncs

Method Rates factors interface will embed SQL code as part of the method

SQL pseudo code Select programmatic_name_1, programmatic value 1 ... where
project = input project

Method Format Standard SQL

Interface #5

Call Name process data

Description call to VDA1 to be used by any request for data pertaining to process

Format process_coefficients (file_path, over_write, qualifiers, security_info, return_code)

Comments Process coefficients require the following qualifiers:
process identifier
manufacturing location

Class Name oraclefuncs

Method Rates factors interface will embed SQL code as part of the method

SQL pseudo-code Select process coefficient name 1, process coefficient 1 ... where
manufacturing_location = input_manufacturing_location

Method format: Standard SQL

Note:

- All embedded SQL will conform to Oracle object library calling sequences. The underlying Oracle network transport will be Version 2.x TCP/IP protocol.
- All Oracle access should be via imbedded SQL.
- All SQL access to data should use the defined views not direct table access.

A.6.2 VDA2

VDA2 will provide the interface between the JGUI and all data sources updated with the result of the estimating tool. Currently this will be limited to the PIM Ingres Database.

A.6.2.1.1 JGUI to VDA 2 (Pass-through to PIM Server)

Interface #6

Call Name	vault product
Description	call to VDA2 to be used by any request to update data resulting from an estimating session.
Data Source	PIM Ingres Database
Comments	IPL key is a structure containing the information necessary to uniquely identify a specific product structure to be updated
Class Name	pim2
Method	vault_product (IPL_key, file_path, qualifiers, security_info, return_code)
Method Format	RPC call to the PIM RPC Server (pass through) to request product cost update. This is a synchronized linkage where the VDA2 will wait until a response is sent back from the PIM server. Similarly, the linkage to the JGUI is also synchronized. Refer to the definitions and specifications for VDA1.

A.7 Database and Communication Servers

All interface calls will be synchronous using RPC. That is, the calling process will wait for the remote procedure to complete execution before it continues.

A.7.1 PIM RPC Server

CSC developed software to interact between the VDA1 Server and the PIM TCP/IP Server. This server is running all the time and waiting for incoming messages. Upon receiving a message, the PIM TCP/IP Server will be connected to and a message will be formatted and transmitted to the PIM Server. This linkage is synchronized until the PIM Server has processed the request and delivered the PIM data to the requesting UNIX machine (see PIM TCP/IP Server description).

Calling Format: pim_data (IPL_Key, file_path, qualifiers, security_info, return_code)

A.7.2 PIM TCP Server

CSC developed software to interact between the PIM RPC Server and the PIM

Database Server. This server is running all the time and waiting for incoming messages. Upon receiving a message, a child process will be spawned and the process owner changed to the requesting user (the User ID embedded in the security_info). The PIM Database will be accessed for security verification. Upon successful connection, the PIM IPL data will be extracted via the Sherpa DMS/EDM API and results written to an output file. At the end of the IPL extraction, the output file(s) are FTPed to the originating UNIX machine per the file spec described in the file_path parameter. A code and a message is then returned back to the PIM RPC Server. This linkage is also synchronized until the all PIM data are processed.

A.8 Exhibits

A.8.1 Exhibit A – CIS Explore Class Structure

Only the first five levels through the electronics component tree are show here. For a complete listing, refer to the class browser in the CIS Explore 2.5.3 application software.

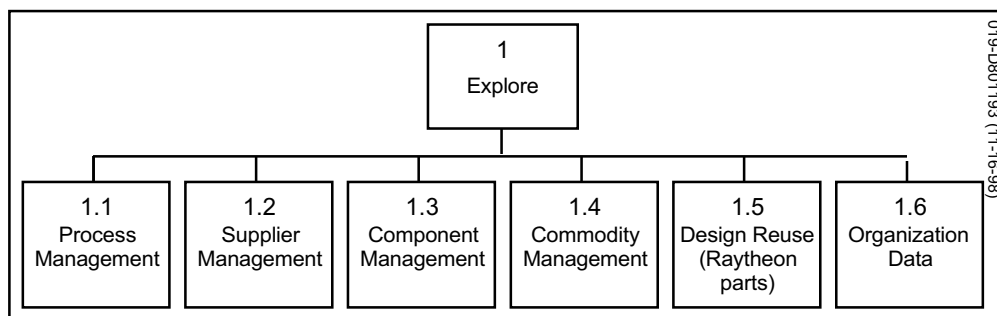


Figure A-7. Level 1 of CIS Class Structure

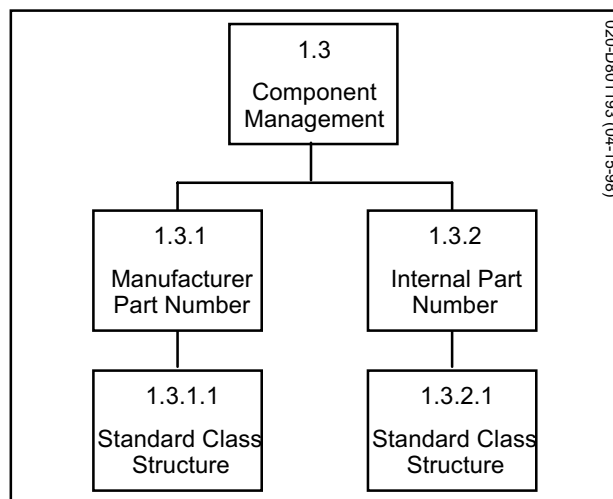


Figure A-8. Levels 2 and 3 of CIS Class Structure

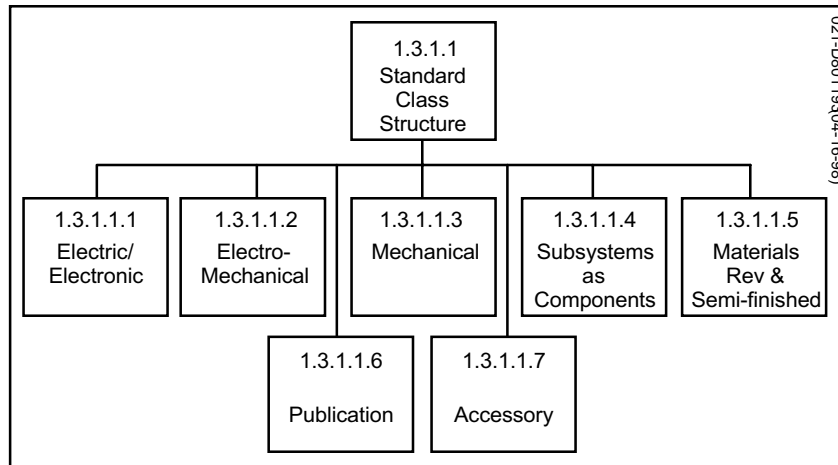


Figure A-9. Level 4 of CIS Class Structure

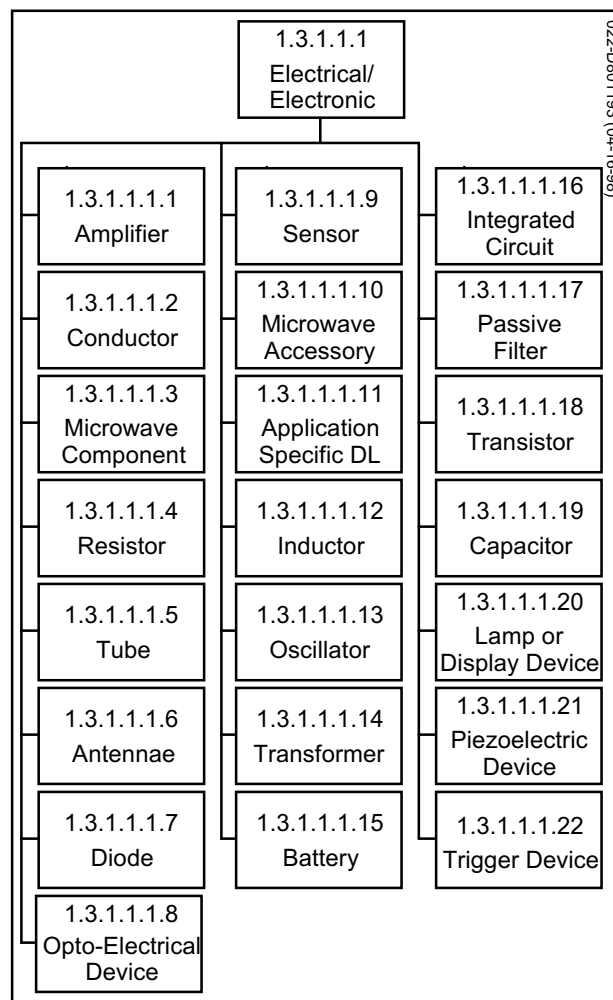


Figure A-10. Level 5 of CIS Class Structure

A.8.2 Exhibit B – Data Dictionary

Table Name	Data Element Name	Key	Oracle Data Type	Lgth	CIS Logical Data Type
supplier/cap	local supplier id	pk	varchar2	6	stext6
supplier/cap	local supplier name		varchar2	35	stext35
cpa	cpa number	pk	varchar2	4	stext4
cpa	cpa start date		date		SDateTime
cpa	cpa expiration date		date		SDateTime
part/cpa	supplier part number	pk	varchar2	26	stext26
part/cpa	cpa number	pk	varchar2	4	stext4
part/cpa	max qty 1				
part/cpa	unit price 1		number	10,2	VNum10_2
part/cpa	max qty 2		number	7	VNum7
part/cpa	unit price 2		number	10,2	VNum10_2
part/cpa	max qty 3		number	7	VNum7
part/cpa	unit price 3		number	10,2	VNum10_2
part/cpa	max qty 4		number	7	VNum7
part/cpa	unit price 4		number	10,2	VNum10_2
part/cpa	max qty 5		number	7	VNum7
part/cpa	unit price 5		number	10,2	VNum10_2
part/cpa	max qty 6		number	7	VNum7
part/cpa	unit price 6		number	10,2	VNum10_2
supplier part	supplier part number	pk	varchar2	26	stext26
supplier part	local supplier id	pk	varchar2	6	stext6
supplier part	supplier part price		number	10,2	VNum10_2
supplier part	supplier part price type code		varchar2	1	stext1
supplier part	supplier part price type qty		number	10,2	VNum10_2
supplier part	supplier part price date		date		SDateTime
internal/supplier part	local supplier id	pk	varchar2	6	stext6
internal/supplier part	supplier part number	pk	varchar2	26	stext26
internal/supplier part	hac part number		varchar2	26	stext26
internal part	hac part number	pk	varchar2	26	stext26
internal part	hac cage code		varchar2	5	stext5
internal part	hac part price		number	10,2	VNum10_2
internal part	hac part price type code		varchar2	1	stext1
internal part	hac part price type qty		number	10,2	VNum10_2
internal part	hac part price date		date		SDateTime
parts lists	parts list name	pk	varchar2	25	stext25
parts lists	part number	pk	varchar2	26	stext26
parts lists	make buy code		varchar2	1	stext1
parts lists	parts list price		number	10,2	VNum10_2

Table Name	Data Element Name	Key	Oracle Data Type	Lgth	CIS Logical Data Type
parts lists	parts list price type		varchar2	1	stext1
parts lists	parts list price qty		number	10,2	VNum10_2
parts lists	parts list price date		date		SDateTime
parts list name	parts list name	pk	varchar2	25	stext25
parts list name	parts list description		varchar2	75	stext75
rates and factors	source sode	pk	varchar2	6	stext6
rates and factors	rate type code	pk	varchar2	1	stext1
rates and factors	rate category	pk	varchar2	25	stext25
rates and factors	valid from year	pk	varchar2	4	stext4
rates and factors	valid from month	pk	varchar2	2	stext2
rates and factors	valid through year	pk	varchar2	4	stext4
rates and factors	valid through month	pk	varchar2	2	stext2
rates and factors	rate		number	12,4	VNum12_4
programmatics	project name	pk	varchar2	15	stext15
programmatics	programmatic type code	pk	varchar2	1	stext1
programmatics	programmatic category	pk	varchar2	25	stext25
programmatics	programmatic name 1		varchar2	25	stext25
programmatics	programmatic value 1		number	12,4	VNum12_4
programmatics	programmatic name 2		varchar2	25	stext25
programmatics	programmatic value 2		number	12,4	VNum12_4
programmatics	programmatic name 3		varchar2	25	stext25
programmatics	programmatic value 3		number	12,4	VNum12_4
manufacturing process	manufacturing location	pk	varchar2	15	stext15
manufacturing process	process name	pk	varchar2	25	stext25
manufacturing process	process description		varchar2	75	stext75
manufacturing process	process coefficient name 1		varchar2	15	stext15
manufacturing process	process coefficient 1		number	12,4	VNum12_4
manufacturing process	process coefficient name 2		varchar2	15	stext15
manufacturing process	process coefficient 2		number	12,4	VNum12_4
manufacturing process	process coefficient name 3		varchar2	15	stext15
manufacturing process	process coefficient 3		number	12,4	VNum12_4
manufacturing process	process coefficient name 4		varchar2	15	stext15
manufacturing process	process coefficient 4		number	12,4	VNum12_4
manufacturing process	process coefficient name 5		varchar2	15	stext15
manufacturing process	process coefficient 5		number	12,4	VNum12_4
manufacturing process	process coefficient name 6		varchar2	15	stext15
manufacturing process	process coefficient 6		number	12,4	VNum12_4
manufacturing process	process coefficient name 7		varchar2	15	stext15
manufacturing process	process coefficient 7		number	12,4	VNum12_4

Table Name	Data Element Name	Key	Oracle Data Type	Lgth	CIS Logical Data Type
manufacturing process	process coefficient name 8		varchar2	15	stext15
manufacturing process	process coefficient 8		number	12,4	VNum12_4
manufacturing process	process coefficient name 9		varchar2	15	stext15
manufacturing process	process coefficient 9		number	12,4	VNum12_4
manufacturing process	process coefficient name 10		varchar2	15	stext15
manufacturing process	process coefficient 10		number	12,4	VNum12_4

A.8.3 Exhibit C – Oracle Data Definition Language

PROMPT Creating JMD obl_rate components
CREATE TABLE JMDDEMO.OBL_RATE_TABLE

```
(
  SOURCE_CODE          CHAR          (006)      NOT NULL,
  RATE_TYPE_CODE       CHAR          (001)      NOT NULL,
  RATE_CATEGORY        CHAR          (025)      NOT NULL,
  VALID_FROM_YEAR     CHAR          (004)      NOT NULL,
  VALID_FROM_MONTH    CHAR          (002)      NOT NULL,
  VALID_THROUGH_YEAR  CHAR          (004)      NOT NULL,
  VALID_THROUGH_MONTH CHAR          (002)      NOT NULL,
  RATE_NUMBER         (012,04)      NOT NULL,
  PRIMARY KEY(
    SOURCE_CODE,
    RATE_TYPE_CODE,
    RATE_CATEGORY,
    VALID_FROM_YEAR,
    VALID_FROM_MONTH,
    VALID_THROUGH_YEAR,
    VALID_THROUGH_MONTH)
)
```

```
PCTFREE 10
STORAGE(INITIAL 5K NEXT 20K)
;
CREATE VIEW JMDDEMO.OBL_RATE
AS SELECT ALL
  SOURCE_CODE,
  RATE_TYPE_CODE,
  RATE_CATEGORY,
  VALID_FROM_YEAR,
  VALID_FROM_MONTH,
  VALID_THROUGH_YEAR,
  VALID_THROUGH_MONTH,
  RATE
FROM JMDDEMO.OBL_RATE_TABLE
;
```

PROMPT Creating JMD Programmatic components
CREATE TABLE JMDDEMO.PROG_TABLE

```
(
  PROJECT              CHAR          (015)      NOT NULL,
  PROG_TYPE_CODE       CHAR          (001)      NOT NULL,
  PROG_CATEGORY        CHAR          (025)      NOT NULL,
  PROG_NAME_1          VARCHAR2     (025)      NULL,
  PROG_VALUE_1         NUMBER        (012,04)   NULL,
  PROG_NAME_2          VARCHAR2     (025)      NULL,
  PROG_VALUE_2         NUMBER        (012,04)   NULL,
  PROG_NAME_3          VARCHAR2     (025)      NULL,
  PROG_VALUE_3         NUMBER        (012,04)   NULL,
  PRIMARY KEY(
    PROJECT,
    PROG_TYPE_CODE,
    PROG_CATEGORY)
)
```

```
PCTFREE 10
STORAGE(INITIAL 5K NEXT 20K)
;
CREATE VIEW JMDDEMO.PROG
AS SELECT ALL
  PROJECT,
  PROG_TYPE_CODE,
  PROG_CATEGORY,
  PROG_NAME_1,
  PROG_VALUE_1,
  PROG_NAME_2,
  PROG_VALUE_2,
  PROG_NAME_3,
  PROG_VALUE_3
FROM JMDDEMO.PROG_TABLE
;
```

PROMPT Creating JMD Manufacturing Rate components

```
CREATE TABLE JMDDEMO.MFG_RATE_TABLE
(
  MFG_LOCATION        CHAR          (015)      NOT NULL,
  PROC_NAME           CHAR          (025)      NOT NULL,
```

```
PROC_DESC             VARCHAR2     (075)      NULL,
PROC_COEFF_NAME_1     CHAR          (015)      NULL,
PROC_COEFF_1          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_2     CHAR          (015)      NULL,
PROC_COEFF_2          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_3     CHAR          (015)      NULL,
PROC_COEFF_3          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_4     CHAR          (015)      NULL,
PROC_COEFF_4          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_5     CHAR          (015)      NULL,
PROC_COEFF_5          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_6     CHAR          (015)      NULL,
PROC_COEFF_6          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_7     CHAR          (015)      NULL,
PROC_COEFF_7          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_8     CHAR          (015)      NULL,
PROC_COEFF_8          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_9     CHAR          (015)      NULL,
PROC_COEFF_9          NUMBER        (012,04)   NULL,
PROC_COEFF_NAME_10    CHAR          (015)      NULL,
PROC_COEFF_10         NUMBER        (012,04)   NULL,
PRIMARY KEY(
  MFG_LOCATION,
  PROC_NAME)
)
```

```
PCTFREE 10
STORAGE(INITIAL 5K NEXT 20K)
;
```

CREATE VIEW JMDDEMO.MFG_RATE

```
AS SELECT ALL
  MFG_LOCATION,
  PROC_NAME,
  PROC_DESC,
  PROC_COEFF_NAME_1,
  PROC_COEFF_1,
  PROC_COEFF_NAME_2,
  PROC_COEFF_2,
  PROC_COEFF_NAME_3,
  PROC_COEFF_3,
  PROC_COEFF_NAME_4,
  PROC_COEFF_4,
  PROC_COEFF_NAME_5,
  PROC_COEFF_5,
  PROC_COEFF_NAME_6,
  PROC_COEFF_6,
  PROC_COEFF_NAME_7,
  PROC_COEFF_7,
  PROC_COEFF_NAME_8,
  PROC_COEFF_8,
  PROC_COEFF_NAME_9,
  PROC_COEFF_9,
  PROC_COEFF_NAME_10,
  PROC_COEFF_10
FROM JMDDEMO.MFG_RATE_TABLE
;
```

PROMPT Creating JMD supplier components

CREATE TABLE JMDDEMO.SUPPLIER_TABLE

```
(
  LOCAL_SUPP_ID        CHAR          (006)      NOT NULL,
  LOCAL_SUPP_NAME      VARCHAR2     (035)      NOT NULL,
  PRIMARY KEY(
    LOCAL_SUPP_ID)
)
```

```
PCTFREE 10
STORAGE(INITIAL 5K NEXT 20K)
;
```

CREATE VIEW JMDDEMO.SUPPLIER

```
AS SELECT ALL
  LOCAL_SUPP_ID,
  LOCAL_SUPP_NAME
FROM JMDDEMO.SUPPLIER_TABLE
;
```

PROMPT Creating JMD supplier cpa components

CREATE TABLE JMDDEMO.CPA_TABLE				UNIT_PRICE_5			
(				NUMBER			
CPA_NUMBER				(009,04)			
CHAR				NULL,			
(004)				MAX_QTY_6			
NOT NULL,				NUMBER			
CPA_START_DATE				(008)			
DATE				NULL,			
CPA_EXP_DATE				UNIT_PRICE_6			
DATE				NUMBER			
(009,04)				NULL,			
PRIMARY KEY(				PRIMARY KEY(			
CPA_NUMBER)				SUPP_PART_NUMBER,			
)				CPA_NUMBER),			
PCTFREE 10				CONSTRAINT CPA_PART FOREIGN KEY (			
STORAGE(INITIAL 5K NEXT 20K)				CPA_NUMBER)			
;				REFERENCES JMDDEMO.CPA_TABLE			
CREATE VIEW JMDDEMO.CPA				)			
AS SELECT ALL				PCTFREE 10			
CPA_NUMBER,				STORAGE(INITIAL 5K NEXT 20K)			
CPA_START_DATE,				;			
CPA_EXP_DATE				CREATE VIEW JMDDEMO.CPA_PART			
FROM JMDDEMO.CPA_TABLE				AS SELECT ALL			
;				SUPP_PART_NUMBER,			
PROMPT Creating JMD cpa part components				CPA_NUMBER,			
CREATE TABLE JMDDEMO.CPA_PART_TABLE				MAX_QTY_1,			
(				UNIT_PRICE_1,			
SUPP_PART_NUMBER				MAX_QTY_2,			
CHAR				UNIT_PRICE_2,			
(026)				MAX_QTY_3,			
NOT NULL,				UNIT_PRICE_3,			
CPA_NUMBER				MAX_QTY_4,			
CHAR				UNIT_PRICE_4,			
(004)				MAX_QTY_5,			
NOT NULL,				UNIT_PRICE_5,			
MAX_QTY_1				MAX_QTY_6,			
NUMBER				UNIT_PRICE_6			
(008)				FROM JMDDEMO.CPA_PART_TABLE			
NULL,				;			
UNIT_PRICE_1							
NUMBER							
(009,04)							
NULL,							
MAX_QTY_2							
NUMBER							
(008)							
NULL,							
UNIT_PRICE_2							
NUMBER							
(009,04)							
NULL,							
MAX_QTY_3							
NUMBER							
(008)							
NULL,							
UNIT_PRICE_3							
NUMBER							
(009,04)							
NULL,							
MAX_QTY_4							
NUMBER							
(008)							
NULL,							
UNIT_PRICE_4							
NUMBER							
(009,04)							
NULL,							
MAX_QTY_5							
NUMBER							
(008)							
NULL,							

A.8.4 Exhibit D – PCT Interface Specifications

A.8.4.1 Introduction/Scope. The purpose of this specification is to describe a database construct that will serve as a communications vehicle between the back-end servers (source) and the client-side data objects that serve as the intermediary (i.e., buffered by the VDA) source of data for the design-to-cost models on the client-side. The functionality of the Process Characterization Tool (PCT) is described in detail in the document “Process Characterization IT Integration Requirements,” dated June 1997.

The database construct is generic and supported both the integration of the Process Characterization Tool (PCT) databases into the JMD IT architecture and future upgrades to the Parts Database characteristics which had initially focused mainly on price characteristics. As a result of demonstrating some of the earliest capabilities of the JMD prototype, parts characteristics needed to address size, weight, physical configuration, etc. It was anticipated that the model developers would require these other parts characteristics in their models for increased model fidelity and design trade space. Since the PCT database typifies a database with an extensive trade space for the manufacturing processes at Raytheon, it supported this natural extension for the update of the interface to the Parts Database characteristics.

It was intended that the implementation of this generic database construct be robust, such that as the model requirements identified new data fields that were required as variables in the model cost equations, the data interface/exchange via the VDA remained unchanged. This was accomplished using a variable record format for an m x n spreadsheet format for both the PCT and the Parts Database communication protocol between the back-end servers and the client-side workstations.

A.8.4.2 Database Construct/Format. Figure A-11 illustrates the relationships between the data generators and the data consumers in the IT architecture for both the PCT and

the other application tools being used on the JMD program. Note that the VDA middleware software is a buffer between the data generators and the data consumers.

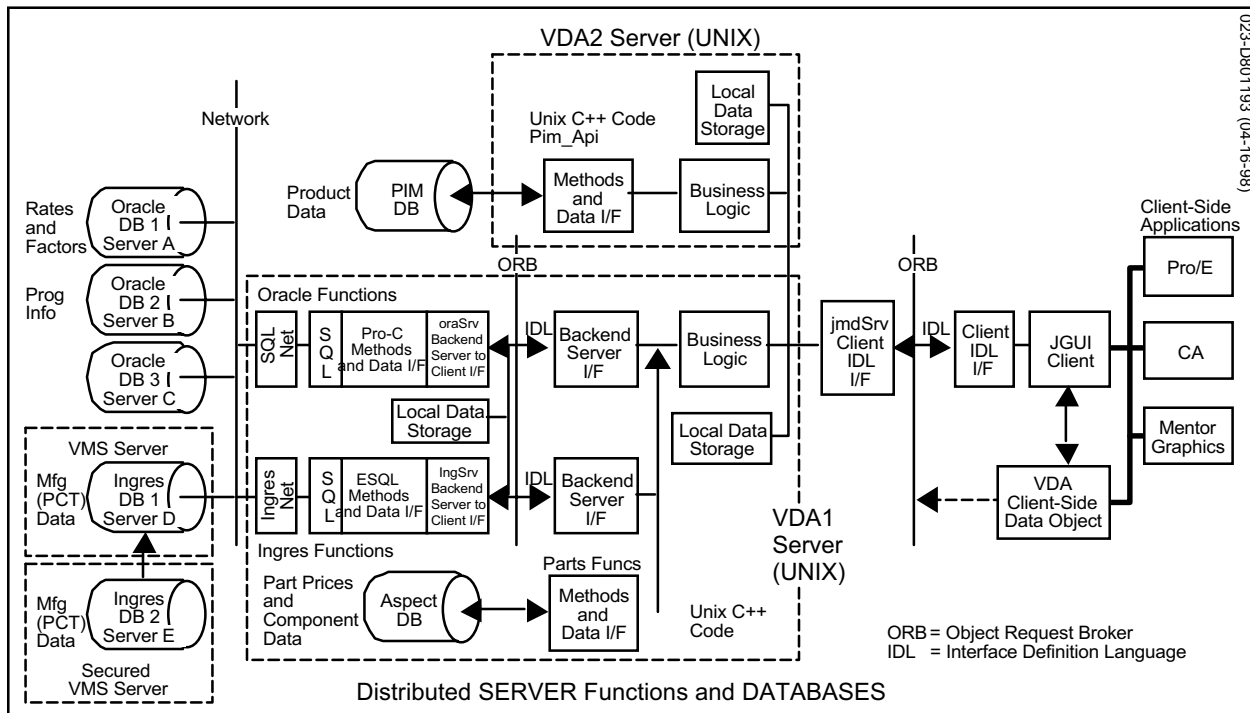


Figure A-11. Prototype IT Architecture

The MFG (PCT) Database shown in the lower left corner of Figure A-11 has been expanded in Figure A-12 to show the data characteristics/format that the PCT application must generate and store in the PCT database, and which the VDA will port/move from the server-side object to the client-side data object. Before discussing the PCT database construction and its utilization in detail, the Server E to Server D Ingres connection shown in Figure A-11 will be discussed using the exploded view of Figure A-12.

Because of the location of the Ingres Server E and the business environment associated with the database and sources of information, a special Ingres-to-Ingres connection was devised to provide a pathway for the data as shown in Figure A-12. Several 'scripts' and a GUI were developed to integrate the PCT data source into the JMD architecture. From outside the VDA, these scripts enable security and regular updates to take place to Server D. This is all transparent to the VDA, and as far as the VDA and the Clients accessing the data are concerned, Server D is Server E. The scripts allow Server D to see a filtered and controlled portion of Server D.

Figure A-13 depicts a four-dimensional function for "load substrate for stencil," each dimension being represented by a pair of records called a record-couplet. There are no limits to the number of values along each dimension, as are there no limits to the number of dimensions or record-couplets.

It can be seen from Figure A-13 that the multi-dimensional function depicted by the PCT MxN data file comprises a set of M record-couplets, one for each dimension of the function. Each record-couplet is a pair of records consisting of an equal number of N fields in each record, where N is the number of data points along each dimension of the function plus a fixed amount of header field information. Each record-couplet can be of variable length from 4 through N. 4 is the minimum number of fields in a record-couplet because there are three fields dedicated to identification of process and variable names and units, and there has to be at least one field for the abscissa and ordinate data. The record-couplet in effect represent the points along a curve for which data has been accumulated. For points in between, there are a number of options available to derive the intermediate values, such as linear interpolation or a “bounded” assignment to one of the points in the variable length record. At this point in the JMD IT architecture demonstration, the latter was chosen not just for its simplicity in implementation, but as an interim implementation until the cost model designers had an opportunity to “push” back some refined requirements based on some initial cost model development.

Full Process Name	X-axis variable Name	Y-axis Variable Name	X1 Value	X2 Value	X3 Value	..	..	..	XN Value
Process Pseudo ID	Units	Units	Y1 Value	Y2 Value	Y3 Value				YN Value
Load substrate for stencil	substrate area	Yield	4	5	6				
364	cm ²	%	A	B	C				
Load substrate for stencil	substrate area	Oper_Support	4	5	6	9			
365	cm ²	minutes	5	0	5	24			
Load substrate for stencil	substrate thickness	machine CT	4	5	6				
366	Mils	seconds	12	10	10				
Load substrate for stencil	substrate thickness	Operator CT	4	5	6	10	25		
367	Mils	seconds	20	17	15	8	2		

Figure A-13. An Illustrative Example of a Variable Record PCT MxN Data File

The record-couplet in bold in A-13 has been repeated in Figure B-14 and will be described in detail to illustrate the expandability of the PCT table to define n-dimensional functions for a manufacturing process.

The second column in Figure A-14 describes the parameters for the x-axis depicted by the last four cells in the first row which are shaded lightly (i.e., 4 represents the abscissa value for 4 cm² of substrate area).

The third column in Figure A-14 describe the parameters for the y-axis depicted by the last four cells in the second row which are shaded slightly darker (i.e. 10 represents the ordinate value for % of MST for Eng Support for 4 cm² substrate area .

In the JMD IT Prototype Architecture, the data contents of Figure A-13 are compiled by Raytheon's PCT tool and stored in an Ingres database. The VDA is tasked with the methods and the intelligence to acquire, format, and port the data from the PCT database in the far bottom left of Figure A-11 are the client-side data-object on the far right of Figure A-11.

Once the data becomes resident in the client-side data-object, there are specially developed Cost Advantage functions that have been created to allow the equations of

load substrate for stencil	substrate area	Eng Support	4	5	6	9
365	cm ²	% of MST	A	B	C	D

Figure A-14. A Typical Record-Couplet of a PCT MxN Data File

the cost model to extract the required data for computing costs as a function of the design features that have been modeled into the cost model. The functionality of the Cost Advantage functions is described next.

A.8.4.3 Preliminary Specifications for “Bounded Table Function” for Cost Advantage. “boundedTbl” is a table accessing function compatible with Microsoft Excel’s ‘.csv’ (comma separated values) format. Its purpose is to allow numeric data stored in a tabular file to be accessed by specifying a row designator and a data variable for which a value is required. The function can be accessed from within a Cost-Advantage equation in the following way:

value = boundedTbl (“filename”, Process_Name, Abscissa_Value_of_interest)

Where **boundedTbl** is the actual name given to the function in Cost Advantage, “**filename**” corresponds to the name of the file contained in client-side data object in one of the following directories:

\$CMTUSERROOT/JMD_DATA

\$CMTGLOBALROOT/JMD_DATA

The **boundedTbl** function expects the data directory to be the same as that used for part price, rates & factors, etc.

The **Process_name** specifier is a text string containing data that must exactly match data in the first two cells of one of the odd rows in the table. This key is used to find an appropriate pair of rows in the table from which to access data. A comma is used to separate the field values in the text string **Process_name**. Note that this approach is scalable to allow a hierarchical access to manufacturing process data (i.e., if there is an indentured process structure to the characterization of the manufacturing processes, it may take several comma separated fields to “find” the correct entry in the table). To illustrate this implementation of the function in Cost Advantage and the characterization capabilities of Raytheon’s PCT itself, the following example is given.

Example:

Assume that the data of Figure A-13 exists in a file called opCost.csv

If a Cost-Advantage model wanted to determine how much Oper_Support was required to load a 6 cm² substrate for stenciling, the following function call would yield the number of minutes which could then be converted into a cost using a labor rate:

Val = **boundedTbl (“opCost.csv”, “Load substrate for stencil, substrate area”, 5.5)**

The routine first locates rows three and four based on the input “**Load substrate for stencil, substrate area**”. It then locates an appropriate return cell by comparing the input value 5.5 against the data fields found in the first of these two rows in columns four through seven. Because $5 < 5.5 < 6$, the returned value for the function lookup is **0 (minutes)** based on the lower bounding value of 5 (hence the name **boundedTbl**). Instead of this arbitrary assignment to the left side of the bounding values, the function could be recoded to use the right side, or better still it could even do a linear interpolation of the values in the table.

Note that there is in implicit **ascending** order to X-axis values, and there are no restrictions on the Y-axis values. An error is reported if an input is less than the first X-

value in the table, or if it is greater than the last X-value given, then the last Y-axis value is returned.

If the **Process_name** specifier is not located, if there are no data fields, or if the data file itself is not located, then the routine will return a 0 value and print an error message on the system console.

Because the **Process_name** specifier can be rather lengthy, an unique alias has been assigned to it so that the cost model developers can keep the equations cleaner in appearance and content. The alias must be unique and the following two function calls are equivalent:

Val = **boundedTbl** ("opCost.csv", "Load substrate for stencil, substrate area", 5.5)

Val = **boundedTbl** ("opCost.csv", "365", 5.5)

The first one is more user-friendly in that it conveys information about the process characterization, whereas the second one is simpler but less informative about the process being accessed.

A.8.5 Exhibit E – VDA Call Format Parameter Definitions

file path	The directory structure leading to the file to be processed, or the directory where the set of files to be processed exists. The file path starts at the root of the UNIX file system on the "calling" machine. For example, the file path for the part price file might be /Export/home/Cognition/DSS.
over-write	Flag indicating whether or not the original contents of an existing file should be replaced with the new requested information (redo the entire request) or just "fill in the blanks." For example, if you are pricing a list of parts and the over-write argument is 'Y,' then the entire file would be repriced. If the argument was 'N,' then only those parts with zeros or blanks would be repriced.
qualifiers	Variable length character string containing a list of request specific variables or arguments. For example, the qualifiers for a product-structure call might be 'proe,ipl' to return the Pro/E files and an indentured parts list.
security info	Structure containing the current login identification and password of the end user. This identification is used in subsequent data retrieval to restrict access to authorized users only.
return code	return code indicating status of request. For example, zero indicates successful completion.

A.8.6 Exhibit F – File Layout Definitions

All files are ASCII delimited by “ “, “:” or “/” as indicated below.

Indicates comment to UNIX, but may be meaningful to VDA.

.IPL - indentured parts list

indenture level char(2)
part number char(32)
quantity 9999999.99
unit of measure char(5)
price per unit of measure 9999999.99

.EST - estimated cost

part number char(32)
cost per unit of measure 9999999.99

Example:

PN0132577: \$A

.MAT - material cost

material number char(32)
price per unit of measure 9999999.99

Example:

6061 Aluminum Alloy per pound
6061-T1: \$11.42

.PGM - programmatics

program keyword char(32)
keyword value 9999999.99

Example:

TotalProductionRun: 25000

.PPR - part price

part number char(32)
price per unit of measure 9999999.99
qty/unit finished product 9999999

Example:

capacitors
C123: .23/ 10

flip chips

A: X/ 4
B: Y/ 8

wirebonded chips

A: X/2
B: Y/4
C: Z/2

.PRC

process identifier (comment line) char(32)
coefficient name char(8)
coefficient value 9999.99

Example:

solder wirebonded components
A1: 1
solder discrete components
A2: 1
epoxy flip chips
A3: 1
Zevatech placement
A4: 1
Raytheon's 3500 placement
A5: 1

.RAT - rates & factors

keyword char(32)
keyword value 9999999.99

Example:

realization_factor: 1.1
utilization_rate: 168
supervisor_rate: A
sust_engr_rate: B
assoc_prct_rate: C
sal_pc_rate: D
assoc_labor_va_rate: E
assoc_supt_labor_rate: F

A.8.7 Exhibit G – Pricing Logic Pseudo-Code

Price Look-up Routine

```
If input project and organization name is
provided
    retrieve price from parts list
# using part, project and organization
name
    if part found
        if price > 0
            use this price for part
        else
            If make buy indicator on parts list =
M,
            look for part in CIS Internal parts
class
                If part found
                    use price
                else
                    return 0.0
            else
                if make buy indicator on parts list =
B,
                look for part on CPA
                    if part found
                        if more than one price
                            use lowest price
                        else
                            use price
                    else
                        look for part in the CIS parts
class
                            if part found
                                use price
                            else
                                return 0.0
                        else
                            perform default lookup
                    else
                        perform default lookup
                else
                    perform default lookup.
```

Default Look-up Routine

```
Look for price on CPA
if found
    if more than one price
        use lowest price
    else
        use price
else
    look for part in the CIS mfg parts class
    if found and price > 0
        use price
    else
        look for part in the CIS internal parts
class
        if found and price > 0
            use price
        else
            use 0.0.
```

A.8.8 Exhibit H – Project Library Structure

PDM GUI Libraries

Source Code: /home/pd13153/JMD/pgui
Executable: /home/pd13153/JMD/bin/jmd_top_level
Config Input: /home/pd13153/JMD/bin/pgui.cfg
Log Output: /home/pd13153/JMD/bin/pgui.log

VDA Libraries

Source Code: /home/pd13153/JMD/vda
Executable: /home/pd13153/JMD/bin/vda_server
Config Input: none
Log Output: /home/pd13153/JMD/bin/vda_server.log

PIM RPC Server Libraries

Source Code: /home/pd13153/JMD/pim/rpcsvr
Executable: /home/pd13153/JMD/bin/pimrpc_server
Config Input: /home/pd13153/JMD/bin/pimrpcsvr.cfg
Log Output: /home/pd13153/JMD/bin/pimrpc_server.log

PIM TCP Server Libraries

Source Code: /home/pd13153/JMD/pim/tcpsvr
Executable: /home/pd13153/JMD/bin/pimtcp_server
Config Input: /home/pd13153/JMD/bin/pimsvr.cfg
Log Output: /home/pd13153/JMD/bin/pimsvr.log
FTP Shell Script: /home/pd13153/JMD/bin/ftp.script

A.8.9 Exhibit I – JMD Operational Test Procedures

A.8.9.1 Purpose/Scope. These procedures step through the various man-machine operations supported by the JMD JGUI software. The procedures were developed to validate the operability of the software as it transitioned through the changes being made by Raytheon and its subcontractors. The purpose of the tests is to determine whether the software under test preserves the functionality and performance of some or all of the software being modified or replaced.

A.8.9.2 Test Environment. These procedures assume that the hardware environment as depicted in Appendix A, Figure A-3 has already been set up and that the following software modules and databases are active:

A. Software

JMD client software, JMD VDA server software, Network and Solaris Operating systems

B. Hardware

RSPIM_C2 (Sparc) server, RSPIM_C3 (Sparc) server, Client (Sparc) workstation

C. Test Data

Small demonstration test databases for each of the VDA functions

A.8.9.3 Features Tested

A.8.9.3.1 Virtual Database Agent (VDA) Servers. A separate test to verify that the two VDA modules perform the connections between the JGUI and the backend database servers as required was not required. Instead, the tests outlined in A.8.9.3.3, “The JAVA Graphical User Interface (JGUI),” were used to test the complete functionality of the two VDA modules in conjunction with the JGUI test procedures.

A.8.9.3.2 RPC & TCP/IP Servers. A separate test to verify that the RPC and TCP/IP Servers perform the connections to the backend database servers was not required. Instead, the tests outlined under, “The JAVA Graphical User Interface (JGUI)” were used to test these connections in conjunction with the JGUI test procedures.

A.8.9.3.3 The Java Graphical User Interface (JGUI)

Login screen

Test Description:

This test verifies that the login screen is properly displayed on the client machine and that the login provides the required security for allowing users to access the JMD system. At this point in the JMD IT architecture development, only a limited number of user ids were utilized to provide access to the entire JMD software. For the purposes of this test description, the pseudo-name ‘orbjmd’ has been utilized as the ‘id’ in the descriptions for the man/machine interface and dialogues.

Starting conditions:

- An up and running client-machine with the JMD client software installed and a “Command Tool window” window active to accept commands.

Test Procedure:

- change directory to /home/orbjmd/cwJMD
- Verify that the makefile ‘Makefile.HACdamio’ exists
- Start up the login screen by typing the following command:

make -f Makefile.HACdamio run_client <Enter>

Expected Results:

- The JGUI login screen displayed on the terminal is as shown in Figure A-15.

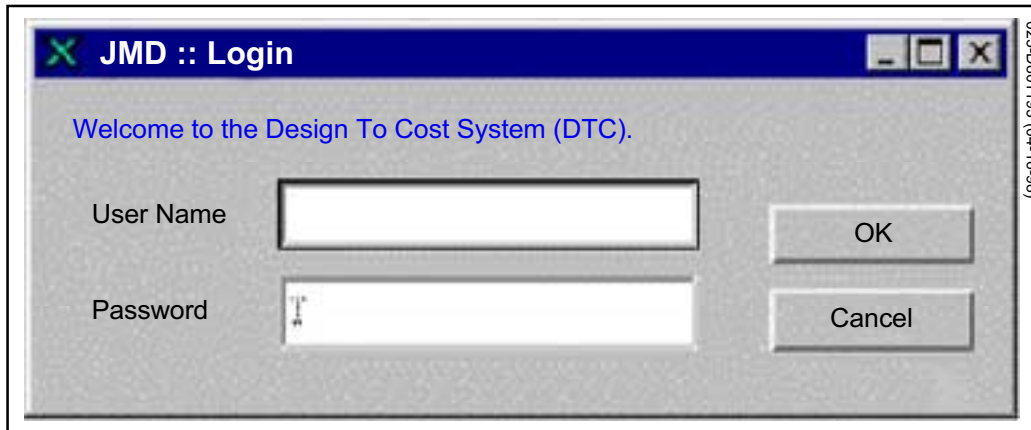


Figure A-15. JGUI's Login Screen

A.8.9.3.4 Access to JGUI's Main Screen

Test Description:

This test verifies that the main JGUI screen is displayed properly on the client machine after the correct entry user name and password.

Test Procedure:

- Perform the Login test (if not already done)
- Enter correct user name for the field "User Name" on the login screen.
- Enter correct password for the field "Password" on the login screen, then click on the "OK" button.

Expected Results:

- The main JGUI's screen should be displayed and should look like the one in Figure A-16.

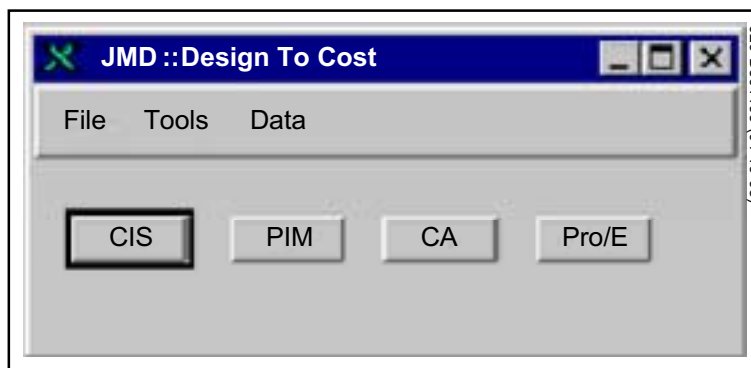


Figure A-16. JGUI's Main Screen

A.8.9.3.5 File Feature

Test Description:

The test verifies that the text editor can be invoked through the JGUI.

Test Procedure:

- Select the selection “Open Text Editor” from the pulldown menu of the “File” from the main screen as shown in Figure A-17.

Expected Results:

- The Solaris text editor will be displayed on the terminal.

Figure A-17. File Feature

A.8.9.3.6 Exit JGUI

Test Description:

This test verifies that user can exit out of the JGUI.

Test Procedure:

- Select the selection “Exit DTC” from the pulldown menu of the “File” from the main screen as shown in Figure A-17.

Expected Results:

- All of the JGUI’s active windows will disappear from the terminal and only those windows that were active prior to initiating the JGUI will remain.

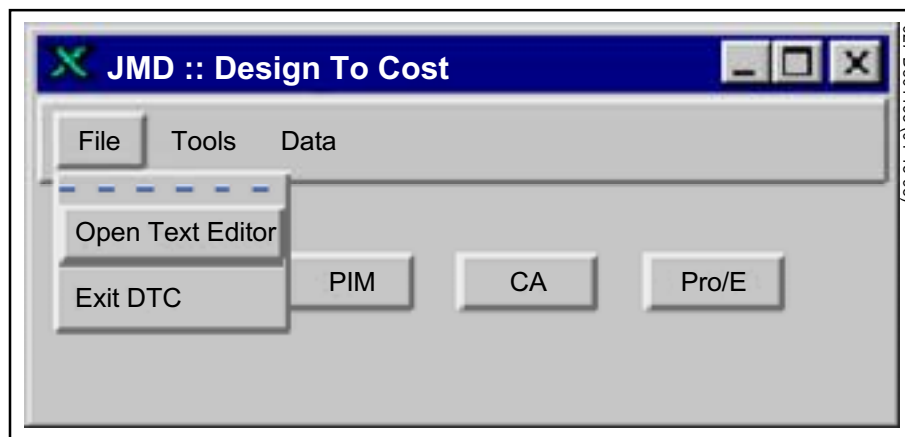


Figure A-17. JGUI’s Login Screen

A.8.9.3.7 CIS (Components Information System)

Test Description:

This test verifies that the CIS application can be invoked from the JGUI.

Test Procedure:

- Double click on the button labeled “CIS” from the main JGUI, or select the selection “CIS” in the pulldown menu under “Tool.”

Expected Results:

- The CIS application window will be displayed on the terminal at which point the User's Manual and CIS operating procedures apply.

A.8.9.3.8 PIM (Product Information Management)

Test Description:

This test verifies that the PIM application can be invoked from the JGUI.

Test Procedure:

- Double click on the button labeled "PIM" from the main JGUI, or select the selection "PIM" from the pulldown menu under "Tool."

Expected Results:

- The PIM application window will be displayed on the terminal at which point the User's Manual and PIM operating procedures apply.

A.8.9.3.9 CA (Cost Advantage System)

Test Description:

This test verifies that the CA application can be invoked from the JGUI.

Test Procedure:

- Double click on the button labeled "CA" from the main JGUI, or select the selection "CA" from the pulldown menu under "Tool."

Expected Results:

- The CA application window will be displayed on the terminal at which point the User's Manual and CA operating procedures apply.

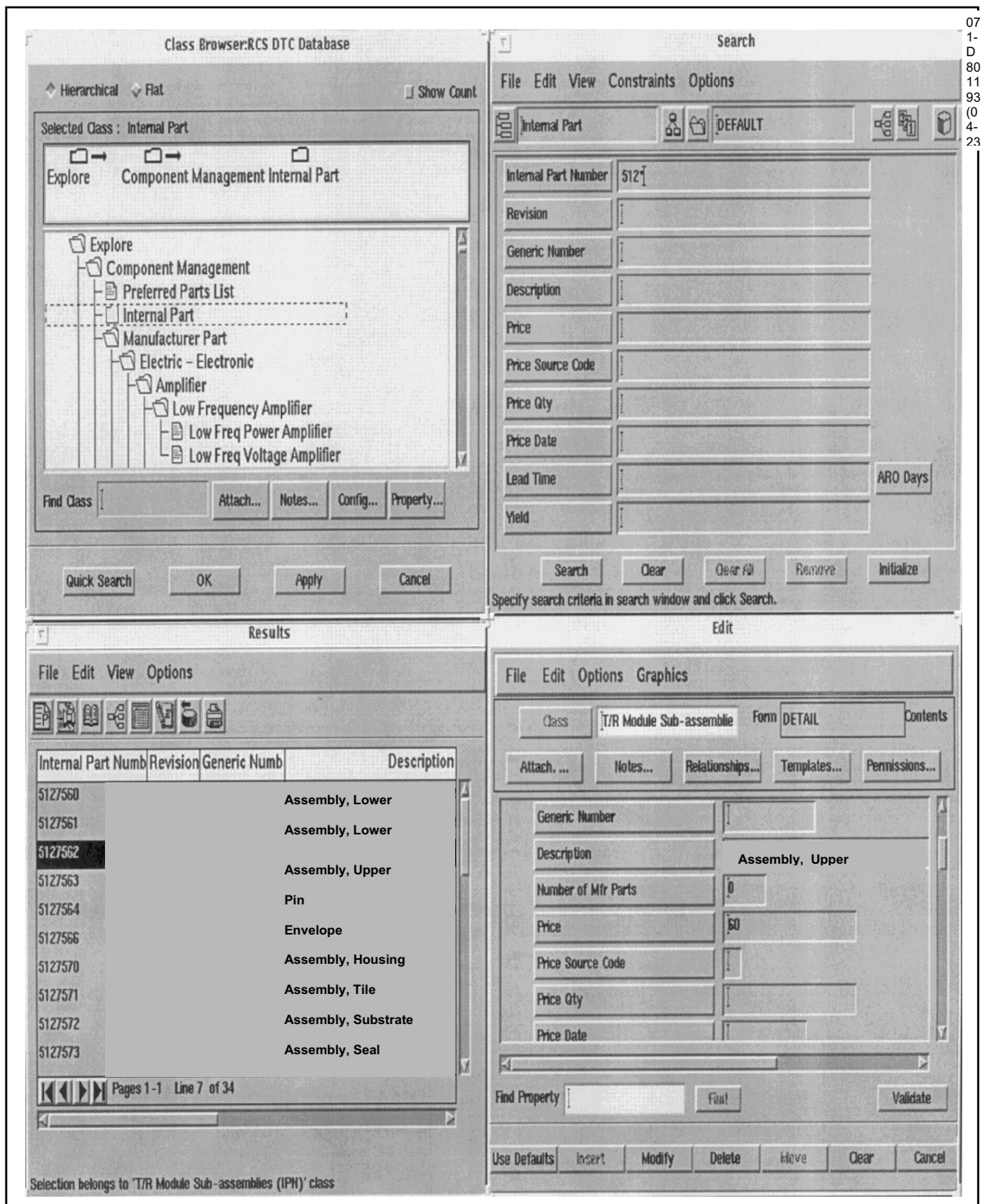


Figure A-18. CIS Application Window

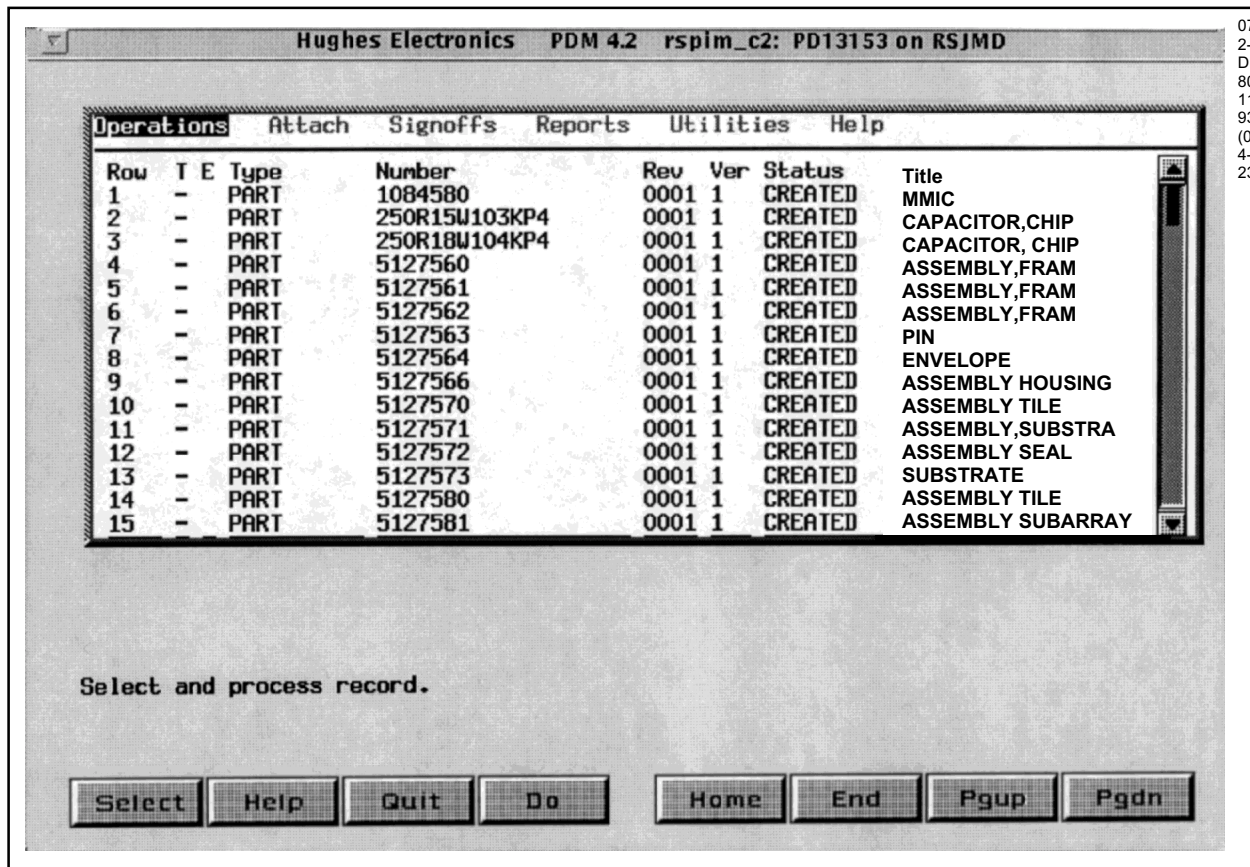


Figure A-19. PIM Application Window

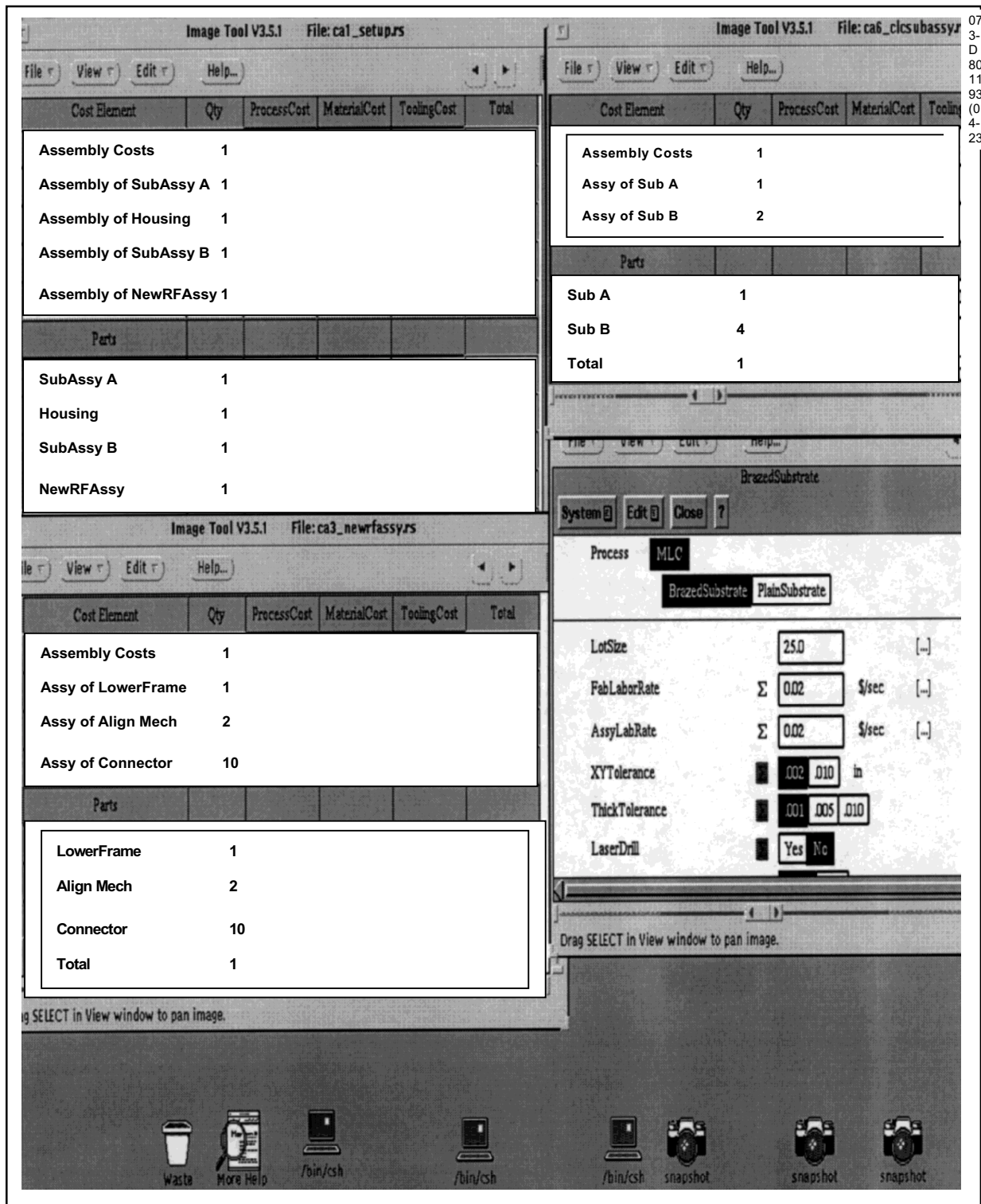


Figure A-20. CA Application Window

A.8.9.3.10 Pro/E (Engineering CAD Tool)

Test Description:

This test verifies that the Pro/E can be invoked from the JGUI.

Test Procedure:

- Double click on the button labeled “Pro/E” from the main JGUI, or select the selection “Pro/E” from the pulldown menu under “Tool.”

Expected Results:

- The Pro/E application will be displayed on the terminal at which point the User’s Manual and Pro/E operating procedures apply.

A.8.9.3.11 Programmatics

Test Description:

This test verifies that the backend Programmatics database is accessed and that the data is properly archived in the client-side data object.

Test Procedure:

- Select the selection “Programmatics ” from the pulldown menu of the “Data” from the main screen. See Figures A-22 and A-23.
- Enter correct information for the Project and Output Location fields.

Expected Results:

- The Programmatics data is accessed and stored in the file specified in the Output Location field and is also automatically presented to the User via the Text Editor as part of the above button sequences. After viewing the data presented by the Text Editor, “OK” returns control back to the main JGUI screen of Figure A-16.

A.8.9.3.12 Rates and Factors

Test Description:

The test is to verify that the Rates and Factors feature will work correctly.

Test Procedure:

- Select the selection “Rates and Factors ” from the pulldown menu of the “Data” from the main screen. See Figure A-24.
- Enter correct information for the Source Code and Output Location fields.

Expected Results:

- The data will be stored and can be viewed from the file that is specified in the Output Location field.

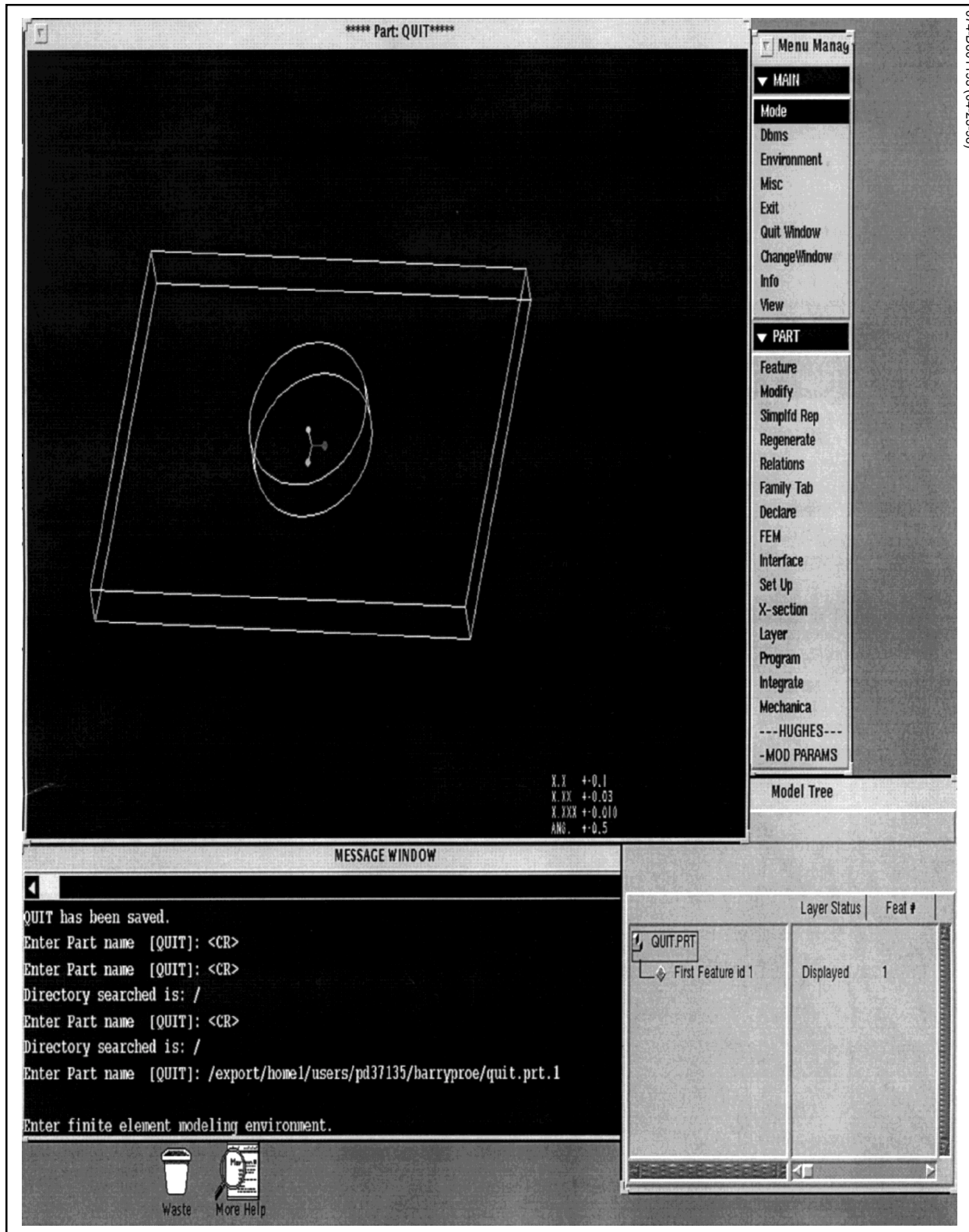


Figure A-21. Pro/E Application Window

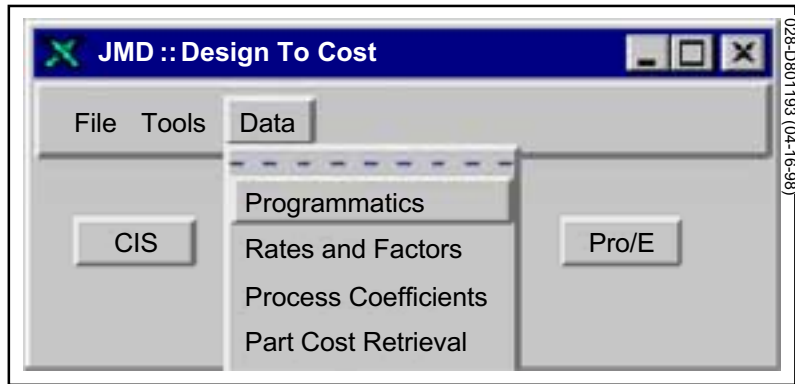


Figure A-22. Pulldown Menu Under “Data”

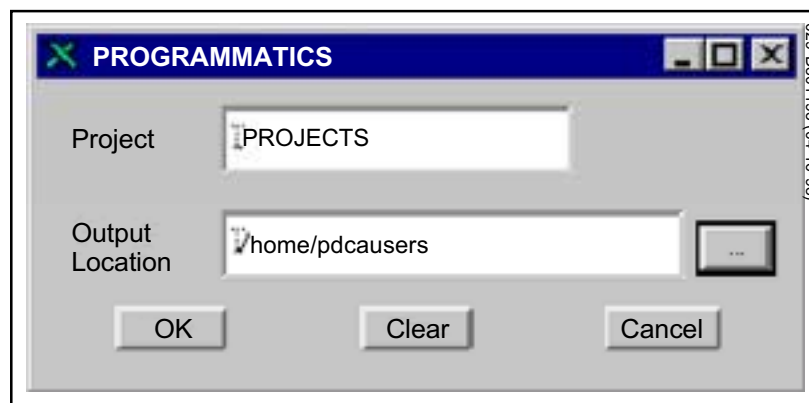


Figure A-23. Programmatics Option Under “Data”

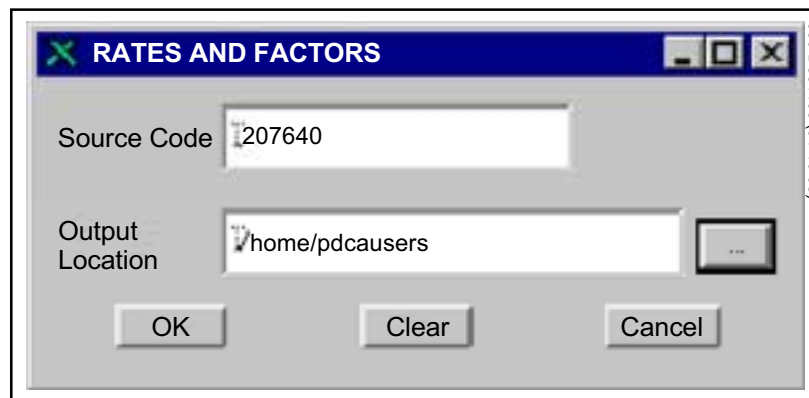


Figure A-24. Rates and Factors Option Under “Data”

A.8.9.3.13 Process Coefficients

Test Description:

The test is to verify that the Process Coefficients feature will work correctly.

Test Procedure:

- Select the selection "Process Coefficients" from the pulldown menu of the "Data" from the main screen. See Figure A-25.
- Enter correct information for the Manufacturer Location, Process and Output Location fields.

Expected Results:

- The data will be stored and can be viewed from the file that is specified in the Output Location field.

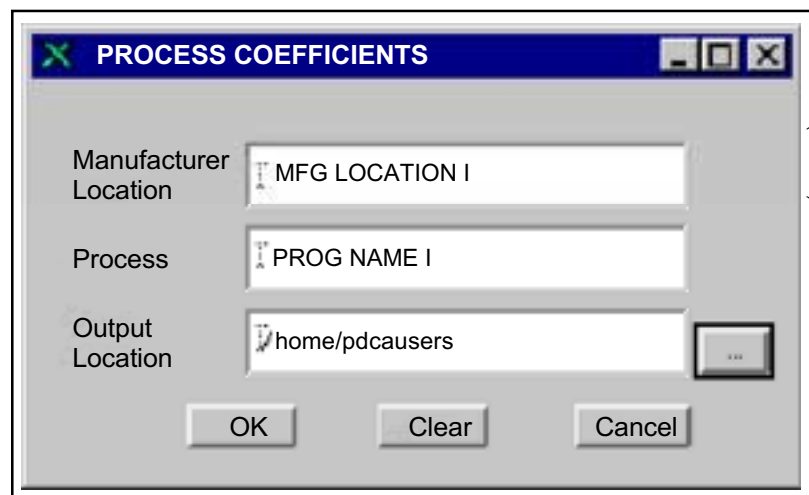


Figure A-25. Process Coefficients Option Under "Data"

A.8.9.3.14 Update

Test Description:

The test is to verify that the Estimate Cost Update feature will work correctly.

Test Procedure:

- Select the selection "Estimate Cost Update" from the pulldown menu of the "Data" from the main screen. See Figure A-26.
- Enter correct information for the Item/Part, Revision, Version, Cage Code, Promotion Status, Part Cost fields.

Expected Results:

- The new cost will be updated inside the PIM database for the candidated part.

032-D801193 (04-16-98)

Item/Part	
Revision	0001
Version	1
Cage Code	82577
Promotion Status	CREATED
Part Cost	0.00

Update Clear Cancel

Figure A-26. Estimated Cost Update Option Under “Data”

A.8.9.3.15 Part Cost Retrieval

Test Description:

The test is to verify that the Part Cost Retrieval feature will work correctly.

Test Procedure:

- Select the selection “Part Cost Retrieval” from the pulldown menu of the “Data” from the main screen. See Figure A-27.
- Enter correct information for the Option, Organization, Part List Name, Input Location, and Output Location fields.

Expected Results:

- The data will be stored and can be viewed from the file that is specified in the Output Location field.

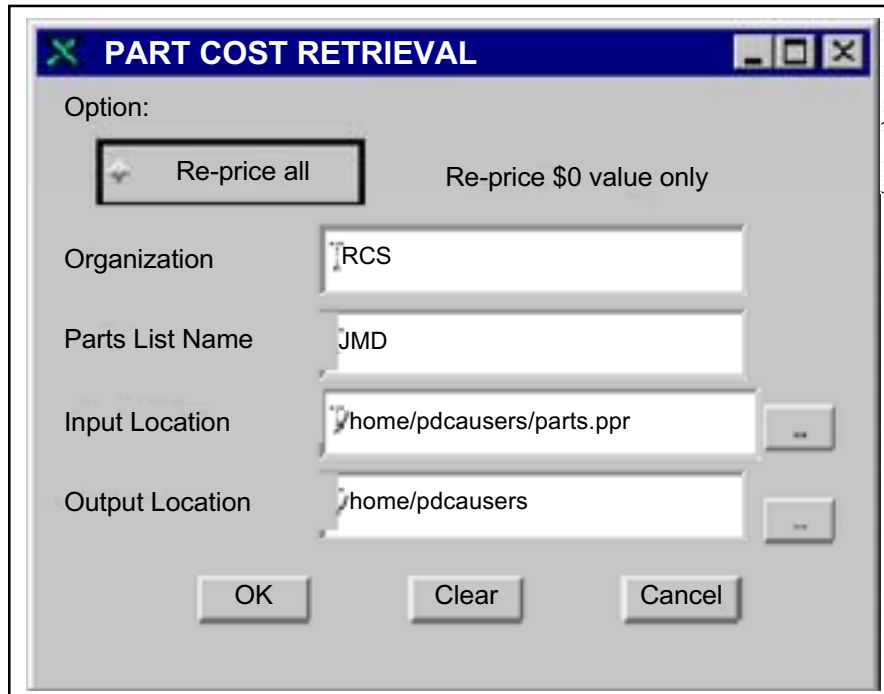


Figure A-27. Part Cost Retrieval Option Under “Data”

A.8.9.3.16 Indentured Part List

Test Description:

The test is to verify that the Indentured Part List feature will work correctly.

Test Procedure:

- Select the selection “Indentured Part List” from the pulldown menu of the “Data” from the main screen. See Figure A-28.
- Enter correct information for the IPL, PROCESS, Pro-E, CA, Item/part, Revision, Version, Cage Code, Promotion Status, Explosion Depth Level, and Output Filename fields.

Expected Results:

- The data will be stored and can be viewed from the file that is specified in the Output Location field.

PRODUCT STRUCTURE

☒ IPL
☐ PROCESS
☐ PRO-E
☐ CA

Item/Part: 5127600
Revision: 0001
Version: 1
Cage Code: 82577
Promotion Status: CREATED
Explosion Depth Level: 5
Output Filename: /home/pdcausers

OK Clear Cancel

Figure A-28. Indentured Part List Option Under “Data”

A.8.9.3.17 Log Files. The following log files exist for tracking the run-time activities of the client and server processes:

/home/JMD/bin directory:

sqlnet.log
pimrpcsvr.log
pimsvr.log
pim_pimapi.log
vda_server.log

A.8.9.3.18 Configuration Files. The following configuration files exist:

/home/JMD/bin
vda.config
pimrpcsvr.cfg
pimsvr.cfg
jmd.cfg

A.8.9.3.19 Starting SQL*NET. SQL*NET V2 is installed on rspim_c3. To start the SQL*NET, do the followings:

1. Log in to rspim_c3 server with
login: oracle
password: oracle7
2. Source /home/JMD/bin/set_orbix_env by doing the followings:

```

ksh
. set_orbix_env
(This should set the correct oracle env variables)
3. Start the listener process:
lsnrctl start
4. Check if the listener process is running by:
ps -ef | grep LIS
The following should be seen:
/oracle/7.2.3/orahome/bin/tnslsnr LISTENER -inherit

```

A.8.9.4 Performance Checklist. The following table contains a list of the man-machine functions for the JMD software. All required features in this table must have a check-mark in the **Verified** column to signify that the behavior was demonstrated through the procedures in Table A-4.

Table A-4. JMD Man-Machine Interactions

#	Man-Machine Operation	Verified
1	The startup script start_pim at /home/JMD/bin works properly and the following servers run on rspim_c2: -pimrpc_server -pimrpc_server2 -pimtcp_server	
2	The startup script start_vda at /home/JMD/bin works properly and the following servers run on rspim_c3: -vda_server -vdatwo_server	
3	The client software (JGUI) can be launched and operational on client machine by: cd /home/orbjmd/cwJMD make -f Makefile.HACdamio run_client	
4	Login screen and logic work properly.	
5	vda_server module works properly.	

Table A-4. JMD Man-Machine Interactions (Continued)

#	Man-Machine Operation	Verified
6	vdatwo_server module works properly.	
7	pimrpc_server module works properly.	
8	pimrpc_server2 module works properly.	
9	pimtcp_server module works properly.	
10	The orbix is launched and operational.	
11	The SQL*NET can be started and operational.	
12	Selection "Open Text Editor Tool" from the File pull down menu of the JMD GUI works properly.	
13	Selection "Exit DTC" from the File pull down menu of the JMD GUI works properly.	

14	Selection "Components Information System" from the Tools pull down menu of the JMD GUI works properly.	
15	Selection "Product Information Management" from the Tools pull down menu of the JMD GUI works properly.	
16	Selection "Cost Advantage System" from the Tools pull down menu of the JMD GUI works properly.	
17	Selection "Pro/E" from the Tools pull down menu of the JMD GUI works properly.	
18	Push button labeled "CIS" works properly.	
19	Push button labeled "PIM" works properly.	
20	Push button labeled "CA" works properly.	
21	Push button labeled "Pro/E" works properly.	
22	Selection "Programmatics" from the Data pull down menu of the JMD GUI works properly.	
23	Selection "Rates and Factors" from the Data pull down menu of the JMD GUI works properly.	
24	Selection "Process Coefficient" from the Data pull down menu of the JMD GUI works properly.	
25	Selection "Part Cost Retrieval" from the Data pull down menu of the JMD GUI works properly.	
26	Selection "Estimates Cost Update" from the Data pull down menu of the JMD GUI works properly.	
27	<u>jmdSrv</u> server process is installed and operational by: <pre> cd /home/JMD/bin ksh . set_orbix_env catit jmdSrv (to see if the jmdSrv exists) rmit jmdSrv (remove existing one) putit jmdSrv /home/JMD/bin/jmdSrv chmodit jmdSrv i+all (give permission to ini) chmodit jmdSrv l+all (give permission to launch) </pre>	

APPENDIX B – ACRONYMS, GLOSSARY AND REFERENCES

Acronyms

A&TS	analysis and trade studies
ABC	activity-based costing
ABM	activity-based management
API	application programming interface
ARPA	Advanced Research Project Agency
ASR	alternative systems review
ASU	Arizona State University
BA	Boeing Aerospace
BOM	bill of materials
CA	Cost Advantage
CAD	computer-aided design
CAIV	cost as an independent variable
CBU	cost build-up
CDR	critical design review
CDRL	Contract Data Requirements List
CER	cost estimating relationship
CIS	Cost Information System
CM	configuration manager or configuration management
CORBA	Common Object Request Broker Architecture
COTS	commercial off-the-shelf
CPA	Corporate Purchase Agreement
CSC	Computer Sciences Corporation
DFMA	design for manufacturing assembly
DPP	detail parts and processes
DPU	defects per unit
DSS	Decision Support System
DTC	design to cost
DTLCC	design to life cycle cost
DTUPC	design to unit production cost
E&MD	engineering and manufacturing development
EDM	electrode discharge machining
EDU	engineering development unit
ESQL	embedded structured query language
FCA	functional configuration audit
FTP	file transfer protocol
GM	General Motors Corporation
GMHE	General Motors Hughes Electronics
GUI	graphical user interface
HDMP	High Density Microwave Packaging
HEM	HE Microwave

IDL	interface definition language
Ingres	interactive graphic retrieval system
IP	Internet protocol
IPL	indentured parts list
IPPD	integrated product and process development
IPPL	indentured parts and processes list
IPR	internal process review
IPT	integrated product team
IT	information technology
JGUI	JMD Graphical User Interface
JMD	JSF Manufacturing Demonstration
JSF	Joint Strike Fighter
KCC	key control characteristic
KPC	key product characteristic
LAI	Lean Aerospace Initiative
LRIP	low rate initial production
MG	Mentor Graphics Corporation
MMIC	monolithic microwave integrated circuit
MST	Management Support Technology Corporation
NC	numerically controlled
NDI	non-development item
NRE	non-recurring engineering
O&S	operation and support
ODBC	open database connectivity
ORB	object request broker
PC	personal computer
PCA	physical configuration audit
PCT	process characterization tool/toolset
PDM	product data management
PDP	product development process
PDR	preliminary design review
PGUI	PDM Graphical User Interface
PIM	Product Information Manager
Pro/E	Pro/Engineer
PTC	Parametric Technology Corporation
RDBMS	relational database management system
RDT&E	research, development, test and evaluation
ROI	return on investment
ROM	rough order of magnitude
RPC	remote procedure call
RSC	Raytheon Systems Company
SAVE	Simulation Assessment Validation Environments
SFR	system functional review
SPC	statistical process control
SQL	structured query language

SRR	system requirements review
SSRB	Source Selection Review Board
STEP	STandard for the Exchange of Product model data
SVR	system verification review
T/R	transmit/receive
TCP	transmission control protocol
UPC	unit production cost
VDA	virtual database agent
WSC	weapons systems contractor

This page has been intentionally left blank.

Glossary

ABM	Activity-Based Management. A technique for measuring costs that differs from that traditionally used, where the overhead of an organization may be used to maintain a capability that cannot be justified by the limited use of that capability. In ABM, all major activities within an major operation are identified and the costs of performing each activity are collected and paid for by each entity using the organization resource. Thus, the cost/benefits of maintaining a capability can be easily determined.
CA	Cost Advantage. The trademark software developed by Cognition Corporation that provides expert-level design guidance and performs predictive cost analysis.
Costlink-PE	Costlink-PE. This is a registered trademark of Cognition Corporation. It provides the integration link between Pro/E and Cost Advantage.
DC	Direct Current. In the context used in this report, DC refers to the control current used to adjust the power and gain of the RF circuitry.
DFMA	Design for Manufacturing and Assembly. A method for analyzing the suitability of a design for manufacture. The analysis is normally completed when a focused group of manufacturing and assembly engineers review the design from those viewpoints.
DPP	Detailed Parts and Processes. A listing of the parts and processes used when completing an assembly. In addition to the IPL, the DPP would cover parts cost, labor, time, and six-sigma reliability figures.
EDM	Electrode Discharge Machining. A machining method in which metal removal is accomplished using an electrode discharge technique.
HDMP	High Density Microwave Packaging. A Raytheon radar array developmental program involving the use of high density multilayer ceramic packaging.
IPL	Indentured Parts List. A list of the parts used to complete an assembly, with the individual parts listed as to when they are used during the assembly process. For example, the parts constituting the level 4 assembly would be listed at level 5, etc.
JMD	JSF Manufacturing Demonstration. The name given by the customer to this study of cost analyses in support of the Joint Strike Fighter program.

LTCC	Low Temperature Cofired Ceramic. An alumina-based ceramic that is fired at low temperature (~800°C); as contrasted to high temperature cofired ceramic, which is fired at a temperature several hundred degrees higher.
PCT	Process Characterization Tool Set. An integrated set of software tools that facilitates the capture and analysis of information detailing manufacturing processes that may initially be contained in detailed process instructions, industrial engineering standards, realization factors, design guidelines, interviews memos, and cost accounts, etc. The output of this tool set, which has been tailored for the JMD program, is to put the details of these various processing steps in tabular form such that they can be readily accessed when using Cognition table linking commands. The PCT contains the information that can be accessed and used to provide the rationale to explain the appropriate CA formulas.
Pro/E	Pro/Engineer. This is a registered trademark of Parametric Technology Corporation, and refers to the computer-aided design capability of that organization.
PWB	Printed Wiring Board. A single- or multilayer organic board structure containing copper conductor traces.
RF	Radio Frequency. In the context used in this report, RF refers to high frequency radar array design considerations.
T/R	Transmit/Receive. The modules of a radar antenna can provide a transmit signal and electronically separate the signal reflection for analysis.
TD6	ToolDesign6. The machining model developed by Cognition Co.

References

1. Cost Advantage User's Guide, Cognition Corporation, 1993.
2. Costlink-PE User's Guide, Costlink-PE Version 2.0, Cognition Corporation, October 1986.
3. Orbix 2: Programming Guide, IONA Technologies Ltd., Release 2.0, November 1995.
4. Orbix 2: Reference Guide, IONA Technologies Ltd., Release 2.0, November 1995.
5. Siegel, Jon, CORBA Fundamentals and Programming, John Wiley & Sons, Inc., New York, 1996.