IDA DOCUMENT D-1439

A COMPARISON OF PRODUCT REALIZATION FRAMEWORKS

Harlow Freitag, *Task Leader*

Brian S. Cohen
Earl Ecklund
Michael Frame
Michael W. Marean
Stephen Yencho

October 1993

Prepared for
Advanced Research Projects Agency

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1993	Final

4. TITLE AND SUBTITLE
A Comparison of Product Realization Frameworks

5. FUNDING NUMBERS
MDA 903 89 C 0003

ARPA Task Number A-163

6. AUTHOR(S)
Brian S. Cohen, Earl Ecklund, Michael Frame, Michael W. Marean, Stephen Yencho

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Document D-1439

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Advanced Research Projects Agency
Software and Intelligent Systems Technology Office
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; unlimited distribution: ~~30 December 1993~~

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

In 1991 DoD established seven key science and technology thrust areas to assist in planning for investment in defense technology. The goals of Thrust 7 are to develop new industrial capabilities that will lower product unit costs and life-cycle costs, shorten the lead time in design and manufacture of products, and increase product quality. This document reviews a set of software products, applications and standards which are potential components of a DoD Thrust 7 product realization infrastructure to support design and manufacturing activities for the Advanced Technology Demonstrations (ATDs). This study is a quick sampling of software products and "frameworks" and is not intended to be either complete or comprehensive. The reviewed frameworks were each designed to meet the needs of a specific engineering discipline. No final recommendations or judgements about particular frameworks were made since a rigorous and detailed analysis of requirements has not been performed. None of the reviewed frameworks met all of the perceived needs for a Thrust 7 and ATD framework, such as the need to design and analyze ATD products both within and across engineering disciplines. The best option for obtaining a product realization framework to accommodate the Thrust 7 needs appears to be the federation of multiple existing framework technologies to achieve the needed capabilities. Future work should iinclude the extension of this study to cover all relevant frameworks and associated technology. The Thrust 7 ATD requirements should be analyzed in order to properly recommend a framework that meets those requirements.

14. SUBJECT TERMS
Product Realization; Frameworks; Computer-Aided Design; Manufacturing.

15. NUMBER OF PAGES
78

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

IDA DOCUMENT D-1439

A COMPARISON OF PRODUCT REALIZATION FRAMEWORKS

Harlow Freitag, *Task Leader*

Brian S. Cohen
Earl Ecklund
Michael Frame
Michael W. Marean
Stephen Yencho

October 1993

INSTITUTE FOR DEFENSE ANALYSES
Contract MDA 903 89 C 0003
ARPA Assignment A-163

PREFACE

This document was prepared by the Institute for Defense Analyses (IDA) for the Advanced Research Projects Agency under the Task Order, Technology for Affordability, and fulfills an objective of the task, "to develop and recommend common approaches and metrics for the infrastructure support of Thrust 7 pilot projects." This document was written in response to a request from the sponsor to examine a selected set of product realization frameworks for application to Advanced Technology Demonstrations.

The following IDA research staff members were reviewers of this document: Dr. Dennis W. Fife, Ms. Deborah Heystek, Dr. Asghar I. Noor, Dr. Karen J. Richter and Dr. Robert M. Rolfe. Dr. Robert I. Winner of the Center for High Performance Computing provided additional review comments.

EXECUTIVE SUMMARY

DoD has established seven key science and technology thrust areas to assist in planning for technology investments related to defense. The Thrust 7 goals are to develop new industrial capabilities that will lower product unit and life-cycle costs, shorten the lead time in design and manufacture of products, and increase product quality. This document describes and compares a number of computer frameworks used to design and manufacture products in the context of the Thrust 7 goals.

These frameworks consist of standards and software which allow computer-aided design systems, analysis tools, and modeling and manufacturing tools to cooperate and appropriately exchange information. The development of frameworks has the potential to significantly improve the cost effectiveness of computer tools by allowing tools to be developed and distributed independently, allowing interoperation of tools from different sources. This study initially examined a small set of frameworks at the request of the sponsor, but was expanded to a wider range of frameworks from three primary domains: electrical, mechanical and software engineering.

The frameworks examined varied in maturity from research efforts to commercial products that have been on the market for several years. The frameworks were historically developed to address issues within specific disciplines, such as electrical engineering, and frameworks strongly clustered around these disciplines. Modern product realization needs the ability to span all of these disciplines and this appeared to be a major problem facing all of these frameworks.

Some industrial trends in the area of framework technology are becoming apparent. One of these trends is standardization on interoperability issues. The CAD Framework Initiative (CFI) is an industrial cooperative effort to standardize on interoperability areas for primarily electrical engineering. Most electrical engineering framework developers are taking part and adhering to the CFI directions. The mechanical Computer-Aided Design (CAD) community has progressed toward standards in the last few years, with the ACIS and Pro/DEVELOP products being adopted by many of the MCAD vendors as supported

frameworks. Overall capabilities of mechanical CAD frameworks lag behind the capabilities of the electrical CAD frameworks.

No single framework appears to meet all the perceived Thrust 7 needs, which include interdisciplinary design and analysis capabilities. There are three possible options for obtaining a framework:

1. Develop a new framework
2. Extend an existing framework
3. Federate several existing frameworks

It appears that based on current information, the best option will be the third, to federate several existing frameworks. The development of a new framework will likely be inordinately expensive and time-consuming. Extension of an existing framework to cover multiple engineering disciplines may be difficult since existing frameworks are heavily oriented towards a specific discipline (in particular due to the exchange of data between discipline specific software), and most are proprietary.

The product realization support found in this study has been limited. Although many of the frameworks address the design phases of product realization, only limited capabilities were shown in conceptualization or manufacturing. To a great degree the support found in this area tended to be tools such as virtual manufacturing simulation rather than support for data exchange with Computer Integrated Manufacturing (CIM).

Further study is needed to identify a frameworks strategy that can effectively meet the Thrust 7 ATD needs, while remaining a cost effective alternative. The unification of the electrical, mechanical and software engineering areas will be a central problem. Innovations in framework technology in the areas of conceptualization and manufacturing will also be important.

Table of Contents

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1 PURPOSE

The purpose of this study is to provide initial evaluations of a set of software products, applications and standards as potential components of a DoD Thrust 7 product realization infrastructure to support design and manufacturing activities for Advanced Technology Demonstrations (ATDs). This study is a quick sampling of software products and "frameworks," and is not intended to be either complete or comprehensive. No final recommendations or judgements about particular frameworks are made since a rigorous and detailed analysis of requirements has not been performed. The study draws some initial conclusions about promising applications and frameworks, and recommendations about additional research that should be performed to support decisions about useful components of a Thrust 7 infrastructure.

This document has been prepared in response to a request for a comparison of a specific set of frameworks. The original list of comparison subjects included the GE/AE DICE framework employed in the GE Aircraft Engines pilot study, the framework described in the Aerospace VEHICLES proposal to ARPA, the CAD Framework Initiative (CFI), the Falcon framework, CV-DORS, the Rapid Response Manufacturing (RRM) framework, and Arcadia. The study team was unable to obtain sufficient information on the RRM framework to support any evaluation.

The study team expanded the initial list of frameworks for review based on a request from the sponsor for increased breadth of the study. The additions appear in the list shown below. The list below does not encompass all of the currently available frameworks. Rather, this list was compiled as a preliminary survey. The following frameworks and sets of specifications are reviewed:

a. Electronic Design Frameworks

   4. Falcon (Mentor Graphics)

   5. Design Framework II (Cadence)

6. OpenFrame (ViewLogic)
7. CFI (CAD Framework Initiative)
8. PowerFrame (DEC)

b. Mechanical Design Frameworks

1. GE/AE DICE (GE)
2. VEHICLE (Aerospace Corp.)
3. ACIS (Spatial Technologies)
4. CV-DORS (Computer Vision)
5. Pro/DEVELOP (Parametric Technologies Corp.)

c. Software Design Frameworks

1. Arcardia (The Arcadia Consortium)
2. ECMA/PCTE
3. CORBA/IDL (Object Management Group)

1.2 APPROACH

This study examines the infrastructure needed to support the automation of the product realization process. The use of a framework to at least partially fulfill the infrastructure requirements is examined. It should be understood that currently the development of advanced products in many areas requires extensive use of design automation tools. The infrastructure allows an automation environment to be created in a manner that is modular, extensible and particular to specific needs while taking advantage of shared resources.

The reviews of frameworks in this document are broad, and made without the benefit of a detailed review of Thrust 7 ATD requirements. This review is not complete; only a representative sample of frameworks was chosen due to time constraints. The authors have not seen actual demonstrations of the software described. Rather, this document was compiled from product information, product reviews, and personal communications with the developers. Additional sources of information include journal articles and product manuals.

1.2.1 Scope

This study is a short survey of the available frameworks. It examines the frameworks from several different points of view. The Engineering Information System (EIS) document [Winner 1986a] is used as a guide to assess the basic features of the frameworks. These EIS concepts are discussed in the context of the product realization process. A comparison of the frameworks based on the analysis of features is performed.

The application of the selected frameworks to various engineering disciplines is examined. The traditional Electrical Design Automation (EDA) frameworks are compared to other frameworks developed for software and mechanical engineering. Finally, a basic market and maturity analysis is performed. This will help in understanding how ready a framework is for immediate use.

A more detailed discussion of the comparison method is given in Section 4.1.

1.3 BACKGROUND

1.3.1 Thrust 7 and ATD Context

The "Strategic Framework for Defense Science and Technology" was enunciated by the Deputy Secretary of Defense in December 1991 as a means of DoD planning for investment in technology related to defense needs. The strategy document establishes 7 thrust areas. Five of these thrust areas are defined to deal with technology directly related to war fighting capabilities. They are global surveillance and communications, precision strike, air superiority and defense, sea control and undersea superiority, and advanced land combat vehicles. Two additional infrastructure thrusts are defined to support these five war fighting thrusts. Thrust 6 focuses on synthetic environments. Thrust 7 addresses technology for affordability.

Thrust 7 addresses the application of technology to increase hardware and software system-life-cycle affordability. The goals of Thrust 7 are to develop new industrial capabilities that will lower product unit costs and life-cycle costs, shorten the lead time in design and manufacture of products, and increase product quality. The goals are being pursued by promoting the expanded use of integrated product/process development (IPPD) methods in design and manufacturing of products for DoD and by efforts to demonstrate flexible dual-use manufacturing that permit production of defense products by factories that concurrently produce commercial products or utilize new design manufacturing techniques. In addition,

the development of information integration both within and among companies will promote effective teaming and reduce overhead expenses [McGrath 1993].

A set of Advanced Technology Demonstrations (ATDs) are planned as part of Thrust 7 to begin to implement these objectives. Several ATDs focus on missile seekers and their components. The Flexible Design and Assembly of Missile and Munitions Seekers (FDAMMS) ATD is focused on mechanical engineering and system integration issues involving seeker components including gimbals, optics and motors, IR sensors and signal processors. The Rapid Prototyping of Application Specific Signal Processors (RASSP) ATD is focused on improved design and manufacture of signal processors employed in seekers. The Infrared Focal Plane Array Flexible Manufacture (IRFPA-FM) ATD addresses design and manufacture of Detector/Dewars required for seekers. The Interferometric Fiber Optic Gyro (IFOG) ATD covers production of IFOG rate sensors that are used in stabilization of the seeker. The Active Electronically Scanned Arrays (AESA) ATD addresses electronic components used in applications such as fire-control radar.

These ATDs are addressing the development of automated tools and work environments as part of their strategy to achieve new efficiencies in design and production of these specific target products and to achieve "first-pass" successes.

The target products of the ATDs involve engineering disciplines of many types, including, at the highest level, mechanical, electrical, optical and software disciplines, with many sub-disciplines at lower levels. Tool and software development to serve these design environments will need to address the ATD-specific requirements stemming from the diverse engineering and product assembly problems.

In the context of the set of ATDs described above, and of the ATDs position as both customer and suppliers of products, the question of shared infrastructure naturally arises. The sharing of resources, tools and standards increases opportunities for efficient exchange of product requirements, design and performance information that will contribute to the improvement of product quality, reduction of product cost and reduction of schedule. Such sharing will also reduce costs incurred as a result of unnecessary duplication of analogous or identical capabilities.

1.3.2 Product Realization

Product realization has developed significantly over the last few decades allowing an engineer to design and manufacture a product by using a computer to create the design, predict its behavior and plan and control its production. In the last decade computer-aided

design tools have become a necessary part of the capabilities used to develop advanced products in fields such as digital electronics and structural design. The number of tools available to the engineer is large and the interaction of all of these tools used together is problematical. A shift in the use of these tools is occurring. That shift is away from designing for a fixed process towards designing processes concurrently as part of the product realization process.

A theme has been developing over the decades as products are designed conceptually rather than experimentally. It is now common practice to have an engineer design certain kinds of electronic products using simulation effectively to achieve "first-pass" success producing the desired product without tuning or adjustment of the fixed manufacturing process. The expansion of this idea into wider application offers significant opportunities for affordable, timely, higher quality and more maintainable products. The challenge is to achieve similar dramatic "first-pass" success in the context of products containing mechanical, optical, and software components in addition to electrical components.

One concept of a product realization process [McGrath 1993] is shown below:

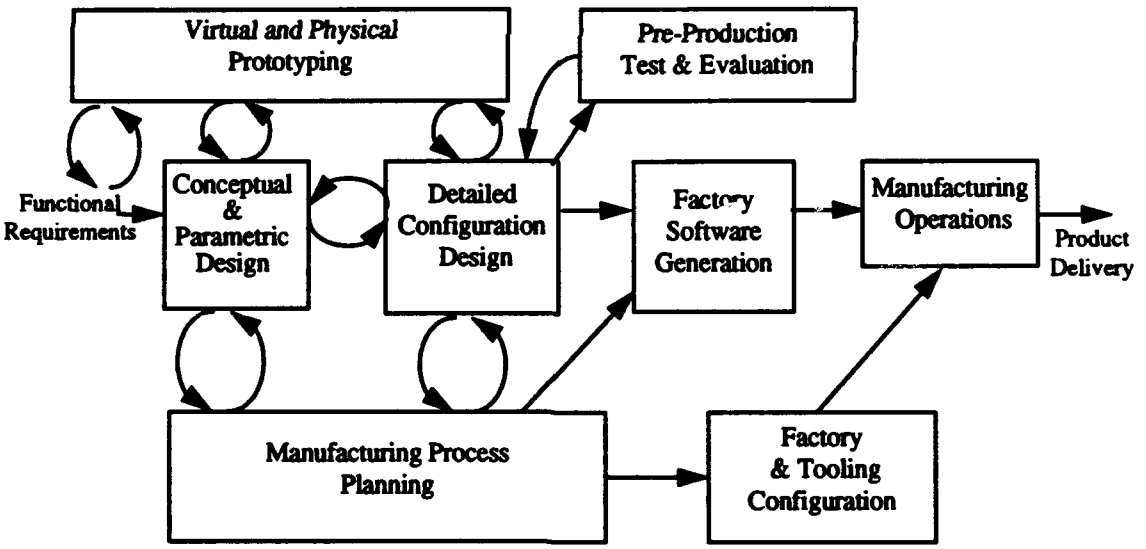

Figure 1. Product Realization Process

This view of the product realization process explicitly recognizes the interactions among product conceptualization, design and manufacturing activities. These activities have traditionally been totally separate. Significant effort has been expended in transition-

ing between them. Major product development costs have been incurred from inabilities to understand the effects of early design decisions on "downstream" engineering and manufacturing operations.

1.3.3 Design Automation Evolution

Cooperation of the separate processes involved with conceptualization, design and manufacture of a product through effective exchange and management of product information is the goal of a successful framework.

One means for achieving the needed sharing of product information is to create a monolithic piece of software that shares its product information data structures with all the algorithms required for each of the engineering disciplines involved with product realization. The difficulties of implementing such a piece of software during a time when product and process representations are evolving and when new software functions are being created have led to an alternative approach: the use of a set of distinct design and analysis tools that can interact efficiently, evolve smoothly over time, and together provide all the needed functions. Once the assumption is made that multiple vendors will write software that must cooperate, the need for standard interfaces and for a common operating environment for their products is established.

This argument for standards and a common operating environment is supported by the history of the earlier evolution that occurred in the Electronic Computer-Aided Design (ECAD) community which has been struggling with many of these issues. Product complexity, the expense of manual design techniques, and shrinking time-to-market windows drove the need for design automation for integrated circuit design. In the mid 1980s strong couplings of particular design automation tools were developed, in particular linking schematic editors, simulation tools and layout tools. These integrations used proprietary transfer formats such as netlists, GDS-II, and Gerber formats. Further developments also sought to encode strict methodologies into the tools so that correctness could be checked with respect to the manufacturing process.

A decade ago integrated electronic design automation systems were constructed by individual vendors. Mechanical design automation systems were created in the same way. These initial integrated systems were monolithic. In the last decade, a number of proprietary ECAD frameworks were developed that used tool and data integration techniques to allow different software tools to interoperate. These systems were generally closed and new tools could only be effectively integrated by the original vendor, or by using special

translators. In the last few years a number of efforts have led to the emergence of both ECAD and MCAD frameworks which are open and extensible.

Initial framework concepts originated in the late 1970s and early 1980s. In the mid 1980s, a group of DoD, government, commercial and academic interests met to focus these framework ideas. This effort resulted in the publishing of a report giving a vision of a framework [Winner 1986a/b], much of which still holds very true. Some of what was envisioned at that time has actually come to fruition and is available commercially. Other parts of the framework vision still lack implementation.

Several of the Electronic Design Automation (EDA) vendors have made significant progress in developing open frameworks. These frameworks generally address the question of how to integrate the vendors' suite of design automation tools, and how to integrate the customer's in-house tools and third party tools into the mainline vendor's environment. Many of these framework products do not address other product realization problems such as process modelling and policy enforcement. Frameworks which support Mechanical Computer-Aided Design (MCAD) have lagged EDA systems partially due to the greater difficulty in representing three dimensions, lack of effective representation abstractions and the problem that there are many more mechanical manufacturing processes than there are electronic manufacturing processes.

A measure of the motivation behind frameworks in the EDA community is the formation of the CAD Framework Initiative (CFI) in 1988. This alliance of commercial interests seeks to develop open common framework standards and interfaces, allowing vendors to develop frameworks and design automation tools that are interchangeable. This effort is a strong influence in the EDA framework community, but is clearly driven by the commercial interests of its participants.

The development of effective data exchange standards has been beneficial beyond the focused application in frameworks. The development of VHSIC Hardware Description Language (VHDL) in the 1980s has recently begun to change the way the products are realized. The success of VHDL has led to a growth in the number of standards efforts. An ever increasing number of data exchange standards are being developed with a better understanding of how a data exchange standard should be developed and managed. The information modelling language Express and Express-G are being used in a number of efforts (specifically CFI and EDIF respectively) to understand the information model and eventually to understand how different languages and formats are related. Despite the deficiencies

of these first efforts, the piecing together of languages and formats to achieve a particular data exchange between applications is still an art today.

MCAD systems historically were developed by vendors implementing the entire software system as a tightly integrated set of proprietary modules. Some vendors even developed their own operating system software, as the only means of obtaining acceptable performance and complete functionality in such areas as file management and display interaction. In recent years, hardware and software vendors have significantly improved the general capabilities of platforms, so no support in these areas is needed. This trend has progressed through uniform operating systems to a wide range of common standards and services that are available today.

Both ECAD and MCAD have benefited by a plethora of independent tools that have been developed to solve specialized application specific problems.

Complementing this improvement by the hardware vendors is a growing number of firms that provide specialized component software packages. Heavy users of design automation began to identify that monolithic design automation environments were too expensive to develop and maintain, and that the availability of modular design automation capabilities was an opportunity. The concept of a framework as a general software architectures needed to support the automation of the product realization process has matured over the last ten years.

1.4 FRAMEWORK CONCEPTS

A framework may be viewed as an architecture for integrating an enterprise. It consists of several major components which are visible from different views of the architecture. One view is the structural view of a framework, showing the relationship between tools and the framework which is shown in Figure 2.

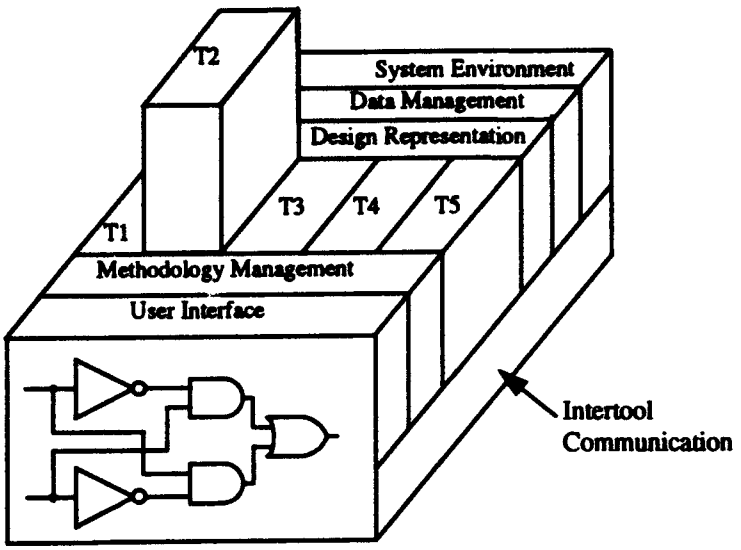

Figure 2. Structural View of a Framework[1]

1.4.1 Motivation for Using Frameworks

The DoD Science and Technology (S&T) Thrust 7 is an effort to make systems affordable, primarily by reducing the cost over the life cycle of systems. The application of affordability concepts to the Advanced Technology Demonstrations (ATDs) is the focus of this paper and the focus of this motivation.

The motivation behind frameworks is described very effectively in [Winner 1986a]. The goal is to create a product with the least cost, the highest quality and in the shortest time. Costs can be reduced by globally reducing the costs over all of the processes involved with the creation and maintenance of that product. Referring back to the product realization, it should be noted that there are a number of basic cooperating processes involved with the product realization. There are several ways to reduce costs; for example:

1. Improve the process yield.
2. Make a single process more efficient.
3. Improve the cooperation of processes.
4. Simplify the product through improved design technology.

---

[1] This view of a framework backplane was developed by George Tatge of HP, and figures of this type have appeared in framework documents from EDA Systems, MCC and others. In particular this figure is found in early CFI FAR documents and in [Brown 1992].

5. Streamline the manufacturing process.

By allowing more and better tools to be utilized, a single process can be made better. If the cost of obtaining and using tools is reduced to improve a process then this will reduce the cost of the process. A successful framework will be reduce the costs for obtaining and using tools.

Processes cooperate better through improved coordination and communication. Frameworks can help by supporting the information exchange between processes. The coordination of processes can be assisted by having uniform mechanisms for data exchange and sequencing. A framework can provide mechanisms for this coordination.

There may be some disadvantages to using frameworks, such as the performance and storage overhead associated with them, or not allowing arbitrary tools to be used for a process without integration. These disadvantages are greatly outweighed by the effective gains through tool portability and integration, and the coordination and communication that can be achieved through the product realization process.

1.4.2 Framework Goals

The original goals developed as part of the EIS effort [Winner 1986a] are still true today. These goals were defined as:

1. Integrate design tools in a cost effective manner.
2. Encourage the portability of tools.
3. Encourage a uniform design environment.
4. Facilitate the exchange of design information.
5. Provide support for design management and the reuse of previous designs.
6. Be adaptable to future changes in engineering methods.

While in more recent years another goal has emerged [Heystek 1987] that is just as much a driving force in the IPPD community, which is:

7. Monitor and control the engineering processes used for product realization.

The recent CAD Framework Initiative (CFI), which is a commercial cooperative effort to develop framework standards, as recently as 1990 affirmed a very similar list of goals [CFI 1990].

Here, for comparison, are the seven top level CFI goals:

1. Facilitate design-in-the-large.
2. Facilitate cost effective, efficient, seamless incorporation of tools into design systems.
3. Facilitate the management, sharing, reuse, and exchange of engineering information.
4. Facilitate tool and framework portability across multiple platforms.
5. Facilitate consistency across user-interfaces in the framework.
6. Facilitate capture and application of local design procedures and practices.
7. Facilitate extension of the framework.

It is important to note that achievement of all of these goals should reduce the cost of product realization. Each of these goals ties very effectively into the features that are evaluated in this paper. For more details on how each of these goals relate to comparison features see section 3.1.1.

1.4.3 Definition of Terms

For purposes of this analysis, the following terms are defined.

A **software framework** is a template set of standards which define how tools and services should interoperate and how information should be represented.[2]

An **infrastructure** is a set of networks, frameworks, and services that enable the flow of information in support of enterprise integration.

A **product realization framework** is software framework designed to provide an operating environment for tools supporting the product realization process.

- A product realization framework supports the product development and manufacturing process by enabling the interoperation of the processes involved with product realization. In particular a product realization framework supports the data and execution interoperability for computer tools and provides a core set of services that are needed across the product realization processes.

---

[2] This definition is a generalization of that used in the CAD Framework Initiative [CFI 1993]. Their definition of a CAD system is: "A CAD framework is a software infrastructure which provides a common operating environment for CAD tools."

The term "framework" is used throughout the remainder of the document to refer to a Product Realization Framework.

1.4.4 Supporting Cooperative Work

A framework should simultaneously support both an individual as well as a team in a coordinated effort for product realization. An individual may use a framework to support consistent look and feel for tools, allowing them to operate on the same platform. Tools may operate within a framework to describe and execute a given task. The framework also provides a mechanism for personalizing the environment for the individual while supporting appropriate interfaces for communication with the rest of the work group.

A framework also allows different groups to cooperatively work together, sharing resources which include tools, information and policies. A framework that supports a small set of groups can provide much tighter integration by only supporting the cooperative needs of those groups. A framework that must support a large organization involved with a variety of activities has many more requirements and is much more difficult to implement. In particular it is important to note that the framework embodies the infrastructure necessary to support all the groups. Figure 3 shows a simplified organization hierarchy. The reader should consider what a framework must support in order that cooperation can occur at the different levels.

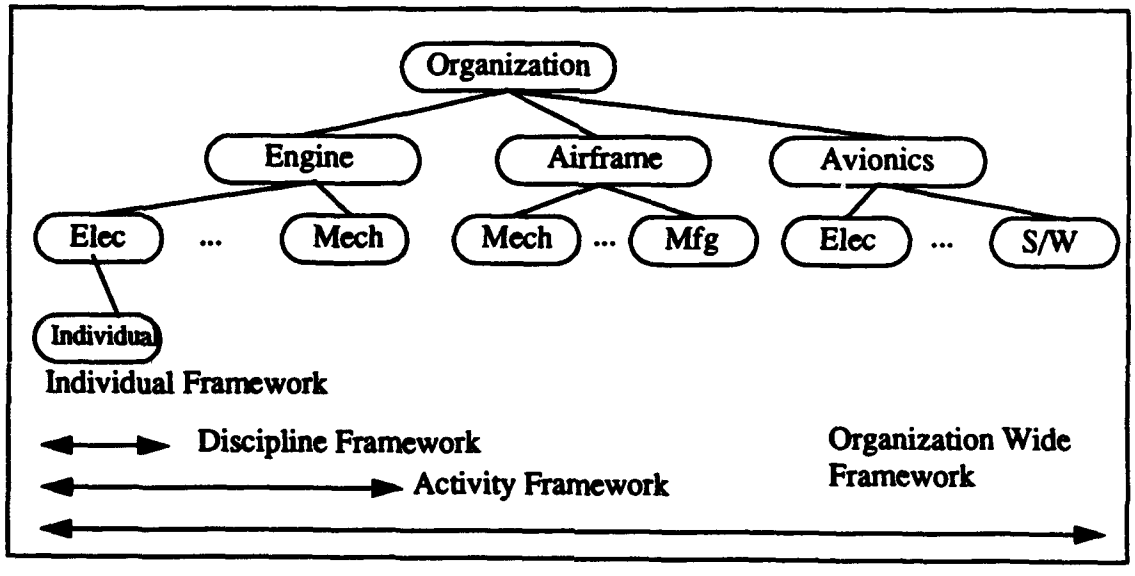

Figure 3. Current-State-of-Practice for Use of a Framework

One would clearly expect that a framework at a department level would only need to support the processes and policies of the department, while a framework at the organization level would need to support all of the processes and policies of the enterprise. The definition of what groups are to use the framework drives the definition of what services the framework provides. Since a framework provides a common set of services, a framework may also contain services that are common to the supported groups, even though that service may not generally support outside groups.

The framework has some core services which provide the central resources, such as the basic operating system facilities. The core services allow the framework to be independent of the underlying hardware and software and to handle problems of communications and distributed systems.

1.4.5 Functions of a Framework

Another view of the framework examines the different classes of functions that are expected in a framework. Figure 4 shows the basic features:

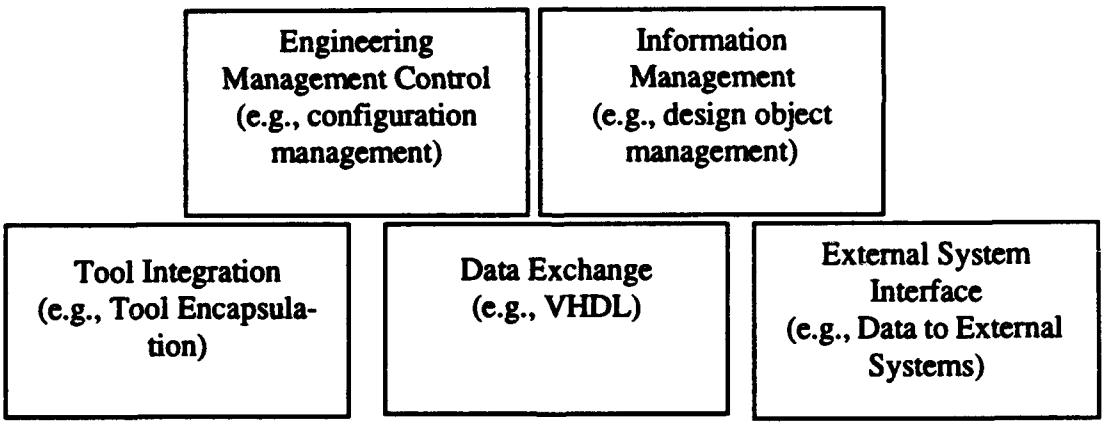

Figure 4. Framework Function Architecture

A framework is expected to allow applications and tools to be integrated. This generally means that they can be executed in the same environment and operated in a concurrent manner. Over the years this execution interoperability idea has grown into a spectrum of capabilities. The most basic ability, sometimes called launch capability supports a common mechanism for setting parameters and executing an application. Only the starting of the application is controlled through the launch capability. The basic mechanism used by most frameworks for providing launch capability is tool encapsulation. This puts a wrapper

around the tool to control the inputs and outputs so that the tool appears uniform (i.e., just like any other tool) to the framework.

More advanced execution interoperability support concurrency and control of the application during execution. Typically these are standards for communication between applications. Capabilities in current frameworks vary, but most allow basic launch facilities. Some support additional concurrent facilities like the "highlight net" inter-tool communication supported in CFI 1.0 standard. The support of actual concurrent operation of tools through advanced inter-tool cooperation is still a future vision.

The other important aspect of tool execution is the support for the user interface. Most typically a standard user interface look and feel is chosen to assure that the user sees a consistent interface. A consistent user interface on multiple platforms is clearly important, and becomes possible through platform independent display standards.

The exchange of information plays a crucial role in allowing design automation tools to interoperate. Data can be exchanged by simply passing the entire set of data as a raw set of numbers or characters, where the meaning of the data is agreed upon between tools. Data which is exchanged in this way is very difficult to use for other purposes or tools. In more advanced data exchange, the data is exchanged as information where the way to interpret meaning from the data is a part of the standard (such as in VHDL). This allows new applications to understand what the meaning of the numbers and characters are independent of the available tools. This distinction is the motivation for the development of standards for data exchange.There are several levels of data exchange, which correspond to different levels of meaning that are attached to the data as shown in Figure 5.

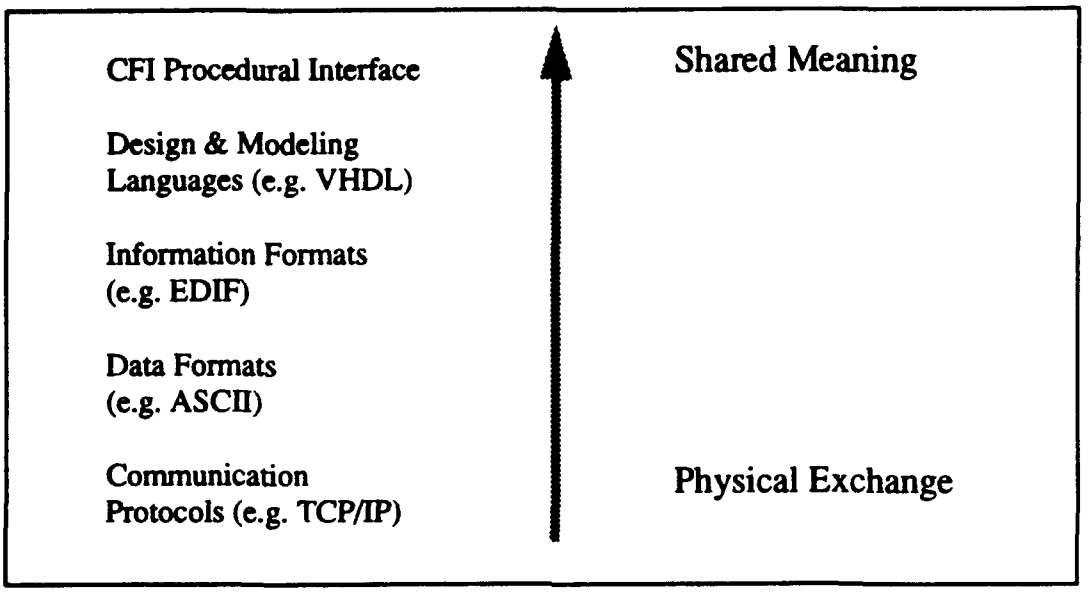

Figure 5. Levels of Data Exchange

A product realization process effectively uses cooperative work and communication between teams to couple the concurrent activities involved with conceptualization, design and manufacturing development. The ability to describe, control, execute and communicate between these processes is crucial. A framework needs to facilitate the control and management of the process of product realization. A framework also needs the facilities to enforce access controls and policy on the process. Some of the basic forms of this include change control, version control and security. More advanced forms actually encode the organizational policies that ensure the integrity of the product being realized. For instance, a design engineer may make a modification to a drawing. An organizational policy may require that a product must be simulated by a solid modelling tool to assure that it meets requirements, and then that certain people in the organization be notified of the change. These policies should be expressed as rules and automatically inserted and enforced by an advanced framework.

Information management is used to organize and effectively use the vast quantities of information that are generated and applied during the product realization process. Basic information management supports product realization by storing the information and providing access and update to it as required. Design information is managed as design objects rather than simple files. Design information includes additional information (meta data) that describes the information itself. Additional areas of support include the handling of the

naturally distributed information and the differences in platforms, which include hardware and software incompatibilities as well as the mix of machine readable information and paper information. Information management supports library functions to manage components of a product and the reusable assets of an organization. The ability to store and maintain reliable information which has accessible versions of information as it evolves, is critical to a realization process.

The framework is also called on to provide external interfaces. The framework needs to be able to let the user access the resources in a manner that makes the location of the information and computation resources transparent to the user. In many instances large resources are tied into central computation facilities, such as parts libraries, and the user needs to be able to effectively access that information. In general, effective interfaces for obtaining and sharing data from various sources is critical to the effectiveness of an integrated design automation system. A framework is expected to make this external interface transparent and effective. The framework is also expected to support the control and management functions needed for information that is imported or exported from the framework.

2. FRAMEWORK DESCRIPTIONS

2.1 OVERVIEW

This section provides overviews of the sampled framework products. These overviews are based on information collected from a variety of sources including marketing literature, published papers, framework documentation, as well as developers involved with these products. While care has been taken to obtain the best information possible within the limited time available, the presented information should be considered in the context reviewed, since some of the available information was limited to somewhat vague marketing literature.

The overviews discuss each of the products in the terms used by the individual framework vendor. At times this may be confusing since quite a number of terms are introduced, and vendors name the components of the framework without regard to any standard conventions. The overview attempts to discuss the goal of the framework, the approach used in its construction, it architecture, application domain, maturity and the application of the framework to the product realization process.

This overview considers frameworks of several types. Some of the frameworks reviewed are software packages, others are software toolkits or suites with which to develop frameworks, and others consist of a set of standards to be used by framework developers. The frameworks reviewed range in maturity from proposals and pilot studies to commercially available software packages. The frameworks reviewed fall into three primary disciplines: ECAD frameworks, MCAD frameworks, and software design frameworks.

The ECAD frameworks reviewed are: Falcon by Mentor Graphics, Design Framework II by Cadence, OpenFrame by ViewLogic, CFI, and PowerFrame by DEC. Falcon and Design Framework II are commercially available software packages. CFI Release 1.0 is a set of framework development specifications from the CAD Framework Initiative, a non-profit group which develops specifications primarily for ECAD frameworks. OpenFrame and PowerFrame are designed to be used together.

The MCAD frameworks reviewed are: GE/AE DICE framework, VEHICLE by the Aerospace Corp., ACIS by Spatial Technology, Inc., CV-DORS by Computer Vision, and Pro/DEVELOP by Parametric Technologies Corp. GE/AE DICE is the result of a pilot study on developing a framework for the design of hollow airfoils. VEHICLE is a software prototype and development proposal by the Aerospace Corp to build an MCAD framework using the technologies of wrappings and weaves. ACIS is a toolkit for developing MCAD frameworks using geometric modelers. Pro/DEVELOP is a library of C functions for extending the functionality of the Pro/ENGINEER MCAD system.

The software engineering frameworks reviewed are: Arcardia by the Arcadia Consortium, ECMA/PCTE, and CORBA/IDL by the Object Management Group. Arcadia is an effort funded by ARPA to develop a framework to assist in the development of software applications. ECMA/PCTE is an operational standard developed to provide tool portability across a wide range of platforms. Although originally started as a European effort, NIST now has adopted the ECMA/PCTE model. CORBA/IDL is an object oriented request mechanism that supports general messaging, and has gained some acceptance as a common means of communication between tools.

The following sections address each of the frameworks described above in more detail.

2.1.1 Falcon Framework (Mentor Graphics)

Mentor Graphics, founded in 1981, is an electronic design automation vendor offering tools for digital and analog design, focusing on PCB, MCM, ASIC, custom IC, and systems. Mentor Graphics 1992 annual sales were $350.7 million, and Mentor employs approximately 2200 people worldwide.

The goal of the Falcon Framework is to provide a superior, open electronic design environment. Falcon's integration technology enables users to seamlessly integrate internal and multi-vendor tool and design data suites into the Mentor Graphics environment. Enhanced with productivity tools, the Falcon Framework helps users achieve increased productivity and decreased time to market.

The components of Falcon Framework are: Advanced Multi-Purpose Language (AMPLE), Common User Interface (CUI), Design Management Environment (DME), Decision Support System (DSS), and documentation software (integrated FrameMaker). Also included are BOLD for on-line documentation delivery, printer/plotter support, and

network licensing support. AMPLE, CUI, and DME are used to integrate tools into the Falcon environment. AMPLE, DSS, and the documentation tools enhance user productivity.

AMPLE is the extension language provided with the Falcon Framework. AMPLE, which has a syntax resembling C, is used to implement wrappers for tool launch, to access CUI, DME and DSS services of the framework, and to implement extensions to the design environment, such as writing functions or macros to tailor the EDA environment to a user's unique requirements. An integrated tool can link C functions to an AMPLE API, as well as allowing integrators access to Falcon Framework services through the AMPLE APIs.

The CUI services provide an OSF/Motif compliant user interface. The AMPLE API enables third party tools to manage windows, menus, dialog boxes, prompt bars, and popup command lines, and achieve an appearance and functionality consistent with Mentor's tools.

DME offers a common desktop for consistent management of tools and data in the Falcon environment. The point-and-click iconic desktop is used for tool invocation, as well as performing design object management. Design object management includes creating, browsing and editing references (associations between design objects) and properties, configuration management, and design release and version control. The Registrar is a DME tool used to encapsulate tools within the framework. The integrated Design Manager (iDM) is used by integrated tools for design object browse/navigation, to move, copy, delete and change references, and to browse design hierarchies.

DSS is a preeminent productivity tool, combining a spreadsheet user interface, links to external data, and a visual control panel building-block toolkit. DSS allows users to build real-time applications that can monitor design data, monitor events, and present the results of complex data analysis. DSS "watchers," an integrated piece of the Falcon Framework, provide the capability to monitor and analyze design data, and then take actions based on status or calculated results. Numerical results of analysis can be presented as text, or visually using meters, gauges, or other gadgets in the DSS control-panel interface. Multi-user applications built with DSS typically consist of C code, AMPLE code, and DSS, which ties everything together. DSS is the most visible of the three since the user interface control panels are built with DSS.

Documentation tools include an on-line information system (BOLD), text editing (Notepad), word processing (integrated FrameMaker), and support for many printing and plotting options. Notepad is an ascii editor that allows access to a consistent, mouse-based

text editor from within an application. FrameMaker is fully integrated with the Falcon Framework to provide consistent documentation capabilities within engineering environments. Mentor Graphic's External Rendering Interface enables text and graphics from Mentor's design tools to be imported into FrameMaker by external reference capabilities.

BOLD delivers searching, viewing and printing of on-line documentation, including full text search, hyperlinks (predefined or user defined) to view across cross-references among related documents, printing hard copy pages of the information viewed on-line, and cut-and-paste text from on-line documentation into various applications. Mentor Graphics now delivers their product documentation via BOLD using a CD-ROM as media.

Mentor Graphics is very active in CFI, and is committed to adopting the CFI standards as they emerge. AMPLE will provide an alternative syntax for Scheme, the CFI extension language. Encapsulated tool launch will be supported using the CFI Tool Encapsulation Standard, and schematic (netlist) data will available through an API that conforms to the CFI Design Representation PI. The latter two products are expected to be available, following CFI certification, in 1994.

Falcon Framework is a two-year old product, yet it continues to evolve. The OpenDoor program offers other vendors access to the Falcon Framework's integration toolkits. The 1993 OpenDoor catalog lists more than 100 tools that are integrated (or encapsulated) into the Falcon Framework. In the product realization spectrum from conceptualization to manufacturing, Mentor Graphic's products primarily focus on design activities, but several OpenDoor partners integrate tools into the Mentor environment that provide links to conceptualization and manufacturing. Examples include, Ascent Logic (product requirements), Fabmaster and Mitron (links to PCB manufacturing), and Metaphase (enterprise product data management).

2.1.2 Design Framework II (Cadence)

Cadence Design Systems, formed by the merger of ECAD Inc. and SDA Systems in 1988, is an electronic design automation vendor, focusing on digital, analog and microwave design. Cadence's 1992 annual sales were $434.5 million.

The goal of Design Framework II is to enable users and third-party vendors to easily connect their tools into the Cadence Design environment, ensuring that data flows smoothly between tools, all within a single environment. Design Framework II supports both encapsulation and integration to connect tools into the framework.

Four components of Design Framework II support tool integration: Communications Manager (CMAN), User Interface Manager (UI), Teamwork Data Manager (DM), and Integrator's Toolkit (ITK). These components provide C language application program interfaces (APIs) to be used by integrated tools. SKILL, the extension language for Design Framework II, can be used to export services of an integrated tool to a SKILL API as well as allowing integrators access to Design Framework II services through the SKILL APIs.

Three components of Design Framework II support tool encapsulation: SKILL, the Open Simulation System (OSS), and the Physical Interface Environment (PIE). SKILL is used to implement wrappers for tool launch, to access CMAN, UI and DM services of the framework, and to implement extensions to the design environment, such as design flows. OSS allows transfer of netlist data to encapsulated tools. PIE allows transfer of physical netlist data and back annotating properties into the design. SKILL language APIs also provide read and write access to the Cadence databases.

CMAN uses a "data bus" architecture. Tools (and the Design Framework II) use CMAN commands to: Express-interest in a specific message or action, Notify others that an event has occurred, Export data, or Import data. Tools can use CMAN to efficiently send and receive small amounts of data. The UI services allow a tool to adopt the Design Framework II look and feel, to achieve consistency with other tools in the Cadence Design Framework II environment.

The requirements for managing individual work-in-progress and enterprise data management are so different that one product cannot satisfy both needs. Cadence provides a data management capability that satisfies this range of requirements of an individual engineer to the enterprise. Design Framework II's TeamworkDM provides the work group solution, while partners (such as Sherpa's PIMS or Control Data Systems' EDL) provide the enterprise level data management.

TeamworkDM is targeted at coordinating the data-sharing and notification needs of the design team. Teamwork Design Manager provides project setup and administration, versioning, configuration management, library management, release control, process management and archiving for multi-user project teams. Designers work within the context of a workarea running tools and creating, accessing and modifying designs. Promotion is the operation used to move (release) a file from a work-in-progress workarea to a release or integration workarea.

Since different enterprises adopt differing policies for data sharing and project management, TeamworkDM provides the building blocks to implement the required data sharing model for each framework installation. As a starting point, TeamworkDM is configured with several workarea use models to support the following project data-sharing models: Dynamic Model, Isolation Model, Formal Release Model and Flexible Release Model.

Data exchange is supported by the ITK. Data can be obtained in netlist, EDIF, VHDL, and Verilog HDL ascii formats, or through the ITK's C language API.

Cadence is very active in CFI, and is committed to adopting the CFI standards as they emerge. SKILL (a LISP dialect) will evolve to Scheme, the CFI extension language. Encapsulated tool launch will be supported using the CFI Tool Encapsulation Standard, and schematic (netlist) data will available through an API that conforms to the CFI Design Representation PI. The latter two products are expected to be available, following CFI certification, in fourth quarter of 1993.

Design Framework II is a mature product, yet it continues to evolve (e.g., Teamwork DM is new this summer). Other vendors can acquire access to Cadence's integration products through the Connections Partners program. The 1993 Connections Partners catalog lists more than 100 tools that are connected (integrated or encapsulated) into Design Framework II.

2.1.3 OpenFrame (ViewLogic)

OpenFrame is the name given to the Viewlogic Framework, which is also referred to as Powerview. It is primarily oriented towards providing a design environment for electrical design automation. Viewlogic is an electrical design automation company focusing on digital and analog design. Viewlogic, established in 1984 had $65 million in annual sales in 1992 and employes approximately 210 people.

OpenFrame links several components for framework technology together. Digital Equipment Corporation PowerDM is used to provide data management, CFI technology is used to provide tool launch integration, and a number of other standards are employed to support a wider range of data integration. A number of additional facilities have been developed by Viewlogic to round out the framework capabilities, such as ViewDoc which provides on line documentation. The conceptual view of OpenFrame is shown below:

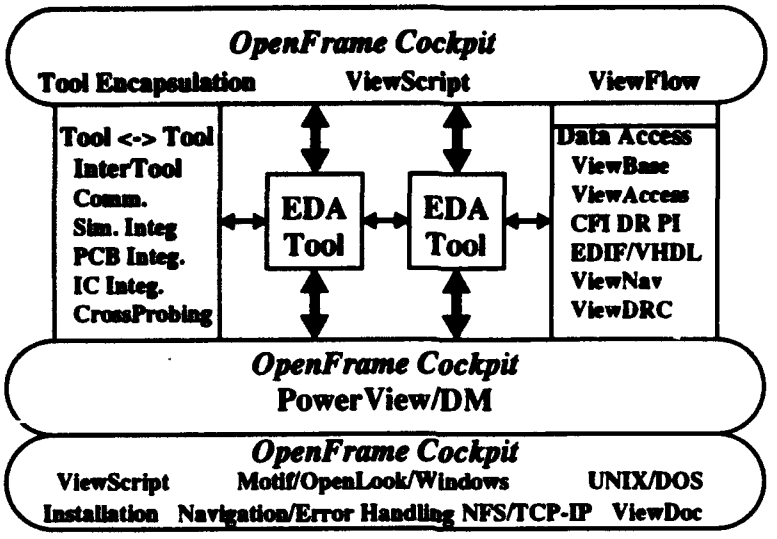

Figure 6. OpenFrame Architecture

OpenFrame currently supports both digital and analog electrical design automation. A wide range of facilities are supported for digital system design, including simulation, layout, design rule check and high level design using VHDL. Some support for analog is available with basic digital like facilities for design, and simulation using variants of SPICE. Limited enterprise wide support is available through PowerDM, but basic version control is provided.

OpenFrame is currently available commercially. It is a new product from Viewlogic and should be extended in coming years to be more CFI compliant and provide a stronger platform for integration of the users environment. OpenFrame currently distinguishes itself by the tools that are available from Viewlogic on the framework. Tools such as ViewDatabook allow corporate parts information to be integrated.

OpenFrame concentrates on the integration of the design environment for the electrical engineer. Limited support is made to integration with manufacturing. Some support is available for conceptualizing through simulation.

2.1.4 CFI (CAD Framework Initiative)

The CAD Framework Initiative (CFI) is an international non-profit consortium of CAD tool users, tool vendors, and research institutions. The CAD Framework Initiative (CFI) started in 1988 followed earlier conceptual work from the Engineering Information System project [Winner 1986a/b, Heystek 1987, Rolfe 1990] in the formation of an indus-

trial non-profit consortium. The CFI mission is to define standards that facilitate the integration and interoperability of design automation tools and design data for the benefit of end users and vendors worldwide.

CFI has been primarily influenced by the commercial vendors who have participated, who themselves are interested in adding value to their markets. As such the products from CFI have focused on the issues of tool integration and data exchange for the sets of tools that currently exist in the largest markets, namely digital electronics. Other framework issues have not in general been addressed, in particular the areas of engineering management control, information management and external system interfaces.

CFI has defined a plan for phased release of standards, the first standard, release 1.0, which is available now, includes the Design Representation, Tool Encapsulation, Inter-Tool Communication, and Computing Environment Services. The second phase will release new versions of the phase one standards and be extended to include Data Management, Simulator Backplane, User-Level Extension Language, and Session Management Support. The third phase will extend to library support and electronic databooks. The future directions in the third phase also include front-end tools with physical design implementation and manufacturing and technology management. The basic architecture proposed by CFI is shown below:

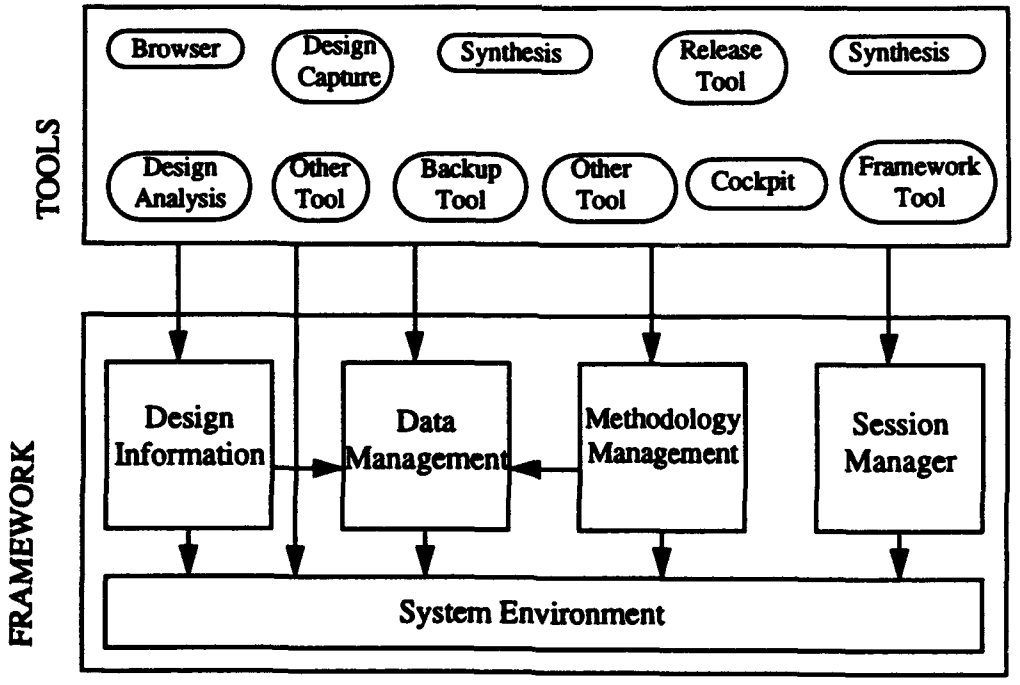

Figure 7. CFI Framework Architecture Reference

There is a subgroup of CFI that is looking at electronic semiconductor manufacturing. Subcommittees are examining two potential standards: the Semiconductor Wafer Representation [CFI 91a] and the Semiconductor Process Representation [CFI 1991b]. The efforts in this area are of particular interest to the product realization area since they support the basic data exchange with manufacturing as well as the necessary information for virtual process simulation.

2.1.5 PowerFrame (DEC)

PowerFrame is an open software framework that supports the creation of a design environment for concurrent engineering [DEC 1991]. It provides the capability to integrate third party tools and to manage design files, processes, and tools. It allows an organization to automate its engineering design guidelines. In addition to providing built-in tools, PowerFrame permits the encapsulation of third party or proprietary tools. PowerFrame is built around the concepts of the CFI "toaster model" (also known as the CFI Backplane Framework Model). DEC is one of the founding members of CFI, continues to be active, and is committed to implementing appropriate CFI standards in PowerFrame.

PowerFrame is built around the concepts of the CFI "toaster model" (also known as the CFI Backplane Framework Model). DEC is one of the founding members of CFI and continues to be active.

PowerFrame is composed of a Design Manager Server which maintains an active meta data database. The meta data keeps track of information about design, library, and tool data. The PowerFrame Executive is an X based GUI for invocation and interaction with tools, and for displaying data maintained by the Design Manager Server. The Frame Administrator utility supports framework administrators in defining which tools, data, and libraries will be used. Each tool integrated into PowerFrame is encapsulated. A definition file, or an optional Tool Agent Program (specially written), or modifications to the source code of the tool provide different levels of "tightness" of tool integration. A Transfer Manager tool is built into PowerFrame to allow data to be moved from one PowerFrame framework to another or even within a framework. PowerFrame provides built-in version control and configuration control.

Generally, PowerFrame is built around the concept of using meta data to track the activities within a project. Since there is no built-in object base, there is no requirement for tools to cooperate by adhering to standard data formats or interface protocols. Therefore, even though many types of tools may be supported by PowerFrame, it is possible that data

from one tool cannot be easily used by another. However, as standards emerge and tools adopt them, PowerFrame users will benefit from the added cooperation of tools that are already integrated.

PowerFrame is called a "design management framework" and emphasizes managing the design process. The development process itself is programmable. Since it supports such a wide variety of tools, there does not seem to be any technical obstacle to PowerFrame supporting the entire product realization process. However, DEC does not use this terminology in describing PowerFrame.

PowerFrame is intended to support electronic and mechanical engineering design. It has been a commercially available framework since at least 1991 and has a number of commercial users. In addition, DEC has established a "PowerFrame Synergy Program" with corporate members representing users, tool vendors, and others.

2.1.6 GE/AE DICE (GE)

This section reviews the framework used at GE Aircraft Engines during the DICE Hollow Airfoil Pilot Project, which sought to apply selected DICE technology to the design of hollow airfoils such as fan blades. The study was consistent with the DICE charter "to develop and field-test a methodology and accompanying integrated information management technology and tools offering a systematic approach to the concurrent engineering development process for design and manufacture, in both mechanical and electronics industries" [Czechowski 1989].

The GE/AE DICE software architecture was intended to be domain-independent. It consists of three major functional layers. The application layer is the highest layer. It includes the GE/AE DICE interface and applications. The management layer allows for management of concurrency, communication, and cooperation. The data layer is the lowest layer. It includes the representation and implementation languages and the Product, Process-activity, and Organizational resource (PPO) database, which is described below.

The GE/AE DICE pilot study had three primary goals:

- Integration of software tools through the use of wrappers
- Controlled sharing of product information through the use of configuration management tools and a product model
- Capture of design intent and product history

The GE/AE DICE architecture used at GE centered on the use of wrappers to integrate applications and data of various design groups involved within product development. [Czechowski 1989]. Wrappers encapsulate non-DICE applications to run within a DICE system. The Application Interface Wrapper tool creates wrappers around pre-existing engineering applications to permit them to read their data directly from the distributed or shared engineering data base, and/or other wrapped applications.

In the GE/AE DICE Pilot Study, software wrappers were used to create the Aster*x Parametric Design framework that was based on a spreadsheet (Aster*x). That framework contained wrapped functions of a CAD/CAM system (Unigraphics), a finite element modeling program (PATRAN), and a finite element analysis program (ANSYS) [Czechowski 1989]. This use of wrappers provided an effective union of multiple software products in order to support rapid iteration in parametric modeling. The efficiencies gained in parametric modeling were the single largest source of cycle time improvement in the concurrent engineering pilot study [Czechowski 1989].

Sharing of design information and coordination of product design activity is supported by use of a Product-Process-Organization (PPO) Model. The model contains product representation information in a number of forms. It contains process models of development activities and manufacturing processes. The capability of modeling organizations was not used in the GE/AE DICE Pilot. In this environment the PPO model contained multiple representations of the air foil product, including spread-sheets data containing parameters that specify the master-model, 2-dimensional CAD representations, 3-D dimensional models, finite element mesh models and numerical control (NC) models. The Testbed Configuration Management System (TCMS) was employed to provide control over multiple versions of designs contained in the master model.

2.1.7 VEHICLES (Aerospace Corp)

The Aerospace Corporation has developed an experimental prototype design environment named VEHICLES over the last six years to support the conceptual and parametric design stages of space-based systems [Bellman 1993]. VEHICLES integrates distributed models and analyses, solves equations symbolically, and automates parametric and tradeoff analyses. Its proponents believe this environment has application in industrial design and manufacturing, and that it is a precursor to a new industrial process that they name "information manufacturing." The Aerospace Corp. defines information manufacturing as the process of collecting, ordering, and integrating information materials into information

products. Their proposal seeks to move the Vehicles environment into industry and to highlight two software engineering technologies: wrappings and weaves. The VEHICLES environment is not now in deliverable form and is not supported by any organization.

VEHICLES is composed of:

- A flexible user interface;
- An information base containing models of subsystem characterization; of data, constraints, requirements, and conditions on use of data; containing context sensitive rules for use of algorithms, variables and sets of equations; and containing knowledge of proper context for use of software packages on distributed platforms;
- Tools for evaluation of designs; and trade-off analysis, including an independent variable sensitivity analysis tool, parametric study analysis tool, design alternatives comparison tool; and
- Report generators covering design comparison, dependencies among design parameters, input-output reports for subsystems, queries allowing for searches of parameter values across multiple designs, and a series of status reports on the state of design analysis.

The VEHICLES framework development employed the software engineering methodologies of wrapping and weaving. Wrapping is an automated process concept for deciding which software resources should be assembled to build a software system, such as a robust version of VEHICLES. A wrapping is an expert interface that describe a resource, such as a parametric design module, in a complex software system. A wrapping contains an explicit description of the resource, used for management of the system architecture. Wrappings are used by the "Study Manager" and planning programs to provide "intelligent user support functions." The Study Manager organizes the problem solving process into a sequence of steps, which are: posing the problem, interpreting the problem, applying resources, and assessing results. The wrapping functions include: selection of resources, assembly and integration of resources to exchange data, adoption of resources to a problem and explanation of how resources were used. No specific wrapping products are planned, though the wrapping approach is intended to permeate the VEHICLES system design. Wrappings are intended eventually to be used to automatically generate weaves. [Bellman 1991].

Aerospace contrasts their experimental wrappings technology from DICE wrapper technology. DICE wrappers are viewed as "code-level encapsulations" as compared to "knowledge-level encapsulations" provided by wrappings. Wrapper resource descriptions are processed off-line by programs to produce wrappers. They also disagree with the wisdom of an architecture that includes a single Product-Process-Organization (PPO) model, stating that the architecture must be able to handle multiple models and modeling paradigms.

Weaves are interconnected networks of concurrently executing tools, which communicate by passing data objects [Gorlick 1991]. Weaves are designed to support rapid and flexible construction of software systems. Weaves are composed of components that are arbitrary deterministic programs and may be written in any common sequential programming language. Components are sent objects, perform a series of functions, and then emit objects. The objects may be complex and may have methods that perform substantial computation. Weave components communicate blindly, that is, they are unaware of the identify of other components with whom they exchange objects. Weave components are supported by threads or light weight processes. Their encapsulation and self-sufficiency permit weaves to be dynamically reconfigured as they are executing. The insertion of new components can be tested without disturbing the proper functioning of existing weave components.

Standard weave objects are defined to permit legacy systems to be incorporated into a weave. Weaves are thus potentially useful as an integrating technology for multiple legacy systems.

2.1.8 CV-DORS (Computer Vision)

CV-DORS is the Computer Vision Developers Open Resource Software. It is a set of object-oriented interfaces which allow third-party software tools to interface with the underlying database, graphics, and geometry of a product model. CV-DORS allows parametric design, feature-based modeling, and has support for assemblies.

CV-DORS is based on four major architectural components: a database, a user interface, graphics, and geometry.

CV-DORS uses the ObjectStore distributed database by Object Design, Inc (ODI). The ObjectStore database is object oriented, and is written in C++. The ObjectStore database is graphically oriented. ODI currently supports several workstation vendors whose systems operate under the Unix operating system, and Microsoft Windows.

The CV-DORS user interface will be available by January 1994. The Computer Vision CADDS 5 Mechanical CAD system currently serves as a user interface for creating part geometries.

CV-DORS uses Ithaca Software's Hoops Graphics system. Graphics are integrated into the CV-DORS kernel, allowing application programs to access the geometric database of the product without the need to use a CAD application program.

CV-DORS/3D Modeler is a geometric modeler which provides integrated wire-frame, surface, and solid modeling using one common data structure. It supports geometric objects including Non-Uniform Rational B-Splines (NURBS) curves and surfaces, and solid primitives. CV-DORS/3D Modeler is double-precision and uses boundary representation to represent topology and geometry. Programmatic access to the modeler is through C++, C, or FORTRAN.

Models may be constructed with these objects through standard modeling operations such as filleting and sculpting. NURBS is an mathematical generalization of the B-spline. It can describe both two-dimensional curves and three-dimensional surfaces. NURBS assists in maintaining data homogeneity throughout the design-to-manufacture cycle. NURBS can represent shapes to arbitrary precision limited only by storage. It therefore facilitates data transfer between dissimilar systems.

CV-DORS uses the UIMX Application Programmer's Interface. This API is a commercially available toolkit which develops Motif-based interfaces. Model information is directly shared between applications, and is maintained intact. Using UIMX, the product geometry may be read from and written to without CADDS being active. As a result, users are able to describe a 3D shape in CADDS, and then transfer this underlying geometry and database between application tools. CV-DORS does not allow programmatic access to the underlying direct object interface for the system, though it does allow access at the object level.

Tool Integration with other systems is not based on the concept of a "wrapper" in which the application tool is simply encapsulated in a layer of system interface code. The disadvantage of wrappers is that they are often not as efficiently written as if the tool itself were re-written. With CV-DORS, application tools need to be re-written to be integrated with the framework. Typically, this rewrite would be done during a major release revision. Computer Vision's expectation is that the rewrite would add an additional layer for the framework interface.

CV-DORS has large potential for integrating third-party application tools. Approximately 100 third-party developers are planning to produce tools for the CV-DORS framework. Table 1 lists vendors of application tools for CV-DORS which will be available by early next year.

Table 1. CV-DORS Application Tools

Vendor	Tool Name	Functionality	Product Status
International Technegroup, Inc.			available by 1/94
PDA	PATRAN III	Finite element analysis.	available
SILMA	CimStation CADDS Interface 2.2	Simulation and off-line programming of automated manufacturing equipment such as robots and CMMs. Also performs verification of NC part programs.	available
Pointcontrol			available by 1/94
Wisdom Systems	Concept Modeler	Represents relationships between components of the product model, allows flexibility.	available
ICAD			available by 1/94
Rasna			available by 1/94

2.1.9 ACIS Geometric Modeler (Spatial Technology, Inc.)

ACIS is an object-oriented 3D modeling kernel consisting of over 300 geometric modeling functions for building geometric modeling systems. It is a toolkit of software around which applications and frameworks can be built according to a data format which is compatible with PDES/Step. ACIS is available for license by application developers to use as a mechanical engineering framework kernel. ARIES Technology's Geometry Bus is an example of a commercially available framework based on ACIS. ACIS is entirely written in C++ and it is produced by Spatial Technology, Inc. Figure 8 shows the ACIS system architecture.

ACIS is a high-performance framework kernel. Typical speeds are approximately five times as fast as CV-DORS. It offers platform support for all major workstations and PC's. ACIS has its own memory manager.

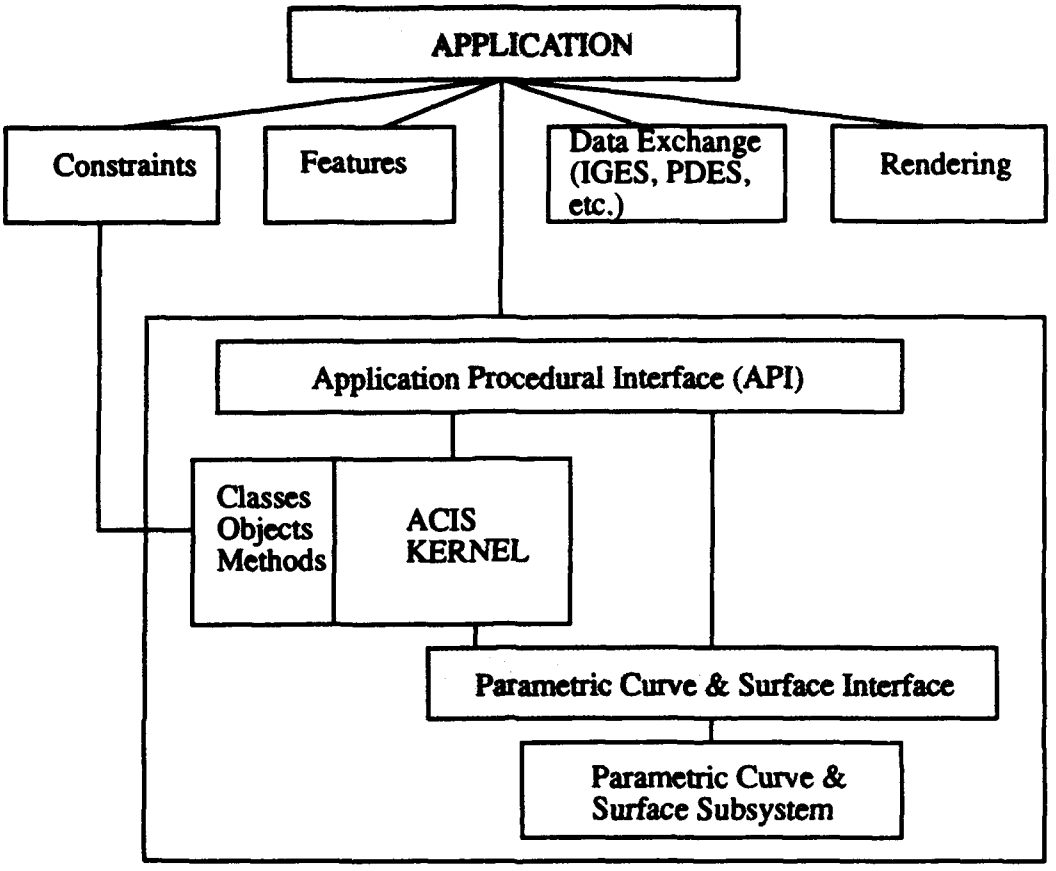

Figure 8. ACIS System Architecture

Several implementation strategies are available for using ACIS. The first of these is to re-write the application code to take advantage of ACIS. A second strategy is to build a wrapper around an existing application in order to make it ACIS-compatible. FEGS, a finite analysis program, is an example of the latter approach. Application software developers can choose to use only a portion of ACIS; an example of this is Hewlett-Packard's 3D modelling system which does not use the surface modeling capabilities of ACIS, but rather uses HP's own surface modeler. Most ACIS applications are developed with Hoops Graphics.

Applications are attached to the ACIS kernel via the Application Procedural Interface (API). ACIS toolkits are available for Microsoft Windows, Windows/NT, and Visual Basic. ACIS uses the SAT format for data exchange between applications. This format is translatable to IGES, PDES/Step, DXF, and other formats. There are three different levels to access ACIS: the first is the API level, where functions are called; the second is the

Object level; the third is the lowest level, at the data structure level, where the application developer can build in their own data structures.

ACIS has no user interface; the application tools which are based on ACIS contain the user interfaces. The development of a user interface is the responsibility of the application developers. A "test harness" is available in ACIS as a user interface for application developers.

Framework developers using ACIS have responsibility for creating the engineering management control. This is an example of the value added by ARIES with their GeometryBus product. Documentation control is also the responsibility of application developers. An object-oriented database, ACIS/DB, is available for use with ACIS. It is sold by Spatial Technology, Inc.

Table 2 lists twelve commercially-available frameworks which are based on ACIS. The ACIS kernel is currently in wide usage, with over 200 licensees outstanding, and 90 application tools being developed for it.

Table 2. ACIS-Based Commercially-Available Frameworks

Company	Product	Ship Date
Advanced Graphics Systems	*Visionael 3D*	June '92
Aries Technology	Concept Station	June '93
Cognition	Mechanical Advantage	August '93
Concurrent Technologies	Rapidcast	June '92
Control Data	ICEM DDN	January '92
Control Data	ICEM PART	September '92
Graftek	GMS/SMART	July '93
Hewlett-Packard	Solid Designer	March '93
Hitachi-Zosen Info. Systems	GRADE/Shape	June '93 (in Japan)
Lujuustekniikka Oy (Finland)	ARGOS	September '92
Strassle Infosystemes GmbH	KONCAD	June '93
Strassle Infosystemes GmbH	RWT 2000	June '93

2.1.10 Pro/DEVELOP (Parametric Technology Corporation)

Pro/DEVELOP is a library of C language functions which may be used by developers to creates application programs to modify or interface with Parametric Technology's Pro/ENGINEER MCAD system. Pro/DEVELOP has three major purposes:

1. Customize the standard Pro/ENGINEER user interface by adding new menus, new prompts, etc.
2. Automate simple or mundane tasks for the user, or automate complex design tasks for which the company has a knowledge base.
3. Integrate Pro/ENGINEER with other existing tools such as MRP schedulers or inventory control programs.

The C language functions which comprise Pro/DEVELOP primarily consist of data extraction and modification functions. Applications developers can extract geometry, features and parametric information from Pro/ENGINEER models and input this information into application tools using the C functions in Pro/DEVELOP. Application tools are made compatible with Pro/ENGINEER by using the C functions in Pro/DEVELOP. The separate application programs created by using Pro/DEVELOP are automatically started when Pro/ENGINEER is started up. Examples of application programs would be to customize the Pro/ENGINEER interface, extract data from Pro/ENGINEER, or modify objects from the Pro/ENGINEER system. Pro/DEVELOP has access to notes, symbols and 2 dimensional graphic entities in drawings. Pro/DEVELOP allows the developer to create a help file to go with new menu items which are added to Pro/ENGINEER.

The Pro/ENGINEER system outputs data in IGES, VDA, SET, or a neutral format file, or the user may write their own format for data exchange.

The user is not aware of the presence of application programs created with Pro/DEVELOP. The user is only aware of the Pro/ENGINEER interface, and the menus which may have been modified with Pro/DEVELOP.

Parametric Technology has a group of software development companies many of whom have used Pro/DEVELOP to make their complimentary software products compatible with the Pro/ENGINEER MCAD system. To date, a total of 50 companies have interfaced their products. The interfaced software products are in the following categories:

1. Document and Image Management
2. Electronic Design Automation

3. Engineering Analysis
4. Industrial Design, and Rendering
5. Information Management
6. Manufacturing
7. Translators / Migration Tools
8. Vertical Applications

There is no engineering management control built into Pro/DEVELOP. However, a separate product from PTC, Pro/PDM (Parametric Data Manager) is designed for engineering management and control capabilities. Pro/PDM uses a state-of-the-art user interface and database for data management. Although a separate application, it is tightly integrated with Pro/ENGINEER, and is supplied with a C library to allow developers to access the Pro/PDM database. Document control is also accomplished through Pro/PDM. Pro/ENGINEER is available for most Unix based workstations, the VMS operating system, and Windows/NT.

2.1.11 Arcardia (The Arcadia consortium)

Arcadia is the name of an ARPA-sponsored research project being conducted by a consortium of the University of Massachusetts, the University of California at Irvine, and Colorado University. The goal of the project is, in general, to carry out validated research on software development environments, and to establish an open architecture for process-centered software environments.

The approach taken in the Arcadia project is to develop prototypes to demonstrate the feasibility of concepts and to integrate these prototypes. A major result of the research includes the lessons learned from the effort to maintain an integrated environment.

The principle components of Arcadia are:

- Capabilities for process definition and execution
- Object management
- User interface development and management
- Measurement and evaluation
- Language processing

- Analysis and testing
- Component composition

Arcadia is intended to support the software development and maintenance. There is no restriction on the type of application it can support. In fact, a number of the capabilities are specifically to support the development of complex concurrent applications. Thus, it could be used to support real-time applications, for example an embedded control system.

Arcadia is a research project and is constantly undergoing revision in order to evaluate new alternatives. The research has been going on since the middle 1980s and many of the original researchers are still actively involved. There does not seem to be any effort to commercialize any of the components of Arcadia at this time.

A major thrust of Arcadia is the development of a process-oriented software development environment. The development process itself is programmable. The environment is primarily built around an event-driven operation. An important aspect of Arcadia is support for measurement and analysis of processes and support for management of the development process.

2.1.12 ECMA/PCTE (The European Computer Manufacturer's Association)

ECMA PCTE (Portable Common Tool Environment) is an application program interface (API) standard. It has been developed by the European Computer Manufacturer's Association (ECMA). The goal of ECMA PCTE is to define a standard interface for a number of services required to support software development and maintenance. It is expected that ECMA PCTE will be used in conjunction with other standards and services to completely support software development.

The approach taken in ECMA PCTE is to define sets of procedure calls and to define the semantics of the services to be provided. There is an interface for an abstract language, a set of C bindings, and a set of Ada bindings. ECMA PCTE is not an implementation and generally does not make any assumptions about how the various services will be implemented.

ECMA PCTE is defined as having four levels of conformance as follows:

a. Level 1 (Core Module) conformance

   1. object management

   2. schema management

3. files, pipes, and devices
4. volumes and archives
5. process execution
6. messaging
7. notification
8. concurrency and integrity control
9. replication
10. networking and distribution
11. discretionary access control (DAC security)

b. Level 2 is Core Module Conformance plus mandatory access control (MAC)

c. Level 3 is level 2 Conformance plus Auditing Services

d. Level 4 is level 3 Conformance plus Accounting Services

The application domain of ECMA PCTE is software development and maintenance. There is no assumption regarding types of applications that may be developed within the environment. However, there are no services specifically intended to support complex types of applications. It is expected that special analysis tools will be integrated by third parties in order to support various types of applications.

ECMA PCTE development was begun in 1990 and, at this time, there are no implementations of the standard. IBM has announced an implementation that partially meets Core Module conformance requirements. The NIST reference model is the same as the ECMA/PCTE reference model. The implementation is in beta testing and is expected to be released for general use in September of 1993. It should be noted, however, that ECMA PCTE is an evolutionary development of earlier PCTE standards. PCTE 1.5 and PCTE+ were developed in 1988 and implementations of PCTE 1.5 exist.

2.1.13 CORBA/IDL (The Object Management Group (OMG)[1])

The Common Object Request Broker Architecture and Specification [CORBA 1991] defines CORBA, the Common Request Broker Architecture, a framework to allow

[1] The OMG is composed of Digital Equipment Corporation, Hewlett-Packard Company, HyperDesk Corporation, NCR Corporation, Object Design, Inc., and SunSoft, Inc.

differing implementations of Object Request Brokers (ORB) to provide common ORB services to clients. CORBA supports portable clients and the implementation of objects. CORBA is structured to allow the integration of a wide variety of object systems running in a distributed, heterogeneous environment.The architecture of CORBA is shown in Figure 9.

A goal of CORBA is to provide interface standards and architectural guidelines to allow implementors of client and object server software to cooperate. In particular, a client should be able to request object services without regard to the number or placement of object servers. In fact, over time objects could move from one server to another without any effect on clients. CORBA is intended to support development of object oriented systems regardless of their application domain.

The approach taken in CORBA is to establish a standard and protocols that participants in an object oriented system would adhere to in order to work cooperatively. CORBA interfaces are defined in terms of an Interface Definition Language (IDL) which can be used to describe the syntax and the semantics of the interface between a client or server and the ORB. A syntax of IDL that is compatible with the C++ language has been developed and will be maintained along with the C++ standard.

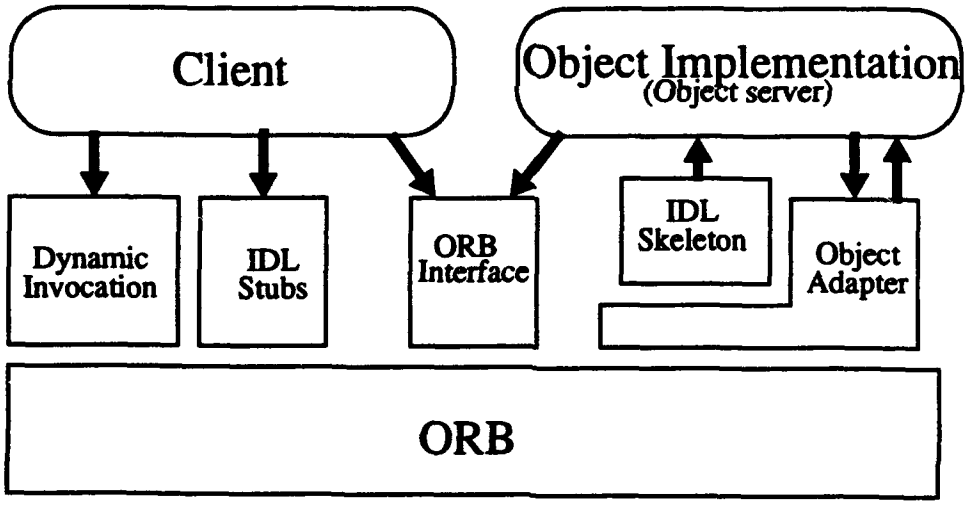

Figure 9. CORBA Architecture

CORBA is built around the concept of clients, object implementations (servers), and the ORB. The ORB provides an interface to clients and servers, IDL stubs, and a dynamic invocation capability for clients, and an IDL skeleton and object adapter for servers. A client may make an object request by using an IDL stub or by dynamically constructing the request and passing it through the dynamic invocation interface. The request is processed by the ORB which determines which server is responsible for the object, then

passes the request to the server through the IDL skeleton. The server may use the object adapter to convert the object to a form that is more usable.

As mentioned above, CORBA is not intended for any particular problem domain, but to provide a way to allow any object oriented systems to work cooperatively. It can conceivably be used to support database applications, design frameworks, or any other application that requires a number of users of different types to work together.

CORBA is not an implementation and, at this time, no implementation has been built. The designers of CORBA argue that many of the concepts upon which CORBA is built are common techniques in computing; in particular, communications protocols.

3. ANALYSES

3.1 COMPARISON METHODOLOGY

3.1.1 Feature Comparison

The features of frameworks will be compared using the concepts developed by the EIS study [Winner 1986a]. The major features discussed are:

- Tool Integration
- Data Exchange
- Engineering Management Control
- Information Management
- External System Interface
- Conformance to Standards

3.1.2 Discipline Comparison

The discipline comparison looks at the way that engineers have been trained to organize their activities. Traditional disciplines include electrical engineering and mechanical engineering. In each of these areas sub disciplines are active such as digital electronics or IR sensor engineering. Each of these disciplines is distinguished by the development of techniques for engineering solutions. These techniques frequently overlap with other disciplines, however in many cases the disciplines have unique problems that are addressed by specific techniques.

For the purposes of this comparison the frameworks are compared against the following major disciplines to determine what general areas of engineering techniques are supported:

a. Electrical Engineering

   1. Digital

   2. Analog

3. Microwave and Millimeter wave
4. Electromagnetics (Antennas)
5. Manufacturing

b. Electro-optics
1. Sensor Engineering
2. Communications
3. Cryogenics
4. Manufacturing

c. Mechanical Design
1. Mechanical CAD 2D/3D
2. Tolerance Analysis
3. Parametric Design
4. Physical Simulation

d. Manufacturing Engineering
1. Manufacturing Process Simulation
2. Statistical Process Control
3. Tachuchi /DOE Analysis
4. Cost Analysis

e. Software Engineering
1. Embedded Systems
2. MIS Applications
3. Testing/Verification/Validation

3.1.3 Market/Maturity Comparison

This comparison examines the current state of the frameworks. The frameworks was analyzed for maturity categorizing each of the frameworks according to whether they are in research, under development of actually available on the market. Where appropriate individual parts of a framework may be in different stages of maturity.

The market for each of the frameworks will be shown in Table 6 with a basic analysis of how widely the framework is applicable. Some frameworks may be so specific as to have little general use.

3.2 DETAILED COMPARISONS

This section compares the frameworks following the methodology given in Section 3.1

3.2.1 Feature Comparison

Table 3. Feature Comparison

Framework	Tool Integration	Data Exchange	Engineering Management & Control	Information Management	External System Interface
Falcon	Encapsulation via AMPLE scripts CFI Tool Encapsulation Standard AMPLE APIs (CUI, DME) DME REGISTRAR to register encapsulated or integrated tools iDM (hierarchy navigator), and Notepad (editor) support for tools	Netlist, EDIF, VHDL, CFI DR PI ERI (graphics to documentation)	AMPLE Extension Language DSS (watchers, control panels) DME (iconic desktop, references, configuration management, versions, release)	BOLD (hyperlinked online documentation) DME (design object encapsulation) DSS (analysis)	AMPLE API to Motif User Interface (CUI) DSS (spreadsheet interface, control panel builder and data integrator)
Design Framework II	Encapsulation via SKILL scripts CFI Tool Encapsulation Standard SKILL APIs (CMAN, UI, DM) OSS, PIE	Netlist, EDIF, VHDL, Verilog HDL CFI DR PI GDS-II	SKILL Extension Language TeamworkDM (versions, configuration management, library management, release, process management)	PIMS & EDL are Partners Workarea use models	SKILL API to User Interface (UI) PIMS & EDL - enterprise level data management
OpenFrame	CFI Tool Encapsulation Standard Inter Tool Communication - C procedural wrapper interface for message passing	ViewAccess - Netlist, schematic and CFI DR data exchange SimBus - Simulation Backplane PowerDM - Integration to Powerframe data manager Standards: CFI, EDIF and VHDL	ViewScript - Interpreted command language (Scheme based) PowerDM - Concurrency, audit, version control, access control, checkpoint and simple policies ViewFlow - Graphical design process modelling	ViewNav - Browse capability	ViewDatabook - Integration of corporate parts information database User Interface - Programming environment, supports Unix and Windows platforms and most workstations

Table 3. Feature Comparison (Continued)

Framework	Tool Integration	Data Exchange	Engineering Management & Control	Information Management	External System Interface
CFI	Encapsulated tool communication with framework via tool wrapper framework control tool *logging of actions* interaction with data design management system sending and receiving of messages via inter-tool communication Integrated tools Tool Ports for communication, data flow and control flow	Messages point to point and broadcast	Session manager, methodology manager	Data manager	
PowerFrame	Tool Abstraction Encapsulation Technology	Only Generic Object support, must be customized to particular discipline areas Transfer Manager supports exchange between PowerFrame frameworks	Configuration Management Version Control Task Invocation and Sequencing	Design Data Files - logical objects support projects, views, users, tools, datapacks, datasets, elements, relationships attributes and annotations	File Management Services
DICE (GE)	Wrapper based architecture Map proprietary extension protocols that applications support to standard or proto-standard protocols	Shared information model that captures complete descriptions of product or system and all associated process activities and organizational resources	Services, methods, tools, advisors that assist concept evaluation, and analysis		

Table 3. Feature Comparison (Continued)

Framework	Tool Integration	Data Exchange	Engineering Management & Control	Information Management	External System Interface
VEHICLE	Wrappings technology, "KE" supported tool integration based on explicit resource descriptions; Weaves - networks of tool fragments that communicate via objects	Through passing of objects (by reference)	Weaves - runtime-configuration of tool fragments Knowledge Bases to assist users in invoking right combinations of tools	Not a single PPO, multiple models	
CV-DORS	Not wrapper-based Tools need to be re-written to be integrated into the framework	UIMX Model data is directly shared between applications	No document control available yet Document control is not set up in developer's toolkit	ODI distributed database	PDES standard Also accepts IGES
ACIS		SAT format Translatable to PDES/Step, IGES, DXF, etc	Responsibility of developer	Responsibility of developer	
Pro/DEVELOP	Tools must be re-written with Pro/DEVELOP C libraries	Supports IGES, DXF, SET, VDA or proprietary format	May be added by application developer or Pro/PDM	Pro/PDM product provides some support	Pro/ENGINEER

Table 3. Feature Comparison (Continued)

Framework	Tool Integration	Data Exchange	Engineering Management & Control	Information Management	External System Interface
Arcadia	User Interface development tools Object Management Common representations Meta-data and translators Generics Abstract Interfaces Interface wrapping Event-based communications heterogeneous transaction support	Uses Object Bases RPC Event-based communications meta-data and translators	Process programming capability Measurement and evaluation capabilities	Object management Support for diverse objects Support for multiple languages	Uses tools developed for the purpose Has aids for developing translators
ECMA PCTE	Three levels of tool integration supported	Uses the object management service	Has some services for process control, notification, discretionary access control	*Object Management*	*Tools must be provided* Loosely integrated tools
CORBA/IDL	Tools cooperate by adhering to CORBA standard CORBA does not address user interface, task sequencing, etc	This is the central feature of CORBA, data in the form of objects can be shared throughout	None	None - The participating object implementations are responsible for information management although ORB does keep track of who the servers are and who is responsible for which objects	None

3.2.2 Discipline Comparison

Table 4. Discipline Comparison

Framework	Electrical	Electro-Optics	Manufacturing	Mechanical Design & Simulation	Software
Falcon	Digital - Strong Analog - Some		Basic Semiconductor Manufacturing Links to PCB Manufacturing	Package Design Thermal Analysis	
Design Framework II	Digital - Strong Analog - Some		Basic Semiconductor Manufacturing Links to PCB Manufacturing	Package Design Thermal Analysis	
OpenFrame	Digital - Strong Analog - Some		Basic Semiconductor Manufacturing		
CFI	Digital - Focused Support, Strong models Analog - Generic support, Little modeling		Active TCAD (semiconductor manufacturing) group, some progress towards data exchange standards		
PowerFrame	Generic product that does not immediately support any particular discipline. Has been used in Electrical and for some Structural work.				
DICE (GE)				Airfoil Design (2D and 3D analysis)	
Vehicle	Capabilities, originally created for parametric design of spacecraft, have applicability in conceptual and parametric design with applicability in multiple engineering disciplines.				
ACIS			Some data exchange support	Strong Support for 2D and 3D models and analysis	

Table 4. Discipline Comparison (Continued)

Framework	Electrical	Electro-Optics	Manufacturing	Mechanical Design & Simulation	Software
CV-DORS			Data Exchange support	Strong Support for 2D and 3D models and analysis	
ACIS			Strong tool and data exchange support	Strong Support for 2D and 3D models and analysis	
Pro/ DEVELOP			Support for MRP, inventory control	Interface tools for Pro/ ENGINEER MCAD system	
Arcadia	Could support the engineering process and integrate tools specific to each discipline, but no work has been done in this area.				Tool integration Object management Tools can be developed to support the entire SDLC (requirements to operation and maintenance)
ECMA PCTE	Could integrate tools specific to each discipline, but no work has been done in this area.				Process control, object management, security are the major services
CORBA/IDL	In the sense that the internal data structures used to representing data in these disciplines can be described by IDL, CORBA can support these areas. Frameworks for these disciplines could incorporate CORBA as the means of integrating tools at the data sharing level.				Coordination of object management. Objects (described by IDL) are restricted to Pascal or Ada-like types: simple types, record structures, and fixed and variable arrays of the others.

3.2.3 Market/Maturity Comparison

Table 5. Market/Maturity Comparison

Framework	Maturity	Market	Tools Integrated
Falcon	Commercially Available for two years	Not available	More than 100 tools integrated
Design Framework II	Commercially Available (mature product)	Not available	More than 100 tools integrated or encapsulated
OpenFrame	Commercially Available	Not Available	Unknown
CFI	Version 1 Standards released Jan 1993. Includes Design Representations, Inter-Tool Communication and Tool Encapsulation and Computing Environment Services standards.	29 US members, 8 European members and 12 Japanese members. 8 of the US members are sponsoring corporations. Some frameworks are CFI compliant in a loose manner, no separate product.	A number of tools are integrated into frameworks which are claiming CFI compliance. In general this means tool encapsulation rather than inter-tool communication.
PowerFrame	Since Jan 1990 EDA Systems originally, had on market for a few years.	Had some difficulty marketing framework. Cannot bundle tools as EDA companies have been able.	ECAD oriented Pro/DEVELOP integrated MCAD SCRC Ideas tools Thermal Analysis
DICE	Research	Not Available	In pilot study, wrappers integrate spreadsheet, CAD systems, finite element modeling, finite element analysis, CNC tool path and Superplastic Forming Analysis Code applications for particular design application.
VEHCILES	Research, for past 6 years; proposal pending to support additional work to permit transition to industry.	Not available	Weaves technology used in industrial setting to test alternative algorithms for target tracking; weaves used to simulate avionics systems.
CV-DORS	Commercially Available Release 3.0 shipping now	120 Development Systems Shipped	By 1/94, application tools will be available from: Intelligroup, PDA, SILMA, Pointcontrol, Wisdom Systems, ICAD, and RASNA.

Table 5. Market/Maturity Comparison (Continued)

Framework	Maturity	Market	Tools Integrated
ACIS	Commercially available for past three years	12 frameworks based on ACIS are now commercially available. 200 development systems shipped 90 applications currently under development	3D MCAD (solid models) FEA, FEM Thermal analysis Surface modeling Design rule processor Parametric Design Kinematic Analysis Material Properties Mgr Inertial & Mass Properties
Pro/ DEVELOP	Commercially available	Not available	Pro/PDM, Pro/ENGINEER
Arcadia	Research	Not Applicable	A few for particular applications. In addition, alternative tools have been developed to evaluate problems related to tool integration.
ECMA PCTE	A published standard. It is an evolutionary development based on earlier versions. An implementation is nearing release from IBM.	Software development environments are being actively marketed. There are three major competing approaches: PCTE based, systems based upon the Atherton Software-Backplane, and message based systems. Large development groups in the US and Europe believe environments can have a significant positive effect on the delivery of software products.	ECMA PCTE does not specify or provide any particular tools. IBM says they are working with tool vendors to integrate tools that support each phase of the software development lifecycle and that these tools will be delivered with the release in September 1993.
CORBA/ IDL	The CORBA specification was published in 1991. At this time there is no commercially available implementation of CORBA. Some vendors, such as DEC, are using CORBA in the design of integrated environments. The status of these projects is not known.	It is conceivable that CORBA could be used in any situation where applications, tools, servers, etc. need to cooperate by communicating via objects. CORBA could be integrated into many of the other frameworks mentioned here.	Not Applicable

4. SUMMARY

4.1 FRAMEWORK COMPARISON SUMMARY

The products that were examined varied in their type. Some of the frameworks reviewed are software packages, others are software toolkits or suites with which to develop frameworks, and others consist of a set of specifications to be used by framework developers. The frameworks reviewed range in maturity from proposals and pilot studies to commercially available software packages. The VEHICLES system, for example, was an early prototype system that was primarily a proposal for further development.

The frameworks reviewed were separable into distinct engineering disciplines. They fell into three primary disciplines: ECAD frameworks, MCAD frameworks, and software design frameworks. No frameworks were reviewed that spanned multiple disciplines effectively. The comparisons between frameworks that were examined are given in Section 3, which provides detailed tables. There are a number of observations that we can make about these frameworks. Some of these comparisons are summarized in Table 6.

Table 6. Framework Comparison Summary

	Type		Maturity				Discipline				
Framework	Software	Standards	Available	Development	Pilot	Proposal	Electrical	Electro-Optics	Manufacturing	Mechanical	Software
Falcon	X	X	X				X		X		
Design Framework II	X	X	X				X		X		
OpenFrame	X	X	X				X		X		
CFI		X		X			X				
PowerFrame	X		X				X				
DICE	X	X			X				X	X	

Table 6. Framework Comparison Summary (Continued)

	Type		Maturity				Discipline				
Framework	Software	Standards	Available	Development	Pilot	Proposal	Electrical	Electro-Optics	Manufacturing	Mechanical	Software
VEHICLE	X					X				X	
CV-DORS	X		X							X	
ACIS	X	X	X							X	
Pro/DEVELOP			X						X	X	
Arcardia	X			X							X
ECMA/PCTE	X	X		X							X
CORBA/IDL	X	X		X							X

4.2 CONCLUSIONS

Based on the authors' understanding of the Thrust 7 and ATD requirements, none of the reviewed frameworks meet all of the perceived needs for a Thrust 7 and ATD framework. These perceived needs include the need to design and analyze ATD products both within and across engineering disciplines.

Several of the reviewed frameworks meet a subset of the perceived Thrust 7 needs. The reviewed frameworks were each designed to meet the needs of a specific engineering discipline. None of these frameworks were designed with the needs of Thrust 7 such as product realization taken into consideration.

Since no single framework appears to meet all the perceived Thrust 7 needs, there are three possible options for obtaining a framework:

1. Develop a new framework
2. Extend an existing framework
3. Federate several existing frameworks

It appears that based on current information, the best option will be the third, to federate several existing frameworks. The development of a new framework will likely be inordinately expensive and time-consuming. Extension of an existing framework to cover multiple engineering disciplines may be difficult since existing frameworks are heavily oriented towards a specific discipline (in particular due to the data exchange), and most are proprietary.

There were a number of trends that were observed in the reviewed frameworks. In the ECAD community, the CFI standards are being readily adopted by all of the major framework vendors. The CFI standards, however, address only a narrow range of the electrical discipline. Future development of CFI standards appear to be forthcoming, but progress is slow.

The MCAD community has progressed toward standards in the last few years, with the ACIS and Pro/DEVELOP products being adopted by many of the MCAD vendors as supported frameworks. Overall capabilities in the MCAD area still lag behind the capabilities of the ECAD frameworks.

During this study no framework was found which supports the electro-optical discipline. It appeared as though no specific frameworks activity supporting that discipline has been done. The authors suspect that existing ECAD frameworks are able to support much of the needs for Electro-Optics, but this has not been verified. Frameworks are configured to an engineering discipline mostly by the choice of data exchange standards. The data exchange standards tend to be applications specific. This has tended to limit the range of applications that specific framework products address.

Software engineering frameworks tend to support an engineering methodology that is very different from frameworks which cover other disciplines. No couplings of software engineering frameworks into other disciplines was found. Weak couplings of electrical and mechanical frameworks occur in thermal analysis, electronic circuit packaging analysis and some simple electromagnetic support (such as interconnect analysis). Some concern is warranted about the coupling of software engineering frameworks with electrical, electro-optic and mechanical frameworks.

No frameworks were examined which spanned multiple disciplines (i.e., electrical and mechanical). Hence, no single existing framework is expected to meet ATD needs. The authors expected that several frameworks would need to be combined to meet ATD needs.

Electromagnetics is an important domain which spans the ECAD and MCAD disciplines, since it employs significant 3-D modelling. Electromagnetics is a potential area to bridge the electrical and mechanical engineering disciplines. No particular frameworks were found to address this bridge, although some electromagnetic analysis tools are available on ECAD frameworks. Further study is needed in this area. No specific framework support for Electro-optics was found, although the support may simply be an extension of current electrical and mechanical capabilities.

The product realization support found in this study has been limited. Although many of the frameworks address the design phases of product realization, only limited capabilities were shown in conceptualization or manufacturing. To a great degree the support found in this area tended to be tools such as virtual manufacturing simulation rather than support for data exchange with Computer Integrated Manufacturing (CIM).

The technology for creating frameworks of frameworks is still not well understood. Products such as OpenFrame use multiple framework products to create a single framework. However the techniques that could be used by the ATDs to create a framework of frameworks is not apparent.

The software engineering technologies of wrappings, wrappers, and weaves as described in the VEHICLES pilot study show promise as a method of creating frameworks, perhaps in an automated fashion. However, these technologies are very immature and would require significant further development before they could be considered for use in building frameworks.

4.3 FUTURE EFFORTS

As stated previously, this study reviewed a varied set of frameworks, but not all frameworks. Future extensions of this type of study should complete the overview of frameworks to consider all of the available relevant frameworks and technologies. There are several frameworks that were identified but not reviewed due to time constraints. Of specific interest are CIMOSA, OctTools, Nelsis and PDFab. Twelve commercially available frameworks, as listed in Table 3,were developed on the ACIS framework toolkit, but none were reviewed for this strategy.

There is a great deal of important work in standards that directly leverage frameworks. The VHDL standard has been very successful for digital electronic information exchange. Correspondingly PDES/STEP shows much the same promise in the mechanical

domain. A more thorough examination of standards should be done in conjunction with the frameworks examination.

This study needs to be extended to make recommendations for frameworks for the ATDs. An analysis of ATD requirements needs to be done in order to properly recommend a framework that meets those needs. The current ECAD and MCAD technology used in any ongoing ATD activities needs to be taken into consideration in any ATD framework recommendations.

A completed review of frameworks technology needs to be performed, extending the work in this document to cover all relevant frameworks and associated technology (such as standards). In order to achieve greater depth of analysis, access to more complete documentation and to the actual software tools should be obtained.

An initial analysis of a set of candidate frameworks should be performed based on the availability of detailed and accurate information. This analysis should identify a subset of promising framework technologies that should be analyzed in greater detail. These detailed analyses could involve steps such as evaluating the products at a site where they are in use, using the products for a benchmark design, evaluating documentation, and interviewing users. Appropriate consideration would need to be given to market directions and emerging technologies.

Based on the statement of ATD needs and this more detailed analysis, recommendations could be made on how framework technologies could meet ATD needs.

It is expected that final recommendations on frameworks will require the integration of multiple framework technologies since no single product appears to be general enough to satisfy ATD needs. A major concern in making framework recommendations is that the selected ATD framework helps to achieve the product realization goals including conceptualization and manufacturing.

LIST OF REFERENCES

[Bellman 1991] Achieving Openness and Flexibility in VEHICLES, K. L. Bellman, A. Gilliam, The Aerospace Corporation, 5 Sep 1991.

[Bellman 1993] Enhancing and Transitioning Vehicles: A Flexible Parametric Design Environment, BAA Proposal: Kirstie Bellman, et al. The Aerospace Corporation, March 12 1993.

[Brown 1992] Software Engineering Environments: Automated Support for Software Engineering, Alan W. Brown, Anthony N. Earl, and John A. McDermid, McGraw-Hill, Company, 1992.

[CFI 1990] CAD Framework Users, Goals and Objectives, Version 0.97, CFI, August 10, 1990.

[CFI 1991a] Semiconductor Wafer Representation Architecture, version 1.0, TCAD Framework Group, Semiconductor Wafer Representation Working Group, CFI, Austin, TX.

[CFI 1991b] Current Concepts in Semiconductor Process Representation, Version 0.2, February 22, 1991, Semiconductor Process Representation, TCAD Frameworks Working Group, CFI, Austin, TX.

[CFI 1993] The CAD Framework Initiative, Enabling CAD Tool Integration, CFI, Austin, TX, March 12 1993.

[Czechowski 1989] Red Book on Functional Specifications for the DICE Architecture, Joseph Czechowski, et al., Joseph Cleetus, eds, February 28 1989.

[CORBA 1991] The Common Object Request Broker: Architecture and Specification, OMG Document Number 91.12.1, Revision 1.1, the Object Management Group and X/Open, 1991.

[DEC 1991] PowerFrame Handbook, Digital Equipment Corporation, 1991.

[Gorlick 1991] Using Weaves for Software Construction and Analysis, Michael M. Gorlick, Rami R. Razouk, The Aerospace Corporation, Proceedings, IEEE 13th International Conference on Software Engineering, May 13-17, 1991.

[Heystek 1987] An Analysis of the DoD Engineering Information System as a Framework for Software Engineering, IDA Paper P-2025, Deborah Heystek and Robert Winner, 1987.

[McGrath 1993] Technology for Integrated Product/Process Design, ARPA Tech Council Review, Michael McGrath, June 14, 1993.

[Nash 1987] EIS Standards and Specifications: Preliminary Study, IDA Memorandum M-281, Sarah Nash, Katydean Price, and Joseph Linn, 1987.

[Rolfe 1990] Interim Status and Recommendations for the Engineering Information System (EIS) Program, IDA Document D-693, Robert Rolfe, Dennis Fife, Edgar Sibley and Herbert Brown, 1990.

[Weisberg 1992] David E. Weisberg, ed., Engineering Automation Report, December 1992.

[Winner 1986a] The Department of Defense Requirements for Engineering Information Systems (EIS), Volume I: Operational Concepts, IDA Paper P-1953, Robert Winner, Joeseph Linn et al, 1986.

[Winner 1986b] The Department of Defense Requirements for Engineering Information Systems (EIS), Volume II: Requirements, IDA Paper P-1953, Robert Winner, Joseph Linn et al. 1986.