On the Development of a Computing Infrastructure that Facilitates IPPD from a Decision-Based Perspective

Mark A. Hale*, James I. Craig†, Farrokh Mistree‡, Daniel P. Schrage§

Georgia Institute of Technology
Atlanta, Georgia 30332-0140

Abstract

Integrated Product and Process Development (IPPD) embodies the simultaneous application of both system and quality engineering methods throughout iterative design processes. The use of IPPD results in the time-conscious, cost-saving development of engineering systems. A computing infrastructure called IMAGE is designed to implement IPPD from a decision-based perspective. IMAGE has four components: designer activities, available assets, agent collaboration, and a computing architecture. IMAGE captures a designer's activities through a timeline partitioning scheme, problem formulation and solution, and comprehensive information management. To support these activities, IMAGE incorporates design resources through the use of agents. Agents are a critical computational enabling technology that provide accountable mechanisms for resource collaboration in an integrated computing environment.

Background and Motivation

Considerable time and effort has been invested in the development of new computing technologies and their associated methods for **I**ntegrated **P**roduct and **P**rocess **D**evelopment (IPPD). These technologies have been applied in systems that emphasize modularity, interdisciplinary program utilization, resource collaboration, and distributed processing. These systems include integrated design frameworks, conceptual design systems, and quasi-procedural systems.[1-9] Though these systems have marked improvements in information processing, their applicability to aiding a designer in making decisions based on new design knowledge remains questionable.[10] Furthermore, the applicability of these systems to continuous, iterative design processes has not been proven and is uncertain, at best.

The characteristics of a computing environment that enable a designer to perform design activities through the use of state-of-the-art computing technologies are summarized in this paper. The development is founded upon several underlying principles concerning the means by which the computing architecture embodies design activities. These principles are as follows:

- A designer would like to do **more with less** throughout all design processes. A designer would like to have more robustness, greater openness, and better efficacy while expending fewer resources, paying less cost, and reducing the number of design cycles.
- The principal role of a designer is that of a decision-maker. Decision-making activities can be marked by discrete milestones in a design process. This designer-centered approach forms the basis of **D**ecision **B**ased **D**esign (DBD) paradigm for IPPD.
- The **D**ecision **S**upport **P**roblem (DSP) Technique is one embodiment of DBD with which a designer can manage design activities. The DSP Technique has two phases. First, a designer partitions a design timeline into manageable sub-problems, called Support Problems. Second, the resulting Support Problems are solved.
- Support Problems are the basic design elements that represent design processes.
- A computing infrastructure will support design activities throughout an entire design timeline. This is true for both product *and* process design.
- A product model must support both form and function representations as well as the process by which this is done. This model is inherently object-oriented and complex inter-relationships and hierarchies exist.
- Computer design environments should be designed to assist a designer in performing design activities; therefore, a designer is "in the loop". Automation should exist only when an appropriate context can be added to design information produced by automated processes.

* Graduate Researcher, Student Member AIAA, Aerospace Systems Design Laboratory, Corresponding Author
mhale@cad.gatech.edu
http://www.cad.gatech.edu/image
† Professor, Member AIAA, Aerospace Systems Design Laboratory
‡ Professor, Member AIAA, Systems Realization Laboratory
§ Professor, Member AIAA, Aerospace Systems Design Laboratory

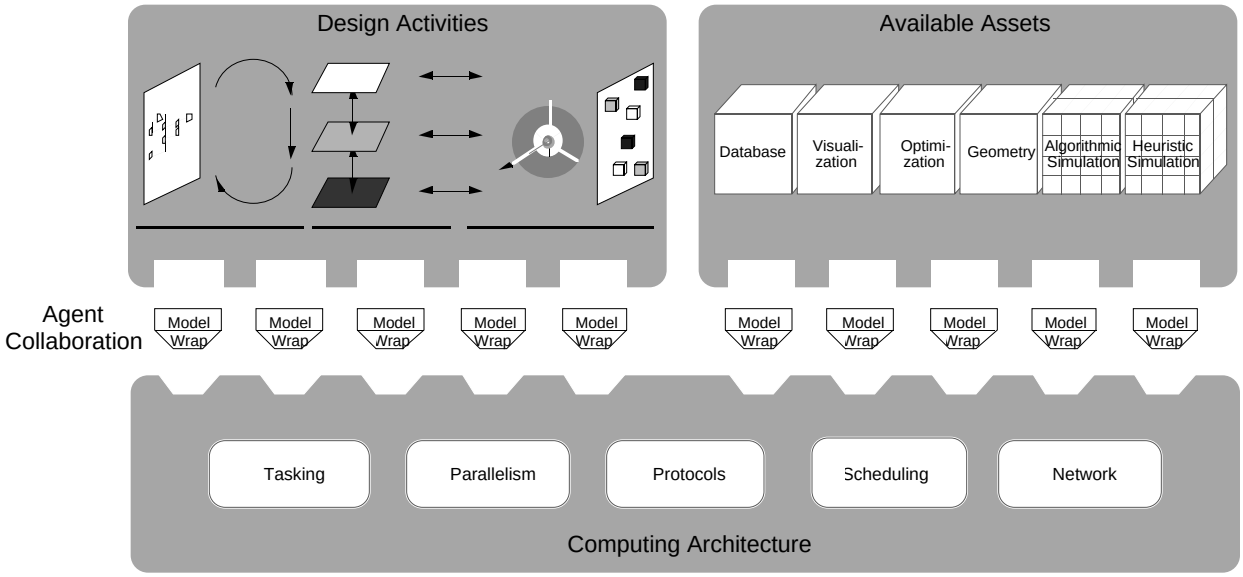


Figure 1. IMAGE Infrastructure

These principles are a foundation of design and information models. Arguments in support of a DBD perspective can be found in references [10-16], those in support of a multi-dimensional product model can be found in [10, 17, 18], and general infrastructure requirements are outlined in [2-7, 10, 18-20]. The computing infrastructure that is outlined in this paper is being developed using these principles as guidelines.

A Computing Infrastructure

A computing infrastructure can be assembled that provides coherent support for design activities. Significant work has been done in the computer science community for establishing computing technologies required for framework integration.[2-4, 9, 19-26] However, these technologies will need to be systematically applied to a coherent design architecture if a framework is to support design activities throughout a design timeline. Expanding on ideas asserted earlier, such a computing infrastructure will have to incorporate:

- A design partitioning process;
- A mechanism for solving the resulting design sub-problems;
- A design information model;
- Information generation in context for informed decision-making;
- Efficient and cost-effective application of design resources; and
- Geographically distributed design activities.

The resulting infrastructure is called IMAGE, an **I**ntelligent **M**ultidisciplinary **A**ircraft **G**eneration **E**nvironment. The infrastructure began as a special project that addressed the lack of facilities for supporting a designer's activities in traditional frameworks.[27]

A diagram of the IMAGE infrastructure is shown in Figure 1. This infrastructure is comparable to the **F**ramework for **I**nterdisciplinary **De**sign **O**ptimization (FIDO), **A**ffordable **S**ystems **O**ptimization **P**rocess (ASOP), and other efforts in the development of underlying computing technologies.[9, 28] However, the IMAGE infrastructure is designed to explicitly support general design activities and an information model within an accountable design context.

IMAGE has the following features:

- *Design Activities:* A designer partitions a problem into activities for solution as well as to provide comprehensive information management;
- *Available Assets:* A variety of design resources are provided to aid in the generation of design knowledge. Resources range from object-oriented databases to CAD packages;
- *Agent Collaboration:* A generic toolkit allows resources to be incorporated into a design infrastructure with minimal effort by the end user. Notice that the incorporation of a model within the toolkit allows for knowledge to be generated in context allowing a designer to interrogate knowledge for the who, what, where, when, and how the information was created.
- *Computing Architecture:* Components that are required for objects to operate in a distributed, homogeneous computing environment are included in an underlying infrastructure.

The characteristics of each component of the architecture will be outlined in the following sections. The implementation of the Design Activities category will be discussed in greater detail since the incorporation of these activities has been lacking in previous computer architecture developments and publications.

Required Functionality	IMAGE Features				
	Design Activities		Available Assets	Agent Collaboration	Computing Architecture
	DSPT Palette	Design Specification Editor			
Partitioning Process	√				√
Solution of Sub-Problems	√	√	√	√	√
Information Model	√	√		√	√
Information in Context		√	√	√	√
Design Efficacy	√	√	√	√	√
Distributed Design Activities	√	√	√	√	√

Table 1. Matrix of Required Functionality vs. IMAGE Components

A matrix that outlines the relationship between the four features of IMAGE and the functionality they support is shown in Table 1. The shaded table elements are topics that are emphasized in this paper.

Throughout this paper, examples will be used to highlight the IMAGE computing infrastructure. IMAGE is built around the Tk/tcl interpretive windowing system with extensions added for drag & drop (blt), object-oriented definitions (itcl), natural language processing (marpa), and message-passing (PVM).*[29] *All of these extensions are publicly available except for the PVM extension which was developed by the first author.*

Design Activities

IMAGE is a computing infrastructure that assists a designer in performing Design Activities. A corresponding architecture is being developed that models the Design Activities required to solve a design problem. This architecture is shown in Figure 2. Based on techniques for modeling a designer's activities along a design timeline, the architecture has three fundamental components:

- *System Partitioning and Execution*: Facilities are provided that permit a designer to partition a design problem into manageable sub-problems and for the solution of the resulting sub-problems;
- *Support Problem Definition and Solution:* A consistent scheme has been developed that allows a designer to define and solve sub-problems, which are called Support Problems; and
- *Design Management*: Utilities are provided to a designer that allow for comprehensive, object-oriented data management throughout design processes.

* Parallel Virtual Machine: An inter-process message passing interface.

Utilizing these three components, a designer can manage complex system design problems throughout all design processes. This process architecture has been given the acronym DREAMS (**D**eveloping **R**obust **E**ngineering **A**nalysis **M**odels and **S**pecifications). Each of the three components will be discussed in turn.

System Partitioning through the DSP Technique

The processes involved in system partitioning can be modeled using a DSPT Palette. A DSPT Palette provides a set of base entities that can be used by a designer to model a design timeline. A designer can use legacy design models (prescriptive models) or create new design models (descriptive models) to describe current design processes and then proceed to solve the design problem.[12]

There are two phases in the DSP Technique:

- *The meta-design phase*, whereby a designer partitions a design timeline with the aid of Support Problems; and
- *The actual design phase*, whereby Support Problems are exercised so that knowledge about a design can be generated and decisions can be made.

During the meta-design phase, a designer explicitly models design processes using Support Problems. Several activities that may occur during a generic design process are shown in Figure 3. The activities are depicted in a manner consistent with a traditional design timeline. A designer can use the DSP Technique to partition this design timeline into Support Problems. The timeline from Figure 3 can be partitioned as shown in Figure 4. Design Phases† and Events‡ are identified by a designer during the partitioning process. A designer also identifies information that is to be accumulated for making decisions. For the generic case represented in Figure 4, a designer has identified four design Phases and three design Events that occur during the *Conceptual Design* Phase.

† Design Phases are distinct stages of a design's development whose boundaries are usually characterized by changes in design fidelity.

‡ A Design Event is a coincidence of design activties that are to occur so that the solution of a problem or sub-problem can be achieved.

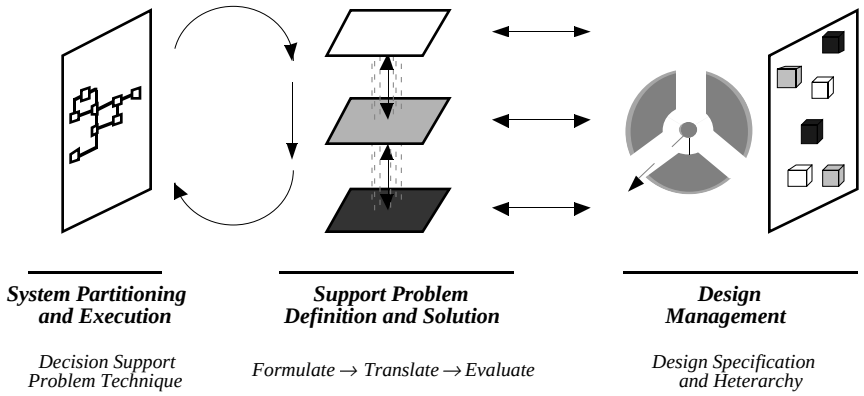


Figure 2. DREAMS Architecture for Supporting a Designer's Activities

Figure 3. A Traditional Design Timeline

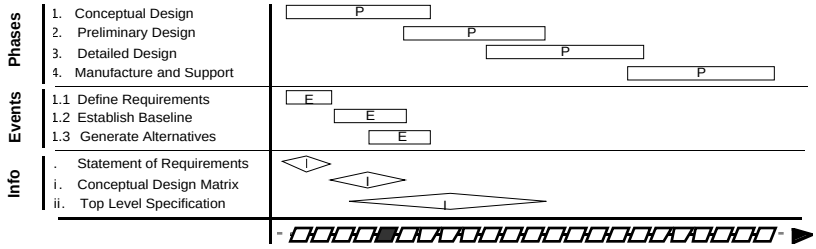


Figure 4. A Partitioned Timeline Corresponding to Figure 4

Support Problems may be formulated that represent design activities that are to occur during design processes. A partial representation of the Support Problems that comprise the shaded timeline partition in Figure 4 are shown in Figure 5. This schematic depicts the transformation of the *Statement of Requirements* into *Top Level Specifications* through two design Phases and three design Events. Notice that the schematic is multi-level. The lower sequence represents the activities that comprise the *Conceptual Design* Phase. Design Events may occur simultaneously, thus requiring the use of multidisciplinary and concurrent analysis techniques. Transmission Entities, shown as Information blocks with a triangular border, encapsulate the respective information requirements that are shared among Support Problems.

Through Phase and Event Support Problems, the first or highest level of design activities that may occur can be represented, as shown in Figure 5. As a design process is decomposed further, other Support Problems may be encountered. The different types of Support Problems and their corresponding function are outlined in Table 2.

Table 2. Support Problems

Support Problem	Icon	Function
Phase	P	Stages of development
Event	E	Coincidence of design activities
Task	T	Activity to be accomplished
Decision	?	Evaluation of a design based on its content
System	◎	Actual design components which may be real or abstract

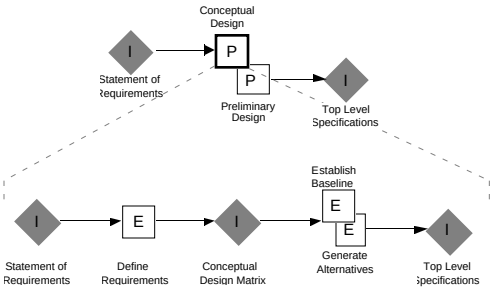


Figure 5. Support Problem Network Corresponding to Figure 4 (See Table 2)

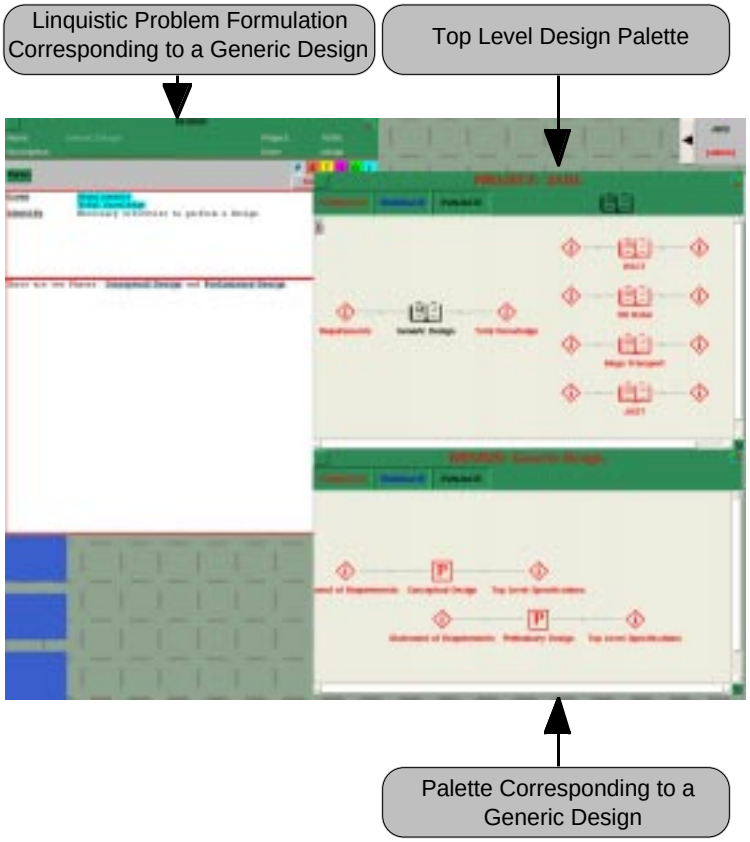


Figure 6. Screen Capture of the DSPT Palette as Displayed in the IMAGE Environment

In IMAGE, the DSPT Palette implementation provides an interface for meta-design activities (partitioning a design timeline) and actual designing (solving Support Problems). The implementation is adapted from earlier work done on a **D**esign **G**uidance **S**ystem.[15] An implementation of the DSPT Palette used in IMAGE is shown in Figure 6. The DSPT Palette supports the following functions:

- A top-level Design Palette for multiple problem management;
- Additional Palettes that allow for problem decomposition;
- An editor for linguistic problem formulation;
- A natural language processor based on grammars for linguistic parsing (not shown);
- A dictionary for English language and design catalogues (not shown);
- Information recomposition from lower Palettes to higher Palettes (not shown); and
- Multiple-user problem definition.

These functions allow a designer or an integrated team to decompose and solve a design problem throughout a design timeline. After defining design processes in meta-design, a designer can use the DSP Technique to generate knowledge used for decision-making by solving the resulting Support Problems. For example, each icon in Figure 5 corresponds to an associated Support Problem. A standard technique exists for describing and solving Support Problems and is outlined in the next section.

Support Problem Definition and Solution

Within the DSP Technique, Support Problems are exercised by a designer to produce knowledge about a design so that decisions can be made based on that knowledge. Support Problems provide standard models for transforming design information into knowledge. There are three steps required in defining and solving Support Problems:

- *Formulation*: The structuring of the problem statement into specific Support Problem models;
- *Translation*: Associating processes, that govern the generation of information into knowledge, with the Support Problems; and
- *Evaluation*: Producing design knowledge through the solution of the Support Problems.

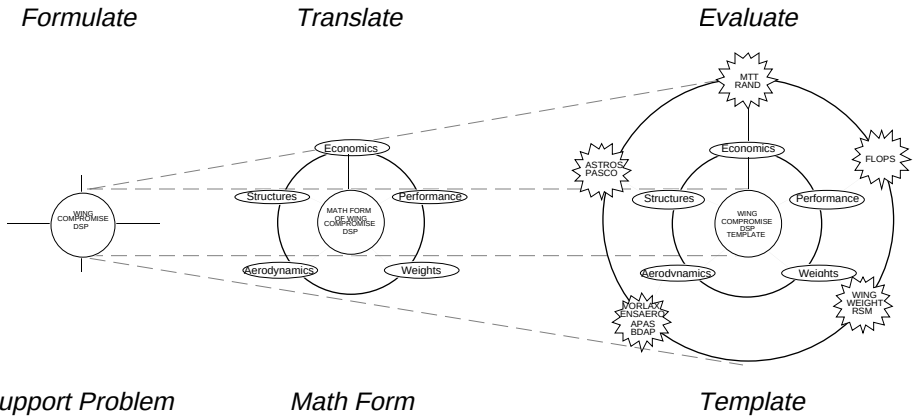


Figure 7. Multidisciplinary Wing Integration Compromise Decision Support Problem

These three steps are illustrated with a multidisciplinary wing integration Compromise Decision Support Problem* in Figure 7. This problem examines the flutter and buckling constraints imposed on a High Speed Civil Transport.[30]

Formulation - Support Problem

Support Problems are defined when a design process is partitioned in meta-design. Support Problems have a defined structure given by keywords. The Compromise DSP has the following form:[15]

Given:	Feasible design and aspiration spaces
Find:	Values of variables
Satisfy:	Systems constraints, bounds, and goals
Minimize:	Deviation between "what I want" and "what I can have"

Support Problems are formulated as linguistic statements, a form natural to a designer and, hopefully, unambiguous in meaning. As Support Problems are formulated, they are embodied by Forms and Functions from a Design Specification.

Translation - Math Form

Once a Support Problem has been formulated, the problem is then translated into an equivalent Math Form. The Math Form provides the process connectivity between Forms and Functions†. At this point, a Form-Function-Model triplet is selected. This triplet is a Process element. For instance, the functions `Lift` and `Drag` are associated with the form `Wing` through the relations:

$$Lift = C_l q S \quad (1)$$

$$Drag = C_d q S \quad (2)$$

where C_l and C_d are the respective lift and drag coefficients, q is the dynamic pressure, and S is the wing area. As the Math Form becomes more complex, equations are typically grouped into engineering models. In turn, models are often grouped into disciplines. Some of the traditional aerospace disciplines that are present in the multidisciplinary wing integration problem are shown in Figure 7. Notice that inter-disciplinary models do exist, as in the case of aeroelasticity, and must be accounted for. Looking again at Figure 7, the problem definition can be visualized as an expanding cone. The Compromise DSP forms the frustum of the cone and the cone expands as the problem is translated into the Math Form.

Evaluation - Template

Finally, the Math Form of the Support Problem can be solved. The Support Problem solution consists of three steps: pairing the Math Form with a suitable Agent, structuring a solution network, and solving the Problem. Agents are used by a designer to generate design information from the expressions found in the Math Form of a Support Problem‡ . As shown in Figure 7, Agents are typically engineering analysis codes. Other Agents include expert systems, hyper-media sources, virtual reality, and the human designer. After the Math Form and Agents have been collected, the new form of the Support Problem is called the Support Problem Template. The SP Template forms the base of the expanding cone. The notion of the SP Template is important for modeling design processes. SP Templates represent a bridge between modeling design processes and systematically employing Available Assets to generate the information required for those processes.

* A compromise decision is a process of determining the "right" values of design variables such that a system is feasible. In contrast, a selection decision is a process of making a choice between a number of possibilities.

† Forms, Functions, and Models are an inherent part of the information model used in IMAGE and are discussed in the Design Management section to follow. These entities are summarized in Table 3.

‡ Agents will be discussed in more detail in the Available Assets and Agent Collaboration sections to follow.

Continuing, a solution network must be generated for Agent collaboration and distribution. Figure 8 shows a solution network corresponding to the multidisciplinary wing integration problem shown in Figure 7.[30] Finally, the Support Problem must be solved. **D**ecision **S**upport **I**n the **D**esign of **E**ngineering **S**ystems (DSIDES) is a suite of tools used to solve Support Problems.[11] Tools in DSIDES are used to solve Selection DSPs (SELECT) and multi-level, multi-goal Compromise DSPs (ALP).

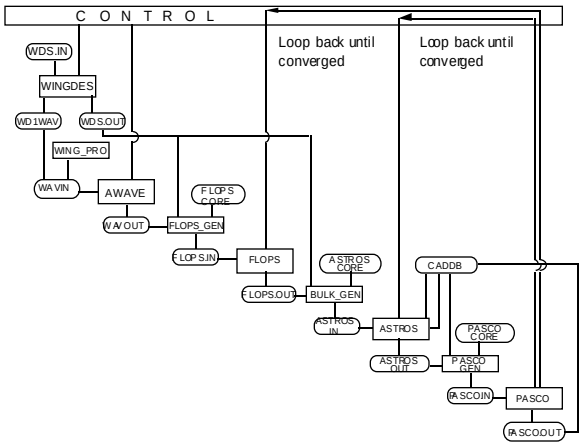
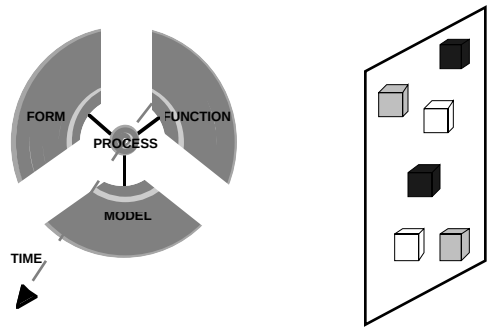


Figure 8. Partial Solution Network for the Multidisciplinary Wing Integration Problem

Design Management

Support Problems are explicitly tied to an information model since they govern the transformation of information into knowledge. The DREAMS architecture provides comprehensive information management for this purpose. As represented by the icons in Figure 9, information can be either structured (an information hierarchy) or unstructured (an information heterarchy).



Information Hierarchy *Information Heterarchy*

Figure 9. Design Information

The information heterarchy refers to unstructured information, or loose information.[15] During design processes, some information will be structured from the heterarchy into the information hierarchy. An example of unstructured information would include local program variables and process id's.

Stephens has shown that design hierarchies can be formed based on Form-Function-Process-Model/ Temporal design sub-spaces.[18] These spaces span nine-dimensions as given in Table 3. These sub-spaces encompass all of the elements that are to occur during a design process, including both the physical artifact and the process of designing itself. The four sub-spaces, Form, Function, Process, and Model, are populated by multiply-connected object hierarchies, giving rise to two dimensions for each category. For instance, a Form sub-space may be populated by the objects and organized into the hierarchy shown in Table 4.

Table 3. Design Hierarchy Entities

Category	Function	Dimensions
Form	A mechanism by which a design can perform an activity	2
Function	An assigned activity a design is to perform	2
Process	The means by which a function is performed by a form	2
Model	An idealization of a process	2
Time	Either real or event-based	1
		$\Sigma = 9$

Table 4. Form Objects

Objects	Hierarchy
Aircraft	Aircraft
Lifting Surface	|-----Centerbody
Vertical Stabilizer	|-----Lifting Surface
Propulsion Unit	|-----Propulsion Unit
Centerbody	|-----Vertical Stabilizer

Looking again at Figure 9, Process elements are formed by a Form-Function-Model triplet. Later, it will be shown that Process elements are implemented as Agents. Because Process elements explicitly contain Models, these Agents can be used by a designer to produce design information in context. Therefore, a designer may interrogate design information to determine who produced it, when it was created, what was used to produce it, etc. Thus, accountable design information may be obtained through the use of Agents.

The Design Specification Editor is a tool for comprehensive information management and implements the notions of both information heterarchies and hierarchies. The Editor is based on an earlier information system called DEFINE implemented as part of a **L**aboratory **E**nvironment for the **G**eneration, **E**valuation, and **N**avigation of **D**esign (LEGEND).[18] The Editor is shown in Figure 10. The Editor includes:

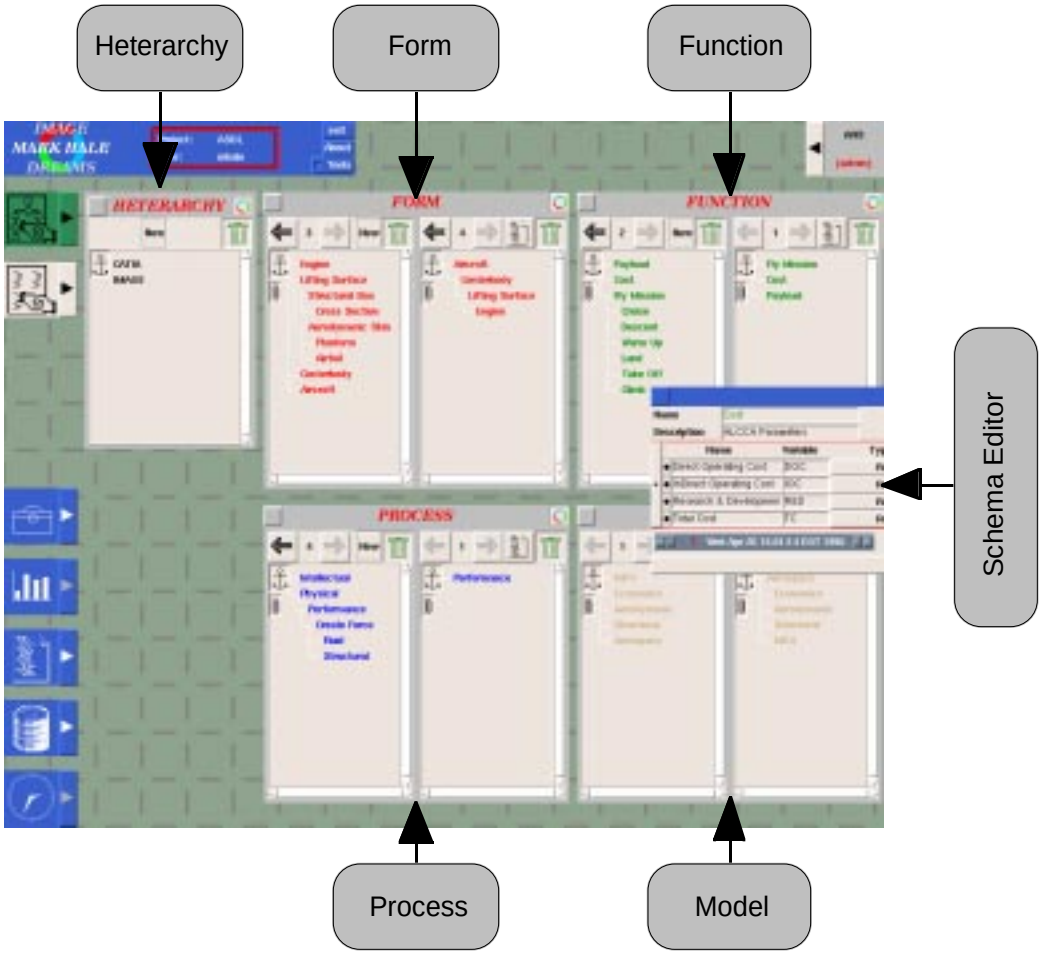


Figure 10. Screen Capture of the Design Specification Editor as Displayed in the IMAGE Environment

- Generation of Heterarchy, Form, Function, Process, and Model objects;
- Generation of Form, Function, Process, and Model hierarchies;
- Heterarchical to hierarchical object migration;
- Varying levels of fidelity;
- Schema development and evolution;
- Instance accumulation provided in context (not shown);
- Object sharing among projects and team members (not shown); and
- An interpretive object-oriented database management system including inheritance, persistence, memory management, and object condensation (not shown).

These capabilities allow for comprehensive design information management based on a common product data model.

As a design progresses, the Information model must support increasing levels of fidelity consistent with increasing completeness of a design. Table 5 shows a `Lifting Surface` Form object described to a tertiary level of fidelity from a structural standpoint. Each object encapsulates one or more schemas*, each potentially having different degrees of accuracy. Table 6 illustrates three schemas that may be used to represent a `Lifting Surface` Form object in conceptual design. The ability to support increasing levels of accuracy within the information model is called schema evolution.

Table 5. Increasing Fidelity for a `Lifting Surface`

Lifting Surface
|-----Inboard Wing Box
|-----Spars
|-----Ribs
|-----Outboard Wing Box
|-----Spars
|-----Ribs

* A collection of attributes and their instances.

Table 6. Three Possible Schemas for Describing a `Lifting Surface`

Schema 1	Schema 2	Schema 3
Root Chord	Aspect Ratio	(X1,Y1)
Span	Span	(X2,Y2)
Taper Ratio	Taper Ratio	(X3,Y3)
Root t/c	Root t/c	(X4,Y4)
Tip t/c	Tip t/c	(X5,Y5)
Sweep	Sweep	Root t/c
Dihedral	Dihedral	Tip t/c
Twist	Twist	Dihedral
NACA Profile	NACA Profile	Twist
		NACA Profile

Available Assets

Available Assets are the second feature of the IMAGE infrastructure shown in Figure 1. These Assets, or resources, are the entities that are inevitably responsible for carrying out design methods. The Assets can be categorized as databases, visualization tools, optimization routines, geometric modelers, and algorithmic and heuristic simulation. The most familiar form of a resource is the computer program. A designer directs computer programs to calculate some desired information based on pre-determined algorithmic procedures. These programs may be combined in automated analysis modules that may incorporate heuristic controllers. Some examples of computer programs used by the aerospace industry include: ASTROS (a structural optimization code), FLOPS (an aircraft convergence code), ACSYNT (an aircraft convergence code), CONMIN (an optimization package), CATIA™ (a three-dimensional geometric modeling, simulation and analysis package), and ORACLE™ (a relational database).

Traditional computer-based design systems utilize resources that primarily operate in the conceptual stage of design. The design resources used in these systems are mostly non-proprietary codes, as illustrated in Figure 11. However, computing environments must also incorporate proprietary resources that often predominate later during a product's design, as also seen in Figure 11. Proprietary resources are generally stand-alone in nature, with limited communications capabilities, and preserve software rights through a number of advanced computing techniques. Together, they present a formidable challenge to implementation of integrated design environments. IMAGE incorporates newly developed technologies that accommodate proprietary resources.

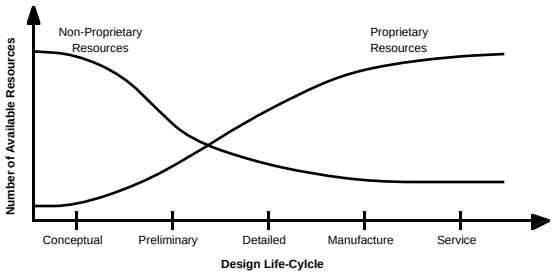


Figure 11. Proprietary Resources

There are a number of other Available Assets that are often overlooked. The first is the design expert. A designer plays an important role in providing expert knowledge, some of which may be captured in knowledge-based systems. Legacy design processes are an Asset that can be incorporated through the prescriptive capabilities of the DSPT Palette. Finally, on-line services, such as World Wide Web (WWW) documents, can be utilized for information dissemination.

Agent Collaboration

Agents are one of the key enabling technologies that bind IMAGE together, as shown in Figure 1. Agents allow for the meaningful creation of design knowledge by Available Assets (resources) during Design Activities. As discussed earlier, the information model used in the IMAGE implementation incorporates the use of Process elements. The instantiation of these elements as Agents results in knowledge generation in context.

The creation of an Agent requires three components: a Resource, a Model, and a Wrap. Resources are the Available Assets discussed in the previous section. Models have two components: the Process Model and the Implementation Model. The Process Model is the model incorporated into the process element. The model may be physical or intellectual. Models are typically based on mathematical formulations, engineering principles, or geometrical constructions.

The Process Model has typically been discarded or included only in external reference documents The use of Agents allows for Process Models to be explicitly defined. For example, a solids construction model used to represent complex solids in CATIA™ is shown in Figure 12. In words, the geometric process model describing the volume transformation would be:

> *In a volume transformation, an object is represented by an approximate solid computed directly from the exact volume. A volume is constructed from faces which, in turn, are defined by the edges that enclose simple or multiply connected regions of planar or complex surfaces.*

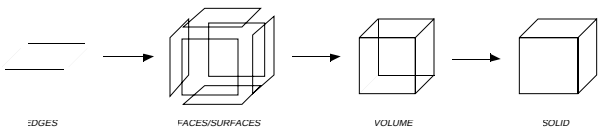


Figure 12. CATIA Solid Representation

The Implementation Model, the second model component, captures the execution characteristics of the resource. Some of the items that are contained in the implementation model include: variable definition, file descriptions, units, execution characteristics, and platform dependencies.

A Wrap enables Agents to work in a collaborative environment. A Wrap is responsible for publishing models so that designers may employ the services of resource contained within, communicating information among resources while conforming to protocols and data exchange standards, and negotiating its services with other agents. In all a Wrap has six components: a Communications Interface, a Protocol Filter, a Model Interpreter, a Resource Interpreter, the (Graphical) User Interface, and a Low Level Compliance layer.

A generic scheme has been proposed that allows for Agents to be developed from existing available assets or new Agents to be developed that incorporate the functionality of new design environments and computing characteristics. References [10, 19, 20] summarize generic Agent implementation schemes. Examples of Agents that incorporate proprietary resources are also highlighted in these references.

Computing Architecture

Design Activities utilize Available Assets in an accountable fashion through the collaborative use of Agents. The Computing Architecture allows for this process to occur using existing computer facilities, see Figure 1. The underlying architecture is a subset of what has come to be known as the National Information Infrastructure. The NII is simply a conglomeration of users and services on the Internet. Users range from end-users performing design functions to developers providing simulation services. Services range from information dissemination, such as on the WWW, to underlying transport mechanisms such as TCP/IP*. In addition, the NII also provides direction for new computing technologies such as fine-grained and parallel computing.

The wrap component of the Agent allows for collaborative efforts to occur through an intimate interface with the NII. Proper integration requires that the following services be standardized and made available: communications support, protocols, data representations, and ontologies. As a testbed, the following services are implemented in IMAGE:

- Transparent X-based communications and message passing using **P**arallel **V**irtual **M**achine (PVM);[23]
- An object-oriented data model that includes multi-fidelity, multi-accuracy, persistence, ownership, and accumulation schemes;
- A dynamic protocol;
- An ontology†.

The NII incorporates rapidly expanding and evolving computing facilities. IMAGE has been designed so that new technologies may be incorporated and tested within the architecture without re-configuration.

Summary

A computing infrastructure called IMAGE has been outlined. The infrastructure consists of four components: design activities, available assets, agent collaboration, and a computing architecture. These four categories were discussed in detail and their respective importance to overall design processes was noted. A formal architecture called DREAMS was presented and provides consistent support for designer activities throughout a design timeline. IMAGE integrates available assets through state-of-the-art computing technologies. A generic agent scheme has been designed that allows for resources to be integrated easily into a design environment and allows for the generation of design knowledge by these resources in context. Using IMAGE, a designer will eventually be able to produce better, more robust designs while expending fewer resources and, therefore, can perform Integrated Product and Process Development.

Acknowledgments

Funding for this paper is provided by the NASA Graduate Student Researchers Program (NGT-51250) under the direction of NASA Langley's High Performance Computing and Communications Program. Software and hardware support is provided by the CAE/CAD Laboratory at the Georgia Institute of Technology.

References

[1] "ACSYNT Overview and Installation Manual," ACSYNT Institute, Virginia Polytechnic Institute and State University, May 1992.

[2] Cutkosky, M.R., *et al.*, "PACT: An Experiment in Integrating Concurrent Engineering Systems," IEEE Computer, vol. 26, pp. 28-37, January, 1993.

[3] Dovi, A.R., G.A. Wrenn, J.-F.M. Barthelemy, P.G. Coen and L.E. Hall, "Multidisciplinary Design Integration System for a Supersonic Transport Aircraft," Fourth AIAA / USAF / NASA / OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September 21-23, 1992. AIAA-92-4841.

[4] Hughes, D., "Generic Command Center Speeds Systems Design," Aviation Week & Space Technology, pp. 52-53, March 8, 1993.

* Transmission Control Protocol / Internet Protocol

† An ontology is a specification of discourse among agents in the form of definitions of shared vocabulary[2].

[5] Jones, K.H., D.P. Randall and C.K. Cronin, "Information Management for a Large Multidisciplinary Project," Fourth AIAA / USAF / NASA / OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September 21-23, 1992. AIAA-92-4720.

[6] Gage, P. and I. Kroo, "Development of the Quasi-Procedural Method for Use in Aircraft Configuration Optimization," Fourth AIAA / USAF / NASA / OAI Symposium on Multidisciplinary Analysis and Optimization, Cleveland, OH, September 21-23, 1992. AIAA-92-4693.

[7] Kroo, I. and M. Takai, "A Quasi-Procedural, Knowledge-Based System for Aircraft Design," AIAA / AHS / ASEE Aircraft Design, Systems and Operations Meeting, Atlanta, GA, September 7-9, 1988. AIAA-88-4428.

[8] McCullers, L.A., "FLight OPtimization System, User's Guide, Version 5.41," NASA Langley Research Center, December, 1993.

[9] Townsend, J.C., R.P. Weston and T.M. Eidson, "An Overview of the Framework for Interdisciplinary Design Optimization (FIDO) Project," NASA Langley Research Center, July, 1994.

[10] Hale, M.A., "A Computing Infrastructure that Facilitates Integrated Product and Process Development from a Decision-Based Perspective," Thesis Proposal, Georgia Institute of Technology, School of Aerospace Engineering, January, 1995.

[11] F. Mistree, O.F. Hughes, and B.A. Bras, *The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm*, Ed. Kamat, M.P., Structural Optimization: Status and Promise, Washington, DC, (pp. 247-286), AIAA.

[12] Bras, B.A. and F. Mistree, "Designing Design Processes in Decision-Based Concurrent Engineering," SAE Transactions Journal of Materials & Manufacturing, vol. 100, no. pp. 451-458, Warrendale, PA, SAE International, 1991.

[13] F. Mistree, W.F. Smith, and B.A. Bras, *A Decision-Based Approach to Concurrent Engineering*, Ed. Paresai, H.R. and W. Sullivan, Handbook of Concurrent Engineering, Chapman & Hall, New York, 1993. (pp. 127-158).

[14] Muster, D. and F. Mistree, "The Decision Support Problem Technique in Engineering Design," The International Journal of Applied Engineering Education, vol. 4, no. 1, pp. 22-33, 1988.

[15] B.A. Bras, W.F. Smith, and F. Mistree, *The Development of a Design Guidance System for the Early Stages of Design*, Ed. Oortmerssen, G.V., CFD and CAD in Ship Design, Elsevier Science Publishers B.V., Wageningen, The Netherlands, (pp. 221-231).

[16] Mistree, F., W.F. Smith, B.A. Bras, J.K. Allen and D. Muster, "Decision-Based Design: A Contemporary Paradigm for Ship Design," Transactions, Society of Naval Architects and Marine Engineers, vol. 98, pp. 565-597, 1990.

[17] Pahl, G. and W. Beitz, Engineering Design: A Systematic Approach. Berlin, Germany: Springer-Verlag. 1992.

[18] Stephens, E., "LEGEND: Laboratory Environment for the Generation, Evaluation, and Navigation of Design," Doctoral Dissertation, Georgia Institute of Technology, School of Aerospace Engineering, September 1993.

[19] Hale, M.A. and J.I. Craig, "Preliminary Development of Agent Technologies for a Design Integration Framework," AIAA / NASA / USAF / ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 7-9, 1994. AIAA-94-4297.

[20] Hale, M.A. and J.I. Craig, "Use of Agents to Implement an Integrated Computing Environment," Computing in Aerospace 10, AIAA, San Antonio, TX, March 28-30, 1995. AIAA-95-1001.

[21] Chapman, B., P. Mehrotra, J.V. Rosendale and H. Zima, "A Software Architecture for Multidisciplinary Applications: Integrating Task and Data Parallelism," Institute for Computer Applications in Science and Engineering, March 1994.

[22] "The Common Object Request Broker: Architecture and Specification," December, 1991.

[23] "Parallel Virtual Machine User's Manual," Version 2.4, June, 1993.

[24] Finin, T., *et al.*, "Specification of the KQML Agent-Communication Language," The DARPA Knowledge Sharing Initiative External Interfaces Working Group, February, 1994.

[25] Frank, G.A., J.B. Clary and B.L. Dove, "Design Automation for Concurrent Engineering," Center for Digital Systems Research, 1989.

[26] Genesereth, M.R. and N.P. Singh, "A Knowledge Sharing Approach to Software Interoperation," January, 1994.

[27] Hale, M.A., "IMAGE: An Intelligent Multidisciplinary Aircraft Generation Environment," Masters Program Special Project, Georgia Institute of Technology, School of Aerospace Engineering, September, 1992.

[28] "Definiton of Requirements for and Aeronautics Affordable Systems Optimization Process," Proposal to NASA Langley Research Center, Madic Team #2, January, 1995.

[29] Ousterholt, J.K., An Introduction to Tcl and Tk. Reading, MA: Addison-Wesly Publishing Company, Inc. 1993.

[30] Röhl, P., D.P. Schrage and D.N. Mavris, "A Multilevel Wing Design Procedure Centered on the ASTROS Structural Optimization System," AIAA / NASA / USAF / ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 7-9, 1994. AIAA-94-4411.