N89-25159

DEVELOPMENT OF A MICRO-COMPUTER BASED INTEGRATED DESIGN SYSTEM FOR HIGH ALTITUDE LONG ENDURANCE AIRCRAFT*

by
David W. Hall** & J. Edward Rogan***

ABSTRACT

In recent years, increasing attention has been given in the aerospace industry to integration of aircraft design disciplines. This approach has been applied theoretically to sailplane design and to solar powered high altitude long endurance (HALE) aircraft design. More recently, it has been applied to the design of microwave powered aircraft. These studies describe attempts to arrive at integrated designs of one class of aircraft using then-existing state-of-the-art computer capabilities. No attempt was made in any of these cases, though, to use new programming techniques derived from Artificial Intelligence (AI) research to develop more flexible systems for the conceptual design of HALE aircraft.

The purpose of this study was to investigate the feasibility of developing a general parametric sizing capability for micro-computers using integrated design methodology implementing an existing HALE methodology as a test case. The methodology described here incorporates some detailed calculations, many qualitative rules-of-thumb and constraints which are not easily quantified except by the accumulation of design experience. In this regard, the resultant software which will be developed in future efforts will be a knowledge-based system for the conceptual design of HALE aircraft.

* This work was sponsored as a Phase I SBIR by the NASA/Ames Research Center, Advanced Plans & Programs Office
** David Hall Consulting, AIAA member
*** The Georgia Institute of Technology, School of Aerospace Engineering, AIAA member

INTRODUCTION

Advances in computing technology have made it possible to create a new kind of design tool. Current generation CAD/CAE/CAM systems are based on an idea of the computer as a "number cruncher"---computations are thought of as manipulations of numbers. This approach has led to an emphasis on geometry representation and numerical techniques in engineering computing. Viewed in this way, computers *aid* design, but are essentially external to the design process.

An alternative view of the computer has been around since Alan Turing's epoch-making work in the 1930's: the view of computing as manipulation of *symbols,* (which might include numbers among other things). The great beauty of Turing's work is in the intimate connection between the scientific definition of what a computer is (it's called a Turing Machine) and aspects of mathematics as a *language* that are among the most important scientific discoveries of this century (Kurt Gödel's completeness and incompleteness theorems).

What does any of this have to do with design or aircraft? After all, designers are notoriously,and admittedly, innocent of mathematics[1]. However, designers, like everyone else, use a language of symbols to solve problems. The idea of a symbolic **programming language for (aerospace) design,** clearly delineated and with most of the main pieces present, is the primary innovation of the work described in this paper.

The application, if not the development, of a programming language for design must be graphical since designers think graphically. A symbolic "design programming language" must provide the designer with the means to describe design concepts and manipulate them in a convenient, but disciplined, manner. Certainly, the language must provide the capability to describe (1) the air vehicle itself ("What it is"), (2) the *functional decomposition* ("What it does") and (3) various approximate theories, analyses and models which collectively describe "How it works".

What does it mean to execute or "run" (the word "interpret", in its everyday sense, rather than in the specialized computing sense, is probably most accurate) computer programs written in such a language? Interpretation of a computer program involves defining contexts (environments) in which names (variables, attributes) are *bound* (assigned, set) to values (which may be numbers or other symbols)[2]. Procedures, in which the named variables appear as parameters, are then evaluated in the environments. On the other hand, execution of an air vehicle design process involves exploring how alternative design ideas fit together by preparing engineering drawings and performing analyses. The process of making an engineering drawing is a decision-making process that results not only in the design specification, but in the designer's *understanding* of how the final concept works and why design decisions were made the way they were. It is this understanding that allows the designer to conscientiously "sign off" that the design is correct in the designer's professional opinion.

The design programming language to be described in this report allows designers to use computers to *do* design, rather than to *aid* design. This means that interpretation of computer programs written in the design programming language must support the design decision-making process. Designers must be able to use computer programs written in the "programming language for design" to gain a technical understanding of the way the aircraft will work and might fail, and to develop and communicate their rationales for design decisions.

Background

A programming language for design is a "knowledge-based system" in the sense that the basic architecture (knowledge-base, inference engine, problem state or context) and the techniques used for structuring and interpreting computer programs (inheritance, constraint propagation, objects/modularity/state, data and procedure abstraction, and metalinguistic abstraction) are basically the same. However, a "programming language for design" is far from being an "expert" system. The idea of an expert system is that once knowledge has been extracted from the expert (or experts), the computer program is able to reach conclusions that can be logically inferred from the rule base with only minimal human intervention. The implication is that novices would then be able to tackle many of the problems that can nowadays only be solved by experts. In contrast, the programming language for design is a tool specifically for use *by* expert aircraft designers. The intention is to accomplish the kinds of productivity gains for aircraft designers that word processing provides for authors and secretaries, or that symbolic mathematics programs (e.g., MACSYMA or SMP) provide for mathematicians and other scientists.

An example is a *decision-oriented* (rather than drafting oriented) user interface for preparing aircraft three-views. An important insight resulting from the research is that *a fundamentally wrong turn is being taken with the introduction of three-dimensional geometry in the very early phases of conceptual design in that the* ***decision-support*** *aspects of the three-view drawing have been overlooked in favor of the* ***product definition*** *advantages of three-dimensional surfaced geometric models.* The three-view drawing is a classic example of the suppression of non-essential detail (required to implement a 3-D model) in order to emphasize critical design aspects: placement of landing gear, pilot visibility, wing planform shape, static stability, propulsion integration, etc. The three-view should properly be used to make these design decisions. Three-dimensional surface definition should then be developed by *interpreting* the three-view drawing. Procedures for interpreting three-views would be implemented using the computer programming language for design.

Successful implementation of the integrated design system for micro-computers will accelerate the use of computers to *do* design, increasing product quality and reducing the length of the development cycle by allowing designers to look at a broader range of design concepts. Additional benefits obtain if the use of a computer programming language for design is carried forward into preliminary design, detailed design, production, and support phases of the system life cycle. Here, the ability of computers to manage complexity and detail would be invaluable. The idea that a product decision history, represented by a portion of the computer programs for the design, could be carried onboard each "tail number" and used to assess the impact of maintenance actions and for configuration control has considerable appeal for long-life, high-value systems such as aircraft and reusable spacecraft.

The advantages of a computer programming language for design are clear. The question remains, "How can a design computing language be implemented?". The answer to this question begins with an assessment of some alternative approaches to applying advanced computing technology to design, comparing and contrasting them with our technical approach. From this discussion will emerge key issues that must be addressed to establish proof-of-concept for the design

computing language. These issues then provide a backdrop for the technical discussions, results and conclusions which follow.

Alternative Approaches

The design computing language being developed at DHC, (called ***"windFrame"***) is one of several design tools that could justifiably be said to represent the next generation. These alternative approaches are in various stages of maturity, ranging from research projects to small, new-start CAD companies and new directions being taken by established CAD vendors. One of the first projects to break ground in design computing languages was the **Paper Airplane** project at the Massachusetts Institute of Technology [3]. Paper Airplane contributed the idea that constraint propagation could be used in conjunction with a symbolic representation of design "attributes" and relationships among those attributes to free the designer from the tendency of FORTRAN synthesis programs to have "hard-wired" design decision paths. Newton's method was applied in a single variable to solve the constraints in the "backward" direction. Paper Airplane, written in the LISP programming language, was envisioned primarily as a tool for conceptual design.

Rubber Airplane, currently being developed by Mark Kolb at M.I.T., was started as a result of a technical exchange between Dr. Elias and Randy Smith, then (December 1985) at The Lockheed-Georgia Company [4]. Smith had been working on a FORTRAN code for parametric geometry and 3-D surface definition, called GRADE. In his Rubber Airplane work, sponsored by the NASA/Ames Advanced Plans and Programs Office, Kolb continues to apply a constraint propagation technique based on the successful Paper Airplane ideas (Rubber Airplane is written in Common LISP). However, Rubber Airplane provides *components,* which are similar (from a design point of view) to the components defined in GRADE (except that the designer has much more flexibility). Rubber Airplane also provides *design links* between components. Paper Airplane was originally developed on a VAX using Franz Lisp. Rubber Airplane is being implemented on Texas Instruments Explorer workstations, although it also runs on Symbolics LISP Machines.

At about the same time Elias was starting work on Paper Airplane, Larry Rosenfeld was developing a parametric geometry modeller that eventually evolved into the **ICAD** system[5]. The details of how ICAD works are rather closely guarded; however, it can be noted that the system provides an advanced parametric surface definition capability and demand-driven propagation of symbolic and numeric constraints. Constraint propagation does not appear to be as powerful as those in Paper Airplane and Rubber Airplane because it works only in one direction. An often-cited problem with ICAD is that the user/designer interacts with the system primarily by programming in a LISP-like design programming language, the ICAD Language. A feature of ICAD that is important for comparison with ***windFrame*** is the structure of the design concept representation: the design is represented as a hierarchical tree of *parts,* linked by relationships that are not given any special structure. Early versions of ICAD ran primarily on Symbolics LISP Machines, but the product is becoming available on other workstations with fast LISP software.

Dr. Gene Bouchard's work at The Lockheed Aeronautical Systems Company should also be mentioned[6]. Details of the system are proprietary, except that the system is LISP-based, runs on the Symbolics, and provides a valuable graphical programming user interface. Bouchard has focused on providing designers with a quick turnaround trade study capability. Parametric surface definition capability is planned for the system but has not yet been implemented.

Intergraph Corporation made the decision about four years ago to implement their Non-Uniform-Rational-B-Spline (NURBS) package using object-oriented techniques in the C programming language. This has provided them with a significant fallout design programming language

capability, although the system is primarily oriented toward geometry definition, rather than design decision-making[7].

The Cognition system attracted considerable attention when it first appeared, offering an attractive object-oriented drawing interface linked with on-line standard handbooks[8]. Constraint propagation was provided through the solution of relatively large multi-dimensional Newton-Raphson methods. The Cognition system was written in MainSail, a LISP-like object-oriented language.

windFrame was conceived to explore areas of design computing languages that were not being emphasized by any of these products or projects. These areas were

1) Apply optimization as a constraint propagation technique and use decomposition and parameter passing to limit the extent of constraint propagation.

2) Base the organization of the language around Systems Engineering discipline, especially *explicit description of design function* .

3) Focus on aircraft design and develop the system through walkthroughs of a proven aircraft design methodology, working closely with an experienced aircraft designer.

4) Develop a system that would run on a relatively inexpensive micro-computer and plan to place the system in the public domain.

5) Base the development of the language on the user interface.

6) Design the computer programming language so that it could be used throughout the life cycle of an aircraft.

7) Focus on design decision-making.

8) Use approximation techniques to provide interactive explanation and "What if?" analysis as part of the constraint propagation.

Issues that have been addressed to date include:

- Is object-oriented programming the "best" way to implement a design language? If not, what replaces or complements it?

- What "objects" (or their equivalents) are needed to organize the language around Systems Engineering disciplines?

- How can Dr. Sobieski's (NASA/Langley Research Center Interdisciplinary Research Office) decomposition techniques be extended to apply to discrete parameters?

- How can constraint propagation problems be formulated as optimization problems?

- Is there a suitable programming environment to support high productivity user-interface-oriented programming on the Macintosh? Can critical capabilities for the design language be demonstrated in such a programming environment?

- What would one experienced aircraft designer want the user interface to look like?

- If the system were available in the public domain, what kinds of users would be interested?[9]

DESIGN OF THE *windFrame* LANGUAGE

Considerable emphasis has been placed on the design of the *windFrame* language itself. The success of this effort can be judged by the fact that the results seem somewhat obvious in hindsight. However, *details of the structure of the language strongly influence how it will be used.* The difficulty of identifying a good set of basic elements for the language can be seen in the wide disparity of choices of these elements in the alternative next generation design tools discussed above. The technical approach taken in this effort has been to select an initial set of basic elements based on ref. 10, and then to evaluate this choice of elements by a "walkthrough" of an existing design methodology in close association with an experienced aircraft designer. Alternative computer programming techniques for implementing the basic *windFrame* language elements were then investigated.

The *Architecture and Integration Requirements for an ULCE Design Environment* study[10] represents one step in the direction of bridging the gap between systems engineering discipline and traditional airplane design practice[11]. This study took a dramatically different approach to the integration of downstream concerns (such as supportability and producibility) into early design decision-making. Specifically, the study took a fundamental look at the *subtext* of what designers do; that is, the things designers do without thinking about them. This aspect of the study is also highly relevant to the design of the *windFrame* language. From this point of view, the design process is not static, but highly dependent on the technical content of the design concepts and on the nature of the requirements. The ref. 10 study also highlighted the decision-making aspects of design, as opposed to the generation of product definition data. One of the conclusions of this study was that the architecture of Unified Life Cycle Engineering *as a process* must include

1) the generation of design alternatives,

2) planning of the decision-making process based on the technical content of these alternatives, and

3) execution of the design decision-making process (Figure 1).

The intermediate planning step thus involves "design of the design process", and is therefore a "meta-design process". Development of design deliverable data items, such as engineering drawings, was viewed as an output of the decision-making process. *Parametric* definition of concepts to be specified by these deliverable engineering drawings was to be established as part of the original description of the design alternative.

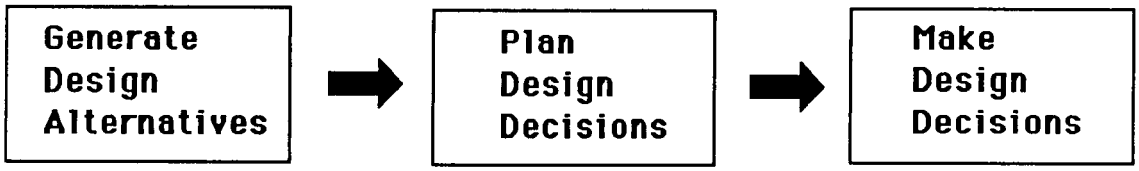

FIGURE 1. ULCE (META) DESIGN PROCESS ARCHITECTURE.

Advances in computing technology were an early motivating force for the USAF Project Forecast II Unified Life Cycle Engineering (ULCE) initiative. The ref. 10 concept was based in part on the use of a parametric design description and specification language for capturing design alternatives in step 1 (Generate Design Alternatives) of the ULCE process. The study was performed by an interdisciplinary team including engineers from preliminary design, supportability, and producibility, as well as design technology. As a result, the need for a systems engineering approach, and specifically for an *explicit description of system function* as a part of the design language was made very clear. Explicit description of function allows an interdisciplinary design team to organize various theories, models, and analytical tools representing different views of how a single system function is performed. This organization is essential for engineers of different technical backgrounds to be able to recognize how these different views interact to make the system work or fail.

The *windFrame* language was specifically developed to address these needs. Basic elements of the *windFrame* language are a precursor to the *system hierarchy* , called the "multiHierarchy", an explicit description of the *functional decomposition* (functionDecomp). The designer defines *theories and models* (theories&Models) at the intersection of the multiHierarchy (which describes alternative system implementations), and the functionDecomp. The theories and models define how the system concept elements associated with specific alternative implementations work to accomplish a given function. Explicit representation of system function allows more than one view of (and associated theories and models for) that function. In casual terms, the functionDecomp provides "hooks on which to hang different ways of looking at the same thing." Similarly, explicit representation of theories and models in the *windFrame* language allows multiple levels of approximation and alternative ways of predicting closely related phenomena, essential in applied aerodynamics (e.g. closed-form approximations, lifting-line, panel, and flowfield methods). The fourth and final basic element of the*windFrame* language is the *design decision* (designDecisions).

One way of thinking about how such a next-generation design environment might be used is that the designer defines the system, performs engineering analyses, and makes design decisions using computer programs that function as integrated "executive", "user interface", and "database" software. Integrating the executive, user interface, and database makes it possible for the software to represent the *state* of the design process, thus such a tool might be viewed as capturing the "design-in-process" (Figure 2).

Evidently, development of a computer program performing all three of these functions is a highly complex software engineering problem, even for a specific domain area such as aircraft design. In this area, our technical approach has been to apply programming techniques such as inheritance; constraint propagation; data and procedure abstraction; modularity, objects, and state; and metalinguistic abstraction[2].

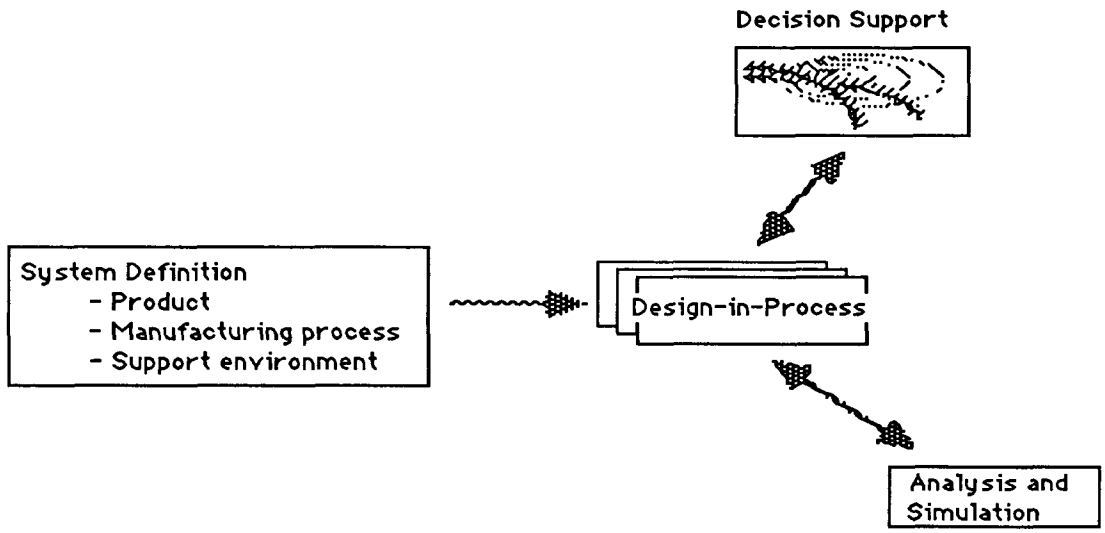

FIGURE 2. DESIGN-IN-PROCESS.

Work began using the MacScheme programming language on the Macintosh SE[12]. The idea was to implement the basic elements of the *windFrame* language using block structure and **make-environment** for modularity. However, this approach was unsatisfactory, primarily because the resulting design computing language was already highly "LISP-like" at the earliest phases of development. Our development philosophy has been, and continues to be, to develop a tool that designers will want to use through intermediate steps that designers would also want to use. Thus far, the LISP programming language has failed to gain wide acceptance among airplane designers, in spite of its obvious advantages. Preliminary experimentation with HyperCard established that (1) the basic *windFrame* elements could be implemented using HyperTalk, (2) instantiation, inheritance, and constraint propagation could be supported, and (3) the resulting design language was *highly* graphical and accessible to aircraft designers. The discussion and results in this report are based on the HyperCard approach.

The *windFrame* language, based on HyperTalk, allows designers to write computer programs using a highly graphical, user-interface oriented programming environment. These programs define design concepts (in terms of "what the concept is", "what it does" and "how it works"). The programs are "object-like" in that they represent prototype design concepts which can be *instantiated*. Based on the technical content of these descriptions, the designer writes more computer programs which manage the user interface for decision support and control instantiation of the design concept "objects". These designDecision programs are also "written" in the highly graphical, user-interface oriented style encouraged and even enforced by HyperCard. Writing the designDecision computer programs will be partially or completely automated once a basic set of decision tools has been established. The design process is then an interactive execution of the design decision programs.

The idea of controlling constraint propagation, instantiation, and inferencing through interactive *designDecision* object-like elements of the *windFrame* language is original with this study and is one of the key innovations of the David Hall Consulting technical approach. The way designDecision elements will work is probably best explained by the following examples. The first example looks at the impact of designDecision-making on the precursor to the system hierarchy (*multiHierarchy*). This example clearly shows how the multiHierarchy evolves into a system hierarchy as a result of executing the design

decisions. It should be emphasized that this process is mapped directly into the *windFrame* design language. The second example illustrates how one type of interactive decision support interface for a designDecision might look. This example also clarifies the role of design- Decisions in managing instantiation of design concept "objects".

A discussion of the impact of design decisions on the *multiHierarchy* must be introduced with some additional background on the basic structure of the *multiHierarchy*. Recall that the *multiHierarcy* is used to define "what the system is". In the *windFrame* langage, the system definition is made up of *conceptElements.* For example, the high altitude long endurance aircraft itself would be a conceptElement in *windFrame*. "*conceptElements*" have *attributes* associated with them. Continuing the high altitude long endurance aircraft example, *attributes* associated with the **aircraft** *conceptElement* would include **takeoff gross weight**, **configuration**, **propulsion**, and so on. In specifying a value for an attribute, the designer can choose from among several *alternatives. Alternatives* for the **configuration** *attribute* of a high altitude long endurance aircraft might include **pusher**, **joined-wing**, **canard**, **three-surface**, and **conventional tailplane** (Figure 3).

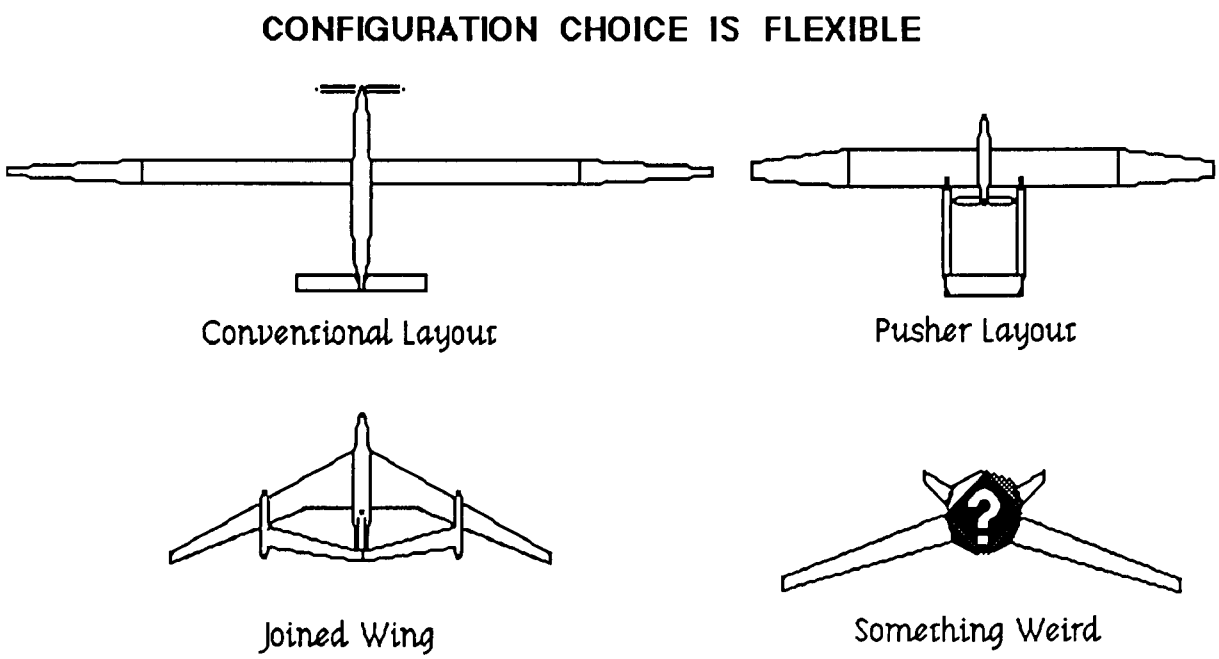

FIGURE 3. ***ALTERNATIVES FOR CONFIGURATION ATTRIBUTE.***

Alternatives for the **takeoff gross weight** *attribute* could be described as "positive dimensioned real numbers". Writing software procedures to handle "positive dimensioned real numbers" (e.g. converting units, finding non-dimensional combinations of parameters, math functions) is completely straightforward, so it makes sense to say that **takeoff gross weight** is completely specified (as an *attribute in windFrame* ---of course no value has been selected for it yet) by saying that it is a "positive dimensioned real number". Not so with the **configuration** *attribute*, which seems to need further description.

How should the *multiHierarchy* description be carried forward for attributes such as **configuration**? The approach taken in *windFrame* is to recognize that the distinctions between *alternatives* for complicated *attributes* (like **configuration**) arise from the fact that different concepts are associated with each alternative. For example, the **canard** *alternative* includes a **canard** *conceptElement* (as does the **three-surface** *alternative*), while the **conventional tailplane**

alternative does not. *windFrame* thus tackles specification of complicated design ideas through a *conceptElement-attribute-alternative-conceptElement* tree structure, the *multiHierarchy* which is shown in Figures 4 through 9.

The *multiHierarchy* evolves into a classical systems engineering tool, the "system hierarchy" [11]. The designer uses the object-like *designDecision* elements of the *windFrame* language to select among alternative design concepts. The *designDecision* elements of *windFrame* manage the partial instantiation of design alternatives needed to apply the engineering analysis procedures contained in the *theories & Models*. The *designDecision* elements of *windFrame* collectively provide the designer with interactive control over constraint propagation, while keeping track of intermediate results and partially defined alternative configurations and accumulating approximations to design goals, requirements, and constraints as surfaces that delimit the design space. This information is used in *windFrame* to provide interactive explanation-oriented constraint propagation.

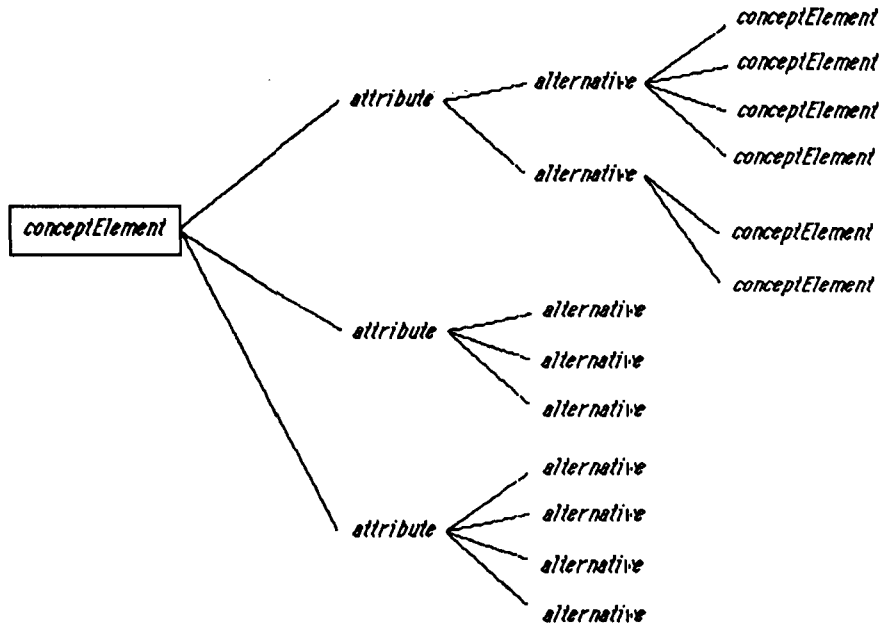

FIGURE 4. *MULTIHIERARCHY* TREE STRUCTURE.

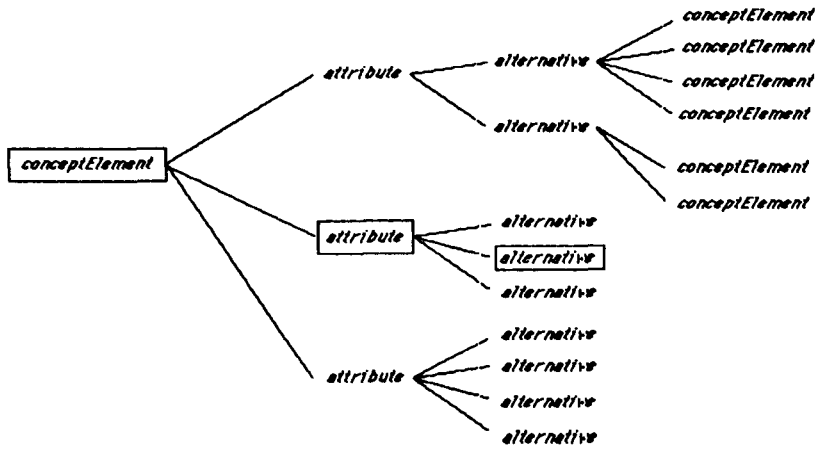

FIGURE 5. *DESIGNDECISIONS* BIND *ATTRIBUTES* TO *ALTERNATIVES*.

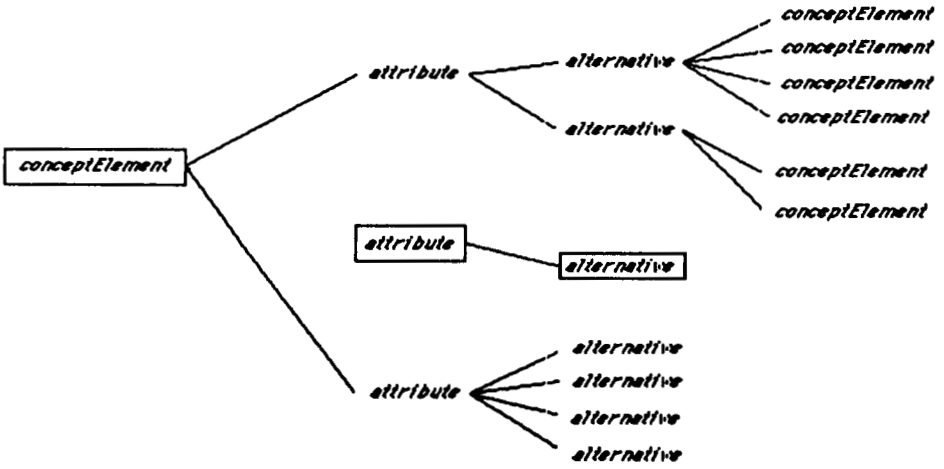

FIGURE 6. "*ATTRIBUTES* THAT ARE COMPLETELY SPECIFIED . . ."

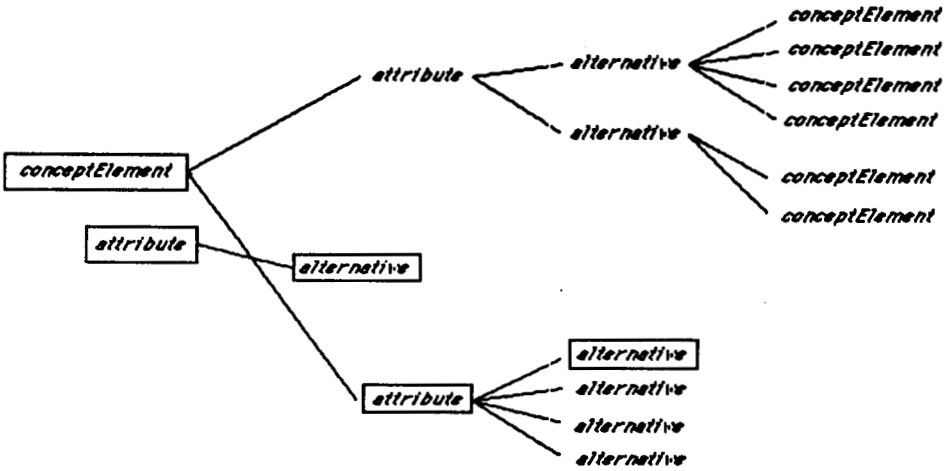

FIGURE 7. ". . . BECOME PART OF THE *CONCEPTELEMENT* SPECIFICATION."

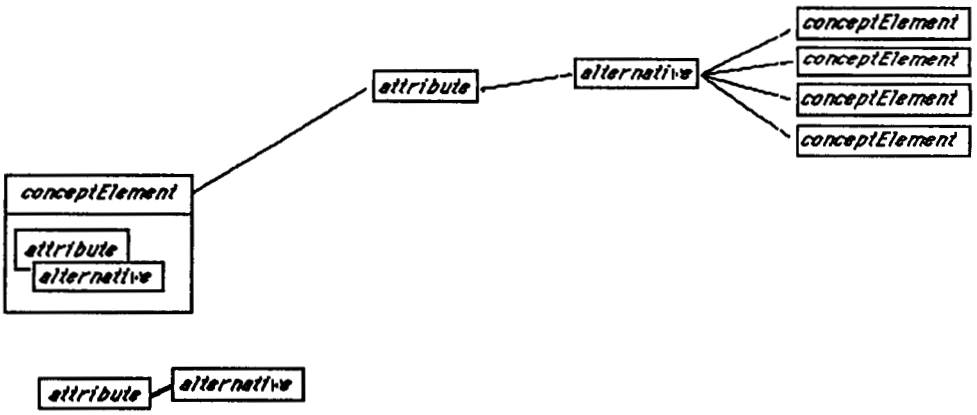

FIGURE 8. "*ATTRIBUTES* THAT FAN OUT TO ANOTHER LEVEL OF *CONCEPTELEMENTS* . . ."

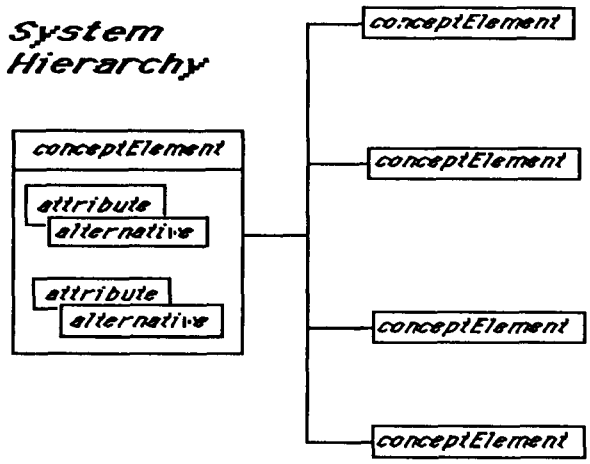

FIGURE 9. " . . . DEFINE THE NEXT LEVEL OF THE SYSTEM HIERARCHY."

One of the results of the design decision-making process supported by *windFrame* is the (aircraft) system specification. Development of the system specification is controlled by the designer through the *designDecisions*. Looked at from the point of view of the *multiHierarchy* as it evolves into the system hierarchy, *designDecisions* bind *attributes* to one of their *alternatives* (Figure 5).

Attributes that are completely specified by this binding (e.g., once a "positive dimensioned real number" has been chosen for **takeoff gross weight**, no additional specification is needed) become part of of the *conceptElement* specification (Figures 6 and 7). Attributes which fan out to another level of *conceptElements* (Figure 8) define the next level of the system hierarchy (Figure 9). An important aspect of the design of the *windFrame* language is that the subtle distinction is clearly made between specifying levels of the system hierarchy through the design decision-making process and specifying levels of design alternatives (by creating a multiHierarchy down to some level of description). This aspect of *windFrame* gives the design language enough expressive power to capture "levels of approximation" which are needed for various *theories&Models*. In fact, the level of approximation of a *theory/Model* is essentially the amount of depth in the multiHierarchy that must be instantiated in order to apply that *theory/Model*.

designDecision elements of the *windFrame* language are used to partially instantiate "object-like" pieces of the design concepts described by the *multiHierarchy* and *theories&Models*. Only those *attributes* of the design concept needed to perform an evaluation of the concept against design goals, criteria, and requirements are bound to *alternative* values. This binding is made in a context (in the sense of interpretation of a computer program discussed above) that is managed by the *designDecision* element. This context is exploratory. Not all the partially instantiated designs in this context will meet requirements. Some of them may be inconsistent or infeasible.

One type of *windFrame* interface that is used by the designer to control this instantiation and evaluation process has the appearance of a set of one-dimensional, two-dimensional, carpet, or other types of plots used for presenting aircraft design and performance information. Another interface, appropriate for discrete, qualitatively different alternatives, is a "trade table" [11]. Both of these classes of decision support tools are familiar to designers. There are limitations to the use of these classical decision support interfaces, however. Their succesful use depends on the designer's ability to set up a design decision process that will rapidly converge on a satisfactory design solution. One of the risks inherent in the integration of new technology is that coming up with a good design process is basically the same problem as coming up with a good design.

The development of the *windFrame* language will make it practical to provide designers with some tools for "designing the design process". The technical approach is to use local optimization and parameter passing to formulate the design process as a network of interrelated decisions. Initial sensitivity and examination of design alternatives are applied to identify sequences of design decisions that have good convergence. As the design space is explored, approximation techniques can be used to build models of complex interactions between design attributes and requirements. The process comes full circle when these approximations are plotted and trade tables are made up using the "classical" decision interfaces. An "explanation" process, using both the "classical" decision interfaces and the design space approximations, can be used by the designer to "design a new design process"---at the same time the designer is designing a new type of aircraft.

The second example illustrating the role of *designDecisions* in *windFrame* is closely linked with these ideas. A prototype decision support interface for exploring the design space would consist of a collection of x-y plots of one attribute against another (Figure 10). For example, the attributes could be **wing loading** and **power loading**. (The values shown in Figure 10 are screen coordinates at the location of the "design instance"). Within allowable limits on *alternative* values for **wing loading** and **power loading**, any point on the plot corresponds to a partial specification of a design. Of course, this information, along with appropriate *theories&Models* may be sufficient to evaluate these design alternatives (e.g., against FAR 23.67). Using the mouse with appropriate pull-down menus and/or HyperCard "buttons", the designer can 1) create an interface to access a partial instantiation of a design alternative in the context of this *designDecision*, 2) interpret computer programs to create the partial instance itself, and 3) evaluate the partial instance using appropriate *theories&Models*. Results of these evaluations can be accumulated in the *designDecision* context and used as the basis to construct design space approximations.

The decision support interface in this example (Figure 10) is based on a technique for constructing multidimensional quadratic splines based on systematic evaluation of design alternatives. The simplex pattern referred to in Figure 10 (drawSimplex---top "button" on right hand side of screen and moveSimplex) is a pattern used in Experimental Design. A simplex is made up of the minimum number of points in n-dimensional space required to construct a first-order (linear) model. A "zero-dimensional" simplex is a point, a one-dimensional simplex is a line (two endpoints), a two-dimensional simplex is a triangle (three corner-points), a three dimensional simplex is a tetrahedron (four corner-points), and so on. The n+1 points in the simplex must be arranged so that substituting the points $x^1, \ldots, x^{n+1}$ (these points are n-dimensional vectors) into the linear model results in n+1 independent equations:

$$a_0 + a_1 x^1_1 + \cdots\cdots a_n x^1_n = F(x^1)$$

$$\vdots$$

$$a_0 + a_1 x^{n+1}_1 + \cdots\cdots a_n x^{n+1}_n = F(x^{n+1})$$

The equations can be inverted to give a unique solution for the coefficients a_i of a linear approximation to an unknown math function F(x). The technique uses the idea that exploration of a design space usually involves evaluating designs on adjacent simplices. If the first derivatives of F are continuous across the boundaries between these adjacent simplices, then a second-order spline approximation to F can be found that fits the design points within each simplex and is continuous across the simplex boundaries. For a pattern of two simplices in two-dimensional (x_1, x_2) space, with the common boundary at x_1=0, the spline equations are

$a_1 = b_1$
$a_{12} = b_{12}$
$a_{22} = b_{22}$
$a_2 = b_2$
$a_0 = b_0$

This gives 5 equations for the 8 unknowns in the two quadratic approximations (one on each simplex):

$$Q_1 = a_0 + a_1x_1 + a_2x_2 + a_{11}x_1^2 + a_{12}x_1x_2 + a_{22}x_2^2$$

$$Q_2 = b_0 + b_1x_1 + b_2x_2 + b_{11}x_1^2 + b_{12}x_1x_2 + b_{22}x_2^2$$

The remaining 3 equations can be used to fit the spline near some design alternatives within the simplex boundary that are of interest. These approximation techniques are more qualitative than quantitative, in the sense that they are used by the designer to guide exploration of the design space and to dynamically estimate stability and convergence characteristics of a design process plan.

An alternative approach, accomplishing the same thing but based on different assumptions, can be derived from the ideas of Fleury [13]. Fleury applies the idea of diminishing returns to conclude that a general parameter optimization problem can be cast (at least locally) in the form:

objective: minimize $w = w_1x_1 + \cdots\cdots + w_nx_n$

constraints: $a^1{}_1(x_1)^{-1} + \cdots\cdots\cdots + a^1{}_n(x_n)^{-1} + a^1{}_0 \geq 0$

A design variable y_i may have to be transformed to $x_i = (y_i)^{-1}$ depending on the local slopes of the constraints and the objective function to ensure that the "law of diminishing returns" can be applied. Fleury has also come up with some ingenious heuristics to extend the approach to discrete design parameters.

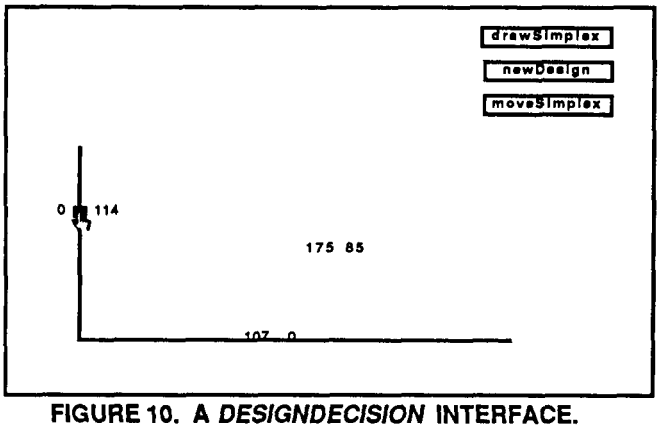

FIGURE 10. A *DESIGNDECISION* INTERFACE.

WALK-THROUGH IMPLEMENTATION OF A HALE DESIGN

Users will have made several system-level choices prior to reaching the aircraft-related portion of the interface which starts with the chart presented in Figure 11 below. This figure is a depiction of the

relationship of historical aircraft categories to one another. Users may choose an aircraft category by clicking on any ellipsoidal area. Under each ellipse is an invisible button which is programmed to take users to the branch of the interface designated specifically for the one particular case. Users may then enter more detailed information about the specific aircraft they wish to investigate. Or, they may wish to investigate several alternatives within one class of aircraft such as high altitude long endurance (HALE). Figure 12 presents several choices within the HALE class. Users must make a decision which will again branch the interface. HALE design considerations for each powertrain vary enough from one another to warrant this further branch.

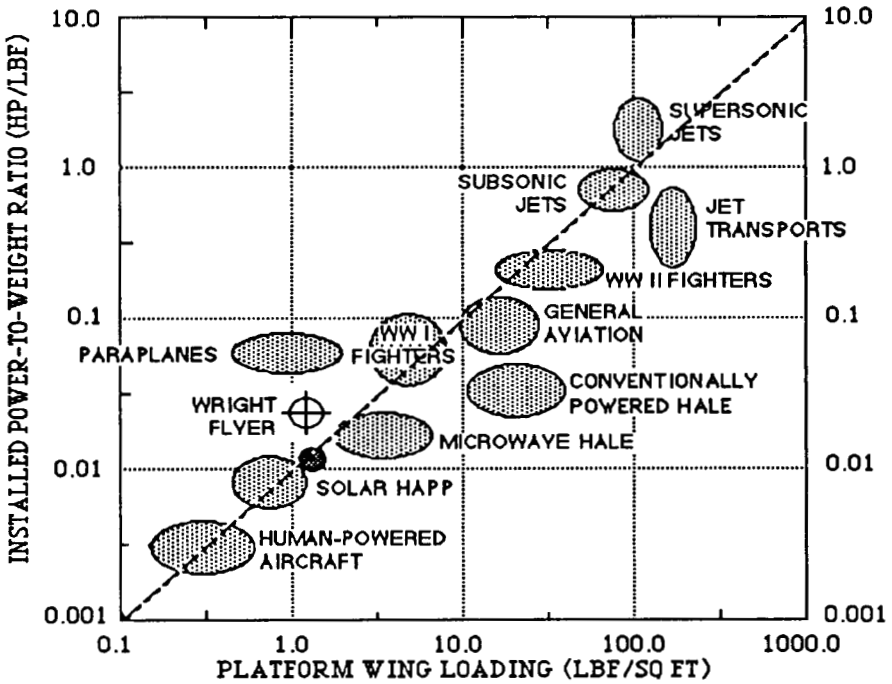

FIGURE 11. HISTORICAL AIRCRAFT CATEGORIES.

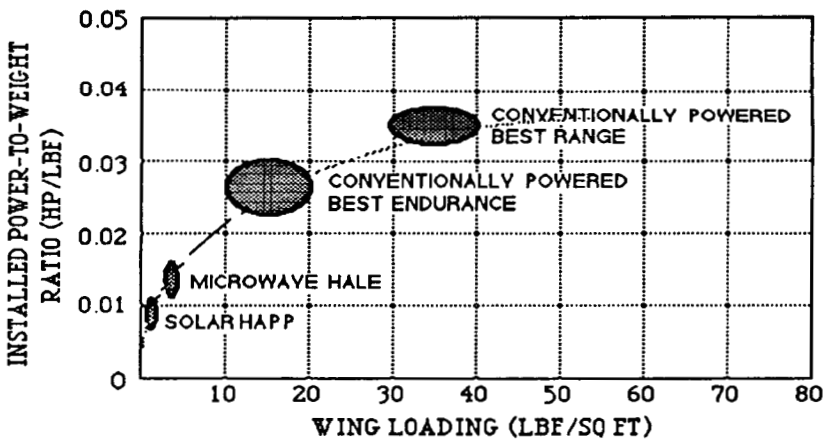

FIGURE 12. HISTORICAL HALE AIRCRAFT CATEGORIES.

Once users have determined which category and sub-category to investigate, they may start entering information about the specific aircraft they wish to model.

As an example, Figure 13 shows how a generic HALE mission profile may be described. This schematic has invisible HyperCard fields adjacent to each mission leg label which users may tab through when entering data. Tabbing is clockwise and returns users to the field in which they started. To exit, users have choices of returning to a previous screen or going to the next by clicking on the "Done" button. Returning to a previous screen voids entries and going to the next screen accepts them.

At this point, users have choices of investigating point designs or doing parametric analyses. Parametric analyses start with defining ranges of design parameters to be investigated by using an entry screen similar to that shown in Figure 14.

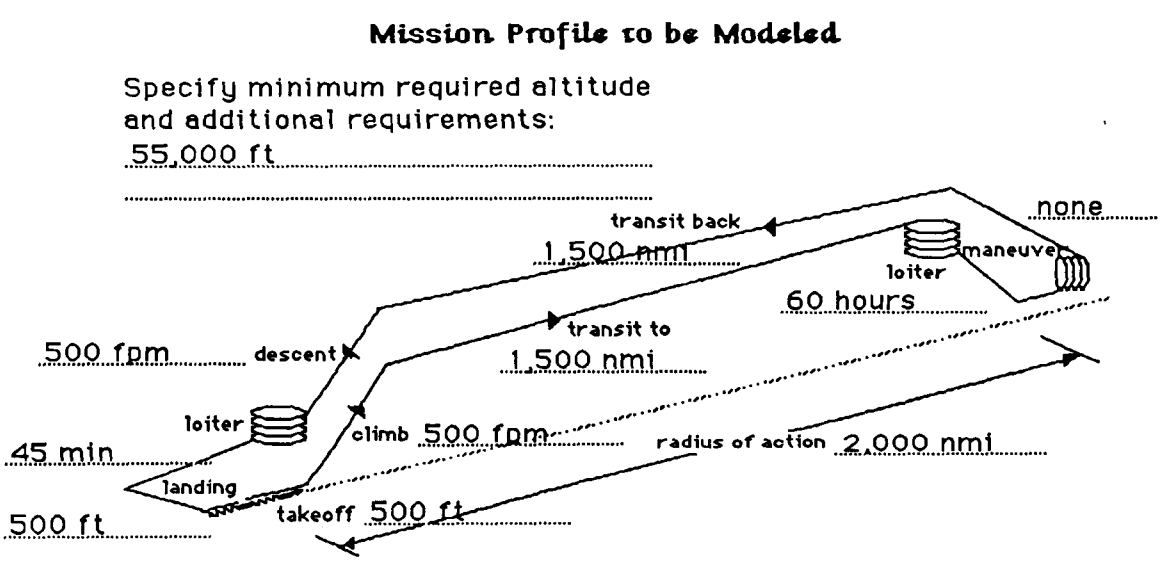

FIGURE 13. TYPICAL HALE MISSION PROFILE.

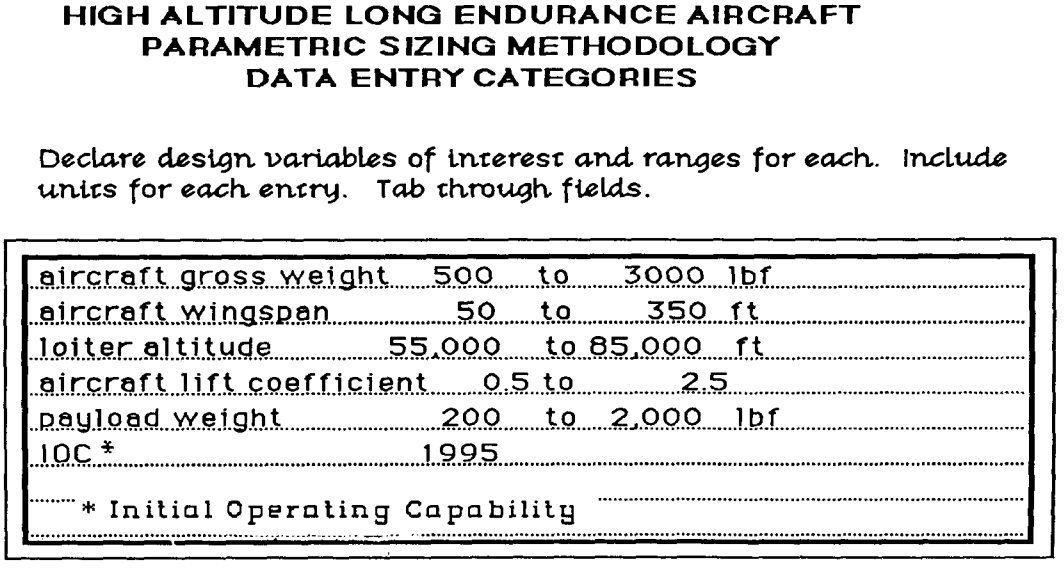

FIGURE 14. PARAMETRIC DATA ENTRY SCREEN.

Output may be presented in several formats, one of which is presented in simplified form in Figure 15 below. At times, users may want to examine the effect of changes in propulsion cycle on figures of merit for their chosen mission. Endurance is an obvious figure of merit with which to examine the effect of propulsion cycle choice. Numeric and symbolic data may be manipulated to provide curves which identify domains of endurance for which each propulsion cycle is best suited. Ideally, users would be able to click on any point on these curves and examine calculated design parameters in more detail.

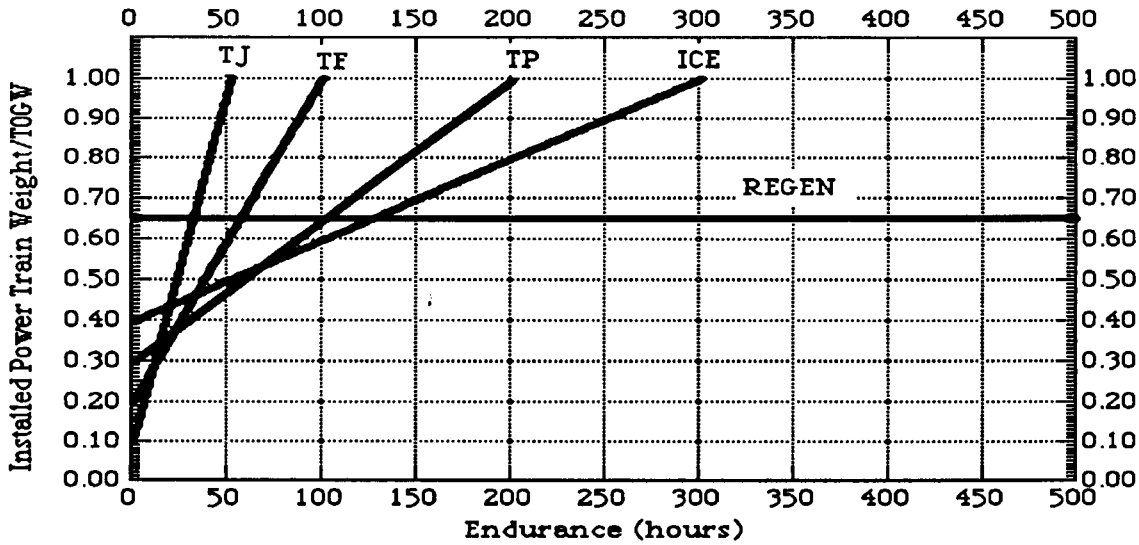

FIGURE 15. EFFECT OF ENGINE CYCLE CHOICE ON ONE DESIGN PARAMETER.

A more useful format, perhaps, for these investigations of domains of design points which satisfy a given set of conditions is a parametric plot such as Figure 16 below which is one of several parametric plots created during a feasibility study of a microwave powered HALE aircraft [14]. This plot would be the result of calculations done after users have made branching choices inside the interface. The choices in this case would be regeneratively powered HALE from Figure 11, microwave powered HALE from Figure 12, and an alternative mission profile screen than Figure 12 since microwave HALE mission tracks tend to be more or less circular when viewed from above. Figure 17 below presents a generic microwave HALE mission data entry screen to act as an additional branch to Figure 13.

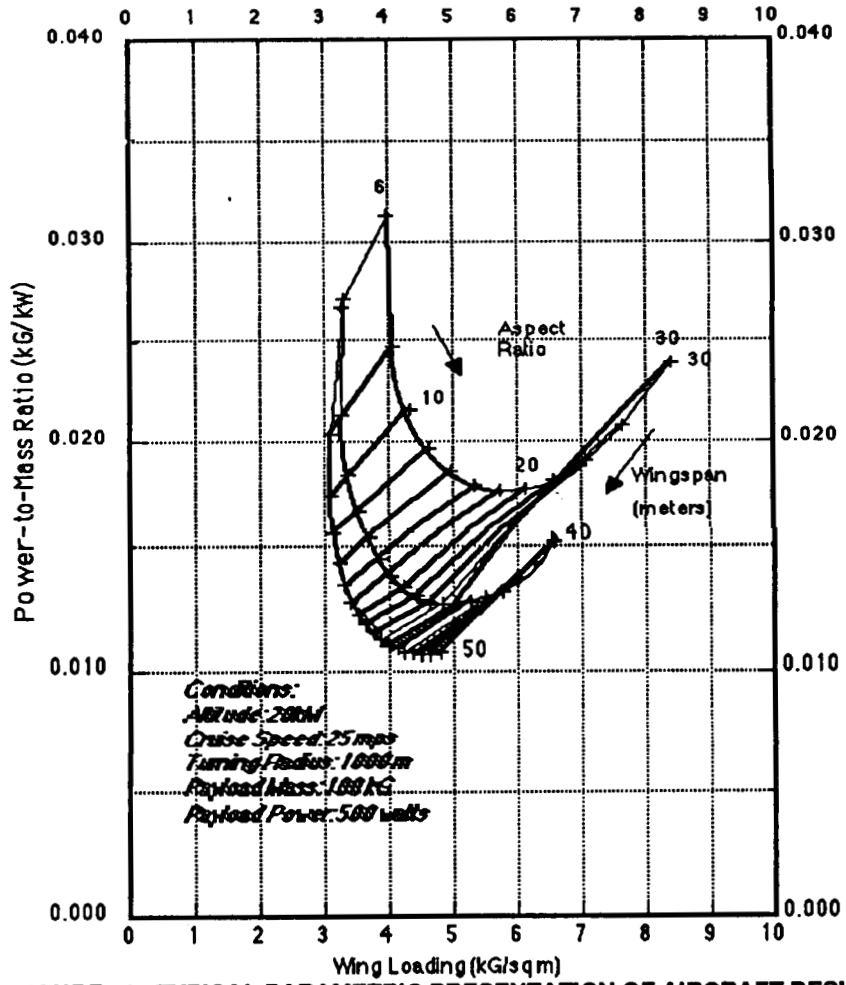

FIGURE 16. TYPICAL PARAMETRIC PRESENTATION OF AIRCRAFT DESIGN DATA.

For a solar powered HALE, an additional mission description screen (Figure 18) branch from Figure 13. Users would identify the part of the world over which a solar HALE would fly by clicking on the general area and HyperTalk would record mouse position at the click. Mouse position could then be converted to latitude and longitude by the card script. Given altitude and time of year, insolation could

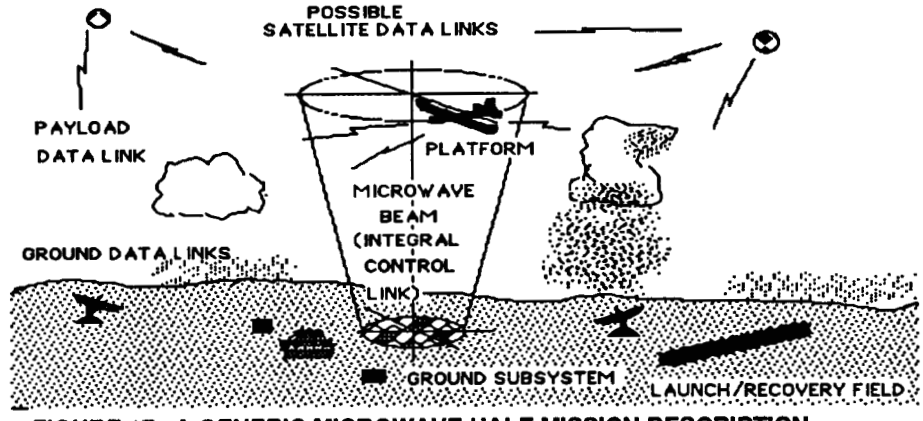

FIGURE 17. A GENERIC MICROWAVE HALE MISSION DESCRIPTION.

ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY

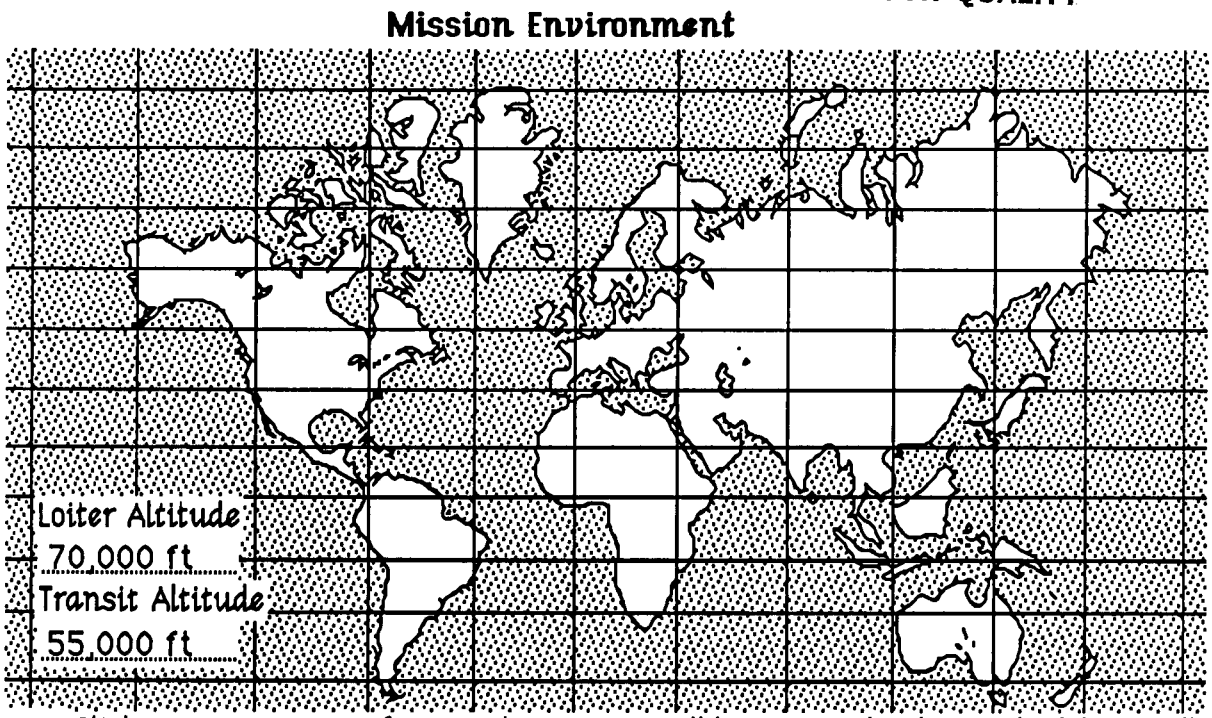

FIGURE 18. BRANCH SCREEN FROM FIGURE 13 FOR SOLAR POWERED HALE AIRCRAFT.

easily be computed using methods similar to those presented in ref. 15. Following mission data entry, the interface could define payload dimensions and constraints with a screen similar to Figure 19. The interface would then deal with the airframe by branching from a screen similar to Figure 3 to Figure 20 to describe the structural concept.

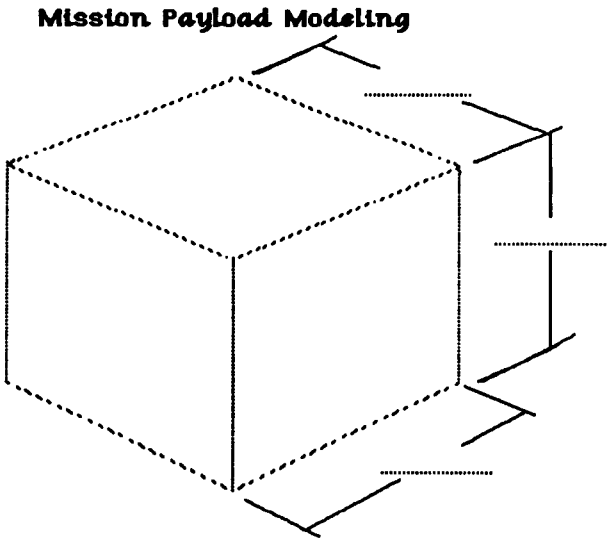

FIGURE 19. PAYLOAD SPECIFICATION.

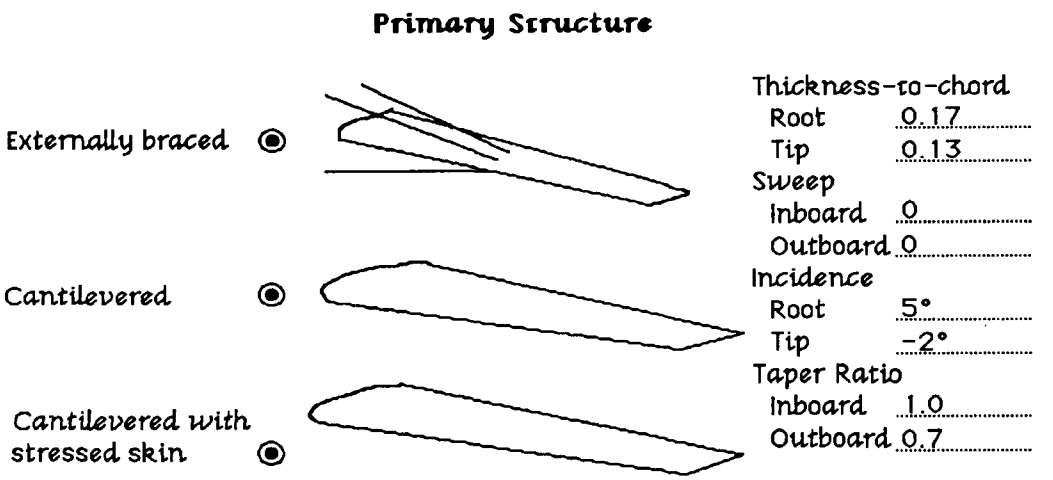

FIGURE 20. PRIMARY STRUCTURE SPECIFICATION FOR WINGS.

IMPLEMENTING THE DESIGN METHODOLOGY IN *windFrame*

Two critical questions arise concerning implementation of the high altitude long endurance aircraft design methodology using *windFrame* . How do the design tasks outlined in the walkthrough.map onto the basic elements of the *windFrame* computer programming language for aircraft design? How would some of the critical pieces of the*windFrame* design interface be implemented in HyperCard? These implementation issues are addressed in this section.

The power loading and wing loading relationships shown in Figures 11 and 12 represent a theoretical relationship between these parameters which all reasonable aircraft must satisfy, at least approximately. The existence of such a relationship suggests that this part of the design methdology should be implemented using the *theories&Models* elements of *windFrame.*

Several interesting issues arise at this point. Recall that the *theories&Models* elements of *windFrame* are conceptually at the intersection between system functions and alternative design concepts. Yet, the wing loading/power loading plots are intended to identify classes of aircraft. The designer is supposed to choose the class of aircraft that is most appropriate for the mission. From this point of view, the Figure 11 and Figure 12 interfaces are *designDecisions.* The *attributes* to be chosen are *ranges* of wing-loading and power loading. Can a convenient *conceptElement* be identified in association with these *attributes* ? An **aircraft** *conceptElement* seems a likely choice. The *function* element should probably be **perform mission** (Figure 13). The **perform mission** *function* will have some *functionAttributes* associated with it, specifying ranges, altitudes, maneuvers and speeds. At the level of the **historical aircraft categories** *designDecision* , the level of approximation employed in a *theory/Model* describing how a given alternative (one of the aircraft categories) performs the **perform mission** function is that gust loads, maneuver requirements and loads, maximum speeds, payload/range/endurance requirements, structural and fuel weight fractions associated with the mission determine an approximate relationship between the relative size of the propulsion system (power loading) and the magnitude of the aerodynamic forces and moments generated by the aircraft (wing loading).

This example indicates that the basic elements of the *windFrame* language provide an excellent way to implement the walkthrough design process in HyperCard. However, the high altitude long endurance aircraft design methodology as it has been developed so far does not make full use of the expressive power of *windFrame* , at least for the **historical aircraft categories** *designDecision*

being considered here. The value of this expressive power can be seen if we consider a slightly different use for the same *windFrame* language elements. What if the designer wanted to assess the impact of a new structural material or design concept, a new set of design criteria for gust loads, or a different approach to flying a certain mission? The same *windFrame* *conceptElements, theories&Models,* and *designDecisions* could be used to determine where the new concept falls on the **historical aircraft categories** plot. This information could help to assess whether an existing class of aircraft might be able to play a new role, or to identify that a totally new class of aircraft is needed.

RESULTS AND CONCLUSIONS

Significant progress toward the development of *windFrame* has been achieved. A carefully thought-out design for the computer programming language has been developed. This design was developed through walkthrough implementations of an existing high altitude long endurance aircraft design methodology. Used with this methodology, *windFrame* provides a convenient way to integrate traditional aircraft design practice with systems engineering discipline. *windFrame* has also been designed to provide support for "metaDesign", a process in which new technologies and evolving requirements are rapidly integrated into design concepts by "designing the design process" concurrently with the design of the aircraft.

Design decision-making is the primary emphasis for the design approach embodied in the windFrame computer programming language for design. The highly graphics/user interface-oriented programming style supported by HyperCard is a particularly compatible approach. Key elements of the decision support interface have been prototyped in HyperTalk and integrated with designer interface prototypes developed as part of the walkthrough implementation. Together, these HyperCard computer programs provide a prototype of the *windFrame* language which can be used for concept demonstration. Interested readers may wish to obtain references 16 through 18. These references are HyperCard stacks containing demonstrations of user interfaces and other aspects of the prototype *windFrame* design language. They are available through David Hall Consulting or NASA/Ames Research Center Advanced Plans and Programs Office.

REFERENCES

1. Stinton, Darrol P., The Design of the Aeroplane, Van Nostrand ReinholdCompany, New York, 1983.

2. Abelson, Harold & Sussman, Gerald & Julie, Structure and Interpretation of Computer Programs, The MIT Press, McGraw-Hill Book Company, Cambridge, Massachusetts, 1985.

3. Elias, Antonio L., "Knowledge Engineering of the Aircraft Design Process", Chapter 6 in Knowledge Based Problem Solving, Kowalic, J.S., ed., Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

4. Kolb, M.A., "A Flexible Aid for Conceptual Design based on Constraint Propagation and Component-modelling", AIAA-88-4427 ,AIAA/AHS/ASEE Aircraft Design, Systems, and Operations Meeting, Sept. '88

5. Rosenfeld, L., "Intelligent CAD Systems - CAEDM for the 90's", 6th National Conference on University Programs in Computer-Aided Engineering, Design, and Manufacturing, June 1988, Atlanta, Georgia.

6. Bouchard, E.E, "Applications of AI Technology to Aeronautical Systems Design", 6th National Conference on University Programs in Computer-Aided Engineering, Design, and Manufacturing, June 1988, Atlanta, Georgia.

7. Floyd, Bryan, "Synergistic Information Systems", 6th National Conference on University Programs in Computer-Aided Engineering, Design, and Manufacturing, June 1988, Atlanta, Georgia.

8. Steinke, G., "Engineering by the Book . . . And On-Line", Mechanical Engineering, November 1985.

9. Anonymous, Apple Macintosh HyperCard User's Guide, Apple Computer, Inc., Cupertino, California, 1987.

10. Brei, M.L., et al., Architecture and Integration Requirements for an ULCE Design Environment, IDA Paper P-2063, Institute for Defense Analyses, Alexandria,Virginia, April 1988.

11. Anonymous, Systems Engineering Management Guide, Defense System Management College, Contract MDA 903-82-C-0339, Lockheed Missiles andSpace Company, Inc., October 1983.

12. Anonymous, MacScheme Reference Manual, Semantic Microsystems, Inc., Beaverton, Oregon, 1986.

13. Fleury, C., and Schmit, L.,"Discrete-Continuous Variable Synthesis Using Dual Methods", AIAA Journal, v. 16, no. 12, December 1980.

14. Bouquet, Donald L., Hall, David W. and McElveen, R. Parker,II, "Feasibility Study of a Carbon Dioxide Observation Platform System", NASA/MArshall Space Flight Center Contractor Report, 1987.

15. Hall, D.W., Fortenbach, C.D., Dimiceli, E.V. and Parks, R.W., "Feasibility of Solar Powered Aircraft and Associated PowerTrains", NASA/Langley Research Center Contractor Report 3699, 1982.

16. Hall, D.W., "A Guided Tour of Our Developing HALE Methodology", HyperCard stack, David Hall Consulting, Sunnyvale, CA, March 1988.

17. Hall, D.W., "Development of an Integrated Design System for High Altitude Long Endurance Aircraft for Microcomputer Systems", HyperCard stack, David Hall Consulting, Sunnyvale, CA, March 1988.

18. Hall, D.W. and Rogan, J.E., "Development of an Integrated Design System for High Altitude Long Endurance Aircraft for Microcomputer Systems", HyperCard stack, David Hall Consulting, Sunnyvale, CA, June 1988.