AIAA JOURNAL
Vol. 44, No. 3, March 2006

Identifying Interfaces in Engineering Systems

M. Price,* J. M. Early,† R. Curran,‡ E. Benard,§ and S. Raghunathan¶
Queens University, Belfast, Northern Ireland BT9 5AG, United Kingdom

The behavior of engineering systems arises as a direct result of interactions and interfaces between components. Whereas many of these intercomponent dependencies are easily identifiable, many others are obscure, often occurring as a result of technical interactions between remote entities. The identification and management of system interactions forms a major part of the systems engineering practices for system architecture, but methodologies for effective identification and management of interactions arising as a result of technical interfaces are still problematic. A methodology is developed for aircraft systems that identifies analysis-dependent system linkages. The result is the capability to assess the effect of analysis fidelity on system design.

Nomenclature

b	=	length
c	=	chord
D_i	=	drag force per unit span
F'	=	lift force per unit span
L	=	lift
t/c	=	thickness-chord ratio
V_{eff}	=	effective velocity
V_∞	=	freestream velocity
w	=	downwash velocity
x_n	=	coordinates in the x plane
y_n	=	coordinates in the y plane
α_i	=	induced angle of attack
α_{eff}	=	effective angle of attack
α_n	=	angle of attack
Γ	=	circulation
ρ	=	density
ϕ_{le}	=	leading edge radius

Generalized System Parameters

$A(\,)$	=	operation on a set
c_s	=	system characteristics
$D(\,)$	=	set of input parameters, ζ
$F(\,)$	=	set of output parameters, ξ
f_{objf}	=	objective function
p_s	=	system parameters
ζ	=	input parameter set
ξ	=	output parameter set

I. Introduction

COMPLEXITY management of the design process is now a requirement for nearly all engineering projects,[1] and in response, system engineering methodologies have emerged,[2–4] providing a set of guidelines for framework development in which systems may be sufficiently classified, examined, manufactured, operated, and supported throughout the entire life cycle, incorporating both traditional and nontraditional engineering considerations into a single design environment.[5]

The SE principles characterize the system structure in a number of hierarchical views[2–4,6] in an effort to fully define the system and all its interfaces, each of the views providing a valid description, but all being needed to fully appreciate the complete system structure.[7] These hierarchical views (or architectures) provide an abstract description of the entities of the system, as well as a description of the relationships that exist between those entities within their current view.

Several different aspects of system architecture[5] may be encountered: 1) the requirements (R) architecture, which provides an ordered list of needs as given by the customer; 2) the functional (F) architecture, an ordered list of activities that are needed to accomplish the system requirements; 3) the physical (P) architecture, which represents the interconnections between the entities of the system within the physical domain; 4) the technical architecture, which provides a set of rules that govern the interconnection and interdependence of the elements of the system so that they will achieve the requirements; 5) the dynamic operational architecture, which describes how the elements operate and interact over time.

Although each of these different views provides essential information about the system structure, one of the major challenges is to understand the linkages between these architectures, both inter- and intrasystem, and how the architectures evolve over time (the system architecture should be able to absorb changes at any point within the life cycle, and these changes should be reflected in all views). For example, physically remote entities may have interactions due to their function, which provides a challenge to the analysis capabilities. This relates directly to the ability to measure any element of the system. The linkages and interactions can be identified only if they can be measured.

The aim of this work is thus to present a methodology that identifies interactions between elements of a system. The approach taken considers, in particular, measurement of the performance of a system and what happens to both inputs and outputs of that performance analysis. To tackle this problem four key issues must be considered: 1) analysis fidelity; 2) reductionist vs holistic system design; 3) simulation driven design environments; 4) real vs perceived system characteristics.

The end result is a tool that can be used for interface management within the engineering design process, enabling informed decisions about the chosen design pathways to be made. The following sections discuss each of the four key issues in turn, highlighting relevant problems and the proposed solution, before describing the implementation of the interface management tool.

Presented as Paper 2004-6457 at the AIAA 4th Aviation Technology, Integration and Operations Forum, Chicago, IL, 20–22 September 2004; received 21 February 2005; revision received 3 June 2005; accepted for publication 30 June 2005.

*Senior Lecturer, Centre of Excellence for Integrated Aircraft Technologies, School of Aeronautical Engineering. Member AIAA.

†Research Assistant, Centre of Excellence for Integrated Aircraft Technologies, School of Aeronautical Engineering. Member AIAA.

‡Lecturer, Centre of Excellence for Integrated Aircraft Technologies, School of Aeronautical Engineering. Member AIAA.

§Lecturer, Centre of Excellence for Integrated Aircraft Technologies, School of Aeronautical Engineering.

¶Bombardier—Royal Academy Chair of Aeronautical Engineering, School of Aeronautical Engineering. Fellow AIAA.

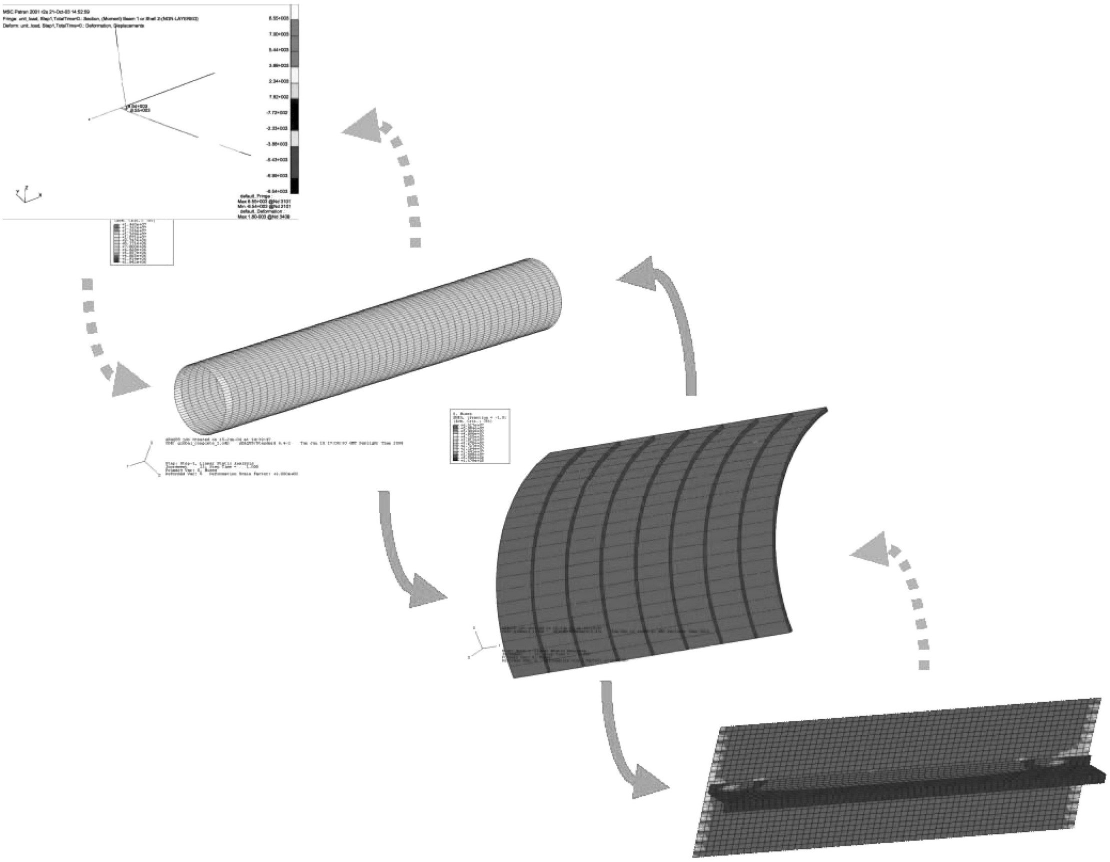


Fig. 1 System abstraction and emergent behavior.[14]

II. System Behavior and Modeling Fidelity

The current application of systems engineering has shown a tendency toward favoring the reductionist approach to the problem of providing a description of the system. This reductionism involves decomposing the system into its subsystems and components and down to the individual parts, evolving a physical architecture, and similarly for the functional and requirements architectures. Although this approach is extremely beneficial, it lacks the essential information that is provided by the holistic system view, which provides details of the system behavior.

The overall system architecture, both its entities and their associated relationships, strongly influences the overall behavior of the system. Complex systems also exhibit behavior that no subset of their elements has, as a result of the interaction of individual components of systems.[8–11] Whether designed for or unanticipated, such emergent behavior is difficult to predict and will further increase the overall perceived complexity of the system. They contain nonlinear interactions between components, resulting in evolving states (which are not always ideal), and are highly dependent on the intrasystem connections and the environment that they are in.[12] Therefore, although individual system components may be analyzed and their behavior understood, within the limits of engineering analysis capability, the interactions between components of a system add such a level of complexity that the resulting systems are inherently difficult to analyze and predict.

The definition of an interface is of interest: "a common boundary or meeting point between two different systems or processes. . . ,"[13] indicating that any commonality between systems can be termed an interface and is not limited only to the more commonly perceived interfaces found in the physical world. The technical design process generates many such interfaces (or relationships), which complement the physical and functional interfaces most readily identified within systems engineering.

In a reductionist approach systems are continually broken down until a single component or manageable subsystem is arrived at. As more levels are added to this hierarchy long chains of dependence arise, and links can become difficult to trace through the overall hierarchy. For a complex system, the arrangement of the system entities and multiple relationships between those entities become more difficult to manage as the design evolves and more detail is added. The main obstacle to reducing the level of unpredicted behavior within system design is a clear understanding of and the ability to predict all the interactions, both physical and functional, that exist within the system design, and not considering only those system interfaces that arise as a result of inheritance.

Whereas the R–F–P system breakdowns provide views of the system and links, they are, predicated on the systems engineer's view of how the product should be structured, informed of course by analysis. However, the fidelity of the analysis available clouds this judgment.

Analysis capability or accuracy can mask or reveal behavior. For example, Fig. 1 shows an aircraft fuselage section with submodels of a barrel segment and a single skin-stiffener element.[13,14] At the highest (most idealized) level, this may be represented by a beam model, and hence changes at more local levels will be masked because beam theory does not naturally account for such detail. If the functionality of the system (considering the system as the barrel fuselage section) is examined, the barrel section displays the property of the number of passengers that it is capable of carrying, although its constituent subsystems (the barrel segments, and at a lower level, the skin-stiffener element) do not exhibit this capability on their own. The level of abstraction will then determine the behaviors of the system that are visible. The issue of fidelity within the design process is also of extreme significance, with the distinction between high- and low-level fidelity models and methods extremely important: high-fidelity models are those with a small area of focus, whereas low-fidelity models are those in which the area of focus is high. Similarly, high-fidelity analysis methods often require detailed geometric information, unlike the lower-fidelity methods, which are more abstract. The issue of fidelity is an important one for the determination of appropriate models/methods to use at each stage of the process, because the examination of high-fidelity models using high-fidelity methods is often computationally intensive and, in some cases, inappropriate. The consequence is that the assumed system behavior is based on the underlying analysis models and as the real product evolves it may exhibit behaviors that are unexpected, and possibly unwelcome.

The challenge within systems engineering now is to provide an environment in which the holistic and reductionist views coexist, to give the best overall picture of the system design and its behavior, in a traceable, convenient format.

The ability to identify which systems and parameters have measurable links will provide benefits in two ways. First, gaps in the linkages can be made visible, and second, the required analysis fidelity can be stated, enabling the decision as to the value or impact of carrying out a very detailed local analysis to be made.

III. Balance Between Reductionistic and Holistic System Design

As stated previously, a complex system (a single entity intended to perform a specific functional purpose) is composed of multiple subsystems, each subsystem having required analyses performed on it to derive its attributes.[2,3,6] These analyses usually translate into a corresponding disciplinary analysis, and neglecting experimental validation studies, these disciplinary analyses have corresponding analysis software (commercial, in-house, or legacy). To perform a system analysis, it is usually necessary for the disciplinary analyses to be executed in some sequence, and as the number of disciplines and the complexity of the system increases, it becomes increasingly difficult to generate this sequencing correctly to obtain the required information.[15] From the perspective of simplifying the system design, "ideal" would enable fully modular architectures to be developed, in which each "module" of the architecture had a distinct function, with each module connected to another via a few well-defined interfaces (reductionism).[16] In the limiting case, all interactions between modules should occur over these predefined interfaces, and all the system behavior should be encompassed by the individual module behavior and interactions across the defined interfaces. This, however, is not usually the case, and most complex system designs are somewhere between fully modular and fully integral, with multiple interactions occurring across often unknown or unidentified interfaces.

In most cases the physical architecture is easily decomposed in the physical domain, in which each entity in the physical architecture is designed to perform a function. Although the physical system entities may be obtained through a logical decomposition, the attributes (e.g., the span, chord, and sweep angle of a wing) are obtained through the results of the associated analyses and can promote interactions between remote systems that have no interface in the physical domain. Although there is sequencing associated with the analyses (the output from analysis A is required before analysis B may be performed, and so on), there is also a certain amount of flexibility in the design process that can allow alternative pathways to be taken through the framework. Further, the fidelity of the analysis methodology can also influence the design pathway that may be taken.

As the system design evolves, many choices may be taken with regard to the manner in which the design process will continue. Consider again the fidelity of the analysis chosen at each stage within the design process [for example, is a full finite element analysis (FEA) required, or will a simple empirical calculation suffice?]. Although these choices may be self-evident for an experienced engineer, the overall architectural effects of such choices may not always be obvious, for example, noninheritance analysis-driven interfaces that may be overlooked. The fidelity of the design analysis will also influence the resulting output parameter set obtained. If too low a fidelity model is used, it is possible that subsequent analyses (intra- or intersystem) will be affected, corrupting the design chain (although this may not be immediately obvious), whereas, in some cases, a high-fidelity model may be unnecessary.

Aside from this, there are several other noteworthy examples of emergent properties that require prior knowledge of the interactions and interfaces which exist within the system design. Cost is often viewed as a method of ensuring reliable capital and operating cost assessment, but alternative methodologies such as the genetic causal costing model[16,17] require knowledge of the system architecture and are a prime example of currently evolving methodologies which are reliant upon an understanding of the entire system design.

Again the ability to identify measurable system interactions enables the appropriate architectural decisions to be made during the systems design process.

IV. Simulation-Driven Design Environments

The design framework within which the systems and analyses evolve is itself not a single entity, but a combination of software and hardware linked and controlled in such a way as to promote an efficient working environment. To make it possible to easily and effectively develop such a framework for a given product, a series of tools need to be made available to the user.[18,19] Open frameworks enabling collaborative multidisciplinary design have been recommended and several implementations of such environments exist. The framework described in Mawhinney et al.[20] presented methods by which the design process could be approached to adopt a design-driven rather than a model-driven design procedure. This open framework is based within a spreadsheet, operating in a heterogeneous working environment (four scripting languages and six different applications), with modeling ranging from three-dimensional solids to one-dimensional beams.[14] It is capable of performing sequential analysis from the simple empirical calculations performed within the spreadsheet environment to more involved calculations performed in Catia™ and Fluent™. Because that approach is most hierarchical and allows evolution from low-fidelity to high-fidelity detail models, it is highly suited for the systems engineering approach. This section now describes how such a framework can be used and its implications within the systems engineering design process.

Any entity within the system may be considered to be composed of a series of attributes: the input parameter set (dependent and independent parameters that constrain the type of analysis that may be performed), the analysis methodology (which imposes limits upon the fidelity of the overall design), and the output parameter set (which will possibly form constraints for subsequent analyses). Each system entity may be governed by one or several of these relationships, and many may be found to be recursive in nature. The definition of these input/analysis/output relations (Fig. 2) enables a consistent design process to be constructed that logically maps to the physical system architecture. The sequencing of this process is also completely independent of the manner in which the physical architecture is decomposed, ensuring that potential interfaces are not overlooked. The relationship between the analysis input set $[D(\zeta)]$ and the output parameter set $[F(\xi)]$ can be described by a function, $F(\xi) = f[D(\zeta)]$. The function f is given by $A(\zeta)$ [i.e., the function f is described by the analysis routine that operates on the input data set to produce the dependent data set $F(\xi)$]. In practice this means that analysis fidelity is taken into account and therefore all measurable interfaces become visible.

For example,[21] considering the flow around a wing section (Fig. 3), the two-dimensional airfoil can be considered as a wing of infinite span, with each spanwise location identical. In contrast,

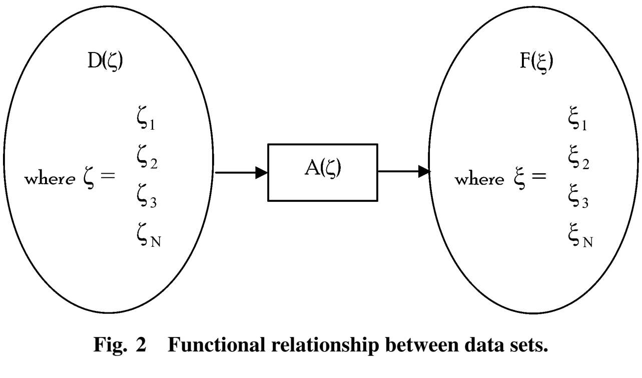


Fig. 2 Functional relationship between data sets.

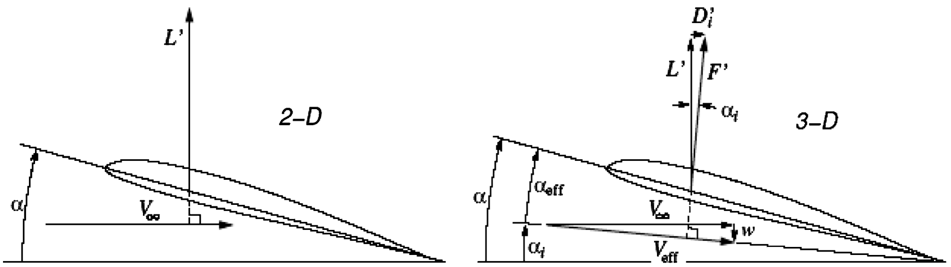


Fig. 3 Differences between the two-dimensional and three-dimensional analyses of an airfoil.

the three-dimensional wing will introduce the presence of wing tips, which will have a marked effect on the flow. Trailing vortices will be formed at the wing tips, giving rise to the downwash. There is a nonzero downwash at the wing itself (i.e., the wing operates in its own downwash), and the apparent freestream velocity is therefore tilted by the induced angle $\alpha_i = \arctan(w/V_\infty) \cong (w/V_\infty)$ (because the downwash velocity should be much less than the freestream).

The effective angle of attack of the three-dimensional wing relative to the geometric angle of attack is significantly reduced compared to that of a two-dimensional wing, and an effective freestream velocity is introduced: $\alpha_{\text{eff}} = \alpha - \alpha_i$ and $V_{\text{eff}} = \sqrt{(V_\infty^2 + w^2)}$.

Given $w \ll V_\infty$ (see earlier), the V_{eff} can be approximated as V_∞.

The presence of a downwash on the finite airfoil, induced by the wing tips, modifies the lift calculation, because the effective angle of attack is no longer the same as the geometric angle of attack. From this, the tilt induced by the downwash will also lead to a tilt in the lift force by the same angle α_i. By considering the lift force per span, $F' = \rho V_\infty \Gamma$, and resolving parallel and perpendicular to the true freestream velocity, the induced drag component, $D_i = F' \sin_i \alpha_i \cong F'\alpha_i$, can be calculated. This induced drag component is very important to aircraft design and forms a very large portion of the total drag on most aircraft. Therefore, even if the flow is modeled as inviscid, this induced drag, which is attributable as pressure drag, will appear. (The three-dimensional lifting wing will have nonzero induced drag.)

Consider this now in terms of the interface analysis, as illustrated in Fig. 2. If the possible input parameter set D is given as $\zeta = \{c, b, \alpha_n, \rho, V_\infty, t/c, \%c_{\max}, x_n, y_n, \emptyset_{\text{LE}}\}$, where x_n and y_n give the coordinates of the points on the surface, assuming (0,0) at the leading edge. Given the initial set, if A operates on the set to calculate the lift produced by the two-dimensional airfoil section, the resultant output parameter set, F, will just give $\xi = \{L\}$. However, if A operates on the set to calculate the lift of a three-dimensional airfoil section, the resultant parameter set will give $\xi = \{F', \alpha_i, \alpha_{\text{eff}}, w, V_{\text{eff}}, D_i\}$.

These two cases are summarized using the notation given in Fig. 2:

For the two-dimensional case, $(\zeta) \to A(\zeta) \to F(\xi)$, where $\zeta = \{c, b, \alpha_n, \rho, V_\infty, t/c, \%c_{\max}, x_n, y_n, \Phi_{\text{le}}\}$ and $\xi = \{L\}$.

For the three-dimensional case, $D(\zeta) \to A(\zeta) \to F(\xi)$, where $\zeta = \{c, b, \alpha_n, \rho, V_\infty, t/c, \%c_{\max}, x_n, y_n, \Phi_{\text{le}}\}$ and $\xi = \{F', \alpha_i, \alpha_{\text{eff}}, w, V_{\text{eff}}, D_i\}$.

In this case, the effect of the analysis fidelity has serious implications for the overall design, because the induced drag due to the three-dimensional flow around the finite wing section is not accounted for. This immediately has further implications for calculation of the total drag, as described in the preceding. In the context of systems engineering, the simple schematic in Fig. 2 can be repeated all over the system and, by considering input and output parameters sets, system linkages can be easily identified. Simple set operations provide this functionality. For example, the union of all the output sets with all the input sets for a given parameter identifies all the system elements involved.

These parameter sets then form the basis for the construction of a relational database that allows a sequence to be discovered for the design procedure, as well allowing the fidelity of the overall design to be altered through the analysis methodology specified. The most easily identifiable benefit of approaching the design in this manner (and one of the most commonly encountered problems in multidisciplinary optimization methodologies[21]) is that coupling within the design is not predetermined, which in turn does not impose constraints on the degrees of freedom of the system.

V. Relationship Between Real and Perceived System Characteristics

The impact of analysis fidelity, system breakdown, and the design environment having been considered, it is now possible to look at the system characteristics and what can be determined about them.

The development of the technical architecture is very different from that of the flow-down R–F–P architectures commonly encountered in systems engineering. The initial high-level system entities in this case are the physical constraints placed upon the system (the independent parameters), which then flow down through the architecture to their subordinate dependent parameters, related through their analysis procedures.

The system characteristics (c_s) are a function of these system parameters (p_s), such that $c_s = F(p_s)$[22] (as shown by Krus[23]). Both the characteristics of the system and the parameters of the system may be subject to constraints (i.e., the constraints on the characteristics of the system will affect the system parameters that may be defined, and similarly, constraints on the system parameters will affect the characteristics of the system). The functional relationship F is dependent upon the defined relationships between the parameters of the set p_s, and thus to define the function in its entirety, it is necessary to discover what the parametric relationships are. If these relationships are not fully defined, the system characteristics c_s will also be ill-defined. Furthermore, the system objective function (f_{objf}) (such as minimization of the operational cost) is in general a function of the coupled system characteristics and the system parameters $f_{\text{objf}} = f_{\text{objf}}(c_s, p_s)$.[21] To fulfil the objective function, it is necessary to be able to define the functional relationship, which may only be done if all of the appropriate system relationships have been identified correctly.

The system "technical" architecture is an extension of the description of the system design. Much in the same way that the functional architecture describes the "function" of each system component and the physical architecture includes a description of the components that perform these functions, the technical architecture contains a complete set of rules that govern the interconnection and interdependence of the elements of the system so that they will achieve the requirements. The development of this technical architecture is a hierarchical process, much the way the other three views are developed. This technical architecture describes the hierarchical design process, linking the parameters that are used within the design process to the systems that they describe (Fig. 4).

It is also important to understand that the level of abstraction will also affect the outcome of any system analysis: if the system is analyzed at a high level of abstraction, then many of the interactions between systems may not be fully explored, and thus the behaviors of the system may be described incorrectly. Conversely, if a low level of abstraction is used, it is possible that although the system will be fully described, the level of detail may be too much, increasing design turnaround time and cost to unacceptable levels. Within the technical architecture, parameters may skip generations (i.e., a dependent/independent variable need not always appear at every stage within the architecture, but may flow down without having any

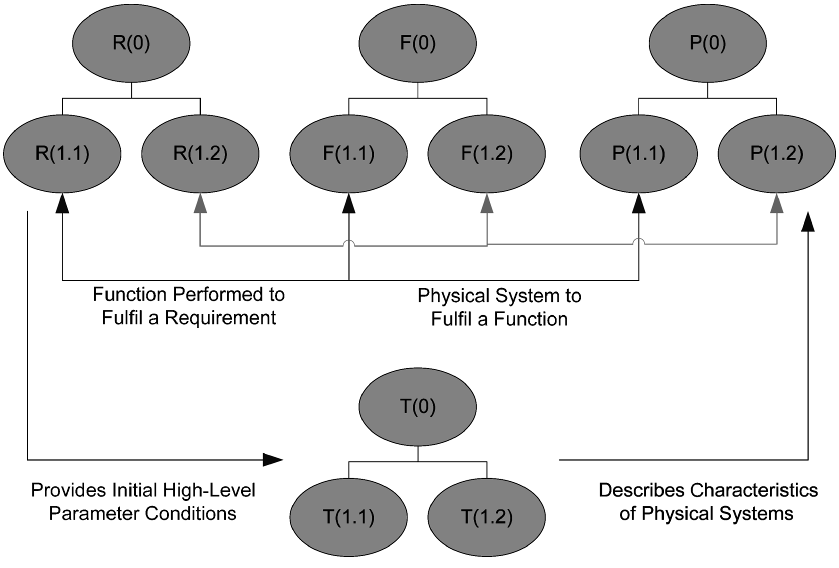

R(0)
F(0)
P(0)
R(1.1)
R(1.2)
F(1.1)
F(1.2)
P(1.1)
P(1.2)
Function Performed to Fulfil a Requirement
Physical System to Fulfil a Function
T(0)
T(1.1)
T(1.2)
Provides Initial High-Level Parameter Conditions
Describes Characteristics of Physical Systems

Fig. 4 Relationship of R–F–P architectures to the "Technical" architecture.

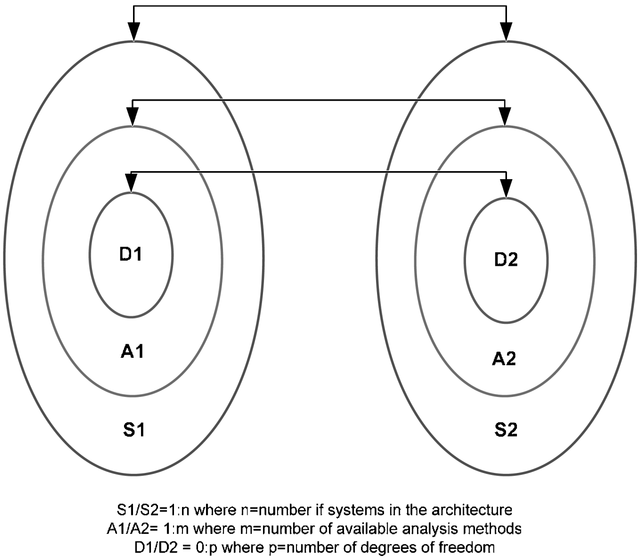

D1
A1
S1
D2
A2
S2
S1/S2=1:n where n=number if systems in the architecture
A1/A2= 1:m where m=number of available analysis methods
D1/D2 = 0:p where p=number of degrees of freedom

Fig. 5 Description of technical intersystem relationships.

impact on the intervening stages), and not all of the lower-level entities are dependent parameters: new independent parameters may be introduced at appropriate points within the architecture process.

The relationships between the "entities" (the design parameter sets: input and output) is determined by the analysis methodology applied to the input parameter set, and the relationships may be altered by changing the type and fidelity of the analysis methodology [simple analytical/multibody/FEA/computational fluid dynamics (CFD) models]. The change in the relationship type will also have an effect on the overall technical architecture, modifying the pathway through the design process that is taken, because different design methods may introduce different dependencies, and as such change the dependency chain. This also opens up the possibility of interactions between systems disappearing as design analyses are changed. The simplest description of the technical relationship is that each system within the physical system architecture will potentially interact with another system in the same hierarchy (Fig. 5). If the relationship between S1 and S2 is equal to 1 (a relationship between the two systems exists), the nature of this relationship is then determined by the nature of the analyses [analysis type (A1) that output the parameters from S1 and passed them to S2 to undergo a further analysis (A2)], and further described by the fidelity of that analysis model used (beam, shell, solid model, etc.). S1 and S2 indicate the interactions between the systems arising from the technical parameters. If there are *m* systems within the architecture, then either two systems interact (1) or there is no interaction (0); if there is interaction, then this interaction will be dependent on the analyses that the parameters are passing between (A1 and A2), and the degrees of freedom of each of the analyses (D1 and D2).

Thus, consideration of the effect of the analysis fidelity on the systems architecture and the identification of measurable links between system elements will provide the capability to identify the impact of design changes and the eventual observed behavior of the system. The form of data used to identify the interactions as described previously leads naturally to the concept of a relational database that contains the systems-elements-associated property sets of input and output parameters. The implementation of this proposed management tool using a database is described next.

VI. Implementation

The development of the system technical architecture grants traceability within the design to be established. This process can be described readily using a relational database, to simplify the process of identifying the linkages. The implementation described used Microsoft Access, because of its ease of use and availability, although the development could be easily accomplished either in similar database development packages or within tailor-made environments.

The design of any system is extremely complex, with many systems intrinsically linked to one another through a series of influencing parameters. These influencing parameters may be of several different types; external environmental parameters, external performance parameters (arrived at as a result of the requirements documentation), and internal design parameters. The first of these two are easily defined from fixed parameters that the system design has no direct influence over, but the final set of parameters are totally dependent upon the design synthesis activities. It is these factors that determine the overall geometry and arise as a result of the system analysis.

These internal design parameters may be broken down roughly into distinct areas. In the case of the generic aircraft model, these may be given as aerodynamic properties, structural properties, geometric properties, thermal properties, cost properties, and manufacturability properties, which may easily be related to the distinct analysis disciplines (CFD, FEA, CAD, etc.) Many of these properties are obviously linked (geometric-structural, geometric-aerodynamic, and so on), with output parameters from one analysis becoming input parameters for another. This is further complicated by the fact that this is spread across multiple levels and multiple subsystems of the

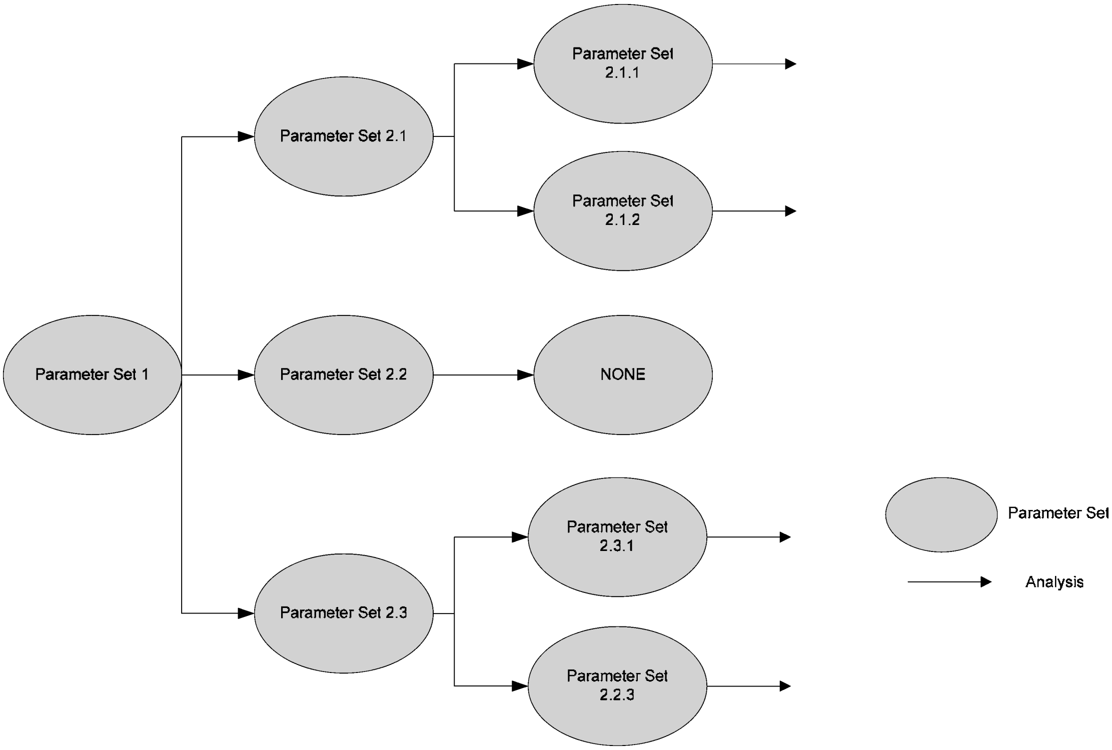


Fig. 6 Development of hierarchy of parameters.

system physical architecture. The recursive nature of the design procedure will also serve to further complicate matters.

Once a physical architecture has been developed from a series of design requirements. The first step toward detailed design is to look at the external influencing parameters to formulate a series of fixed (i.e., unchangeable), high-level design criteria, which may be used in the definition of the high-level system design. In the case of the generic aircraft, parameters such as the number of passengers and maximum range, as well as take-off and landing distances, are (usually) predefined, allowing a high-level design to be easily formulated (cabin dimensions, basic wing dimensions, etc). This new set of design-dependent (internal) parameters may then be used to further expand the design into the subsystems as a series of analysis inputs, and so on. By carefully considering the system level, the input parameters and where they came from, the analysis methodology employed, and the resultant output parameter at each level, the system design may be strongly linked to the physical system architecture.

The first step is to identify the relationship hierarchy for the design process. The form of the hierarchy is determined by the type of analysis which is performed upon a an input set of parameters. The relationship that exists between the input and output parameters is dictated by the calculation that is performed. As each of these analyses is executed, a new level of dependent parameters are introduced into the hierarchy (Fig. 6). These linkages of the system to the design methodology can be traced out within a Microsoft Excel environment, enabling a spreadsheet to be developed that describes the relationships that occur.

Using this as a basis, a database is developed, in which the input parameters are related to the appropriate analysis methodology and the resultant output parameter (as defined in the spreadsheet). Dependent upon the particular analysis used, the resultant output parameter set may vary for a given input parameter set. Because this will then influence the future analyses that can be performed, the database allows a traceable environment to be established, utilizing Standard Query Language.

Figure 7 shows the first step in the database construction. The development is also greatly simplified by the availability of an appropriate library for the system under consideration. By developing an extensive library of analysis parameters (both independent and dependent) that describe the system (as far as is known), the description of the input/output relations is greatly simplified. Tasks such as this emphasize the need for multidisciplinary input right from the start of the developmental stage.

The process of creating the relationships between the input/analysis/output sets can be greatly speeded up through forms that populate the relationship table (Fig. 8). Reference between the input/output sets is maintained through ID numbers, ensuring that the relationships are maintained correctly. This will then develop an environment that can easily be interrogated.

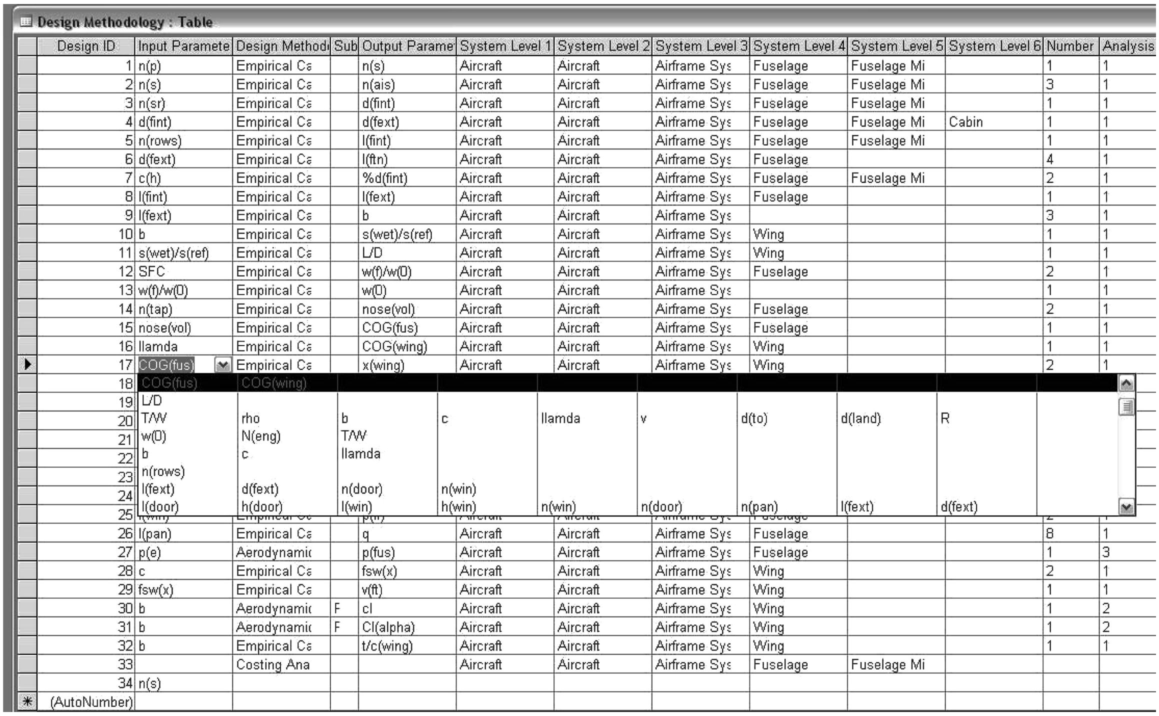
VII. Example: Generic Aircraft Development

Aircraft are complex engineering products consisting of many systems and components acting in concert. The behavior of the final flying vehicle results from a combination of individual system behavior and collective behavior thats results from their combination. The challenge of predicting aircraft behavior has traditionally been met with sophisticated engineering analysis tools, which have evolved over the years to provide ever-increasing accuracy and reliability.

However, the complexity of aircraft has now developed to the point that better understanding of behavior requires knowledge and understanding of the interactions between systems. In fact, this is common to many engineering products.

Complex aircraft systems have, due to the high risk and cost associated with their development, become a prime candidate for the adoption of systems engineering methodologies.[6] Aircraft conceptual design has been well documented in many texts,[23,24] and the interdisciplinary nature of the system is immediately apparent (Fig. 9).

Initial requirement documentation forms the basis of the aircraft requirements architecture, as well as providing the independent parameter set for the system technical architecture. In the same way that the physical system architecture is dependent on the functional requirements for each component of the architecture, the technical architecture is dependent upon the design requirements. From this, whereas the physical architecture will identify the airframe, propulsion, avionics, environmental systems, and so forth as high-level physical components[6] that can be mapped across to the

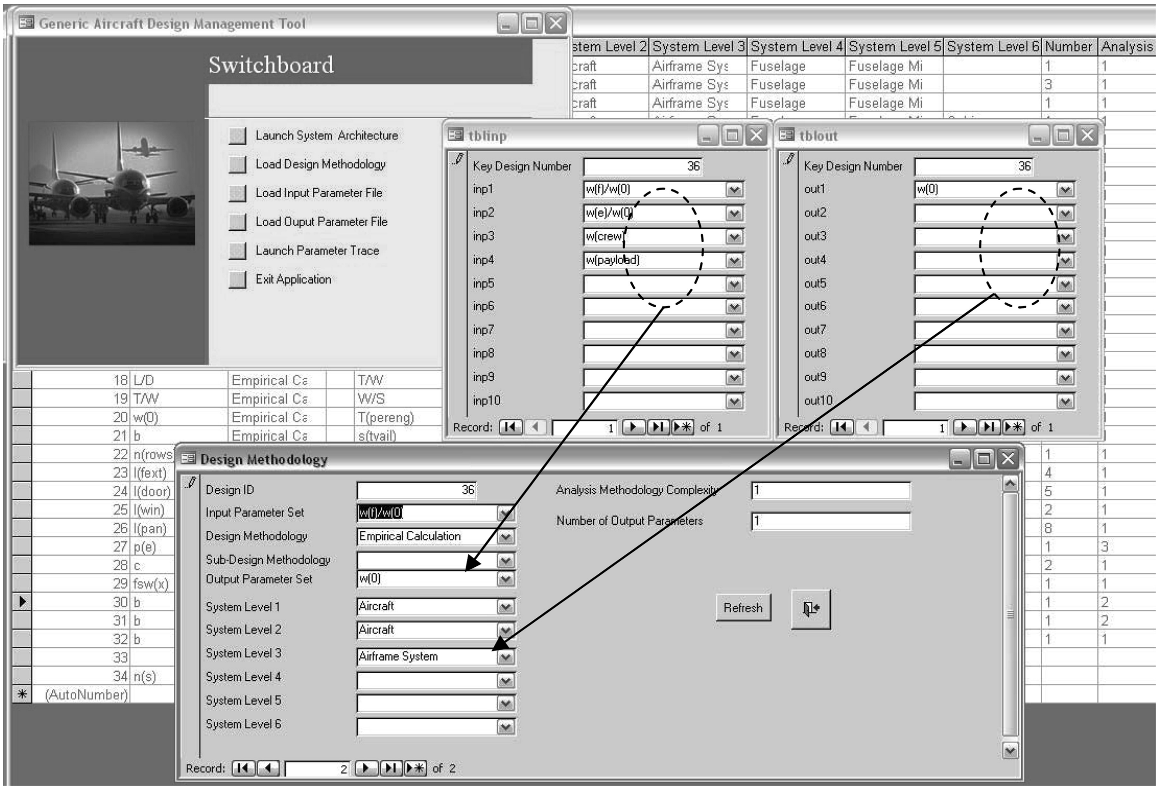


Design Methodology : Table

	Design ID	Input Paramete	Design Method	Sub	Output Parame	System Level 1	System Level 2	System Level 3	System Level 4	System Level 5	System Level 6	Number	Analysis
	1	n(p)	Empirical Ca		n(s)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi		1	1
	2	n(s)	Empirical Ca		n(ais)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi		3	1
	3	n(sr)	Empirical Ca		d(fint)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi		1	1
	4	d(fint)	Empirical Ca		d(fext)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi	Cabin	1	1
	5	n(rows)	Empirical Ca		l(fint)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi		1	1
	6	d(fext)	Empirical Ca		l(ftn)	Aircraft	Aircraft	Airframe Sys	Fuselage			4	1
	7	c(h)	Empirical Ca		%d(fint)	Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi		2	1
	8	l(fint)	Empirical Ca		l(fext)	Aircraft	Aircraft	Airframe Sys	Fuselage			1	1
	9	l(fext)	Empirical Ca		b	Aircraft	Aircraft	Airframe Sys				3	1
	10	b	Empirical Ca		s(wet)/s(ref)	Aircraft	Aircraft	Airframe Sys	Wing			1	1
	11	s(wet)/s(ref)	Empirical Ca		L/D	Aircraft	Aircraft	Airframe Sys	Wing			1	1
	12	SFC	Empirical Ca		w(f)/w(0)	Aircraft	Aircraft	Airframe Sys	Fuselage			2	1
	13	w(f)/w(0)	Empirical Ca		w(0)	Aircraft	Aircraft	Airframe Sys				1	1
	14	n(tap)	Empirical Ca		nose(vol)	Aircraft	Aircraft	Airframe Sys	Fuselage			2	1
	15	nose(vol)	Empirical Ca		COG(fus)	Aircraft	Aircraft	Airframe Sys	Fuselage			1	1
	16	llamda	Empirical Ca		COG(wing)	Aircraft	Aircraft	Airframe Sys	Wing			1	1
▶	17	COG(fus)	Empirical Ca		x(wing)	Aircraft	Aircraft	Airframe Sys	Wing			2	1
	18												
	19												
	20												
	21												
	22												
	23												
	24												
	25												
	26	l(pan)	Empirical Ca		q	Aircraft	Aircraft	Airframe Sys	Fuselage			8	1
	27	p(e)	Aerodynamic		p(fus)	Aircraft	Aircraft	Airframe Sys	Fuselage			1	3
	28	c	Empirical Ca		fsw(x)	Aircraft	Aircraft	Airframe Sys	Wing			2	1
	29	fsw(x)	Empirical Ca		v(ft)	Aircraft	Aircraft	Airframe Sys	Wing			1	1
	30	b	Aerodynamic	F	cl	Aircraft	Aircraft	Airframe Sys	Wing			1	2
	31	b	Aerodynamic	F	Cl(alpha)	Aircraft	Aircraft	Airframe Sys	Wing			1	2
	32	b	Empirical Ca		t/c(wing)	Aircraft	Aircraft	Airframe Sys	Wing			1	1
	33		Costing Ana			Aircraft	Aircraft	Airframe Sys	Fuselage	Fuselage Mi			
	34	n(s)											
✳	(AutoNumber)												

COG(fus)	COG(wing)								
L/D									
T/W	rho	b	c	llamda	v	d(to)	d(land)	R	
w(0)	N(eng)	T/W							
b	c	llamda							
n(rows)									
l(fext)	d(fext)	n(door)	n(win)						
l(door)	h(door)	l(win)	h(win)	n(win)	n(door)	n(pan)	l(fext)	d(fext)	

Fig. 7 Access development environment.


Generic Aircraft Design Management Tool
Switchboard
Launch System Architecture
Load Design Methodology
Load Input Parameter File
Load Ouput Parameter File
Launch Parameter Trace
Exit Application
Refresh

tblinp

Key Design Number	36
inp1	w(f)/w(0)
inp2	w(e)/w(0)
inp3	w(crew)
inp4	w(payload)
inp5	
inp6	
inp7	
inp8	
inp9	
inp10	

Record: 1 of 1

tblout

Key Design Number	36
out1	w(0)
out2	
out3	
out4	
out5	
out6	
out7	
out8	
out9	
out10	

Record: 1 of 1

Design Methodology

Design ID	36
Input Parameter Set	w(f)/w(0)
Design Methodology	Empirical Calculation
Sub-Design Methodology	
Output Parameter Set	w(0)
System Level 1	Aircraft
System Level 2	Aircraft
System Level 3	Airframe System
System Level 4	
System Level 5	
System Level 6	
Analysis Methodology Complexity	1
Number of Output Parameters	1

Record: 2 of 2

Fig. 8 Automatic generation of relationships within the system hierarchy.

corresponding requirements and functional architectures at the same level, the technical architecture will flow down from the requirements (independent parameters) to generate the system attributes through linked analysis procedures. In simple terms, the initial fuselage dimensions need to be determined from the requirements before the wing sizing and landing gear systems are finalized, which is not immediately apparent in the system physical architecture.

In the case of generic aircraft design, parameters such as the number of passengers, maximum range, and takeoff and landing distances are usually predefined, allowing a high level of design to be formulated easily (cabin dimensions, basic wing dimensions, etc.). This new set of design-dependent internal parameters may then be used to further expand the design into the subsystems as a series of analysis inputs, and so on. By carefully considering the system level, the input parameters and where they were derived from, the analysis methodology employed, and the resultant output parameters at each level, the system design may be strongly linked to the physical system architecture.[24,25]

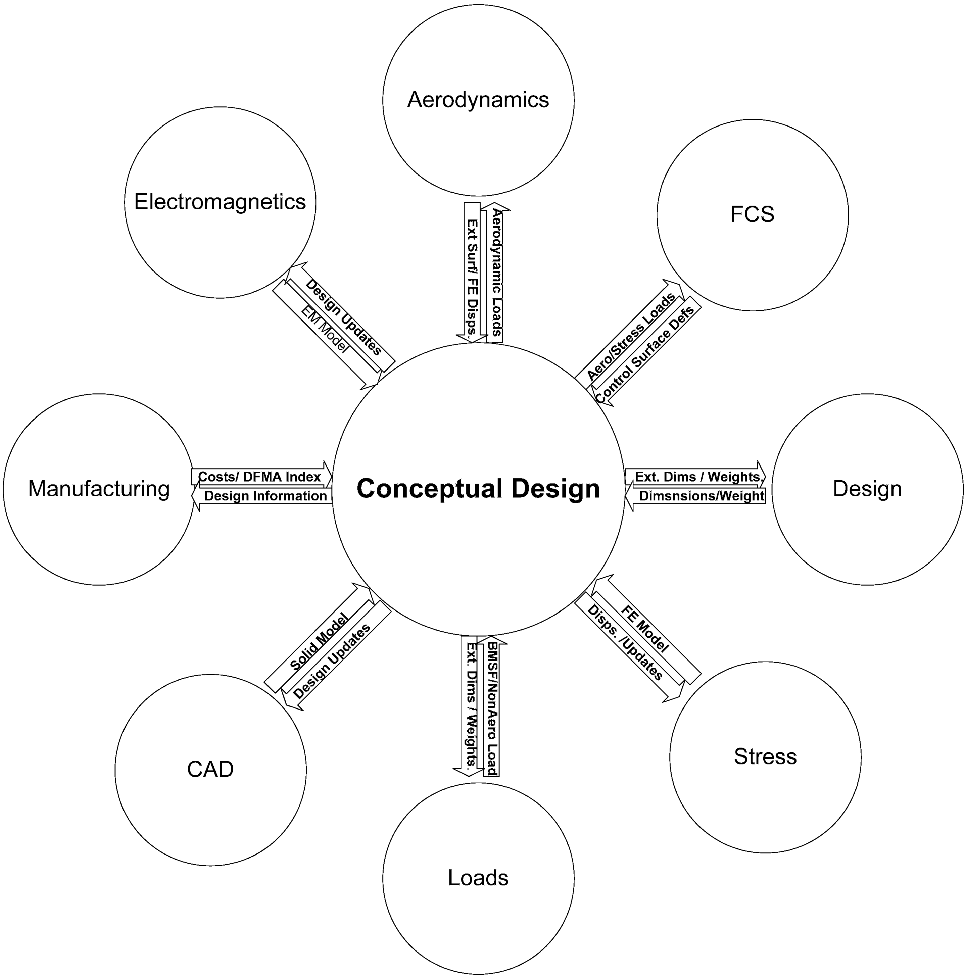


Fig. 9 Relationship between analysis disciplines in a simulation-driven design environment.[20]

The strong linking of the system design to the physical architecture serves a series of purposes:

1) Ease of construction: most products will have detailed requirements for each level of the system design, assigned to individual design teams. By considering the design in this manner, individual system designs may be built up by the appropriate specialist groups, and the overall architecture brought together by an integration team.

2) Traceability of effects: the effect of a modification upon the overall system design can easily be accounted for and traced throughout the system

3) Impact assessment: dual impact assessment on the physical architecture and on the design procedure (not only how much redesign, but also what kind of redesign).

The simple case of wing–fuselage design for the generic aircraft model (assuming a circular cross-section fuselage and low-mounted wing design) was initially modeled in both a spreadsheet environment (Microsoft Excel), and a database (Microsoft Access) to demonstrate these relationships. Influencing parameters are logically arranged for each of the systems' requirements and then related through the analyses that are to be applied.

The strong relationships between the analyses and the influencing parameters allow definite, traceable relationships to be constructed. In the case of the fuselage–wing design, the initial major parameters are drawn almost completely from the external system influences (operational and performance requirements). These parameters, when entered into the appropriate analysis, give rise to a supplementary set of internal parameters (the outputs of the analysis). The construction of the framework in this manner automatically introduces a level of inheritance. (All parameters within the system design may be traced to the point where they enter and leave the system.) It is this inheritance that is critical to the application of these tools in risk assessment and change management tasks.

Figure 10 indicates the requirement for this kind of traceability: as the system is modified, traceability within the system design ensures that it is possible to account for all systems that have been impacted. In this case, the chord length at the wingtip has been modified (marked by 1). By tracing the relationships that this parameter contributed to (for example, wing taper ratio), the parts of the system that are affected directly can be identified (marked 2). Following on from this, parameters such as the taper ratio and wing area contribute to additional relationships, which further describe other systems, which are remote in the physical breakdown (marked 3). It is ensuring an effective manner of enabling "secondary" traceability that makes this technique for developing the system architecture so successful.

By identifying a parameter for modification, the resultant impact of this can be seen in the surrounding subsystems. In the example, the fuselage skin thickness is selected (given here as t_{sk}), and using the process outlined in the previous sections, the relationships in which this parameter partakes are identified.

When the parameter for modification has been selected within the GUI (Fig. 11), the query searches the database to find all instances of that parameter occurring as an input parameter within the predefined relationships. (The library will contain only those parameters that are currently active within the system architecture.) By doing this, those relationships that will be potentially modified first are identified, with the forms returning data about the relationships that are potentially going to be changed (Fig. 12): the entire input data set for that the identified parameter is a part, the analysis that it is undergoing, and the output parameter set that is created as a result. The query also returns information about the location of the analysis within the overall system architecture. The initial query searches the database to determine which relationships are initially affected by the parameter. For this simplified case shown in

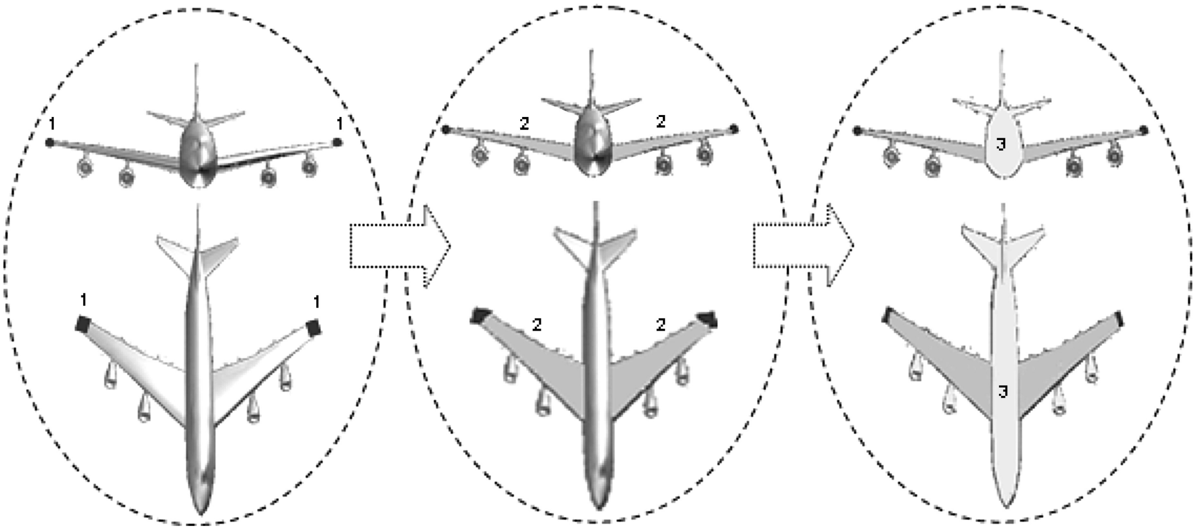
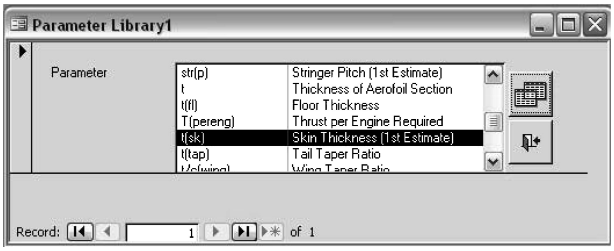


Fig. 10 Impact traceability within system design.



Parameter

str(p)	Stringer Pitch (1st Estimate)
t	Thickness of Aerofoil Section
t(fl)	Floor Thickness
T(pereng)	Thrust per Engine Required
t(sk)	Skin Thickness (1st Estimate)
t(tap)	Tail Taper Ratio
[illegible]	Wing Taper Ratio

Record: 1 of 1

Fig. 11 Selection of parameter to be modified from predefined library.

Fig. 12, the parameter t_{sk} participates in two relationships as an input parameter. The query returns the total input parameter set, the analysis undergone, the resultant output parameter set, and information about the location of the analysis within the system hierarchy.

Once these data have been returned, potentially each parameter calculated through the identified relationships can be subjected to a "knock-on" modification, propagating the change throughout the technical architecture. The follow-on query then searches in a similar manner for all instances in which the output parameters are feeding into future analyses (Fig. 13). This change of dependence may be mapped in this way in a continuous cycle until the results are returning null values (i.e., there are no further knock-on effects), and at that point, the chain may be terminated.

To quantify the effect of the modification on the future analysis simply, quick measures of the impact of the change on the analyses may be performed. Each of these analyses may be graded in terms of the analysis method complexity (a multibody analysis will be much more time-consuming and expensive to rerun than a simple analytical model) and the number of parameters that are affected (the output parameter sets): in cases in which there are fewer parameters affected by the analysis, the potential knock-on effect within the system hierarchy is likely to be less than in a case where multiple relationships and output parameters are affected. These can easily be calculated for consideration at each level.

This capability to identify chains of influence within both the physical and technical architectures is critical to ensuring traceability within the design. Although the above example demonstrates application to an aircraft system, the methodologies are equally applicable to any system.

VIII. Case Study: Lighter-Than-Air Systems

The development of the technical architecture for a much simpler system, the hot air balloon, will now be considered. By applying the process outlined in the previous section, a hierarchical description of the system design process is evolved. This is initialized from the high-level requirements for payload (weight and size), and subsequent analyses give rise to additional parameters that develop the system description (Fig. 14).

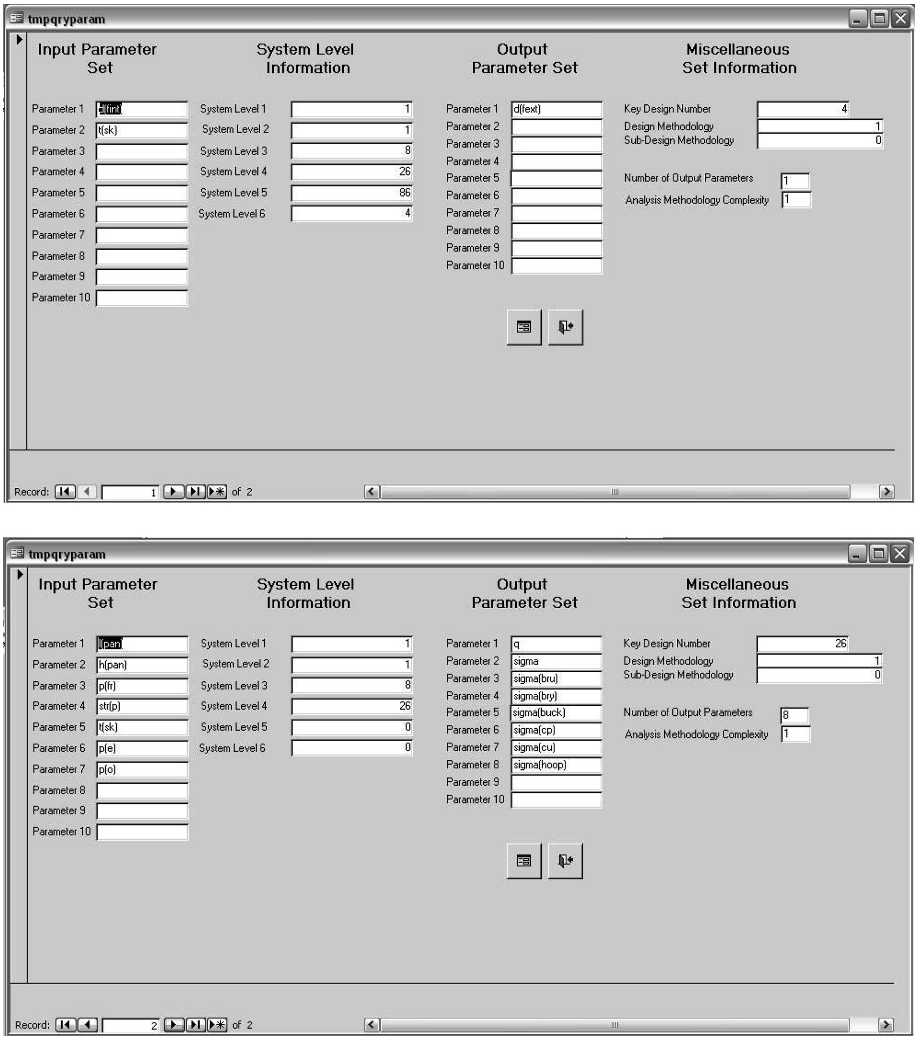
A hot air balloon[26] can be decomposed into three major subsystems: the envelope, the gondola, and the burner. The envelope (the "balloon" section) is coated internally with a plastic that aids heat retention during inflation and flight. The gondola is (usually) woven with a tight, vertical weave (wicker) and coated with urethane coating both inside and outside to prevent the wicker from becoming brittle or beginning to rot. The gondola contains the propane tanks and an instrument panel (which usually consists of a compass, an altimeter, a rate of climb indicator, a fuel gauge, and a pyrometer). To attain the required rise in temperature, a burner is usually positioned over the pilots head and is operated through a hand valve.

The envelope design for the hot-air balloon is nearly entirely based on the payload requirements. Modeling fidelity again becomes an issue, because the shape of the envelope will vary during the inflation process, under external environmental influences and so forth, and will very rarely be the conical, hemisphere-topped shape. The decision to simplify of the model design is again a balance between computational intensity and the potential for missing crucial linkages. This overall sizing will then determine the size of the burner that will be needed to raise the temperature of the air by the required amount (whether a single/double/greater burner is required, hence determining the number of propane tanks that will be required). The size of the envelope will then need to be readjusted to account for the additional mass of the burner and associated equipment and the weight of the envelope material.

Because the basket will be substantially smaller than the balloon, the drag calculation will be based upon the volume of the envelope. The stresses on the envelope skin may then be calculated using the material properties, the internal pressure (calculated from the envelope volume and properties of the heated air), and the external pressure (ambient air pressure and the force due to the "virtual" air mass).

By modeling these parametric relationships and linking them in the design space, the technical architecture for the hot air balloon may be constructed using the methodology outlined earlier. Construction of this technical architecture highlights the importance of these considerations within the design of the aircraft. As the parameters pass through the design process, the significance of the applied analysis becomes more evident at the lower end of the design process, with many of the selections made early in the overall design having a direct effect on the design pathway that can then be pursued.

Although many of the larger design parameters will obviously have a large effect on the design of the balloon (e.g., if the radius of the balloon is increased, this will have far-reaching effects on the overall design), what is of more interest is changes that may seem at first to be insignificant, but actually contribute in a significant way to the design. In Fig. 15, a schematic of the output from the interface



Input Parameter Set

Parameter 1	d(fint)
Parameter 2	t(sk)
Parameter 3	
Parameter 4	
Parameter 5	
Parameter 6	
Parameter 7	
Parameter 8	
Parameter 9	
Parameter 10	

System Level Information

System Level 1	1
System Level 2	1
System Level 3	8
System Level 4	26
System Level 5	86
System Level 6	4

Output Parameter Set

Parameter 1	d(fext)
Parameter 2	
Parameter 3	
Parameter 4	
Parameter 5	
Parameter 6	
Parameter 7	
Parameter 8	
Parameter 9	
Parameter 10	

Miscellaneous Set Information

Key Design Number	4
Design Methodology	1
Sub-Design Methodology	0
Number of Output Parameters	1
Analysis Methodology Complexity	1

Record: 1 of 2

Input Parameter Set

Parameter 1	l(pan)
Parameter 2	h(pan)
Parameter 3	p(fr)
Parameter 4	str(p)
Parameter 5	t(sk)
Parameter 6	p(e)
Parameter 7	p(o)
Parameter 8	
Parameter 9	
Parameter 10	

System Level Information

System Level 1	1
System Level 2	1
System Level 3	8
System Level 4	26
System Level 5	0
System Level 6	0

Output Parameter Set

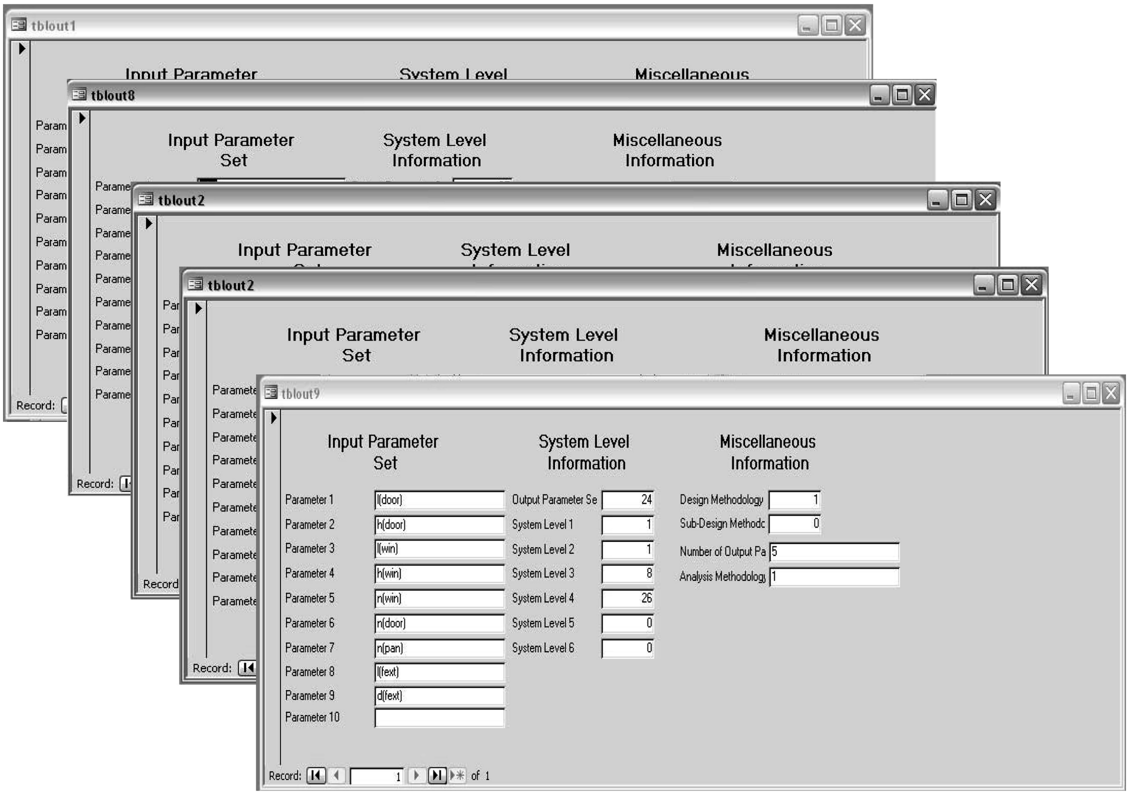
Parameter 1	q
Parameter 2	sigma
Parameter 3	sigma(bru)
Parameter 4	sigma(bry)
Parameter 5	sigma(buck)
Parameter 6	sigma(cp)
Parameter 7	sigma(cu)
Parameter 8	sigma(hoop)
Parameter 9	
Parameter 10	

Miscellaneous Set Information

Key Design Number	26
Design Methodology	1
Sub-Design Methodology	0
Number of Output Parameters	8
Analysis Methodology Complexity	1

Record: 2 of 2

Fig. 12 Identification of relationships involved in query.



Input Parameter Set

Parameter 1	l(door)
Parameter 2	h(door)
Parameter 3	l(win)
Parameter 4	h(win)
Parameter 5	n(win)
Parameter 6	n(door)
Parameter 7	n(pan)
Parameter 8	l(fext)
Parameter 9	d(fext)
Parameter 10	

System Level Information

Output Parameter Se	24
System Level 1	1
System Level 2	1
System Level 3	8
System Level 4	26
System Level 5	0
System Level 6	0

Miscellaneous Information

Design Methodology	1
Sub-Design Methodc	0
Number of Output Pa	5
Analysis Methodology	1

Record: 1 of 1

Fig. 13 Cascading influence of parameter modification.

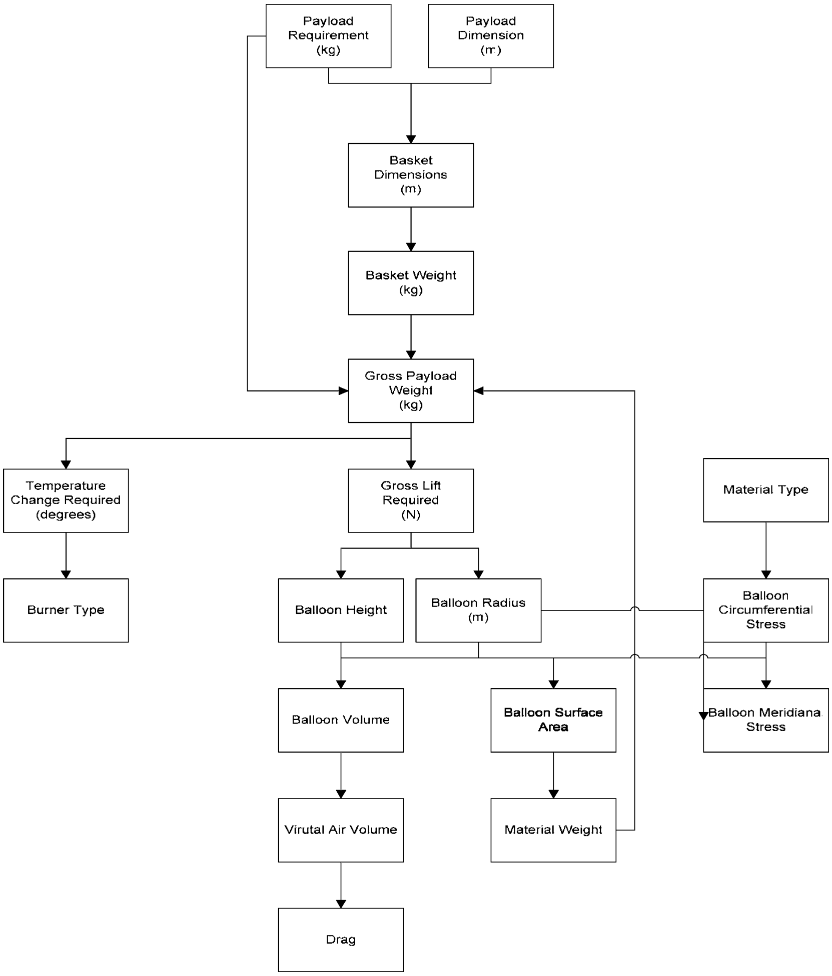


Fig. 14 Basic envelope design process: arrows between elements indicate that an operation has been performed on the data.

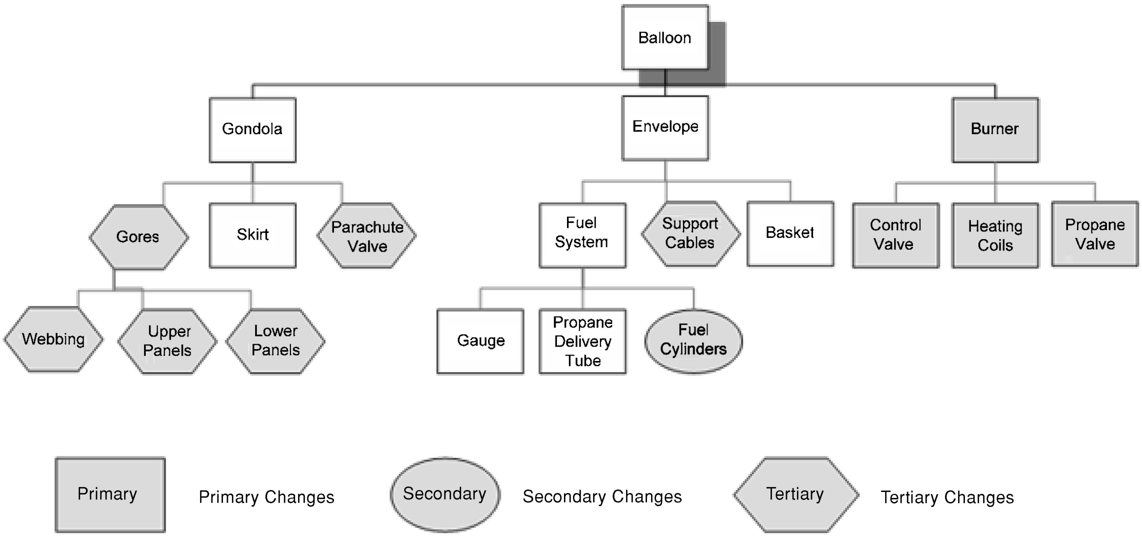


Fig. 15 Physical system architecture of the balloon design.

tool is shown for a change in the burner system. Interactions as far as the tertiary level were immediately identified, as shown. If the burner type is changed from a single to a double burner, this will use significantly more propane in the trip, leading to the necessity for additional propane tanks. This increased payload will then have a knock-on effect on the stresses in the envelope suspension cables and subsequently affect the calculated stresses in the load tapes. The calculation of the loading in these tapes is needed to ensure that the parachute is being held adequately, so that no unexpected loss of air from the envelope can occur. Because the load tapes run along the gores to the crown of the envelope, the loading in the tape will vary with gore position and height. This loading calculation, for a complete analysis, will require the stresses from the suspension cables to be calculated for an accurate analysis. Figure 15 indicates the remote location in the truncated physical architecture for the balloon system, hence demonstrating the usefulness of this additional viewpoint.

Apart from the static case, the nonlinear dynamic analysis of the envelope structure under loading becomes even more complex. With the adoption of the dual-burner system, the impulsive forces applied to the loading tapes as the valve is opened will increase with greater rate of increase in temperature due to the larger burner system. As the impulsive forces increase, the in- and through-plane stresses in the panels that make up the gores will also increase.

Alternatively, to demonstrate corruptibility within the system design due to the modeling, the temperature gradients within the envelope and the envelope design are considered to demonstrate the multiple ways in which the design process may be corrupted through the design pathway chosen. If the fidelity of the design methodology is considered (e.g.), a simple rather than multibody analysis), if the temperature gradient within the envelope is modeled simply, assuming a constant temperature increase throughout the envelope during inflation and flight (not taking into account mixing with the ambient air), this modeling assumption will have a knock-on effect on the calculated stresses in the load tapes and the in-plane stresses within the gores. If the loading within the gores is incorrectly calculated, this will in turn lead to the tensions in the parachute being incorrectly calculated, leading to the possibility of the parachute not operating correctly. Similarly, the modeling of the envelope shape during this inflation process is also going to affect the overall design: if it is modeled as an empty hemisphere and cone, with variation in shape during the inflation process, the stresses between the panels that are promoted during the inflation process will not be accounted for. If no information is available about the variation of these stresses during inflation as the envelope fills with hot air, predicting the maximum rate of inflation allowable without compromising these seams becomes impossible. This demonstrates two possibilities for corruption of the design process: either the accuracy of the results in earlier calculations introduces too high a level of uncertainty in future analyses to be reliable, or the model being used is of too low a fidelity to provide all the information required for subsequent analyses.

IX. Conclusions

As the system complexity increases, the definition of all of the interactions becomes increasingly challenging, with multiple interfaces between local and remote systems that are not always immediately obvious. Many of these interfaces are promoted by analysis, leading to the development of a fourth, technical architecture, which describes the transfer of data intra- and intersystem. The technical architecture is developed by a process through which input/output parameters are linked through the analysis performed on them, with the hierarchy of these relationships used to develop a database application. By formulating the environment in such a way that the system design is strongly linked to the physical architecture, it is possible to assess the effect of modifications at the local level on the global system design, maintaining traceability throughout the design process.

The technical architecture provides a simply, easily implemented solution to the problem of interface management within complex systems, while addressing the four key issues of adaptability for analysis fidelity, integrating the holistic and reductionist views of system design to aid identification of system characteristics, and enabling the use of simulation-driven design environments.

Acknowledgments

The authors acknowledge the support of Invest Northern Ireland under the PEACE II initiative with support from Bombardier Aerospace, the Northern Ireland Aerospace Consortium, the Department for Employment and Learning, and the research team in the Centre of Excellence for Integrated Aircraft Technologies.

References

[1] Faulconbridge, R. I., and Ryan, M. J., *Managing Complex Technical Projects: A Systems Engineering Approach*, Artech House Publishers, Boston, 2002.

[2] Blair, J. C., Ryan, R. S., Schutzenhofer, L. A., and Humphries, W. R., "NASA Systems Engineering Handbook," NASA SP-6105, June 1995.

[3] INCOSE, "Systems Engineering Handbook," INCOSE-TP-2003-016-02, INCOSE Technical Product, Seattle, WA, 2004.

[4] *Systems Engineering Fundamentals*, Defence Systems Management College, Fort Belvoir, VA, 2001.

[5] Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., and Whitney, D., "The Influence of Architecture in Engineering Systems," Engineering Systems Div., Massachusetts Inst. of Technology, Engineering Systems Symposium, Boston, March 2004.

[6] Jackson, S., *Systems Engineering for Commercial Aircraft*, Ashgate Publishing Ltd., Brookfield, VT, 2000.

[7] Hsu, J., "Introduction to Systems Engineering," MAE 408/508 Lecture Notes, California State Univ., Long Beach, CA, 2003.

[8] Poon, J., and Maher, M. L., "Emergent Behavior in Co-Evolutionary Design," *Artificial Intelligence in Engineering*, Vol. 11, No. 3, 1997, pp. 319–327.

[9] Jankovich, L., Jankovich, S., Chan, A. H. C., and Little, G. H., "Emergent Modelling for Structural Design," Univ. of Warwick/EPSRC Manufacturing Complexity Network/ESRC Business Processes Resource Centre/ESRC Network in Sustainable Complex Systems, Conf. on Complexity and Complex Systems in Industry, Warwick, England, U.K., Sept. 2000.

[10] Crick, F., *The Astonishing Hypothesis*, Simon and Schuster, London, 1994.

[11] Damper, R. I., "Emergence and Levels of Abstraction," *International Journal of Systems Science*, Vol. 31, No. 7, 2000.

[12] Ottino, J. M., "Engineering Complex Systems," *Nature*, Vol. 427, No. 399, 2004, p. 399.

[13] *Chambers 21st Century Dictionary*, Chambers Harrap, Edinburgh, 1999.

[14] Mawhinney, P., "CAE Integration for Aircraft Structural Design," Ph.D. Dissertation, Queens Univ. Belfast, Belfast, Northern Ireland, 2005.

[15] Isaacs, A., Sudhakar, K., and Mujundar, P. M., "Design and Development of MDO Framework," AeSI/ISSMO International Conf. on Modeling, Simulation, Optimization of Multi-Disciplinary Engineering System, Sept. 2003.

[16] Curran, R., Rothwell, A., and Castagne, S., "A Numerical Method for Cost-Weight Optimization of Stringer-Skin Panels," *Journal of Aircraft* (to be published).

[17] Castagne, S., Curran, R., and Rothwell, A., "A Generic Tool for Cost Estimation in Aircraft Design," AIAA Paper 2004-6235, Sept. 2004.

[18] Clymer, J. R., "The Role of Reductionism and Expansionism in System Design and Evaluation," 4th International Symposium of the International Council on Systems Engineering, Seattle, WA, Aug. 1994.

[19] Wujek, B. A., Koch, P. N., McMillan, M., and Chiang, W. S., "A Distributed, Component-Based Integration Environment for Multidisciplinary Optimal and Quality Design," AIAA Paper 2002-5476, Sept. 2002.

[20] Mawhinney, P., Price, M., Armstrong, C., Raghunathan, S., Ou, H., Murphy, A., and Curran, R., "Using Idealised Models to Enable Analysis Driven Design," AIAA Paper 2003-6747, Nov. 2003.

[21] Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary Optimisation Methods for Aircraft Preliminary Design," AIAA Paper 94-4325, Sept. 1994.

[22] Houghton, E. L., and Carpenter, P. W., *Aerodynamics for Engineering Students*, Butterworth Heinemann, London, 2003, pp. 159–169.

[23] Krus, P., "Systems Engineering in Aircraft System Design," 11th Annual International Council on Systems Engineering, July 2001.

[24] Howe, D., *Aircraft Conceptual Design Synthesis*, Professional Engineering Publishing, London, 2000.

[25] Raymer, D. P., *Aircraft Design: A Conceptual Approach*, AIAA, Reston, VA, 1999.

[26] Mawhinney, P., Price, M., Armstrong, C., Curran, R., Murphy, A., Early, J., Raghunathan, S., and Ou, H., "Design and Analysis Integration Using Systems Engineering for Aircraft Structural Design," AIAA Paper 2004-6204, Sept. 2004.

E. Livne
Associate Editor