

Memorandum M-665

**X-pert Conceptual Synthesis
of
Airplanes**

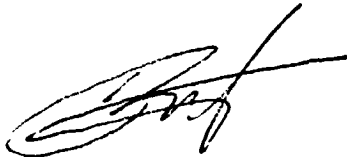
**Naresh Sharma, (M.Tech.)
October 1992**


Approved by:


prof. ir. E. Torenbeek



dr. ir. C. Bil



e-: nash@dutlfs1.tudelft.nl
vlvonar@dutrex.tudelft.nl

: (+31)-15-783992 (0)

X-pert Conceptual Synthesis of Airplanes

CONTENTS

One	<i>Introduction</i>	3
Two	<i>Software Engineering</i>	4
Three	<i>Design</i>	9
Four	<i>Knowledgebase</i>	12
Five	<i>Path Inferencing Processor</i>	16
Six	<i>Statistical Methods Processor</i>	17
Seven	<i>Sizing and Analysis</i>	18
Eight	<i>Statistical Methods Processor</i>	24
Nine	<i>X-cessor and X-amine Processor</i>	25
Ten	<i>The Graphics User Interface</i>	27
Eleven	<i>X-pert Processor</i>	30
Twelve	<i>Conclusions</i>	32
References		33

(A Proposal)

Aircraft Conceptual Design has evolved a long way from the days of spruce and fabric mache that was perhaps sketched on a piece of paper by our daring pioneers to a state-of-the-art technology that uses computers performing millions of calculations per second. Most of the conceptual and preliminary design is done on computers using complex software specially developed for the same. In the recent times with revolution in computing and software development the entire aircraft development process seems to have been integrated into computing environments.

This boom in computing which started in the 1970's has already seen usage of several conceptual design software. Some of which have developed into really complex and reliable design systems. It is imperative to view the proposal of a new design system in the light of availability of existing software. The primary issues to be viewed in favour of such a proposal are, "*can new software for aircraft design be written which, as compared to existing codes, is more*":

1. User-friendly
2. Fast
3. Clearly-organised
4. Easy to understand and implement (POV of developer)
5. Extensive in capabilities
6. Extendible to the "unconventional"
7. Accurate
8. Intelligent

Before expanding into the above one also needs to consider the possibility to upgrade the existing systems, but the question is for how long and at what cost (cumulative)? The effort required to just translate a loosely written program with the 70's technology into the-state-of-the-art languages can be more than just creating a new program. That is, if the tedium of such an effort is considered worthwhile. Computer scientists toil on such matters as programming assistants that do part of such exercises, this only proves that we should review our processes from a *revolutionary rather than an evolutionary approach*.

An approach is suggested here for an "*X-pert Conceptual Synthesis of Airplanes*," XCSA, program written specifically with a view of above criterion. It will be developed from the scratch to draw out the most from known concepts of Artificial Intelligence and their application to aircraft design. With interfacing capabilities to optimizers such as the existing ADAS system, this should prove to be of service as a design, research and development, and teaching aid for the mid 1990's.

A part of this is proposed as a project for Ph.D. research at the faculty of Aerospace engineering, TU. Delft, The Netherlands.

(An Overview)

Before we traverse the field to the subject of designing our software a critical eye should be cast on the-state-of-the-art in software engineering as well. The primary issues here are somewhat different and one need to seek harmony with them too. These issues culminate in the following trends observed on software development and usage in scientific as well as commercial fields.

SOFTWARE LIFE:

Clearly a design system, such as the one proposed, demands enormous amount of effort that needs justification. It may be estimated that the total life span of such software is in the range of ten years. This can be broken down into design development and testing that may last for about 3 to four years. Usage with subsequent version updates may last for another six to seven years depending upon the development of competing software. Since it is realised that this software will see usage starting in a few years hence, it must be understood that this must incorporate the state-of-the-art in present day technology.

ARTIFICIAL INTELLIGENCE IN PRACTICE:

The initial concepts that had set the goals for Artificial Intelligence (AI), the mimicry of human brain, are far removed from the presently held view that maintains that AI's primary aim is to make machines smarter. AI is an essential tool that develops computational techniques for simulating intelligence and discovers their properties and inter-relations. This is of utmost importance in Aircraft synthesis where some really feasible plans can be left untouched by normal thought process that may seem logically probable, such reasoning may come out as a result of AI methodology. Though many-a-times such logic may also seem completely garbage since the method is only good till it has the right information coded. Though even after year's research, AI has yet to come out of infancy, it is a hot topic of research and one can safely rely on its maturing in very near future. Thus it will be safe to expect systems with AI concepts and smart databases to spring up in near future. With this in mind we must incorporate AI principles or have the facility to do so in our system.

According to Roger Schank, the ten most important details that loosely define AI systems are:

Representation:

At the heart of AI is the question of how to best represent the knowledge in a computer.

Decoding:

How do you get real-world knowledge into an internal representation? In

some AI fields, such as voice recognition or machine vision, decoding is the central issue. Schank thinks that decoding systems and representation systems must be considered together when building a system.

Inference:

AI systems must be able to extract meaning from input. They must be able to infer meaning from limited clues.

Control of Combinational Explosion:

Putting it simply, an intelligent program should know when it knows enough about a subject and when to stop.

Indexing:

Schank sees knowledge retrieval not as a search problem but as the optimization of the organisation and labeling of memory. AI systems must be able to get at what they want.

Prediction and Recovery:

An AI system must be able to make prediction about events in its area and be able to explain when the predictions go wrong.

Dynamic Modification:

In Schank's opinion, the ability of the system to change its internal representation based upon experience -- in other words, learning -- is the quintessential AI issue.

Generalisation:

AI programs need to be able to make generalisations from different experiences.

Curiosity:

Schank believes that truly creative computers could surpass human beings and that AI must become familiar with investigations of creativity in other fields.

CROSS PLATFORM PORTABILITY:

With the emergence of several platforms and associated operating systems, each one being a shade different from the other, this factor is of the utmost importance. One can no longer write software for one platform and ignore the rest. The spectrum of end-users and developers help in deciding the compromise one needs to be made about the operating system and platforms on which such a system should run. In the present day

this is an obligatory requirement expected from even the simplest software. One is almost forced to have cross-platform portability if one wishes to harness the potential of modern computers, especially of the computers at the low end of price range. The capabilities of such systems seem to have steadily grown with equally fast rate of reduction in their prices.

The advantages of such development can be seen in data-transparency, multi-user access, inbuilt security systems, network operability, etc. Each one of these in themselves can be expanded to more detail but what brings more surprise is the observation of Andersen and Sherwood on C. They conclude that though C in itself is not portable, the compilers have to be different each with their platform specific enhancements, the combination of C, X-11 and OSF/Motif has achieved a high degree of portability on UNIX machines. Our target machines being of the UNIX (if possible lower end machines as well), this observation is to be viewed with great concern.

There are arguments against interoperability too; they circle around the fact that most platform vendors supply interoperability by means of some interface programs. It is accepted that such appendages reduce speed, however, with the exponential growth in computational speed these reductions in speed may seem insignificant. Furthermore evolution of recent standards in the software field will dictate the design of platforms enhancing interoperability and performance.

GRAPHICS USER INTERFACE:

Graphics User Interface (GUI) has created the maximum impact on present day software technology. This being the interface between the user and the computer it has gained wide acceptance among users. One of the reasons that many well designed and functionally superior codes that interface in numbers rather than with pictorial representation are not as popular as their lesser counterparts. It also acts as a buffer between the user, who may not be interested in the workings of the program, and the complexities of a synthesis code. Thus a well-designed GUI with pleasing appearance is a necessity for any software written for the future. With the growing number of GUI's and their applications, users are put to inconvenience with as many GUI's to master. Hence it is imperative that today's software should be built around a GUI, but the question is which one? If we consider a UNIX based developing environment as a default then the requirement of GUI forces one to carefully consider the X-11 with the OpenLook compatible interface. With the acceptance of X-11 as an international standard in windowing environments, it is available on almost all UNIX machines. The OpenLook interface, formulated by AT&T and SUNSOFT, has been designed using ergonomics and has been accepted as a standard for GUI development.

Essentially OpenLook GUI is a specification for the "look and feel" of a windowing environment for a multitasking computer system. The look and feel include the types of objects' users see on the screen and the basic conventions for how users work with those objects. It is designed around "objects and actions" and addresses the commonly ignored problems of simplicity, good visual design, clearly labelled controls, familiar framework and consistency. Such an interface will substantially speed-up and ease the

work of a user and thus it must be considered an extremely important part of XCAS.

MODULAR LANGUAGES:

With the choices made above requirements for a programming language also need to be set forth. Object oriented languages lead one across another long list of languages for instance, SIMULA, SMALLTALK, MODULA, C++, lisp, etc. Stefik and Bobrow deal in detail on this subject, however, the question of which language to use must be answered with deliberation. Among the important points to be evaluated are:

Compiler Speed:

It is of prime importance that the language chosen must have fast compilers available on a wide spectrum of machines. It is also essential that the standard based functions of such compilers operate at fast speeds.

Modularity:

Language must promote modularity, expandability, must be Object Oriented and should support the basic paradigms of AI. Modularity must extend to over compiled code too and must not sacrifice speed in recursive searches.

Maturity:

A good compiler that sees more usage has least bugs in it and will be preferred. Maturity is important from the point of view of life of the software also. We must though outrule those languages that are mature but are in the process of decay.

Cost:

Compiler cost and associated software development tool kits, windowing tool kits, etc., can be prohibitively expensive if obtained for different platforms. Thus this is also one of the important aspects since we plan to port the system on several platforms.

As we attempt to consider all the above issues, fully realising that the above list is rather incomplete, we conclude that no one language will meet our criteria. Thus we make our first compromise here. "C++," the modular version of "C," supports object oriented programming and is widely available on a competitive basis. The advantage of C lies in the fact that "UNIX" is essentially written in it. This makes most C compilers faster than comparable compilers. As C lies between assembly level languages and high level languages, e.g., FORTRAN, PASCAL, Lisp, etc., it gives the advantages of low-level access to computer's through standard-based routines. Most graphics tool kits and software libraries are available in C. Since C++ is a superset of C it inherits the advantages of C by default. We must use C++ as a basis for development but should not outrule the possibility of using Lisp in the future.

INTEGRATED ENVIRONMENT:

With the raising expectations of users and the enthusiasm of the designers to cope

with it, the size of the codes becomes unbearably large. As outlined before, codes of this size are not simple or easy to handle and most developers resort in unifying their codes with existing programs. Such unification may be made workable with extensive effort by the developer, but they always remain a compromise. The compromise is in speed, extendibility, upgrade-ability, etc. All of these issues make it imperative to critically judge the possibility to design the system by incorporating the essential features without the need of previously compiled packages.

A STANDARDS BASED GRAPHICS:

Jayaram et.al. have done extensive work on standard based graphics for Aircraft design. It is an accepted fact that in the present day, with the advancement in computer graphics, we can afford to have high resolution displays which are capable of handling the visual graphic representations of Aircraft. It is proposed to use Phigs based graphics system for the development of our design environment.

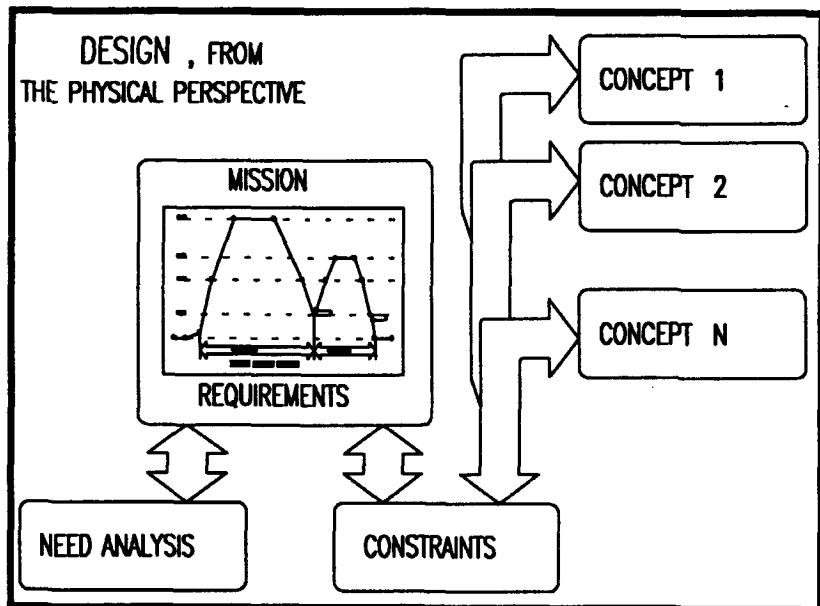
USAGE ACROSS THE NETWORK:

An X-window based interface that is consistent across the LAN makes life so much easier for the user since the user can now concentrate on the development of his design rather than worry about the irritating subjects of file mismatch-match plotter or printer configuration, etc.

(The Conceptual Aircraft)

There is some controversy about the stage until which conceptual design plays a significant role. Design is usually considered to be the achievement of certain goals based upon some inputs meeting a specified set of constraints. This is rather loose and interpretations generate large variations; however, the definition of design by Bouchard seems to be slightly different. He maintains that the design is the set of instructions that describe how to build a system that maximises a measure of merit subject to satisfying some set of requirements. Though Bouchard's definition of design process is radically different, we feel the need to view it from a different viewpoint.

Design, should be considered to be the "formulation and execution of a set of instructions that describe how to build a system that maximises a measure of merit subject to satisfying some set of requirements." Thus we include the realisation of the concept as an inherent aspect of design, which is in contrast to Bouchard's definition. Taking this new definition as the starting point of our concept design system, we can view the process as consisting of formulation of instructions and execution of the same.



Following is a small discussion of the design process from its physical perspective, ending with a recommendation of a design methodology.

REQUIREMENTS:

These are of paramount importance since this drive the design and define the measure of merit. Requirements themselves come from a detailed need analysis, which includes market survey, study of technological state, etc. Need analysis is considered to be out of the scope of present work but it's important to mention that the level of detail in requirements depends upon the amount of effort put into the need analysis. As a result, requirements lie in a wide spectrum ranging from rather woolly to point precise definitions.

From the design engineering point of view, a detailed and precise set of requirements is preferred. The design constraints are derived from them moreover the search

methods operate under these constraints. Thus precise requirements narrow down the search domains and assist in faster and more accurate designs. However, our system should be able to accept rather loose requirements and generate, from this, a detailed and precise set that is the minimum required for a feasible design process to proceed. This will need a powerful search and cross-referencing mechanism with existing designs as part of its search domain. The search domain of such proportions must have the ability to manipulate data based upon inferential statements such methods force one to consider object oriented databases with intelligent search methods.

Once the requirements are formulated in sufficient detail to start the design and synthesis system, further requirements may be evaluated insitu based upon the state of the system. It must also be possible to cross-check the requirements from different paths for any conflicting expectations.

If one may be in a position to design a system from a set of requirements it must also be possible to extract the set of requirements based upon which it was designed. This is of great help in comparing existing designs. Thus if one requires the mission profile of a design which exists in the database, the profile about which such an aircraft was designed must be presented. A proper design cycle may be able to synthesise several airplanes based on the same set of requirements but only one set of requirements can be generated from a design. Therefore such a facility is feasible and helpful and must form a part of such a design system.

MEASURE OF MERIT:

This part of the design process, which is most effected by technological and economic progress drives the design goals. It is very important to be aware of the various functions that model such merits as Direct Operating Cost (DOC), block time, etc., very carefully. The errors due to approximations made here directly add up into the errors of the design system. It is imperative that in such a case recourse must be taken to the most recent market trends. A design system must be in a position to re-evaluate these figures from such recent data and make statistical comparisons and derive schemes for statistical extrapolations.

FORMULATION:

The formulation referred to here is not of the base design process but of the set of instructions that describe how to build a concept. As referred to here, formulation is that dynamic process at the end of which, design execution for its realisation may begin. This process in the real world integrates technologies from all disciplines that effect the design process. Thus it is the most involved stage in design.

Some even divide this process into an array of analysis, evaluation and iterative cycles. This form of view of formulation is fundamentally different from ours in which repetitive iterative cycles, in which sequences of operations are performed are not considered to be the optimum method. Our assertion is that the sequences of operations may not remain the same in each cycle and we must know which processes the design must go through at each stage. This becomes extremely complex in the case of

aircraft design, since the number of processes and variables is rather large. To be able to direct the design process at each stage is the crux of the problem and needs extremely intelligent and independently working processes.

EXECUTION:

An interesting aspect of design methodology that is proposed here is the, "*execution of a set of instructions* ... In the real sense, execution of the said instructions must provide us with feasible aircraft concept(s). Here, however, our feasibility is measured by *execution in the form of a representation*, the representation being in CAD. This implies that the part of execution is the, "Interface of the system with various CAD outputs."

The representations of such a design which may be considered in lieu of a physical object may be considered to be a fairly detailed CAD model, its associated properties, e.g., pressure distributions on surface, location of major structural members, its associated performance curves, etc.: these too with a facility at this stage to compare the concept(s) with other designs. Possibility to edit such parameters will be extremely important too with the possibility to evaluate the changes in the inputs as a result of such changes. An independent and integrated approach to such representation should be considered to be important.

The most important conclusion that one draws from the above mentioned and loosely stated design principles is that design should not be viewed as the process that one follows, by analysis, evolution, or deduction, it is in fact the "concept" itself. Thus to predetermine the analysis, evolution, or deduction path that one needs to follow must be driven by the design itself. This takes into account the iterative nature of design but rules out the sequential nature.

From the above we also can now lay down the components that may be able to perform the tasks laid down, and the expectations from a design system. We must realise that this system must be integrated and independent and must consist of:

X-pert Object Oriented Knowledge Base
X-pert Path Inferencing Processor
Statistical Methods Processor
Sizing Processor
Analysis Processor
X-cessor and X-amine Processor
Graph Processor
Graphics Processor
Graphics User Interface

The above components must be integrated into a feasible design system. Following chapters present the detailed requirements of each part with the last chapter describing the integration.

Four

KNOWLEDGE BASE

(X-pert Object Oriented)

A knowledge base plays one of the most important roles in design. Just as an expert uses experience, gained during years of work as an aircraft designer, an aircraft design system such as the one proposed must have expertise stored in its domain of reference. Such a system is called the knowledge base. At the same time it is important to differentiate between a Database and a Knowledge base. A knowledge base is that subset of a database whose stored entities are used actively in inferencing statements and whose entities may be used to draw deductive conclusions about the species or subject that the knowledge base constitutes. Among the requirements of storage we also have the need to store raw data which is the primary goal of a database.

Once a design concept has been generated it remains unverified unless it is compared with existing designs. This is normally done with the use of an engineering database of existing designs. As the comparison progresses we find some concepts to be feasible and realistic, in reference to such designs it is preferred that they be added to the database. A database must by itself succeed in such a task. Another important situation comes when one wishes to input the characteristics of a design into the database, to be used as a reference later, a situation when the minimum inputs which define the existing aircraft are more than are available.

As we review the requirements of a database, we can make one type of characterisation about the stage of evolution of a database, this is important from the viewpoint of active search paths which need to be considered and directly reflects on the speed of such a search. Each of these is explained in short.

Core Storage
Synthetic Storage
Interim Storage
Project Storage
Insitu Storage

IN-SITU:

As the highest level of storage, this will be most accessible to the end-user and be most prone to manipulations. This will be generated in-situ during the progress of each conceptual design study. This storage will be the most easily destructible. The most important data it will contain will pertain to graphics and to graphs. Unless requested by the user the data will get destroyed at the termination of the program. This area in the data storage will also cater to the requirements of scratch-swap space.

PROJECT:

The user will normally begin the conceptualising exercise in this area. It will inherit some characteristics of interim and in-situ databases. This will also be erasable, but in

parts. The most important information this will store is the computational sequences that are requested by the XCSA. It will also be possible to save such information in the interim database.

INTERIM:

Interim data will contain that data which will be used for various specific projects in progress and which may be incomplete. Such data will be more specific and will be sufficient to restart the system in case of an aborted execution. It may be as detailed as the CORE but may not have statistical correlation's for each variable. This will be the data that may be protected by passwords due to its specific nature. The natural way to proceed to lower levels of data will be via the interim data. Furthermore the conversion to user accessible data format will be via this data level since the core or synthetic data will be in the binary format. The most important characteristic of such data will be because of its interfacing ability with the ASCII level data.

SYNTHETIC:

This will be exactly like the CORE with one exception that this will constitute the designs that are evaluated and conceptualized by the XCSA. As the name suggests, these designs will all be synthetic. The level of detail will be the same as in core, however, this will contain correlation's between the existing class of aircraft and the specific concept. Synthetic and Core will be reference databases in binary format to save vital storage space. Both will have high level of protection to maintain data security.

CORE:

Core must contain the data that has been input from known sources and is reliable. It will, by default, also include the data missing from the input which is the minimum required to define a design. This missing data will be calculated by the design program and will be marked respectively. Thus the data that is insufficient to design an aircraft to the level of detail that our system will be able to do will not be stored in CORE. As each aircraft is input, the stored values will contain not only the input data, and the back-calculated data but also will contain statistical correlation's between the these for the same concept. This database capability which we may call *learning* is further explained in the chapter on X-cessor and X-amine processor.

We must realise that data classification about the type of data is also very important. This may not be important directly in an object oriented database, but plays a prominent role in defining the hierarchical position of specific data type. These data, as each name suggests, will contain their respective type information on specific objects. The detail of information that will be stored will depend upon the status of evolution of the design and upon specific procedures under use. The data types that may be used for such a classification are:

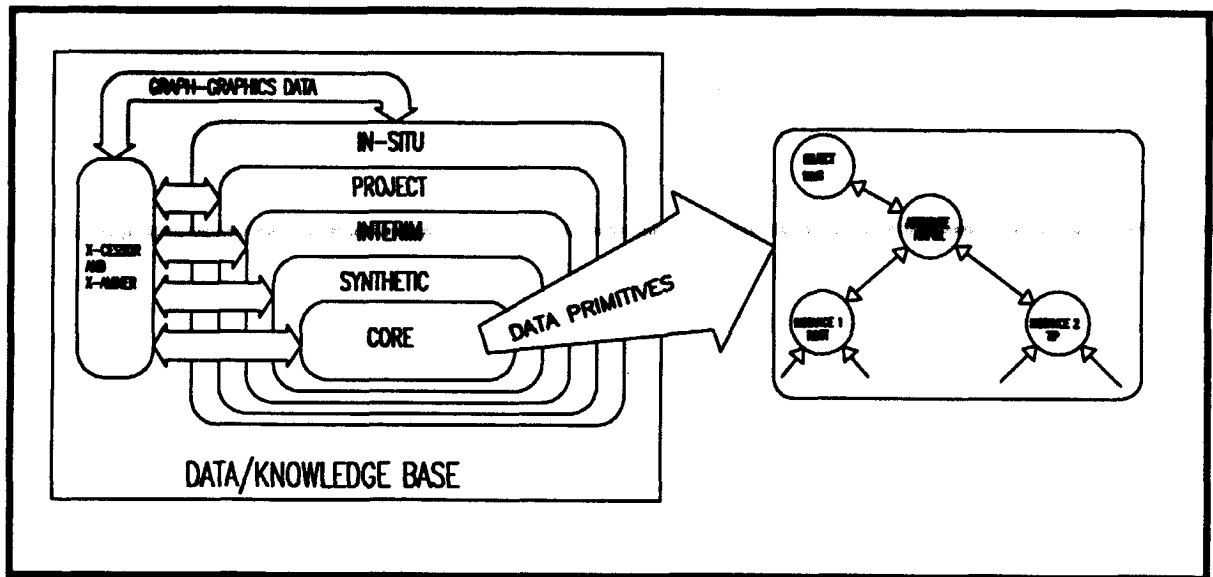
Geometry data
Aerodynamic Performance data

Engine Performance data
 Flight stability and Control data
 Economics data

As we develop the various classifications and sub-classes, we realise that the most important subject that crosses the mind is the basic nature of our database. We must choose our database to be of the object oriented type. Among the multitude of reasons that can be set forth, the most important is that object oriented search in a knowledge base has been found to be the most powerful among search methods used so far. It is the fastest for design oriented search in which object classification is paramount.

Let us start by assuming that we choose a Data Base Management System to accomplish our tasks. We land ourselves into problems of cross-platform-portability, software independence and all the associated problems that will come-by as the chosen DBMS is subsequently updated with new versions. Most of our application will only need to search the database, and for that, a full blown DBMS will be an overkill. All of the above are very important for a successful design system and thus force us to search alternatives to a standard DBMS.

Database libraries are used to custom-craft our database applications by providing us with a set of C or C++ routines in source format. These are used to build database blocks around the application that uses them and incorporates the lowest level access to database files. Thus they can be compiled and run as a part of the system itself. It is important to use such a database library in building our system. A very good review of some state-of-the-art libraries is given by Grehan et.al. in their, "Database Building Blocks."



X-PERT KNOWLEDGE BASE:

With the above discussion we can now picture our database. It is centered around the X-cessor and X-aminer processor. The embedded database is of the object oriented type as shown in the next sketch. The objects are manipulated with their attributes and instances.

None of the embedded database components can have cross-talk without the X-cessor coming into the picture. The X-cessor is controlled by the X-pert processor, which also controls the entire system's sequencing and execution.

The details about independent modules and their computational paths will evolve as the system advances.

Five

PATH INFERENCE PROCESSOR

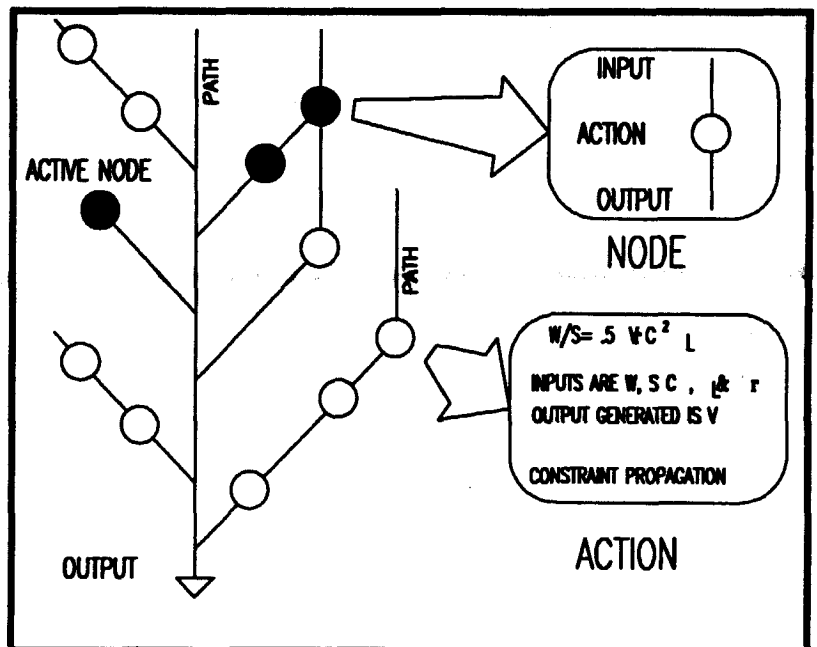
(Encapsulated)

If we have a system of equations that result in a solution from multiple inputs. We also have a set of incomplete inputs and incomplete outputs which are required to evaluate the set of equations. However, assume that the system is complete. This can be possible if the outputs of some equations can be considered to complete the set of other inputs. Further, assume that the order of the system is unknown. How is it that such a system will be evaluated?

Kroo, suggests an approach to solve such a problem using a quasi-procedural method. In his method the equations are sequenced in the proper order such that the results of preceding equations are used in the succeeding equations. Such a system can be very useful when we need to size the aircraft. Kroo's methodology is very detailed and thorough however it lacks one important aspect as has been suggested by Klob.

Klob approaches the problem of sizing with constraint propagation. This has been used to allow declarative statements to be understood as designating mathematical relationships among all the variables of the equation. This implies that one may use minimum number of equations on declarative form and internal representation will allow the system to decipher or invert the equation to evaluate the zero's of any unknown.

The most interesting aspect is that both of these approaches are novel, analogous and powerful if both of them are used together they will compliment one-another. We will make an attempt to have a system of path inferencing using constraint propagation and a quasi-procedural method with consistency maintenance. This will give us the powerful advantage of both methods. The path inferencing processor is encapsulated with the function library in one unit. This has a big advantage in terms of the system's speed. Once a new quasi-procedural function has been defined it will be stored in coded form which can be best handled by an encapsulated system.

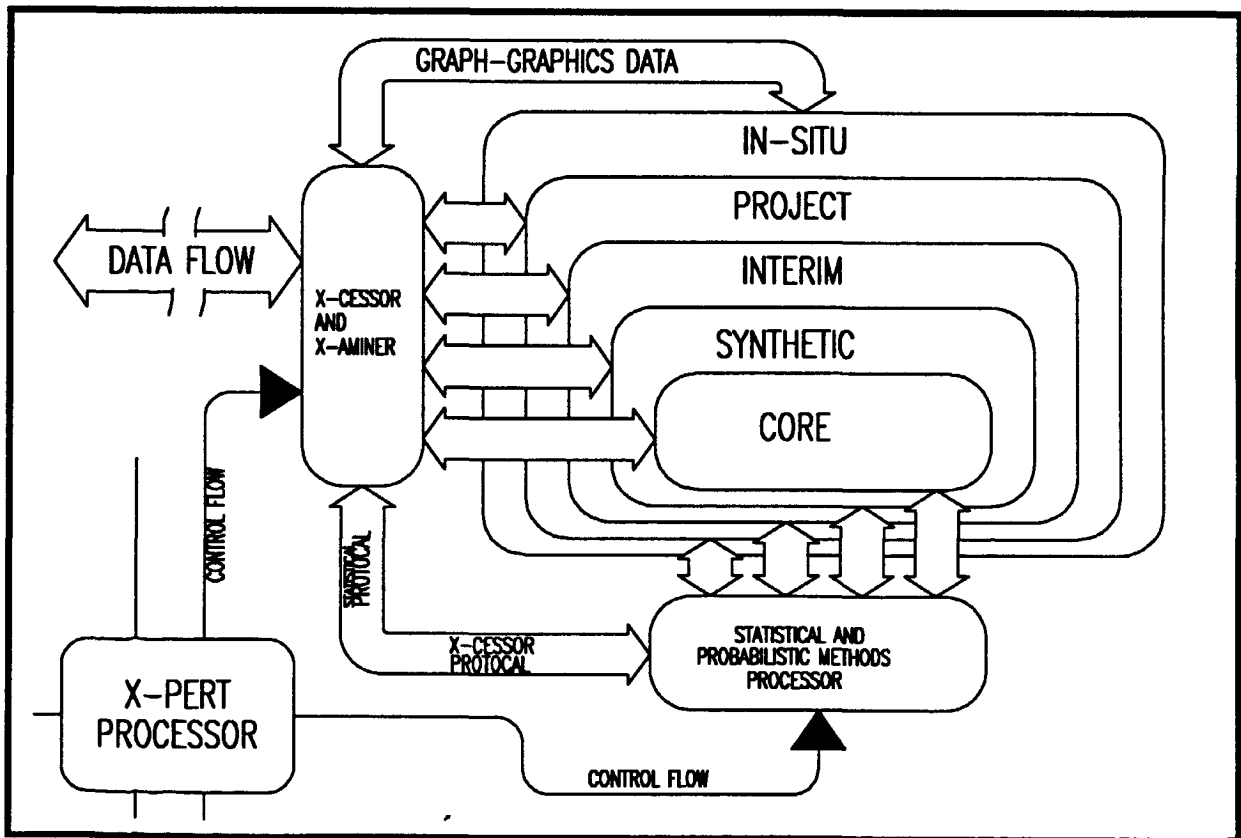


(Embedded)

For making comparisons within designs stored in the database and to make probabilistic estimations from the knowledge base one needs a statistical methods' library. During the design process, a sample extraction based on similar designs is also an important task of such a library.

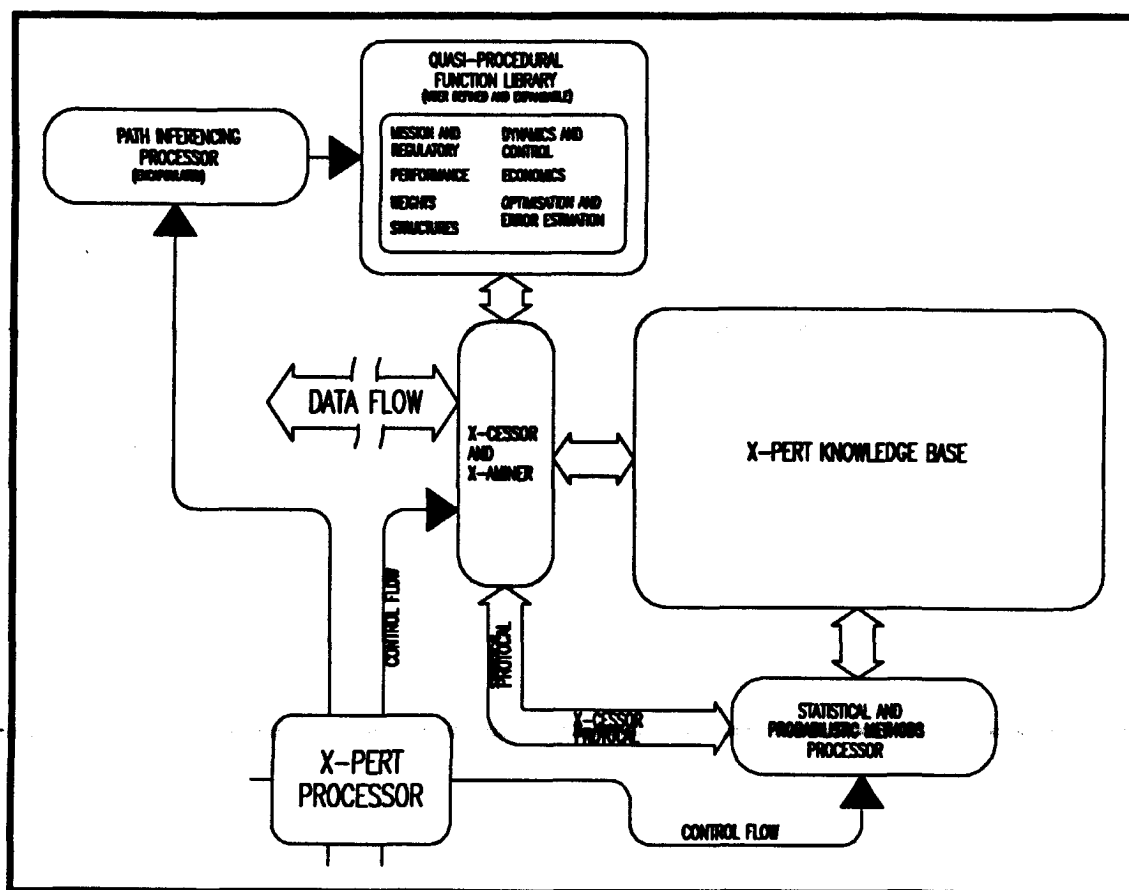
We are contemplating an embedded system of statistical and probabilistic routines which interface directly between the X-cessor and X-aminer, and the X-pert Knowledge Base. They will be responsible for generating the comparison data, estimation data and data for prediction. For the designs active in the in-situ database, and the project database, they will be able to sample design data from the rest of the database. Queuing of messages, control of the statistical processor and access routines will be done by the X-pert processor. Thus the statistical methods' processor must stay as a buffer between the access, examine and storage cycles.

In future, it is proposed that an estimation using fuzzy sets must be incorporated in the system. A suggested information flow diagram of the proposed system is presented here.



(Quasi-Procedural)

Sizing is the most important aspect of computational path. If an expert system shell is used it will be most beneficial in this area. The accuracy provided by these methods determines the accuracy of the design estimation. We have already proposed to use the quasi-procedural and constraint propagation approaches with a path inferencing processor integrating these approaches. An alternate method which can be considered to be equally powerful (if



not more) can be implemented with the use of powerful mathematical inferencing engines such as those used in off-shelf packages as mathematica. These can generate the required code and emulate a quasi-procedural method; it must be noted that quasi-procedural approach has its roots in the mathematical inferencing engine used in Macsyma,[©] which is essentially written in Lisp. Until a suitable quasi-procedural engine can be written, it is proposed to use off-shelf software as suggested; this will allow in testing the concept and in formulating the requirements of a more powerful tool for mathematical manipulations.

The sizing processor is subdivided into seven most prominent parts. These are in turn controlled by the X-pert processor and link directly with the X-cessor and X-aminer. They are modular and object oriented in their interaction with the Database entities. This is possible because of the inherent nature of the X-cessor and X-aminer which allow an object oriented

search and store.

Entities are treated as objects and each one can be influenced by all the modules of the analysis program. A symbolic representation of this approach is depicted in the next figure. Control and data flow orientation of the type suggested in the figure does not allow analysis routines to directly mate with the database. This prevents any spurious calculation to result in a change or evolution of a concept. This aspect is further elicited in the chapter on X-cessor an X-amine processor. Following are the most important sections on the sizing processor:

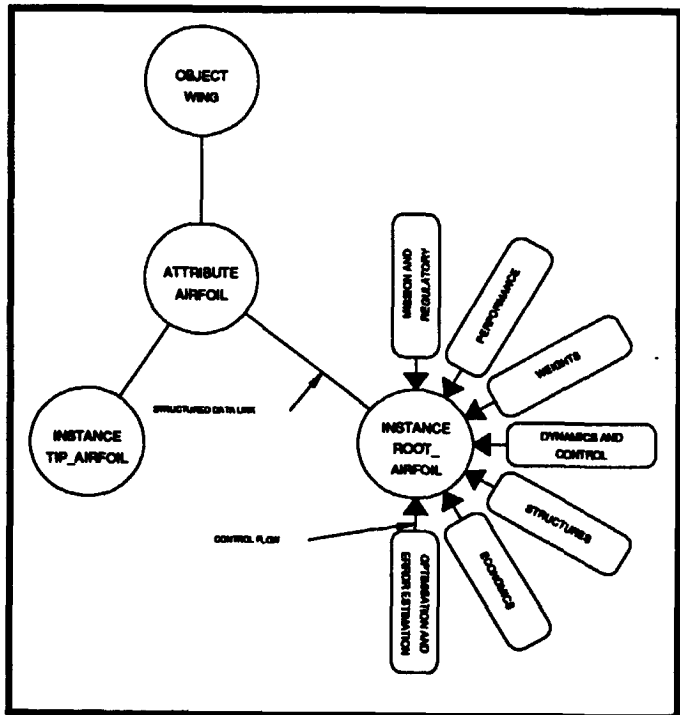
MISSION AND REGULATORY:

Requirements are most important in design and drive the design towards the goal of synthesizing several concepts to meet the laid down requirements. These can be expressed in loose jargon, like, "design an aircraft to carry five persons from A to B," or it may be, "design an aircraft, for the specified mission, optimised for the following stages..., etc.". The implication of detailed design requirement from the point of view of a designer is:

most of the important variables can be defined before design synthesis begins. These can be a limitation since the requirements limit the creativity of the designer. On the other hand, a very loose definition of requirement can overdrive the designer's effectiveness.

Even for highly generalised requirements, the system should be able to set a design mission and progress in design. It will be extremely advantageous if the system helps in the initial setup of requirements by asking intelligent questions from the designer. The question must be based upon forms and must link with an approach which an experienced designer may have taken. While requirements are input, it must be possible to input values in mixed units. Internal units' conversion must be transparent to the user and unless requested by the user the output units must be the same as the units in which inputs were specified. The necessity of this must not be undermined since this helps in bridging the gap between designers, operators and manufacturers, all of who address the requirements from different viewpoints and their decisions or preferences are equally important.

Regulations act as a sieve for requirements, thereby generating the constraints that are imposed on the design. For the same set of general requirements, aircraft designed for FAR's will have a different mission profile as that for JAR's or MIL-



SPEC's. The system should automatically draw out the mission profile that conforms to a specific requirement. Some times custom restrictions are placed by MIL-SPEC's for special purpose aircraft, such requirements come with their own sizing constraint functions and the resulting shapes may be rather different from what may normally be expected. Our system must be in a position to use its intelligence to synthesise such concepts too. A brute force method to execute this can be to work out each and every promising concept to the level of detail that suggests it's state as being infeasible, furthermore, the status report may contain the series of decisions that make the concept infeasible. A-priori routing of control flow by the user should allow the program to evaluate (or not to evaluate) the type of concept selected by the user.

PERFORMANCE:

This portion essentially consists of an aerodynamics and a propulsion combination. Our sizing routines will be based upon the reports of Prof. Torenbeek's methods quoted in the references. They are very easy on the computational effort and for initial estimates provide us with sufficient accuracy. Engine models of each class will be input to the system which will take into consideration sizing and growth factors.

Furthermore, this module will be able to perform operations in two modes, independently, and as a front end of its respective performance analysis modules. There will be complete transparency with regard to the user. The complexity of sizing routines will depend upon the stage of evolution of the design; taking the most current concept as the reference point. This, by default, implies that the system takes cross effects into consideration and thus a gradual updates of concept (object oriented) will proceed effortlessly.

WEIGHTS:

Clever selection of weights reduces the CG travel and allows a smaller static margin, thereby allowing more performance to be drawn from the same wing. Thus, we must provide some possibility of fast and accurate weight prediction; this can be accomplished by statistical or probabilistic means, empirical means, and by analytical means. Neither of these schemes give complete solutions and one must have a harmonious combination of each one of these. For the stage of sizing, an empirical scheme along with a statistical scheme will be sufficient but as the concept evolves it will be necessary to look into analytical schemes too.

DYNAMICS AND CONTROL:

Sizing routines for dynamics and control are equally important; these allow sizing to take dynamic requirements into consideration. With the evolution of Fly-by-Wire-flight-Control-Systems, FWCS, it has steadily become more important to size

aircraft based on dynamic requirements.

Most procedures used in these estimations are empirical in nature and we will also make use of such procedures to size the aircraft from the stability and control aspects. Basic criteria, e.g., stability margins will be used in the sizing module. This area too will have an inverse capability, i.e., for sizing, it will be possible to specify a dynamic requirement, and from general requirements based sizing we will be in a position to derive a dynamic criterion that the aircraft will satisfy.

STRUCTURES:

From the requirements it must be possible to determine the proper loading diagrams, structural flight envelopes, etc., these are of great importance in this module. Entire structural sizing procedure bases its estimates upon the structural flight envelope that it has to meet. The flight envelope too keeps changing due to requirements, regulations, etc.

Statistical estimations with empirical corrections have been used so far for such sizing routines and they provide sufficiently accurate results. However, we will also have the possibility to use simple beam theory, shell theory, expressions to determine the structural strength. A facility to correlate it with the weight module will also be available. When the concept evolves to sufficient accuracy, it will be possible to output its results into a FEM module, e.g., the DATADECK of MSC Nastran.

At the present level, it may not be practical to use anisotropic (or quasi-isotropic) materials in their truly mathematical form of representation, however, an increased strength level due to the use of advanced materials can be reflected by the overall increase in strength or overall reduction in weight for the same size. We will use this simple methodology for structural sizing mechanisms. Use of flutter criteria of any kind for strength sizing is far too advanced for our level of analysis thus it will not be delved into.

ECONOMICS:

Most of the important estimates are concerned directly with our optimising mechanisms. Procedures for standard statistical, and empirical evaluations will be used here for detailed calculations of DOC, cost per passenger seat per mile (for individual operators along with the facility to implement the cost functions used by the operators. An attempt will be made to estimate the cost of new design development; development of a class of aircraft based upon an existing design, e.g., jetliner, regionaliner, etc.

We will attempt to implement the prediction of tooling costs, project feasibility based upon requirements, sale, growth of traffic, growth of facilities, increase in living cost index, expected sector lengths in future, etc. Most of the requirements will be dictated by the designer and thus the designer will have almost complete control over the economic predictions.

An interesting variation of the theme of economic modelling is that the

changes in the design at any level of development will force it into an economic sizing cycle, as a result, reflecting the effect that the change in design will have on change in economic status of the concept. This should be considered to be one of the important advantages of using the object oriented approach, with quasi-procedural scheme.

OPTIMISATION AND ERROR ESTIMATION:

The problem of conceptual design is extremely complex, especially since its global optimum is either not known or it is very difficult to determine due to the high dimensionality of search space. Hence, we need sophisticated optimisation tools, the performance of which is measured by the average quality of best answer that can be computed in a fixed amount of time.

Bramlette and Cusic, have tested seventeen such methods which have been compared for the specific task of aircraft conceptual design, out of which five algorithms are discussed by them. According to them, better solutions may be possible from stochastic iterated genetic hill climbing (SIGH), which is an algorithm combining the elements of other methods.

It is highly recommended that further work be done in the modeling of optimisation schemes and a suite of several such schemes be available for the design system. This will certainly increase the versatility of the system since it will now be possible to use optimisation and search method tuned (or biased for that matter), to our particular design.

It is further required to have an intelligent assignment of error-values which may allow separate searches/optimisations in separate areas.

QUASI-PROCEDURAL WITH CONSTRAINT PROPAGATION:

Quasi-procedural approach is being used in the sizing processor, this implies that the various procedures that need to be executed shall be in form of loose, compiled and independent routines. These will be available in a library; the path generation program will instantaneously generate a code which will execute the procedures in the required manner performing consistency check for each one of them. This approach proposed by Kroo will be complimented by using constraint propagation in each of the procedures. This implies that each one of the procedures will be treated as a mathematical function instead of as an assignment statement (as is the case in most programs).

EXPANDABILITY:

The user will have the freedom to assign alternate procedures: these will also be treated in their implicit forms. These external procedural relations will allow expandability of the system without hampering the ease of usage. The external procedural relations will also assist in customising the system for specific usage. Expandability will also extend to analysis routines and the same transparency of procedures from such external links will exist.

ANALYSIS PROCESSOR

The analysis processor, in essence, will be a sort of back-end of the sizing processor; just suited for the high end analysis routines which may use off-shelf packages. Thus the analysis processor will be in essence an interfacing unit between analysis packages and the XCSA.

The most important of these will be the preliminary design system, ADAS; the functionality of ADAS in design optimisation will compliment the XCSA system. ADAS links automatically with CAD systems like AutoCAD, panel code like 0-215 NLR, FEM analysis code like B3000, etc. This will be exploited by the XCSA system once the level of maturity of the solution warrants high level analysis mechanisms.

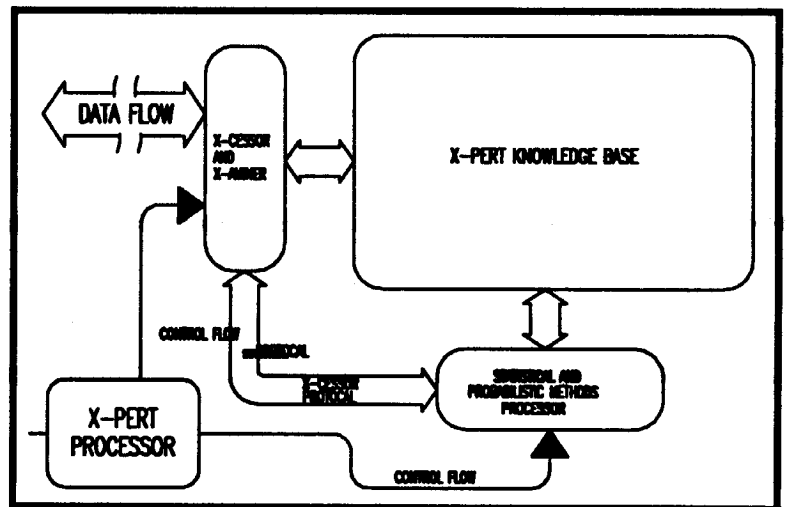
An option for the future must contain a general interface with FEM codes like MSC Nastran, nonlinear panel codes with wake relaxation like the VSAERO, stability and control systems, material databases, etc.

(Probabilistic)

In conceptual design, recourse is taken to existing systems, available knowledge, and available expertise. This is in order to hasten the complicated process by providing best guess estimates, subsequently narrowing down the search domain. Present day representations of such knowledge utilises the database technology. We have already elicited upon the importance of both databases and their intelligent counterparts, the knowledge-bases. In this section we throw some light on the importance and usage of statistical methods along with databases in our system.

Statistical and probabilistic methods' processor will be an important part of our system, especially since it acts as a buffer between the x-pert processor and the knowledge-base. The adjoining figure gives the detail control flow and data flow schemes.

The x-pert processor controls both the x-cessor and x-aminer processor, and the statistical processor. The importance of the shown structure can be explained with regard to data operations as follows:



DATA EXTRACTION:

Suppose the system has to perform a data extraction operation, the route followed by the data will depend upon the status of data update level, the type of data, e.g., synthetic data on extraction will almost never pass through the statistical processor. The data that most certainly will pass through the statistical processor will be one that is requested by the quasi-procedural library. This data will be compared with the existing entries in the quasi-procedural library.

DATA-STORAGE:

All the data that will be stored will necessarily go through the phase of statistical processing, this is to make certain that spurious data does not enter the system. Furthermore the statistical and probabilistic processor in conjunction with the x-cessor and x-aminer processor will give valuable inputs about the correctness of input data.

(Embedded)

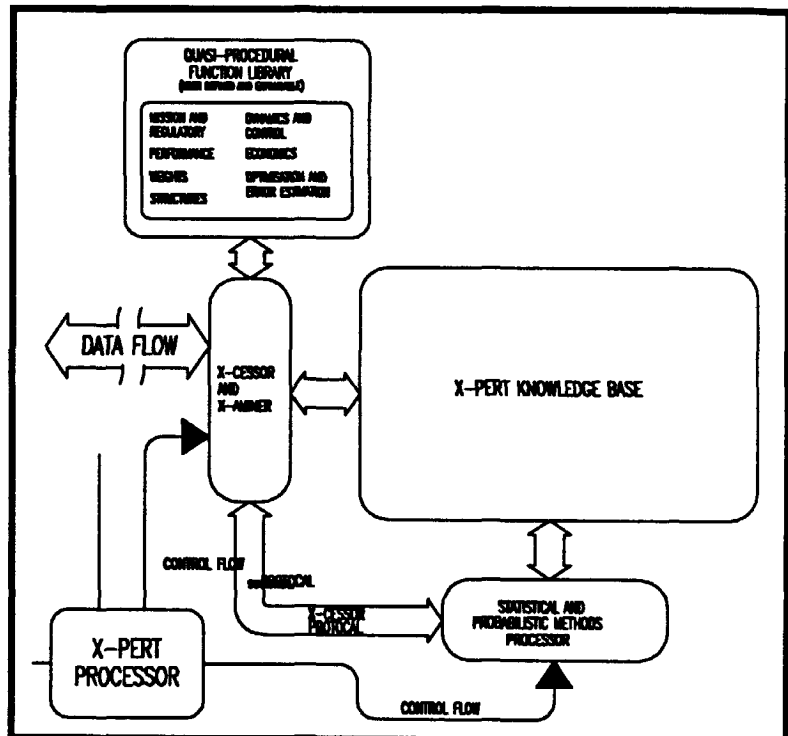
The x-cessor and x-amine processor are one of the components of our system which provide our system the capability of being artificially intelligent. Here, we will list the requirements of our system. The x-cessor and x-amine processor are that section of our system that has the following primary functions:

- 1 Labelling of data
- 2 Routing of data
- 3 Retrieval of data
- 4 Data Storage

LABELLING OF DATA:

This unit performs the task of a data gateway. Thus it is here that the data is first examined for its correctness and validity. If the data is presently being processed by the sizing and analysis modules, it will be labeled at a different level as compared to

data that is stored in the knowledge base as reference data. The decisions about the update level of data is taken by this module. The rules of labeling will develop as we progress along with the design system, however, at this stage an example can be given as follows; if we wish to design an entirely new type of aircraft, in the mission requirements we give some data. This data is immediately passed via the data flow stream to the x-cessor and x-aminer for checking its completeness. The x-aminer will realise that the data is un-labeled and will label it as new project data and further modules will respectively process the data.



ROUTING OF DATA:

From labelling of data, we move to its routing. There are three routes that our data can take, to-and-fro from the user interfaces, the output devices and the input devices. Second route can be from the sizing and analysis processor, and the third is to-and-fro from the database. The path from the database is divided into two portions, one goes directly to the database and one follows the path through the statistical processor. The route that the data takes will be decided by the x-cessor and x-aminer processor with certain inputs coming in from the x-pert processor. It is beyond the scope of this report to expand in detail on the reasons for the particular choices of data

path and that of these particular configurations. However, it was realised that this path breakdown has several advantages of which only the most important are as follows; ability of the system to be able to run on its own to automatically update its data. i.e., to run the existing the data through mock runs by the analysis processor; data transparency between the user input data and the system data; data security.

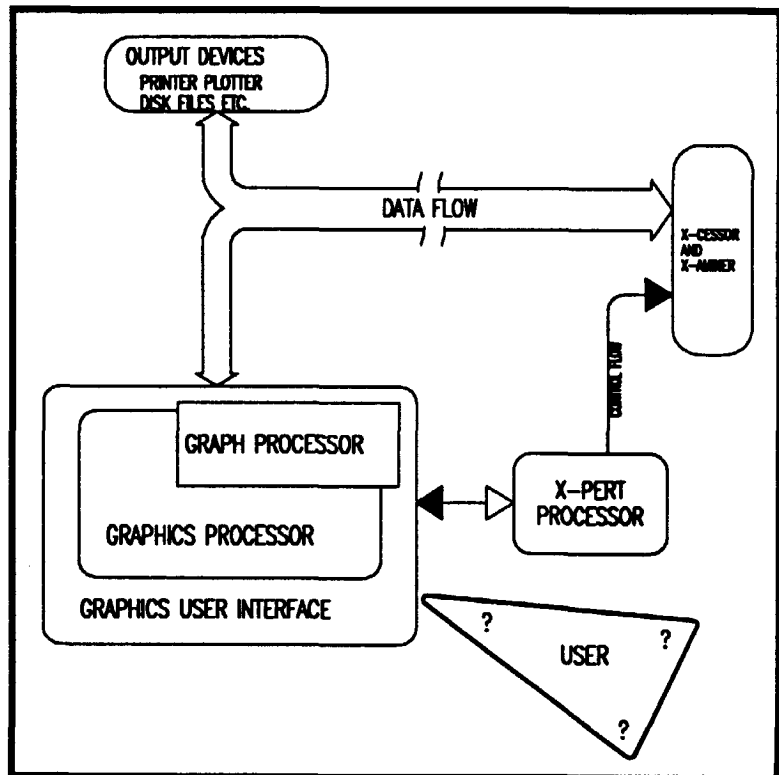
DATA RETRIEVAL AND STORAGE:

Data retrieval or storage is another interesting aspect of this break down, it can be accomplished via three paths and the x-pert processor can independently provide for the required control inputs for each path. The data of highest confidence level, that is required by a pre-verified user can be directly accessed by the x-cessor and x-aminer from the knowledgebase. In case the data is requested by a sizing module then most probably it will pass through the statistical and probabilistic processor. In case it does so, then the data passes through the x-cessor and x-aminer via a statistical protocol,

(Graph and Graphics)

As we had seen earlier, we realise that to make our system easy to use and acceptable for a wide spectrum of users, there must exist some form of buffer between the user and the design system.

The graphics user interface as proposed here, serves the purpose adequately. Though this may seem to be an unnecessary addition, however, its importance must not be underestimated. In the process of conceptual design, we normally work at a relatively high error level, where design starts with *guesstimates* here the accuracy of results is of lesser importance as compared to the sequence of operations that generated the result. Such results are difficult to distinguish and to appreciate in fine detail. Hence it is preferred to have with us, a method of representing the data to the designer in the most compact representation. The logical choice turns to graphics output devices. These not only provide an approximate representation that is easy to grasp but they also can be used to input approximate values of input data from bars or sliders, to present a less confusing interface to the user, etc.



These not only provide an approximate representation that is easy to grasp but they also can be used to input approximate values of input data from bars or sliders, to present a less confusing interface to the user, etc.

The graphics user interface will contain apart from textual interface, suited for ASCII inputs, a graphics processor and a graph processor. Thus the GUI will be merely a shell for it's three component processors. The ASCII mode will be the basic GUI and graphics and graph will be able to operate as popup utilities from within the GUI. Following descriptions about the proposed system are noteworthy:

THE GRAPHICS USER INTERFACE:

The basic purpose of this interface is to shield the user from the intricacies of a complicated program. To provide with complete functionality of the system, this type of user interface must consist of at least some of the available input devices. In our case we must consider to add the functionality of a graphic user interface with the facility of both keyboard's and mouse's inputs. The type of data that is processed by this kind of interface can be broadly subdivided into firstly textual input and output, which may

further be divided into plain textual input, table type textual input and form type textual input, and secondly into graphic input and output. Due to the complexity of the graphic process we have divided it into two sections which will be subsequently explained.

The plain textual input, which in some systems is referred as *command line input* plays the role of most important input, this gives the root level commands that control the system, provides most of the starting parameters and finally is the only way to surpass the graphic user interface hierarchy. A menu structure which will suit our particular process will be stored and available from this level.

In some applications, hypertext, or form's, are very beneficial. Most of them are used in providing options required from the user, these normally pop-up at the point when such inputs are necessary. For a system that is truly user-interactive such a facility is important.

Input by tables is an accepted method with the databases, spreadsheets, data files, etc.; our system must mesh with a number of tools, like analysis programs, must be able to receive and give data related to databases, in such a case it is imperative that it contains input facilities by tables or spreadsheets and the like.

THE GRAPHICS PROCESSOR:

Embedded in the GUI, the graphics processor, executes as a popup utility when CAD usage is deemed necessary. This may be considered to be a full fledged CAD package designed with a bias toward conceptual aircraft design. There are several reasons to design a CAD package for our use, some of which have been explained in the second section. In this section we will devote ourselves more to the modelling aspect of the graphics processor.

Creation and manipulation of objects, entities, and their instances in the graphics area will be the main function of the graphics processor. The representation will consist of sufficient detail which is necessary for adequate data translation between various analyses and CAD packages. Furthermore, it must have the facility to model aerodynamic objects with considerable ease. The representation must be accurate and must be amenable to parametric variation. The capacity of parametric variation will allow one model representation to represent a wide variety of objects without loss of excessive storage area.

This processor will mature as the project progresses, however, it is best to design it after the major design system is complete. Parallel work can progress in the definition of menu structure and the type of representation of objects, since the same representation will eventually be used in the database.

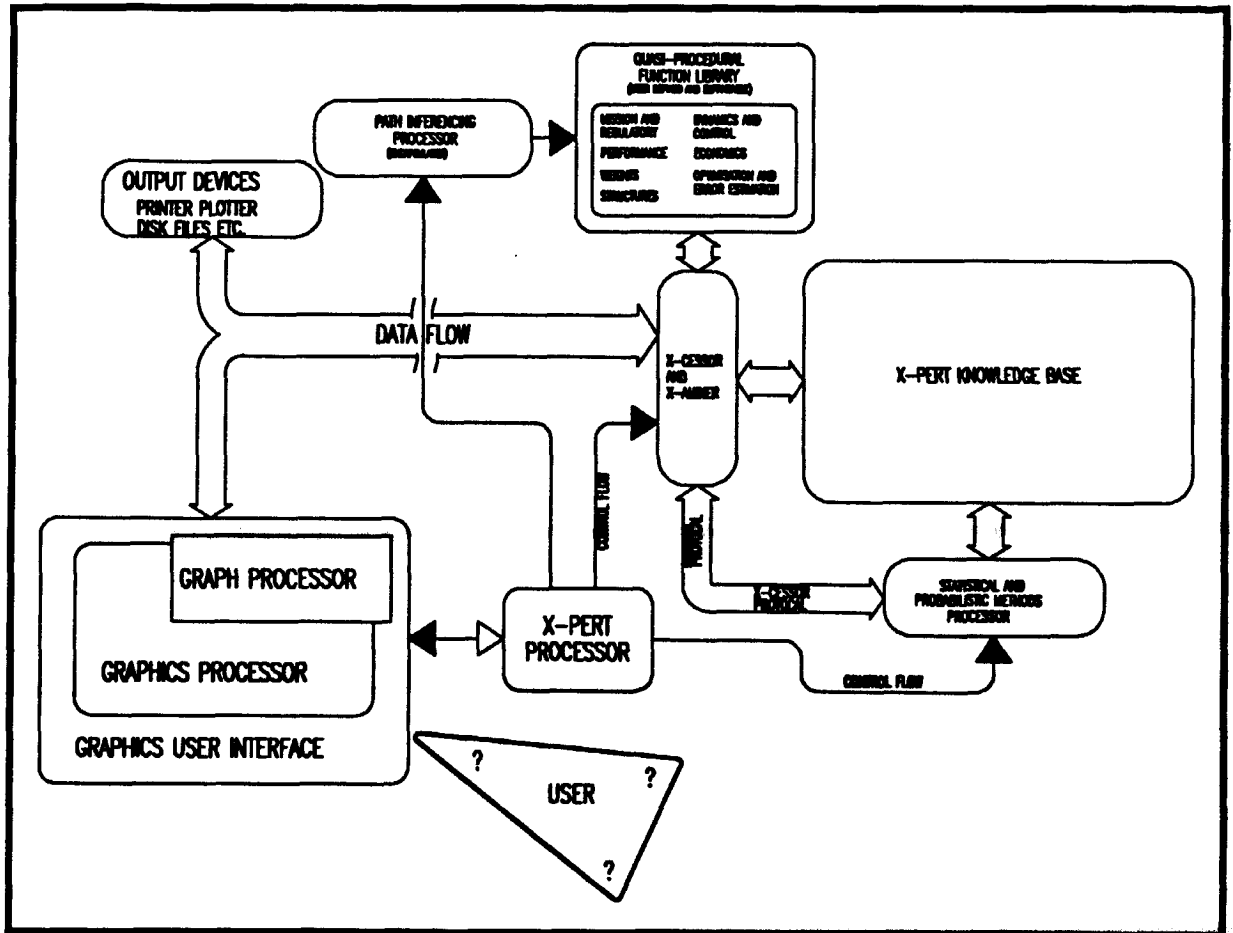
THE GRAPH PROCESSOR:

The graph processor is also a component that is embedded into the GUI as a popup utility. One may question the need of having two independent units like graphics and graph processor? The graph processor gives a specialised type of

representation which is most needed while graphics operations are in progress, for instance to review the changes in the thrust to weight ratio with the change in diameter of fuselage, etc. The only practical way to achieve the previous functionality, we need a graph processor. Now to the question, why can not the graphics unit serve the same purpose, the answer to this is that using the graphics processor for the same will be an over kill for the required function.

With the advent of computer graphics such facilities are feasible and realisable within reasonable period.

(The integrated design system)



A simplified block diagram of our system illustrates most of the integrating paths. The x-pert processor being the one that controls the entire system is certainly the most important among the rest of the subsystems.

X-pert processor consists of artificial intelligence principles, which are used to facilitate its functions; most notable among them are:

- 1 Routing of control flow protocols to various sub modules
- 2 Decision on control transfer to other sub modules
- 3 Observation of the entire process status via the GUI

The system essentially acts upon objects and instances, by giving different instances to same object and by saving the instance with most recent update level. Thus it is imperative that each of the objects must be processed individually. The method of processing individual objects is scheduled by the x-pert processor.

There can be times when some sub modules may demand attention of other sub modules simultaneously; the x-pert processor resolves such conflicts and addresses the problem of such conflicting protocols. It also acts as the scheduler of jobs to the independent modules for auto updates and design of the system. This type of configuration has the power to design, compare existing designs and to create its design database, automatically. This facility gives the x-pert processor the ability to act as a designer, or rather simulate one, to proceed in its tasks. For our system, it must be able to provide the required answers, inputs, make observations about the on-going design process, without interference from the user.

(The Future?)

As of today, such a system is not known in the aircraft design literature and thus it is all the more important to work on it. We are not attempting to create a clone of a designer, but to make a more intelligent design system that will be fast, accurate, intelligent, and will take most tedious jobs from the hands of an actual designer. We must not also forget that such a system will be fun to use!

The purpose of this report was to show how a system can be built that would use the concepts of artificial intelligence, smart databases (knowledgebases), artificial intelligence optimisation techniques, etc. The proposed system requires several inputs and is an enormous task for an individual. There are two paths that one may take, namely, to research on one aspect of the design system in detail or to research in detail on the entire system by working out in partial detail individual modules. The first choice makes one run the risk of an advanced subsystem with almost negligible testability. With the second we fear an overall compromise. We will, however, prefer to work on the entire system as this provides all the answers to our professional curiosity, "can a system be designed which is ..."

For furthering this work, it is preferred to have the following system available for a wide period of study with the following software:

- | | |
|----------|---|
| 1 | <i>SPARC compatible workstation</i> |
| 2 | <i>C++ compiler, with object-classes tools</i> |
| 3 | <i>Software development tools for OpenLook interface</i> |
| 4 | <i>C++ object oriented, database library routines (db Vista)</i> |
| 5 | <i>Phigs library with development tools</i> |
| 6 | <i>Mathematica or Macsyma, on SPARC</i> |

The above mentioned facilities at the end of three years will, hopefully, result in a design system that we foresee.

REFERENCES

- 1 **Ackley David H.**, "A Connectionist Machine for Genetic Hillclimbing," Kluwer Academic Publishers, 1987.
- 2 **Alsina J., et.al.**, "Progress towards a commercial aircraft design expert system", Computer Applications in Aircraft Design and Operation (editors, Murthy T. K. S. and Fielding J. P.), Springer-Verlag, 1987.
- 3 **Andersen David M. and Shrewd Bruce A.**, "State-of-the-art, Portability and the GUI", BYTE Nov. 1991, pp 221-226.
- 4 **Barnum J., et.al.**, "Advanced Transport Design Using Multidisciplinary Design Optimisation", AIAA-91-3082, Sept 1991.
- 5 **Bil C.**, "Development and application of a computer-based system for conceptual aircraft design", PhD Thesis, Delft University Press, Delft, The Netherlands, 1988.
- 6 **Bil C.**, "Aircraft Design and Analysis System (ADAS) Users Manual", Handleiding LR-110, Faculty of Aerospace Engineering, TU Delft, The Netherlands, May 1992.
- 7 **Bouchard E. E.**, "Concepts for a Future Aircraft Design Environment", AIAA-92-1188, Feb 1992.
- 8 **Bramlette Mark F. and Cusic Rod**, "A Comparative Evaluation of Search Methods Applied to Parametric Design of Aircraft", Proceedings of The Third International Conference on Genetic Algorithms (editor, Schaffer David J.), June 1989.
- 9 **Buckley M. J., et.al.**, "Design Sheet: An Environment for Facilitating Flexible Trade Studies During Conceptual Design", AIAA-92-1191, Feb 1992.
- 10 **Cebeci T., et.al.**, "An Approach to the Design of Wings; The Role of Mathematics, Physics and Economics", AIAA-92-0286, Jan 1992.
- 11 **Chalfan Kathryn M.**, "A Knowledge System that Integrates Heterogenous Software for a Design Application", The AI Magazine, Summer 1986, pp 80-84.
- 12 **Conklin Jeffrey and Richter Charles**, "Support for Exploratory Design", , 85-5090, pp 238-241.
- 13 **Dener C. and Hirsch Ch.**, "IGG - An Interactive 3D Surface Modelling and Grid Generation System", AIAA-92-0073, Jan 1992.
- 14 **Dhingra A. K. and Rao S. S.**, "Application of Fuzzy Theories to Formulation of Multi-Objective Design Problems", AIAA-88-4430, Sept 1988.

- 15 **Elias Antonio L.**, "**Knowledge Engineering of the Aircraft Design Process**", **Knowledge-Based Problem Solving** (editor, Kowalik J. S.), Prentice-Hall, 1986.
- 16 **Elias Antonio L.**, "**Computer-Aided Engineering: The AI Connection**", **Astronautics and Aeronautics**, Jul/Aug 1983.
- 17 **Evans S. B., et.al.**, "**Design and Development of a Generic Graphical User Environment**", AIAA-92-0070, Jan 1992.
- 18 **Fulton R. E. and Yeh Chao-pin**, "**Managing Engineering Design Information**", AIAA-88-4452, Sept 1988.
- 19 **Goedhart Bob and Swaan Arons H. de**, "**Delfi3 User Manual**", Faculty of Technical Mathematics and Computer Science, Delft University of Technology, The Netherlands, Apr 1991.
- 20 **Grehan Rick, et.al.**, "**Database Building Blocks**", BYTE Jan 1992, pp 204-224.
- 21 **Haberland C, et.al.**, "**Computer-Aided Conceptual Aircraft Configuration Development by an Integrated Optimisation Approach**", ICAS-90-2.6R, Sept 1990.
- 22 **Harrell T. L.**, "**The Design of a Sport Aircraft Configured to Emulate Jet Fighter Characteristics**", AIAA-90-3244, Sept 1990.
- 23 **Hollowell S. and Bitten R.**, "**Multidisciplinary Optimisation of Conceptual Aircraft**", AIAA-92-1196, Feb 1992.
- 24 **Jayaram S., et.al.**, "**ACSYNT - A Standards-Based System for Parametric Computer Aided Conceptual Design of Aircraft**", AIAA-92-1268, Feb 1992.
- 25 **Kidwell G. and Eskey M.**, "**Expert Systems and Their Use in Augmenting Design Optimisation**", AIAA-85-3095, Oct 1985.
- 26 **Kiss Peter A.**, "**Standards for Artificial Intelligence**", AIAA-92-1584, Mar 1992.
- 27 **Klob Mark A.**, "**Constraint-Based Component-Modelling for Knowledge-Based Design**", AIAA-92-1192, Feb 1992.
- 28 **Kolb Mark A.**, "**A Flexible Computer Aid for Conceptual Design Based on Constraint Propagation and Component-Modelling**", AIAA-88-4427.
- 29 **Kroo I. M.**, "**An Interactive System for Aircraft Design and Optimisation**", AIAA-92-1190, Feb 1992.
- 30 **Kroo I. and Takai M.**, "**A Quasi-Procedural, Knowledge-Based System for Aircraft Design**", AIAA-88-4428, Sept 1988.

- 31 **Levy David M.**, "Prediction of Average Downwash Gradient for Canard Configurations", AIAA-92-0284, Jan 1992.
- 32 **Maughmer M. D. and Sommers D. M.**, "Figures of Merit of Airfoil/Aircraft Design Integration", AIAA-88-4416, Sept 1988.
- 33 **Munck Robert G.**, "Toward Large Software Systems that Work", AIAA-CP-858, Paper 85-5057, Oct 1985, pp 96-99.
- 34 **Newman D. and Stanzione K.**, "Aircraft Configuration Design Code Proof-Of-Concept: Design of the Crewstation Subsystem", AIAA-91-3097, Sept 1991.
- 35 **Paisley D. J., et.al.**, "The Aerodynamic Assistant", AIAA-CP-8910, Paper 89-3132-CP, Oct 1989, pp 1052-1062.
- 36 **Plant Robert T.**, "The Verification and Validation of Knowledge Based Systems", , 89-2982-CP, pp 150-156.
- 37 **Pouraghabager Reza, et.al.**, "Concurrent Engineering - into The Curriculum", SAE 912102, Sept 1991.
- 38 **Rand R. H.**, "Computer Algebra in Applied Mathematics: An Introduction to MAC-SYMA", Pitman Advanced Publishing Program, 1984.
- 39 **Ricci P. and Hale J.**, "Cost-Conscious Concurrent Engineering", AIAA-91-3152, Sept 1991.
- 40 **Rogers J. L.**, "The Potential Application of the Blackboard Model of Problem Solving to Multidisciplinary Design", AIAA-CP-8910, Paper 89-3071-CP, Oct 1989, pp 686-695.
- 41 **Roskam J. and Gomer C.**, "A Revolutionary Approach to General Aviation Airplane Design", AIAA-91-3126, Sept 1991.
- 42 **Roskam Jan and Malaek Seyyed**, "Automated Aircraft Configuration Design and Analysis", SAE-891072,
- 43 **Roskam Jan**, "Airplane Design Parts I to VIII", Roskam Aviation and Engineering Corp., 1989.
- 44 **Rowell L. F., et.al.**, "Software Tools for the Integration and Execution of Multidisciplinary Analysis Programs", AIAA-88-4448, Sept 1988.
- 45 **Ryan Bob.**, "State-of-the-art, Artificial Intelligence, AI's Identity Crisis", BYTE Jan 1991, pp 239-246.

- 46 Schank Roger C., "What is AI Anyway?", *The AI Magazine*, Vol 8, no 4, 1987.
- 47 Schildt Herbert, "C++ The Complete Reference", Mc Graw-Hill, 1991.
- 48 Sibley Egdar H., "Functional Analysis and Simulation of Complex Systems Involving Large Information Systems - A Fusion of Embedded System and Competitive Advantage Concepts in the Design Process", AIAA-CP-8910, Paper 89-3114-CP, Oct 1989, pp 928-935.
- 49 Steele Guy L. (Jr.), "Common Lisp The Language", 2nd Ed., Digital Press, 1990.
- 50 Stefik Mark and Bobrow Daniel G., "Object-Oriented programming: Themes and Variations", *The AI Magazine*, pp 40-62.
- 51 Sun Microsystems Inc., "OPENLOOK Graphical User Interface Application Style Guidelines", Addison-Wesley Publishing Company Inc., Jun 1990.
- 52 Sun Microsystems Inc., "OPENLOOK Graphical User Interface Functional Specification", Addison-Wesley Publishing Company Inc., Jul 1990.
- 53 Swaan Arons H. de, "Delfi - Design, development and applicability of expert systems shells", PhD Thesis, Delft University Press, Delft, The Netherlands, Nov 1991.
- 54 Swift Richard A. and Batill Stephan M., "Damage Tolerant Structural Design Using Neural Networks", AIAA-92-1097, Feb 1992.
- 55 Tong S. S., "Design of Aerodynamic Bodies Using Artificial Intelligence/Expert System Technique", AIAA-85-0112, Jan 1985
- 56 Tong S. S., et.al., "Integration of Artificial Intelligence and Numerical Optimisation Techniques for the Design of Complex Aerospace Systems", AIAA-92-1189, Feb 1992.
- 57 Torenbeek E., "Introduction to Preliminary Aircraft Design Volume 1", Handleiding LR-102, Faculty of Aerospace Engineering, TU Delft, Holland, Apr 1988.
- 58 Torenbeek E., "Fundamentals of Conceptual Design Optimisation of Subsonic Transport Aircraft", Report LR-292, Faculty of Aerospace Engineering, TU Delft, The Netherlands, Aug 1980.
- 59 Torenbeek E., "On the conceptual design of subsonic transport aircraft for cruising flight optimised for different merit functions", Report LR-451, Faculty of Aerospace Engineering, TU Delft, The Netherlands, Dec 1985.
- 60 Torenbeek E., "Introduction to Preliminary Aircraft Design Volume 2", Handleiding LR-103, Faculty of Aerospace Engineering, TU Delft, The Netherlands, Apr 1988.
- 61 Torenbeek E., "The initial calculation of range and mission fuel during conceptual

- design*", Report LR-525, Faculty of Aerospace Engineering, TU Delft, The Netherlands, Aug 1987.
- 62 **Torenbeek E.**, "Synthesis of Subsonic Airplane Design", Delft University Press, Delft, The Netherlands, 1981.
- 63 **Trainer A. and Gabriele G.**, "Computer-Aided Conceptual Design of Rotorcraft", AIAA-91-3099, Sept 1991.
- 64 **Ullman Ellen**, "State-of-the-art, You can't run on everything", BYTE Nov 1991, pp 255-264.
- 65 **Ullman Ellen**, "State-of-the-art, Developing Applications Across Platforms, Let the system do the porting", BYTE Jan 1992, pp 191-199.
- 66 **Van Laarhoven P. J. M. and Aarts E. H. L.**, "Simulated Annealing: Theory and Applications", Kluwer Academic Publishers Group, Dordrecht, Holland, 1987.
- 67 **Vaughan-Nichols Steven J.**, "State-of-the-art, Transparent Data Exchange", BYTE Nov 1991, pp 211-216.
- 68 **Volk J. A.**, "Multidisciplinary Design Environment Development for Air Vehicle Engineering", AIAA-92-1113, Feb 1992.
- 69 **Vucinic D., et.al.**, "CFView - An Advanced Interactive Visualization System based on Object-Oriented Approach", AIAA-92-0072, Jan 1992.
- 70 **Wampler S. G., et.al.**, "Improving Aircraft Conceptual Design- A PHIGS Interactive Graphics interface for ACSYNT", AIAA-88-4481.
- 71 **Wang A. James and Miecznikowski Jan S.**, "Crew Station Design and Optimal Analysis Using AI", AIAA-CP-8910, Paper 89-3097-CP, Oct 1989, pp 843-849.
- 72 **Yang Hunik and Hoeltzel David H.**, "Automatic Finite Element Mesh Generation for the Automation of Parametric Conceptual Design", SAE 912219, Sept 1991.
- 73 **Zadeh L. A.**, "A Formalization of Commonsense Reasoning Based on Fuzzy Logic", AIAA-CP-858, Paper 85-5044, Oct 1985.



Memorandum 665



60142051352