AIAA 2002- 5613

An Extensible, Interchangeable and Sharable Database Model for Improving Multidisciplinary Aircraft Design

Risheng Lin and Abdollah A. Afjeh
The University of Toledo
Toledo, Ohio

9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization
4 - 6 September 2002
Atlanta, Georgia

AN EXTENSIBLE, INTERCHANGEABLE AND SHARABLE DATABASE MODEL FOR IMPROVING MULTIDISCIPLINARY AIRCRAFT DESIGN

Risheng Lin* and Abdollah A. Afjeh†
The University of Toledo
2801 West Bancroft Street
Toledo, Ohio 43606, USA

ABSTRACT

Advances in computer capacity and speed together with increasing demands on efficiency of aircraft design process have intensified the use of simulation-based analysis tools to explore design alternatives both at the component and system levels. High fidelity engineering simulation, typically needed for aircraft design, will require extensive computational resources and database support for the purposes of design optimization as many disciplines are necessarily involved. Even relatively simplified models require exchange of large amounts of data among various disciplinary analyses. Crucial to an efficient aircraft simulation-based design therefore is a robust data modeling methodology for both recording the information and providing data transfer readily and reliably. To meet this goal, data modeling issues involved in the aircraft multidisciplinary design are first analyzed in this study. Next, an XML-based, extensible data object model for multidisciplinary aircraft design is constructed and implemented. The implementation of the model through aircraft databinding allows the design applications to access and manipulate any disciplinary data with a lightweight and easy-to-use API. In addition, language independent representation of aircraft disciplinary data in the model fosters interoperability amongst heterogeneous systems thereby facilitating data sharing and exchange between various design tools and systems.

INTRODUCTION

Improvement in aircraft design involves research into many distinct disciplines: aerodynamics, structures, propulsion, noise, controls, and others. Due to the inherent complexity and coupling of the disciplinary design issues, simulation-based analyses of aircraft design will naturally evolve to complex assemblies of dynamically interacting disciplines where each of the disciplines interacts to various degrees with the other disciplines (Figure1). The multidisciplinary couplings inherent in aircraft design not only increase computational burden but also present additional challenges beyond those encountered in a single-disciplinary simulation of aircraft. The increased computational burden simply reflects the massive size of the problem, with enormous amounts of analysis data and design variables adding up with each additional discipline. As a result, designing and implementing a new simulation methodology that supports the multidisciplinary aircraft design process can be an impractically expensive and time-intensive task. Currently reasonably well-developed and validated software tools exist within individual disciplines. Hence, a key requirement for the success of a practical multidisciplinary aircraft simulation is to provide the tools necessary to support efficient integration of these computer simulation codes. This approach demands a well-constructed data sharing and validation environment, which includes a robust data modeling and/or the use of a data exchange standard.

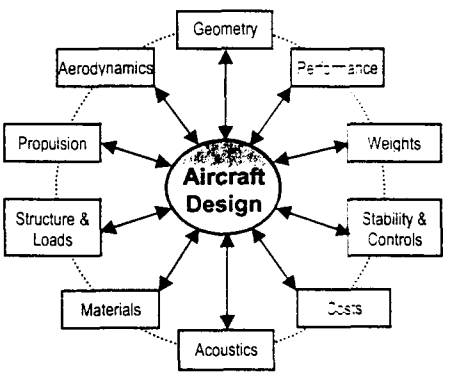


Figure1. Typical disciplines in an aircraft design

Traditional preliminary design procedures often decompose the aircraft into isolated components (wing, fuselage, engine, etc.) and focus attention on the individual disciplines (geometry, propulsion, acoustics, etc.). The common approach is to perform disciplinary analysis in a sequential manner where one discipline may synthesize the results of the preceding analysis

* Research Associate, Student Member AIAA, Department of Mechanical, Industrial and Manufacturing Engineering, E-mail: rlin@eng.utoledo.edu
† Professor and Chair, Department of Mechanical, Industrial and Manufacturing Engineering, Member AIAA, E-mail: aafjeh@eng.utoledo.edu

during the simulation run-time. The current practice emphasizes the multidisciplinary nature of the design of an aircraft through the use of integrated product teams. However, integrated and sharable aircraft design databases are not yet common in industry. One reason for this is because aircraft system simulation typically requires complex numerical algorithms and coupling models between dominant disciplines. Accordingly, developers can barely afford to build propriety data storage models around successful design applications. With the distinction, each discipline focuses on activities related to its own concerns. The designers typically provide each discipline with only those data which are required in performing the specific task of that discipline, and often, they spend 50-80% of their time organizing data and moving it between applications [1]. A very common problem with this kind of data exchange is data consistency. It is not uncommon to find that during the design phase, a particular discipline's updated calculations have not been effectively communicated with other disciplines involved in the design effort. This breakdown in the data exchange process results in inconsistent predictions among the various disciplines and could cause, for example, an "optimal" aerodynamic design that can not contain a sufficient supportive structure.

Other factors that can make the design process less efficient are data redundancy and the lack of a standard data format. To synthesize and evaluate aircraft designs, numerous software packages for analysis, post processing or data visualization are often employed. Because the aircraft simulation computing environments are typically heterogeneous, with platforms ranging from personal computers to UNIX workstations, to supercomputers, their internal data representations are normally not the same, these tools in general use different, possibly proprietary, data formats. Moreover, data are often duplicated in a slightly different format for the various disciplines' use. This lack of portability of data in different file systems greatly hinders sharing and exchanging of interdisciplinary data. In addition, the multiplicity of representation of disciplinary datasets not only wastes storage media capacity and CPU time, but it also generates an enormous overhead in terms of data translator development, additional software and data management. Although in some cases, custom translation tools are available to "massage" the data into the appropriate format; users still spend considerable time and effort tracking and validating data. As the analysis and design tasks become more distributed, communications requirements become more severe. Advances in aircraft disciplinary analyses and the growing trend in the use of high fidelity models in the last two decades have only aggravated these problems, increasing the amount of shared information and outpacing developments in interdisciplinary communications and system design methods [2].

Improving the simulation-based aircraft design process, therefore, requires the development of an integrated software environment which can provide interoperability standards so that information can flow seamlessly across heterogeneous machines, computing platforms, programming languages, and data and process representations [3]. In particular, emphasis should be placed on the generation of a database management system specifically crafted to facilitate multidisciplinary aircraft design. The subject of this paper is to provide a sharable and interchangeable database model for multidisciplinary aircraft design, with the intent to promote the interdisciplinary information sharing.

DESIGN REQUIREMENTS

The Multidisciplinary Optimization Branch (MDOB) at NASA Langley Research Center (LaRC) recently investigated frameworks for supporting multidisciplinary analysis and optimization research. The major goals of this program were to develop the interactions among disciplines and promote sharing of information. This section outlines several design requirements related to the data modeling that are particularly evident in the aircraft multidisciplinary analysis and optimization, based on the experience gained from the Framework for Multidisciplinary Design Optimization (MDO) project [4,5].

1) *Standards.* Use of standards in a database model preserves investment, results in lower maintenance costs and also promotes information sharing. It ensures that there are no interoperability problems between design teams that use the open standard.
2) *Sharable.* Data must be shared between disciplines and within disciplines with all the applicable quality, consistency and integrity checks [1]. Information sharing can reduce discipline isolation and encourage the use of the most advanced techniques while increasing the awareness of the effects each discipline has upon other disciplines and for reduced design cycle time [6].
3) *High-level interface.* Database model should allow the user to use and modify aircraft data in complex MDO problem formulations easily without low-level programming. By raising the level of abstraction at which the user programs the MDO problems, they could be constructed faster and be less prone to error.
4) *Extensible.* Advances in aircraft design will have new disciplines to appear, such as maintainability, productivity, etc., therefore database model should

be extensible and should provide support for developing the interfaces required to integrate new disciplinary information into the system easily. As a result, the user would avoid having to wait for the needed features to appear in new releases.

5) *Large data size.* Since aircraft design involves a lot of disciplinary analysis variables, database model should be able to handle large problem sizes. Supporting techniques should allow database to grow and shrink dynamically, but do not degrade the database performance dramatically.
6) *Object-oriented.* Database model should be designed using object-oriented principles. Object-oriented design [7] has several advantages in aircraft design. For example, object-oriented principles provide *polymorphism* for analysis or optimization methods at run time. Object-oriented software design has been employed as a tool in providing a flexible, extensible, and robust multidisciplinary toolkit that establishes the protocol for interfacing optimization with computationally-intensive simulations [8].
7) *Distributed.* For large problems, the designers in different disciplinary teams need to be able to conveniently work together by *collaborative design* [9]. It is desirable that a database model could support disciplinary code execution distributed across a network of heterogeneous computers.

The implementation of a database to meet all these requirements is a major challenge. In the following sections, we focus on the design and development of a XML-based database model as a first step toward meeting that challenge.

XML FOR AIRCRAFT DATA

XML [10] is a generic, robust syntax for developing specialized markup language, which adds identifiers, or tags, to certain data so that they may be recognized and acted upon during future processing. Several good features inherent within XML would make it well suited to the task for satisfying multidisciplinary data requirements.

As indicated in the Design Requirements section, data sharing is an essential element in preventing design isolation between various aircraft disciplinary components. XML provides a hierarchical container that is platform-, language-, and vendor-independent and separates the content from any environment that may process it. It is normatively tied to an existing ISO standard, ISO 8879 (SGML) [11], and is an acceptable candidate for full use within other ISO standards without the need for further standardization effort. By accepting and sending aircraft data in plain text format, the requirement to have a standard binary encoding or storage format is eliminated, allowing aircraft applications running on disparate platforms to readily communicate with each other. Aircraft design applications written in any other programming language that process XML can be reused on any tier in a multi-tiered client/server environment or distributed computing, offering an added level of reuse for aircraft data. The same cannot be said of any previous platform-specific binary executables. Because XML is or will be fully supported in Web browsers, it should be possible to use Web technology to communicate disciplinary data entities in a collaborative aircraft design environment.

When using XML, it not only allows input of the data, but also permits one to define the structural relationships that exist inside the data. The hierarchical structure in XML combined with its linking capabilities [12, 13] can encode a wide variety of aircraft data structures. The element's name, attributes and content model are closely related to data class name, properties and composition associations in object-oriented aircraft simulation. By using XML to represent aircraft data, it is possible to faithfully model any structural aircraft data of a chosen component in their design context.

In traditional aircraft multidisciplinary analyses, validating data format and ensuring content correctness is another major hurdles in achieving data exchanges of aircraft data. XML also provides facilities for the syntactic validation of documents against formal rules. This can be achieved through Document Type Declaration (DTD) [10] or XML-Schema [14], which defines the constraints and logical structures that an XML document should be constructed. A data file written in XML is considered valid when it follows the constraints that the DTD or XML-Schema lays out for the structures of XML data. XML Schema also offers a number of other significant advantages over DTD, such as more advanced data types and a very elaborate content model. Without XML, any validation of aircraft data has to be implemented at the expense of work by application developers. When using XML to encode aircraft design data, XML parser can be used readily to check the validity and integrity of the aircraft data stored in XML documents. This guarantees the data producer and consumer exchange the aircraft design data correctly.

The various advantages outlined above present compelling reasons to use XML for aircraft design data representation. However, the solution is not as easy as it might at first appear. While XML is a useful technology, it is, ultimately, simply serialization syntax. In particular, just putting aircraft data into XML form does not make it any more interchangeable than it was before, because the recipient of the data must still have an understanding of what the design-specific data are

inside XML file *semantically* in order to process them correctly. Semantic interoperability is of vital importance between different aircraft disciplines and simulation components, as it enables them to agree on how to use aircraft data and how to interpret application data for different disciplinary designs. In addition, there are still several other requirements (for example, large datasets, object-oriented, high-level interface, etc.) to meet in order to use XML to communicate aircraft design data between disciplines efficiently.

In the next section, we will provide the design of an extensible aircraft data object model. This model will be used to interpret aircraft design data between design disciplines, and serve as a foundation to implement a XML-based aircraft database to meet all the design requirements.

DATA OBJECT MODEL

The aircraft design process [15] can be divided into three phases: *conceptual design*, *preliminary design*, and *detailed design*. Since aircraft design by its nature is a very complicated process and involves vast amounts of data, for the purposes of this paper, we will only demonstrate the data model in the aircraft conceptual design. Conceptual design involves the exploration of alternate concepts for satisfying aircraft design requirements. Trade-off studies between aircraft conceptual designs are made with system synthesis tools, which encompass most of aircraft components and a broad range of disciplinary interactions.

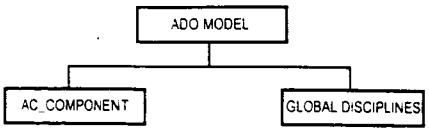
In order to effectively represent aircraft design data using XML, a set of data object structures was first designed. Figure 2 shows an overall layout of a simplified data model. The designed database model is composed of aircraft components and other disciplinary data objects (Fig. 2a). The overall model is organized in a strict hierarchical manner in accordance with the XML topology. Each node in the data structure shown here is represented as an *Aircraft Data Object* (ADO). These objects hold no complex design logic, but they contain typed data and preserve the logical structure of the model. The ADO model precisely defines the intellectual content of aircraft-related data, including the organizational structure supporting such data and the conventions adopted to standardize the data exchange process. The functional model identifies a common process in order to ascertain what data are required for a typical aircraft design process. Figure 2 also indicates (informally) what data, if any, are encapsulated within each node object.

Aircraft Components

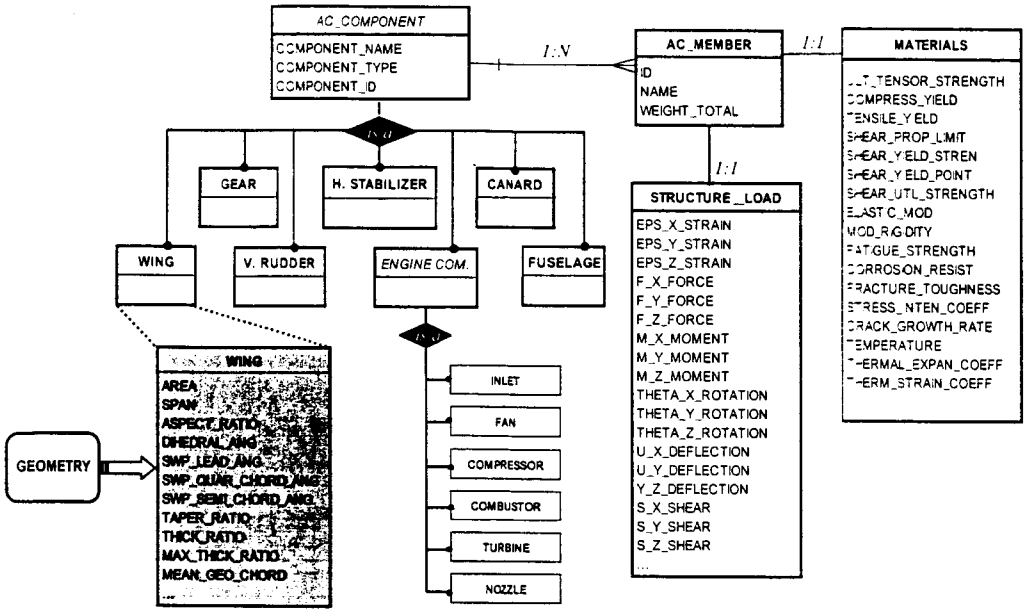
An aircraft component (*AC_Component*) object can be an engine, fuselage, landing gear, canard, horizontal stabilizer, vertical rudder, or wing (Figure 2b). Every component has a user-defined name and unique *component type*, which characterizes the nature of its usage. For practical purposes, a component type is characterized as a set of possible values, such as WING, ENGINE, etc. There is a special data type, called *object identification (Component_ID)*, whose value is the unique identifiers of encapsulated objects to be referenced in the aircraft design.

Each aircraft component itself may be made up of physically distinguishable *subcomponents* or *parts*. For example, an engine is made up of inlet, fan, compressor, combustor, turbine and nozzle subcomponents. Likewise, most landing gears have parts like wheel, tire, brake assembly, etc. Every subcomponent is represented by a data object, with member properties and subtypes (not shown here for simplicity) encapsulated in it. Each part is modeled as a *component member* object and encapsulated as a child in *AC_Component*. An important feature to note from Figure 2b is the local inclusion of several disciplinary data. Since each member object has its own materials requirements (e.g., modulus of rigidity, fatigue strength, etc), structure and loads characteristics (e.g., strain, stress, displacement, etc), these disciplinary data are naturally considered as parts of a member object. The local inclusion of component disciplines prevents design data isolation, and promotes data sharing and exchange during the design process. Aircraft propulsion system (not shown here) is considered as a member type for the engine components. A more detailed demonstration for aircraft propulsion model can be found in Ref. [16].

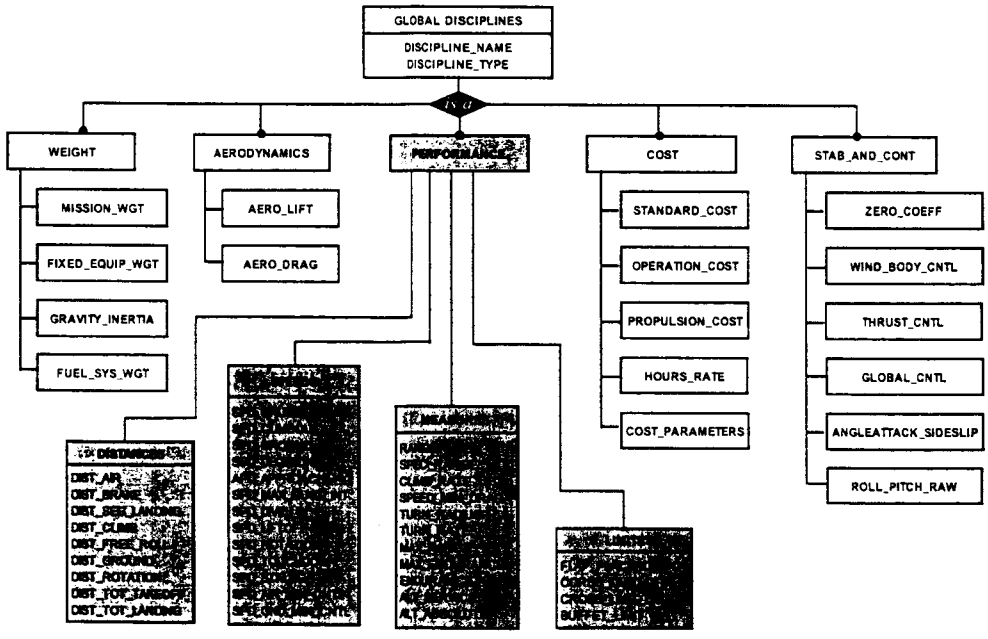
Besides the hierarchical layer of the data objects structure, the designed model also encourages the use of data object *abstraction*, *inheritance*, and *composition*. Returning to Figure 2b, we can see that each of the specific aircraft components is patterned as an "*is/a*" relationship with *AC_Component*, therefore each specific component data model automatically inherits all the data member proprieties and subtypes (materials, structure and load) of its parent. In this sense, *AC_Component* provides a data abstraction for all its component children, allowing each single element to be treated the same way as the assemblies of elements in its internal data representation. For each specific aircraft component data modeling, we can represent the hierarchical structure of the data properties, substructure and their disciplinary data using *recursive composition*. For example, we can combine multiple sets of rotor and stator blade data objects to form a fan component data. This technique allows us to build increasingly complex aircraft data components out of simple data object models. The designed database model gives us a convenient way to construct and use arbitrary complex aircraft data model and makes the



(a) Top level children of ADO model



(b) Aircraft components data object model



(c) GlobalDisciplines data object model

Figure 2. Aircraft data object model

model totally extensible for future enhancements.

Geometry Modeling

Component geometry modeling is somewhat unique in aircraft design. All disciplines share the same geometry. Strong interactions between the disciplines are very common and complicated. For example, during operation, the geometry of a flexible structure (e.g., wing) may change due to the aeroelastic effects. Geometry modeling must, therefore, be accurate and suitable for various disciplines (e.g. deflection and load). For a multidisciplinary optimization problem, the application must also use a consistent parameterization across all disciplines. Thus, an application requires a common geometry dataset that can be manipulated and shared among various disciplines [17].

STEP Application Protocol *AP 203* — Configuration Controlled 3D Designs of Mechanical Parts and Assemblies [18] — is a set of standards that defines the CAD geometry, topology, and configuration management data of solid models for mechanical parts. AP203 supports wireframe, surfaces, solids, configuration management, and assemblies. The STEP modelers have undertaken the very difficult job of defining mappings between the different representations of the same information. For example, a curve on the surface of fuselage can be represented as a B-spline, as a list of curve segments, or as NURBs. In our aircraft database, a placeholder has been designed to support various aircraft components' geometry disciplinary data that conform to the STEP-based model. Because different components normally have very different geometry requirements, the geometry disciplinary data are considered local to every concrete component. Different fidelity geometry models can be chosen for use in the design process.

Global Disciplines

Other disciplinary data, such as stability and control, aerodynamic, performance, cost, and weight data, are currently modeled as *global* objects (and grouped together as *GlobalDisciplines*) of the aircraft database (Figure 2c). This seems a little unnatural, however, these calculations have been traditionally grouped by discipline in aircraft design, and they probably will continue to be associated in this manner for some time to come. The relationship between these disciplinary data and aircraft database is also modeled as parent to child. For example, one of the relative important design parameters on the conceptual vehicle design is system performance. This disciplinary category in our design is currently made up of different criteria data objects, such as *distance, speeds, limits, measures,* etc., as shown in Figure 2c. The figure also gives the sample data that may be included in the discipline. New data will be added in as the data object model evolves in the future.

SCHEMA DESIGN

Aircraft Schema establishes a bridge between XML-based description of aircraft data and the ADO model. A set of aircraft Schema has been designed in XML Schema language that specifies how the constituents of the ADO object are mapped to an underlying XML structure. It associates each piece of information defined in ADO to a precise location in the XML structure.

Each aircraft data object defined in ADO is mapped to one or more nodes. For the most part, the aircraft Schema closely follows the ADO model. Aircraft-schema file must be ADO-compliant in order for other applications to be able to properly interpret aircraft data. This is particularly important when trying to transfer data between different disciplines and different storage models, as there must be agreed-upon data structure and syntax for different systems to understand each other. The rules in ADO model will guarantee that the schema description of aircraft data is syntactically correct and follows the grammar defined within it. An important feature of the ADO data model is the hierarchical structure, which allows the aircraft data file to be structured as a rooted directed graph, so it is necessary to map the directed graph of aircraft data in XML onto a tree of aircraft data objects specified in ADO. However, when a given piece of information is listed as being "under" a node, there are actually two possibilities: the information can be stored as data in the current node, or it can be stored as data under a separate child node. The aircraft schema also determines which of these two possibilities are best for each situation.

An example of aircraft schema design is demonstrated in Figure 3. Based on the ADO model, an aircraft database model includes several kinds of component data objects (such as Wing, Fuselage etc.), which can be contained in an aircraft, and a *GobalDisciplines* data object. To create aircraft component constructs, we start by creating a basic aircraft component complex type, `AircraftComponent_t`, which contains a single `AircraftMember` element. An `AircraftMember` is constrained by its complexType `AircraftMember_t`, where `AircraftMember_t` itself contains `Name` `TotalWeight`, `Materials`, and `StructureLoad` elements, and in turn, are constrained by their corresponding built-in string type, double type and similarly-defined complexTypes separately.

An aircraft component also contains a set of desired data attributes – componentType, name, identification – that are encapsulated in the *AircraftComponent* object.

```
<?xml version="1.0"?>
<xsd:schema targetNamespace="http://mems1.ni.utoledo.edu/aircraft"
    xmlns="http://mems1.ni.utoledo.edu/aircraft"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">

    <xsd:include schemaLocation="materials.xsd" />
    <xsd:include schemaLocation="structureload.xsd" />
    <xsd:include schemaLocation="globaldisciplines.xsd" />
    <!-- other basic schema components are included here-->

     <xsd:complexType name="AircraftComponent_t">
          <xsd:sequence>
               <xsd:element name="AircraftMember" type="AircraftMember_t"
                  maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attributeGroup ref="ComponentAttributes"/>
     </xsd:complexType>

     <xsd:attributeGroup name="ComponentAttributes">
        <xsd:attribute name="componentType" use="required">
          <xsd:simpleType>
              <xsd:restriction base="xsd:string">
                 <xsd:enumeration value="WING"/>
                 <xsd:enumeration value="LANDINGGEAR"/>
                 <!-- other aircraft types are continued here-->
              </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="identification" type="xsd:ID" use="required"/>
     </xsd:attributeGroup>

     <xsd:complexType name="AircraftMember_t">
          <xsd:sequence>
              <xsd:element name="Name" type="xsd:string" />
              <xsd:element name="TotalWeight" type="xsd:double" />
              <xsd:element name="Materials" type="Materials_t" />
              <xsd:element name="StructureLoad" type="StructureLoad_t" />
          </xsd:sequence>
          <xsd:attribute name="memberID" type="xsd:ID"/>
     </xsd:complexType>

     <xsd:complexType name="Wing_t">
          <xsd:complexContent>
               <xsd:extension base="AircraftComponent_t">
                    <xsd:sequence>
                         <xsd:element name="Area" type="xsd:string" />
                         <xsd:element name="Span" type="xsd:string" />
                         <!-- other geometry data are continued here-->
                    </xsd:sequence>
               </xsd:extension>
          </xsd:complexContent>
     </xsd:complexType>

      <xsd:element name="Aircraft">
        <xsd:complexType>
          <xsd:sequence>
               <xsd:element name="Wing" type="Wing_t" minOccurs="2"
                         maxOccurs="2"/>
               <!-- other aircraft components are defined here -->
               <xsd:element name="GlobalDisciplines"
                         type="GlobalDisciplines_t"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

Figure 3. A sample aircraft schema

These attributes are grouped together, represented by `ComponentAttributes`, and referenced by name in the *AircraftComponent*'s complexType declaration. For example, the `componentType` attribute is restricted to a set of predefined type values, such as `WING`, `LANDINGGEAR`, etc, these types are constrained by enumerations definition in the simpleType definition.

Then a set of concrete aircraft components is built based on the `AircraftComponent_t` complexType. The technique here is to derive new (complex) aircraft component types by extending an existing type. For example, when building data schema for the wing component, we define the content model for Wing element using new complex types, `Wing_t`, in the usual way; in addition, we indicate that the concrete wing component (Wing) is extending the `AircraftComponent_t` base type. When a complex type is derived by extension, its effective content model is the content model of the *base* type plus the content model specified in the type derivation. In the case of `Wing` element, its content model `Wing_t` is the content model of *AircraftComponent* plus the declarations for the wing's local data elements and attributes.

Other aircraft component and disciplinary data schema can be designed in a similar manner. Finally, the whole aircraft schema is composed of different aircraft components and *GlobalDisciplines* data. Note that when designing aircraft schema, all the basic components and disciplinary schemas do not need to be coded in a single file during the design time. For example, Figure 3 does not explicitly show the disciplinary schema such as material, structure and load, components other than wing, etc. instead, it uses `include` element to indicate that these schemas exist outside the aircraft schema file. In this way, each schema can be designed separately by different disciplinary groups, and then "included" together during the run time. This kind of flexible design will allow for modular development and easy modification of aircraft schema as its data object model evolves in the future.

Because of the important nature of aircraft geometry disciplinary data, our database model currently uses STEP AP203 standard to encode all the aircraft geometry data. STEP models are written using the EXPRESS language [19]. EXPRESS provides a rich collection of types and inheritance organizations to capture data structure and to describe information requirements and correctness conditions necessary for meaningful data exchange, therefore makes it easier to describe an accurate aircraft geometry model. However EXPRESS does not dictate how the models should be implemented using various database technologies. Implementers must convert an EXPRESS information model into schema definitions for the target database. This conversion requires a mapping from the EXPRESS language into the data model of the target database system. EXPRESS information models describe logical structures that must be mapped to a software technology before they can be used.

Given an EXPRESS schema that specifies aircraft geometry information, it is possible to define a set of schema languages (such as DTD or XML-Schema) that are used to encode geometry information specified in EXPRESS schema. Several researches have been done to encode EXPRESS schema by DTDs, among which

the most important one is STEP Part 28 (XML representation of EXPRESS-driven data) [20], which includes a set of standard DTD declarations to represent any EXPRESS schemas in XML as well as data corresponding to an EXPRESS schema. Therefore, it is convenient to take advantage of this standard to encode all aircraft geometry data. This is done by designing a STEP CAD Conventer, which can convert from a valid aircraft geometry STEP model constrained by DTD to XML-Schema. In this way, the general schema design techniques provided in this section can still be applied to aircraft geometry data. Moreover, by using XML-Schema to represent STEP CAD model, it can benefit the good features in XML-Schema, such as modular schema inclusion (xinclude), and also offer a uniform data schema formalism for database implementation. A simple illustration is given in Figure 4.

```
<!—DTD for 3D point-->
<!ELEMENT Cartesian_point EMPTY>
<!ATTLIST Cartesian_point
      x-id ID #REQUIRED
      X CDATA #REQUIRED
      Y CDATA #REQUIRED
      Z CDATA #REQUIRED
>

<!—XML Schema for 3D-point after conversion-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Cartesian_point">
            <xsd:complexType>
                  <xsd:attribute name="X" type="xsd:string" use="required"/>
                  <xsd:attribute name="Z" type="xsd:string" use="required"/>
                  <xsd:attribute name="Y" type="xsd:string" use="required"/>
                  <xsd:attribute name="x-id" type="xsd:ID" use="required"/>
            </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

Figure 4. Example STEP CAD Converter for 3D point

AIRCRAFT DATABINDING

Multidisciplinary design of aircraft systems is a complex, computationally intensive process that combines discipline analyses with intensive data exchange and decision making. The decision making is based on the overall design optimization but is greatly assisted by data sharing and automation [5]. Aircraft data encoded by XML provides a means to share disciplinary data between aircraft design teams, but their physical storage form on the external storage medium is still not intelligible or easily accessible. *Aircraft databinding* provides an implementation for the designed data object model. Meanwhile, it also encapsulates a convenient way for conversion between the aircraft data in XML file and their object representations *automatically* and provides a lightweight and easy-to-use API, which facilities the design applications to access, modify and store any aircraft data object using a high-level object interface.

Aircraft Databinding includes two components: an *aircraft schema compiler* and a *marshalling framework*. It was written in Java; thus the software can be run on different design platforms.

Schema Compiler

The *aircraft schema compiler* is designed to automatically translate the aircraft schema into a set of derived aircraft data class source codes. It maps instances of aircraft schemas into their data object models, and then generates a set of classes and types to represent those models (Figure 5).

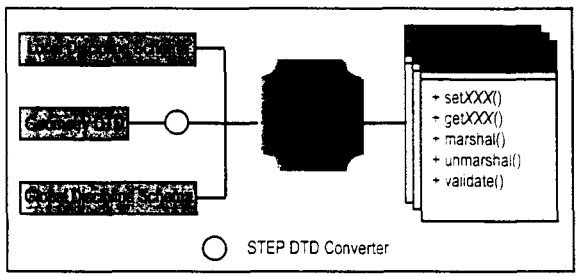


Figure 5. Schema compiler in Aircraft databinding

Let's consider how the data class is generated by schema compiler with input of the schema defined in previous section. With the "Aircraft" schema defined, attributes represent simple Java types, usually primitives. Thus, *name* and *componentType* attributes in the *AircraftComponent*'s complexType are compiled into Java type of String, and *identification* attribute becomes Java primitive of type *int*, respectively. All elements (along with its type information which specifies the content model), such as *Aircraft*, *AircraftComponent* etc, become Java Classes, which can then have class instance properties themselves, again represented by attributes. In this way, a recursion occurs: an element becomes a new class, and each property of it is examined. If the property is an attribute, a simple Java primitive member variable is created for the object; if the property is element, a new data object type is created, added as a member variable, and the process begins again on the new object type, until all classes are created. All other aircraft components and disciplinary data can be similarly created. A Unified Modeling Language (UML) diagram for generated Java class (only wing component is shown) is illustrated in Figure 6. The generated classes also ensure that all the hierarchical data object structure and their internal relationships are properly maintained. For example, the figure shows that Wing is a *subclass* that extends *AircraftComponent*, therefore, it inherits all states and behaviors from its ancestor.

In addition, the generated classes provide methods

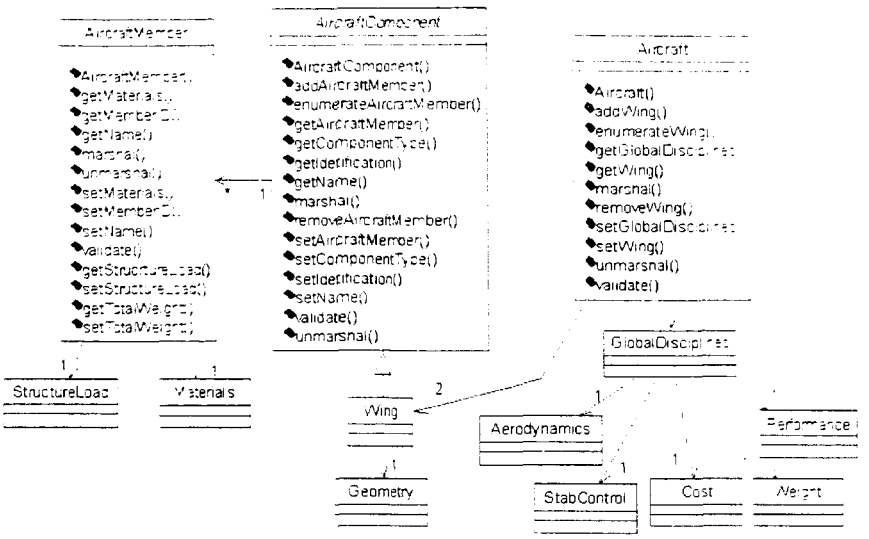


Figure 6. UML for Data class structure generated by aircraft databinding

to access and modify the properties defined in the aircraft Schema. These methods closely follow the JavaBean Design Pattern [21]. The main guideline for the design pattern is that all publicly accessible fields have proper getter (accessor) and setter (mutator) methods. For a given field, a getter method is quite simply a method that returns the value of that field, while a setter method is one that allows us to set the value of the field. Each method signature specifies the name of the operation, which is sufficient for design tools to obtain information about the fields of data classes by examining the method signatures of a given class. This examination process is called *Introspection*.

For each aircraft data class that is automatically generated (e.g. *AircraftComponent*), there is also included a set of *marshal*, *unmarshal* and *validate* methods, with their method signatures like:

```
public boolean validate()
public void marshal(Writer out)
public AircraftComponent unmarshal(Reader reader)
```

The `validate` method is used to check whether the aircraft data contained in XML file is valid, i.e. conform to its corresponding data schema; `marshal` and `unmarshal` methods can be used to map directly to the data of elements and attributes within the XML document and also affect the underlying aircraft data. This is achieved through underlying Marshalling Framework design.

Marshalling Framework

The marshalling framework supports the transportation (unmarshal) of aircraft data in XML files into graphs of interrelated instances of aircraft objects that are generated during schema complier and also converts (marshal) such graphs back into XML file. For example, when XML-based wing data is correctly unmarshaled into aircraft Java codes, the Wing node in the XML file becomes an instance of the Wing class that was generated by aircraft Schema Compiler, i.e. Wing Data Object. The aircraft design system can then interfaces those objects, and all interactions and manipulations of aircraft disciplinary data in a design system can be described as invocations of operations on those objects. In particular, the aircraft design application can use the corresponding methods devised with a set of mutator and accesor methods to work with the aircraft data in the underlying design data file. Therefore, it provides a convenient way to access and modify the aircraft data where all underlying files are transparent to the user. The end result is aircraft data binding.

Distributed Access

As the argument in `marshal` is a general "writer" object, it can be piped to or wrapped into many other different writers or streams, such as a network connection, or another program. This means marshaling can be done remotely from aircraft disciplinary design team servers (Figure 7). The same applies to unmarshaling process where a general "Reader" is used. A set of sample disciplinary drivers have been written that use HTTP socket connection, Java Servlet, CORBA, RMI technology to allow the databinding to be called from different client working environments. These discipline drivers can serve as a 'plug-in' for aircraft disciplinary simulation codes and enables them to use XML-based aircraft data easily and remotely.

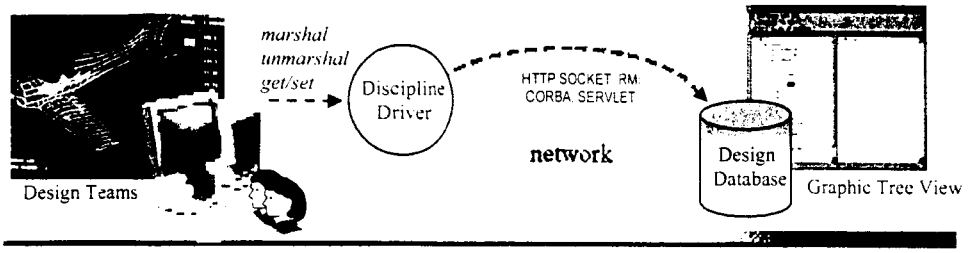


Figure 7. Design teams use Marshal Framework to access aircraft data remotely

According to the design requirements, more than one discipline data may need to be called by a driver. The interfaces between the discipline codes and their drivers must be accurately specified in order to provide proper communications. The disciplinary drivers can also serve as templates or examples for more complex problems.

Dynamically Schema Add-in

With advances in aircraft design process, there has been an increased realization that new disciplines, such as maintainability, productivity, etc., should be addressed in order to optimize the aircraft design process. Aircraft databinding also provides a service that can dynamically add-in new aircraft disciplinary schemas. These schemas can be either in XML-schema format or in DTD format, but they must conform to a set of newly designed disciplinary data object models. *STEPConverter* is an example of this service that provides a set of tools and libraries to read and write STEP Part28 compatible DTD file and to be used for aircraft geometry modeling (Figure 5). By using add-in support to aircraft disciplinary schema, the databinding code itself is kept generic and does not need any special coding for a new problem.

Performance

Since XML description of aircraft data are by their nature potentially large in size, in order to improve aircraft database performance, the databinding internally integrates another service, through XInclude [12] and XLink [13], that further allows users to split an arbitrary large aircraft data file into a sequence of sufficiently small subfiles during the marshalling process, and resemble all these pieces together when unmarshaling XML-based aircraft data to their data objects. This kind of flexibility allows an aircraft data file to span multiple physical files reside in different computers by referencing as URI, and also make possible a portion of one aircraft data file to be referenced by several other aircraft files. The individual files are more portable due to their reduced size, and make use of less memory to represent the whole necessary layered tree of the aircraft data nodes. In addition, a *ZipArchiver* is included in the aircraft databinding, which will compress the aircraft data in XML subfiles into different Zip entities in an aircraft archive when transferring aircraft data objects to data files. By using text compression algorithms, the XML data file size can be much smaller than the original size and even smaller in size than binary representation of the same data. This reduces file I/O access times and improves performance required for large aircraft dataset.

CONCLUSION

In this work, a XML-based database model for use in multidisciplinary aircraft design has been designed, which meets design requirements of diverse disciplines. The database consists of data object models, database schemas, and data binding. Aircraft Data Object (ADO) model encompasses most of common components involved in multidisciplinary aircraft design, as well as various pertinent disciplines, such as aerodynamics, structures, cost, materials, performance, stability and control and weights. STEP AP203 standard is used to describe each component's geometry data. The ADO model precisely defines the organizational structure supporting aircraft design data and the conventions adopted to standardize the data exchange. This is particularly important when trying to transfer data between different disciplines and different storage models, as there must be agreed-upon data structure and syntax for different systems to understand each other.

In order to store and validate XML-based aircraft data, a set of database schemas was designed based on ADO model. By using XML Schema to represent aircraft Schema, a set of constraints establishes how domain-specific data should be constructed, which can then be used to further schema-validate the aircraft data, ensuring that the contained data are valid. The database schema follows a modular design pattern such that it is extensible for future addition and/or modification. By using and developing focused aircraft disciplinary schema for specific aircraft component

object types, users can benefit by an increase in application reusability.

The aircraft databinding provides an object interface to various aircraft disciplines, allowing automated storage and retrieval of XML-based aircraft design results within and across disciplines. Most of the data manipulation services are transparent to the aircraft designer and simulation codes. This higher level database development with automation support provides a common working environment, which would enhance the productivity of multidisciplinary projects.

Since all disciplinary data in the binding process are stored in XML documents, they bypass the requirement to have a standard binary encoding or storage format. Additionally, the language independent representation of various aircraft component and disciplinary data can foster interoperability amongst heterogeneous systems, and thereby greatly facilitates the multidisciplinary aircraft design.

ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial support from NASA Grant NAG-1-2244 under the direction of Mr. Wayne K. Gerdes, NASA Langley Research Center.

REFERENCE

[1] Current State of the Art on Multidisciplinary Design Optimization, An AIAA White Paper, September 1991. *http://endo.sandia.gov/AIAA_MDOTC/sponsored/aiaa_paper.html*

[2] Kroo, I., Altus, S. et al., "Multidisciplinary Optimization Methods for Aircraft Preliminary Design", AIAA 94-4325, *Fifth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Panama City FL, September 1994.

[3] Reed, J. A., Follen, G. J. and Afjeh, A. A., "Improving the Aircraft Design Process using Web-based Modeling and Simulation", ACM Transactions on Modeling and Computer Simulation, Vol. 10, No. 1, 2000, pp. 58-83

[4] Ramki Krishnan, "Evaluation of Frameworks for HSCT Design and Optimization", NASA/CR-1998-208731, October 1998

[5] Salas, A.O. and Townsend, J. C. "Framework Requirements for MDO Application Development," *7th AIAA/USAF/NASA/ISSMO Symposium on Multi-disciplinary Analysis and Optimization*, St. Louis, Missouri, AIAA 98-4740, September 2-4, 1998

[6] Jones, K.H. et al, "Information Management for a Large Multidisciplinary Project," AIAA-92-4720, *4th AIAA/AF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Independence, Ohio, September 21-23, 1992.

[7] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W., *Object-Oriented Modeling and Design*, Prentice Hall, Englewood Cliffs, NJ, 1991.

[8] Eldred, M., Hart, W., et al, "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," *Proc. 6th AIAA/USAF /NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, paper AIAA-96-4164, 1996.

[9] Monell, D. W. and Piland, W. M., "Aerospace Systems Design in NASA's Collaborative Engineering Environment", *50th International Astronautical Congress*, Amsterdam, The Netherlands, October, 1999

[10] Extensible Markup Language (XML) 1.0, W3C Recommendation, Feb. 1998

[11] "Information Processing -- Text and Office Systems - Standard Generalized Markup Language (SGML)", ISO 8879:1986.

[12] "XML Inclusions (XInclude) Version 1.0," W3C Working Draft 16 May 2001

[13] "XML Linking Language (XLink) Version 1.0," W3C Recommendation, 27 June 2001

[14] "XML Schema Part 0: Primer, W3C Recommendation," 2 May 2001, http: www.w3.org/TR/xmlschema-0

[15] Raymer, D. P., *AIRCRAFT DESIGN: A Conceptual Approach, 3rd Edition*, AIAA Education Series, New York, NY, 1999

[16] Lin, R., Afjeh, A.A., "Interactive, Secure Web-Enabled Aircraft Engine Simulation using XML Databinding Integration", AIAA-2002-4058, *38th AIAA/ASME/ SAE/ASEE Joint Propulsion Conference & Exhibit*, Indianapolis, Indiana, 2002

[17] Samareh, J. A., "A Survey of Shape Parameterization Techniques", *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics 1999*, Williamsburg, Virginia, 1999

[18] ISO/WD 10303 STEP, the International Standard for the Exchange of Product Model Data, ISO TC184/SC4 committee

[19] ISO/WD 10303-11, Product data representation and exchange: EXPRESS Language Reference Manual

[20] ISO/WD 10303-28, Product data representation and exchange: Implementation methods: XML representation of EXPRESS-driven data

[21] Watson, M., *Creating Java Beans: Components for Distributed Applications*, Morgon Kaufmann Publishers, Sept, 1997