JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, March 2004

An XML-Based Integrated Database Model for Multidisciplinary Aircraft Design

Risheng Lin* and Abdollah A. Afjeh†
The University of Toledo, Toledo, Ohio 43606

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 simulations, typically needed for aircraft design, will require extensive computational resources and database support for the purpose 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 exchanging data efficiently and reliably. To meet this goal, data modeling issues involved in the aircraft multidisciplinary design are first examined in this study. Development and implementation of an XML-based, extensible data object model suitable for multidisciplinary aircraft design is then discussed. The model, which incorporates aircraft databinding and aircraft persistence engine, allows the design applications to access, manipulate and manage 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 exchanging between various design tools and systems.

I. 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 (see Fig. 1). 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, 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.

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 during the simulation run-time. The current practice emphasizes the multidisciplinary nature involved in an aircraft design through the use of integrated product teams. However,

Presented as Paper 2002-5613 at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, 4 September 2002; received 19 April 2003; revision received 5 November 2003; accepted for publication 10 December 2003.
*Mechanical, Industrial, and Manufacturing, Engineering Department, 2801 West Bancroft Street, rlin@eng.utoledo.edu
†Mechanical, Industrial, and Manufacturing, Engineering Department, 2801 West Bancroft Street, aafjeh@eng.utoledo.edu

integrated and sharable aircraft design databases are not yet common in industry. One reason for this is that aircraft system simulation typically requires complex numerical algorithms and coupling models between dominant disciplines. Accordingly, developers typically build propriety data storage models around successful design applications, with each discipline focusing on activities related to its own concerns. The analysis tools provide each discipline with only those data, which are specific to that discipline. Users often spend 50-80% of their time organizing data and moving it between applications.[1] A prevalent 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 is not consistent with the supportive structure.

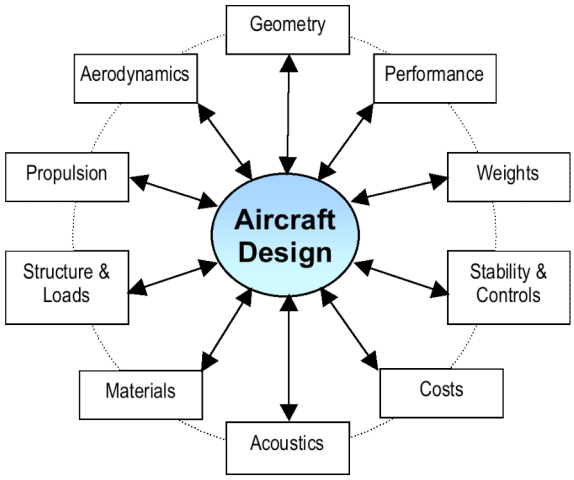


Fig. 1 Examples of disciplines involved in an aircraft design. (For clarity not all disciplines are shown.)

Other factors that can impede the multidisciplinary analysis 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. Aircraft simulation computing environments are typically heterogeneous, with platforms ranging from personal computers to UNIX workstations, to supercomputers. The internal data representations in these systems are normally different. Users have many options to choose a code/language/application, which in general limits automatic data exchange in various computing environments. Furthermore, some analysis tools use data formats, possibly proprietary, which are custom-designed for efficient use in specific disciplines. Collectively, all these factors lead to a lack of portability of data in different file systems and hinder 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]

Data modeling is also an important issue in aircraft certification and operation. Currently, more than 80 large databases of aviation safety data are being used by industry and the Federal Aviation Administration (FAA). Examples include:[24] Aircraft Type Certificates, Aircraft Registry (AR), BASIS (British Airways Safety Information System),[25] etc. There is a good deal of duplication; data collection efforts are fragmented, and databases are not coordinated. For example, when investigating an accident, one or more links in the accident chain may be in some cases associated with several disciplines in the aircraft design, but the relevant data may be scattered over separate, unlinked databases. In this environment, developing a comprehensive and accurate understanding of system malfunctions is difficult. For these reasons, government and industry are now exploring ways to construct advanced databases, such as the Global Analysis and Information Network (GAIN),[26] which would serve as a single global database. The development of GAIN has been hampered, however, by how the data should be standardized, how

they should be shared and disseminated, and so on. The use of standardized formats for aircraft data would facilitate data collection, database management, and data dissemination. To be complete, aircraft database should include data from manufacturers' design, regulatory inspectors and test engineers, etc.

This paper will focus on the development of a sharable, extensible database framework for aircraft multidisciplinary design. The ideas advanced and methods employed are applicable to certification applications as well. Considering the critical role of data sharing and management in aircraft design process, it becomes apparent that one of the approaches to improve the simulation-based design process is 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 creation of a database management system specifically crafted to facilitate multidisciplinary aircraft design.

XML, due to its structured, platform and language independent, highly extensible and Web-enabled nature, has rapidly become an emerging standard to represent data between diverse applications. The opportunity now exits to take advantages of recent developments in XML and its related information technology to potentially liberate the flow of data that are trapped within boundaries of disparate disciplinary data sources in the multidisciplinary aircraft design, thereby promoting efficient interdisciplinary information sharing and data management. In the remainder of this paper, we examine how this development might be achieved. In Section 2, we briefly outline some of the design requirements relevant to the aircraft multidisciplinary analysis and optimization. Advantages of applying XML to aircraft database design are analyzed in Section 3. Section 4 and Section 5 introduce the design advanced in this paper, namely, aircraft data object models and aircraft schema design, two semantic data components in aircraft design domain, based on object-oriented and XML based views, respectively. Section 6 describes the components of an aircraft databinding framework developed in this research to allow easy conversion between the XML-based aircraft data and their data object model. The related aircraft database persistence engine is described in Section 7. Finally, conclusions appear in Section 8.

II. 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 increase the interactions among disciplines and promote sharing of information. Based on the experience gained from the Framework for Multidisciplinary Design Optimization (MDO) project,[4,5] this section outlines several design requirements related to the data modeling that are particularly relevant to the aircraft multidisciplinary analysis and optimization.

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 that each discipline has upon other disciplines and help reduce design cycle time.[6]

3) High-level interface. The 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, the programs could be constructed faster and be less prone to error.

4) Extensible. Advances in aircraft design will permit inclusion of new disciplines, such as maintainability, productivity, etc., therefore a database model should be extensible. Furthermore, the model should provide support for developing 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 software releases.

5) Large data size. Since aircraft design involves many disciplinary analysis variables and large datasets, the database model should be able to handle large problem sizes. Supporting techniques should allow a database to grow and shrink dynamically, but not degrade the database performance dramatically.

6) Object-oriented. The database model should be designed using object-oriented principles. Object-oriented design[7] has several advantages in aircraft design applications. 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.

7) Distributed. For large problems, it is highly desirable that designers in different disciplinary teams be able to conveniently work together by *collaborative design*.[9] This can be facilitated by a database model that could support disciplinary code execution distributed across a network of heterogeneous computers.

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

III. A Survey on XML for Aircraft Data

A. Overview

XML[10] (eXtensible Markup Language) is a standard "Recommendation" adopted by the World Wide Web Consortium (W3C) in 1998. The language defines a generic robust syntax that allows user to add identifiers (or tags) to certain data, and construct the structural relationships between them, so that they may be recognized and acted upon during future processing. XML by itself specifies neither preconceived semantics nor a predefined tag set for data.[27] However, several XML-related standards and technologies provide means for defining content and semantics of XML documents. Inherent within these standards are many good features that would be well suited to the task of satisfying multidisciplinary data requirements.

As indicated in the previous section, data sharing is an essential element in preventing discipline isolation. XML provides a hierarchical container that is platform-, language-, and vendor-independent, and separates the content from any environment that may process it. XML is normatively simplified from an existing ISO standard, ISO 8879 (SGML),[11] and is an acceptable candidate for full use within other ISO standards without 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 programming language that processes 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.

When using XML, it not only allows input of 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.

Currently, Web-based technology for automatic generation and parsing of XML content is abundantly available and can be used to leverage the development of an infrastructure that will alleviate the data/model exchange and integration in a collaborative aircraft design environment. Like HTML, XML is a Web-enabled specification and standard that both Netscape™ and Microsoft™ have endorsed and are included in their latest browser products. However, XML is not HTML. In HTML, both the tag semantics and the tag set are fixed. A key difference between XML and HTML is that XML itself is a *metalanguage*—a language can be used to develop an unlimited number of special-purpose languages. One means to supply the XML data *semantic* can be achieved through the use of Document Type Declaration (DTD)[10] or XML-Schema,[14] which defines the data contents and logical constraints on how an XML document should be constructed. An added benefit from DTD and XML-Schema is that they enable data validation support. A data file written in XML is considered "valid" when it follows the constraints that the DTD or XML-Schema lays out for the XML data. The 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, any *validating* XML parser can be used readily to check not only the syntax of data, but also the validity and integrity of the aircraft data provided that a DTD/XML-Schema is offered. This guarantees that the data producer and consumer exchange the aircraft disciplinary data correctly, and plays an essential role in applying new Electronic Data Interchange (EDI) technology to aircraft design.

EDI is defined as[52] "the computer-to-computer exchange of business data in standard formats." Traditionally, electronic business messages are normally interchanged via standard EDI protocols (including the X12 (Ref. 53),

*There are also three common ways to store binary data in XML documents: Base64 encoding (a MIME encoding that maps binary data to a subset of US-ASCII [0-9a-zA-Z+/] — see [RFC 2045]), hex encoding (where each binary octet is encoded using two characters representing hexadecimal digits [0-9a-fA-F]), and unparsed entities (where the binary data is stored in a separate physical entity from the rest of the XML document).

and EDIFACT[54] series), and their communications are carried out over proprietary Value Added Network (VAN). The EDI protocols' brittle syntax has repeatedly been criticized for poor design, confusing or absent semantic, etc.[55] XML, with its ability to transfer open, standard and semantic data over the Web, has opened a new paradigm, called XML/EDI, for business data exchange. XML/EDI shines as it allows a system to encode the document's information much more precisely and in a very rich structure using XML. Moreover, XML/EDI transactions can take place over the Internet, which are more economical than traditional EDI messages, while being easier to validate and translate into the formats needed by applications at each end of the exchange.[56] In traditional aircraft multidisciplinary analyses, validating data format and ensuring content correctness are among the major hurdles in achieving exchange of aircraft data.

XML documents are also easily committed to a database (relational or object) or any other kind of native XML databases. There are many XML-enabled (such as Oracle,[28] Microsoft SQL Server 2000,[29] etc.) or XML native databases (such as dbXML,[30] Xindice,[31] etc.) available that deliver this type of functionality. This lends XML to be used in the large-scale distributed aircraft design process where the information is used by different components and/or systems groups. XML documents normally fall into two broad categories: data-centric and document-centric.[32] Data-centric documents are those where XML is used as a data transport. They include scientific data, such as CFD, FEM, CAD, etc. Their physical structure (e.g., the order of sibling elements, whether data is stored in attributes or PCDATA-only elements, etc.) is often unimportant. Document-centric documents are those in which XML is used for its SGML-like capabilities, such as in design manuals, or static aircraft maintenance log. They are characterized by irregular structure and mixed content and their physical structure is important. To store and retrieve the data in data-centric documents, an XML-enabled database is often used, as it is best suited for maintaining the integrity of data, but some sort of data transfer software are needed. The query of data can be done by standard Structured Query Language (SQL).[33] To store and retrieve document-centric documents, a native XML database or content management system normally is a better approach. Both of these are designed to store content fragments, such as procedures, chapters, etc. In this case, XML-based query languages, such as (XPath,[34] XQuery,[35] etc.) are used to question the databases.

B. XML-based Scientific Specifications Pertinent to Aircraft Disciplines

XML has been enthusiastically adopted by many industry work groups and companies to develop their standard engineering application models. Many applications became available very quickly and are very relevant to the engineering data exchange endeavor. Examples include: XML/Data Extension tool[36] from Autodesk Inc. for its AutoCAD 2000i family of products; eXT, an XML-based 3D MCAD interoperability standard around UGS's Parasolid XT format;[37] and 3D Instant Website[38] from Solidworks, which enables users to easily create, modify and display 3D data on a Web site using XML standards and XSL[39] templates.

In addition to these proprietary efforts, some science disciplines (including mathematics, chemistry and biology, etc.), and the scientific computing community (e.g., XSIL,[40] MatML,[41] femML,[42] STEPml[43]) are developing XML-based data exchange languages. The core part of these languages is the use of DTD/XML-Schema to define their data semantics. Some of these schemas are very useful in the interchange and modeling of aircraft disciplinary data, and are briefly described below. These schemas can be dynamically incorporated into the research work presented herein (see Section 6).

MatML[41] is an XML-based materials-oriented specification. Being coordinated by the National Institute of Standards and Technology (NIST), MatML targets multiple industries for facilitating the exchange of a wide variety of material properties. The current version (MatML3.0) ported the language specification from DTD to XML Schema, and has many refinements over pervious version, including redundancy elimination, units content model; measurement uncertainty, etc. The ultimate goal of MatML is to provide for direct program-to-program interoperability, efficient data processing, and rapid, reliable, and useful response to searches for materials data over the Web.

femML[42] is an XML variant to aid in the data interpretation and interoperability in the Finite Element Modeling domain. The project was initiated by members of the Composite Materials and Structures group at the Naval Research Laboratory and the International Science and Technology Outreach Society. The focus of this work was to utilize the XML's power of semantic encapsulation along with the existing and continuously improving associated technology to develop a dialect for exchanging FEM data across various codes with heterogeneous input formats. MatML is used as a Namespace[44] in the specification. The current DTD implementation (v. 1.02) follows a strict FEM data file structural architecture.

STEPml[43] takes the data models from STEP,[18] the international product data representation and exchange standard, and publishes them as XML specifications using DTDs and/or XML Schemas. The resulting specification

brings together the rich semantics of STEP and the widespread adoption of XML technology. STEPml is sponsored by PDES Inc, an international industry/government consortium.

Another use of XML was recently proposed by NASA Ames Research Center, U.S. Naval Air Systems Command, etc., in their DAVE-ML project.[45] The goal of the project is to develop an XML application to encode complete aircraft flight vehicle dynamic models in a language-independent, consistent way to expedite model exchange and validation between different simulation facilities and tools. Jackson et al.,[46] who introduced the concept of DAVE-ML, estimated potential savings of over $6M in productivity gains and reduced simulator down-time for a single aircraft type in one year from adopting a standard such as XML for vehicle model updates. Currently, as reported in the literature, DAVE-ML is still in its initial development status and not yet implemented.

C. Further Issues in XML-based Aircraft Database Design

The various advantages outlined above present compelling reasons to use XML for aircraft design data interchange and database management. However, the solution is not as easy as it might at first appear, as outlined next.

1) The interdisciplinary and multidisciplinary coupling inherent in MDO tends to present additional challenges beyond those encountered in a single-discipline optimization. The independent industrial initiatives mentioned above generally focus on describing single disciplinary data, and most of them do little to facilitate interaction across industry and functional boundaries. In addition, a vendor has obvious incentives for describing its offerings in ways that highlight its competitive advantages and obscure comparison on features where it lacks an advantage.

2) 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. Although some industry standards have been designed that allow schematically interpret single disciplinary data, still many other important disciplinary specifications (such as maintainability, control, etc.) involved in aircraft design are required. Even if they will become available soon, XML by itself does not enable plug-and-play to easily use and share data. This presents another challenge in interchanging the aircraft disciplinary design data.

3) Current discipline-pertinent specifications often fall short in adequately uniting several other data exchange design requirements (for example, large datasets, object-oriented, high-level interface, distributed, etc.) in order to effectively use XML to communicate aircraft design data between disciplines. Increasing disciplinary interactivity is not accomplished effectively by just putting more and more data and functionality onto clients. Good multidisciplinary aircraft database design means making application implementation easier, allowing for quicker reach of disciplinary data into affected disciplines, improved performance when processing transactions, and most importantly, ensuring aircraft design data consistency and security using proven database mechanisms.

A successful collaborative aircraft database modeling system should at least combine a good level of interoperable data with enough extensibility for data interchange and management. We propose that an XML-based database model employing reusable semantic components common to many aircraft design domains will meet those requirements. Such a model can be easily understood by aircraft designers, while it also provides a common mechanism for linking to the current, unique industry specifications, thus enabling users to benefit from any and all available specifications. In the next section, we will present the design of such an aircraft data object model. This model is also used to interpret aircraft design data between design disciplines, and serve as a foundation to implement an XML-based aircraft database to meet all of the above design requirements.

IV. 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 specifications. 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 structure was first designed. Figure 2 shows an overall layout of the static model, simplified for brevity. 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.

A. Aircraft Components

An aircraft component (*AC_Component*) object can be an engine, fuselage, landing gear, canard, horizontal stabilizer, vertical rudder, wing, etc. (Fig. 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.

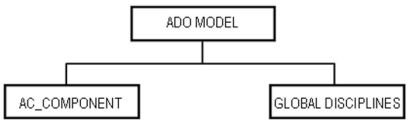
Each aircraft component itself may be made up of physically distinguishable *subcomponents* or *parts*. For example, an engine is made up of inlet, fan, compressor, combustor, turbine and nozzle subcomponents. Likewise, most landing gears have parts like wheel, tire, brake assembly, etc. Every subcomponent 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 Fig. 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 structure layer of data objects, the designed model also encourages the use of data object *abstraction*, *inheritance*, and *composition*. Referring to Fig. 2b, it can be seen 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 object. 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 model totally extensible for future enhancements.

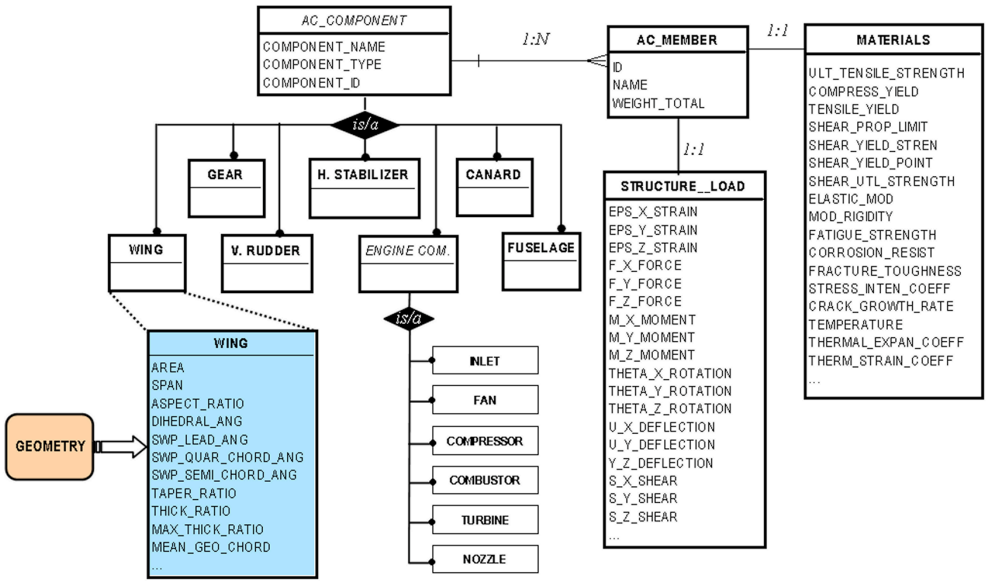
B. Geometry Modeling

Component geometry modeling is somewhat unique in aircraft design. All disciplines-specific analyses share the same geometry. Strong interactions between the disciplines are very common and thus necessary in analysis. 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 would benefit from a common geometry dataset that can be manipulated and shared among various disciplines.[17]

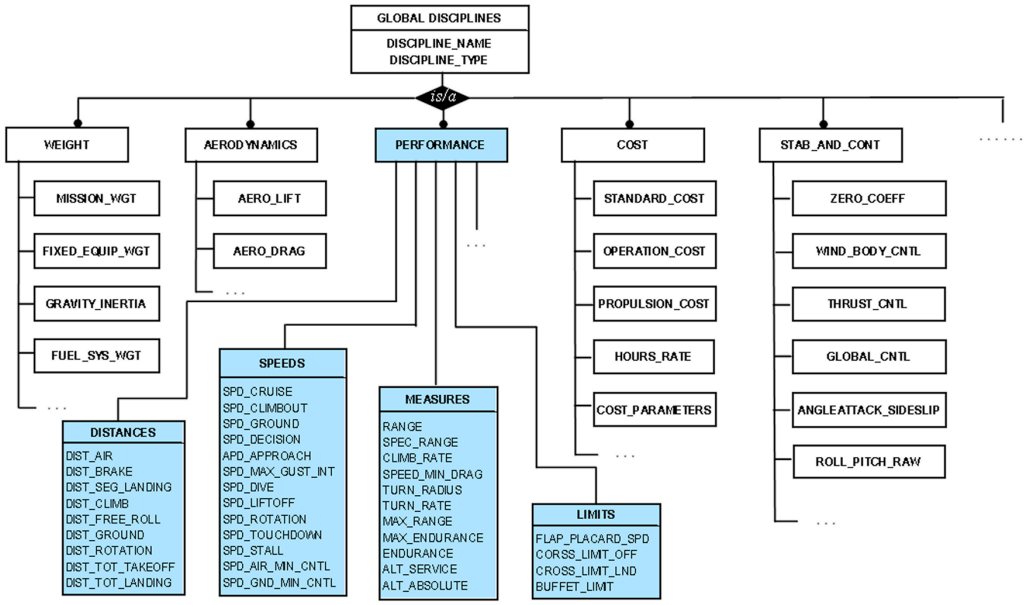
STEP Application Protocol AP209 - Composite and metallic structural analysis and related design[57] - is a set of standards that specifies computer-interpretable composite and metallic structural product definition including their shape, their associated finite element analysis (FEA) model and results, and material properties. AP209 supports not only CAD geometry modeling, such as topology, assemblies, and configuration management data of solid models, but also the associated finite element representations for mechanical parts. 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. The placeholder can also accommodate other CAD/FEA-related industry models, such as the model defined in femML. 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.



(a) Top level children of ADO model



(b) Aircraft components data object model



(c) GlobalDisciplines data object model

Fig. 2 A simplified Aircraft Data Object (ADO) static structure. Each ADO may hierarchically encapsulate many other children ADOs.

C. 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 (Fig. 2c). This may seem 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 of 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 Fig. 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.

V. 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 root-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 Fig. 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 *GlobalDisciplines* 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. 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.

Next, 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_t` 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. For example, *DataArray* complexType is an atomic type designed in the schema, using the list simpleType definition (xsd:list) combined with union type (xsd:union) to model arrays, vectors or multiphysics fields in the simulation. 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 at the design time. For example, Fig. 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.

Our database model currently uses STEP AP209 standard to encode aircraft geometry disciplinary data (CAD and FEA data). In addition, the database allows other industrial standards, such as femML and MatML (see Section 6.3), to be added in dynamically. STEP models are created using the EXPRESS language.[19] It 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 CAD geometry and/or FEA information, it is possible to define a set of schema languages (such as DTD or XML-Schema) that can be used to encode the information specified in EXPRESS schema. Several works 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 the aircraft CAD and FEA data. This is done by designing a *STEP_DTDConverter*, 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 disciplinary data. Representation of the STEP model using XML-Schema also benefits from the good features in XML-Schema, such as modular schema inclusion (XInclude),[12] and also offers a uniform data schema formalism for database implementation. A simple illustration is given in Fig. 4.

VI. Aircraft Databinding

Simulation-based 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 an external storage medium is still not intelligible or easily accessible. Aircraft databinding provides an implementation for the data object model designed in the present research. 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;[21] thus the software can be run on different design platforms. Aircraft Databinding Framework closely follows the common XML databinding architecture. As a result, any of the available software that adheres to the architecture (such as JAXB,[47] Caster,[48] Zeus,[49] etc.) is pluggable into our design. There are several reasons why the available software is not used in the present work. First, when the present work began in 1999, JAXB was still a Java Specification Request (JSR-031) (Ref. 50), thus the authors had to design and develop their own. Second, most of the other databinding, such as Zeus, are DTD-based and are not XML-schema based. JAXB was also DTD-based for a long time. Our databinding also supports many advanced data types in XML-schema, which are heavily used in our aircraft schema, but are not yet fully implemented by other data databinding software. For example, *list* and *union* types are very important for DataArray type and are currently not supported in Castor. To the best of our knowledge, the current JAXB 1.02 does not support schema features such as *abstract*, instance *type substitution* (xsi:type), etc., which are essential in modeling inheritance in the ADO model. Additionally and most importantly, we believe performance is an important consideration in aircraft design applications and hence a great deal of attention has been, and will continue to be, paid to our databinding framework with the objective of improving performance. The databinding together with other modules designed in our aircraft database are shown in Fig. 7, and are further described next.

```
<?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="idetification" 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>
```

Fig. 3 A sample aircraft schema.

```
<!—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-id" type="xsd:ID" use="required"/>
            <xsd:attribute name="X" type="xsd:string" use="required"/>
            <xsd:attribute name="Y" type="xsd:string" use="required"/>
            <xsd:attribute name="Z" type="xsd:string" use="required"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Fig. 4 An example of using *STEP_DTDConverter*, showing conversion from DTD to Schema for three-dimensional point representation.

A. 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 (see Fig. 5).

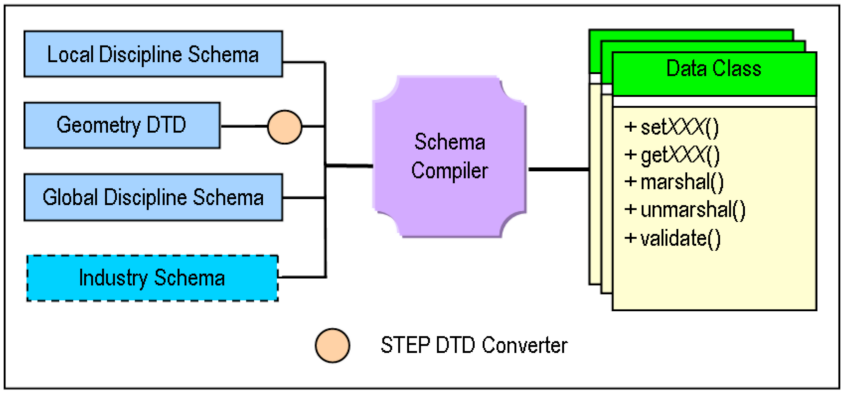


Fig. 5 Schema compiler in aircraft databinding.

Consider how a 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. For *DataArray* type attributes, *java.util.Vector* is generated, where each member type is the actual data type inside the *DataArray*. 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. Clearly, 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)[22] diagram of generated Java class (only wing component is shown) is illustrated in Fig. 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 to access and modify the properties defined in the aircraft Schema. These methods closely follow the JavaBean Design Pattern.[23] 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.

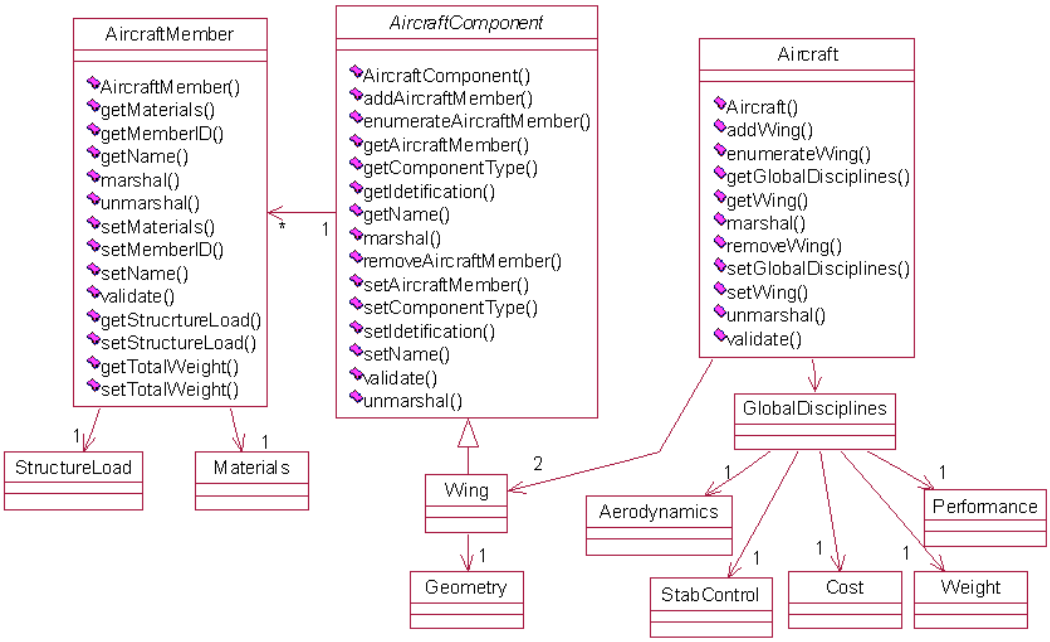


Fig. 6 UML for data class structure generated by aircraft databinding.

B. Marshalling Framework

The marshalling framework supports the transportation (unmarshal) of aircraft data in XML files into graphs of interrelated instances of aircraft data objects that are generated during schema complier and also converts (marshal) such graphs back into XML file. For example, when XML-based wing data are correctly unmarshaled into 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 interface aircraft data 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 data objects devised with a set of mutator and accesor methods to work with the aircraft data. Therefore, the marshalling framework 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.

C. Dynamic Addition of External Schema

Aircraft databinding also provides a service that enables dynamically add-in new aircraft disciplinary schemas, such as maintainability, productivity, etc. 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. For example, *STEP_DTDConverter* developed in this work, 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 and FEM modeling (Fig. 5).

Industry specifications that are currently available (such as MatML, femML, etc.) are valuable within the aircraft schema development context. The current framework allows users to readily utilize these specifications. Figure 7 shows an example of integrating MatML 3.0 schema into the designed database in aircraft databinding. In general, different disciplinary schemas are included as separated namespace[44] for the entire aircraft schemas, and the aircraft databinding will generate separate Java packages to hold the different disciplinary data objects. For example, MatML schema was compiled into Java package *ado.material.external* in Fig. 7. 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.

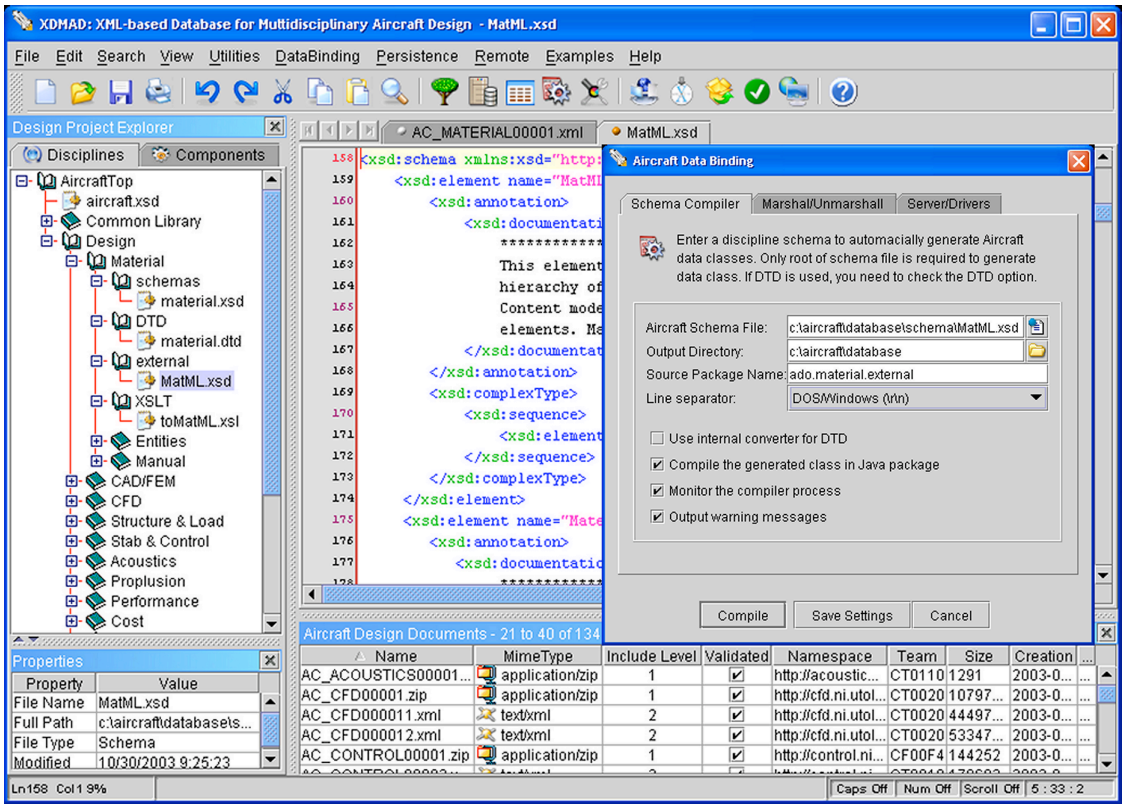


Fig. 7 XML-based database for multidisciplinary aircraft design, showing: 1) the integration of aircraft discipline schemas with the designed database management system; 2) dynamically new disciplinary schema add-in - MatML 3.0; 3) ZipArchiver and XIncluder/XSplitter use to improve database performance.

D. 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 (Fig. 8). 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. 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.

E. Performance

Since XML representation of aircraft data is likely 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. An example is shown in Fig. 7 for the simulation results of a CFD case - AC_CFD00001, which was split into 'AC_CFD000011.xml' and 'AC_CFD000012.xml' subfiles. These two files will be resembled together later when unmarshaling XML-based aircraft data to their data objects. This kind of flexibility allows the entire aircraft data files to span among multiple physical files, possibly residing in different computers by referencing as Uniform Resources Identifiers (URI). In addition, this feature makes 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 use less memory to represent the entire layered tree of the aircraft data nodes. In addition, a *ZipArchiver* is included in the aircraft databinding, which compresses 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 data file size can be much smaller than the original XML file size, and even compatible in size with binary representation of the same data. This reduces file I/O access times and improves performance for large aircraft datasets.

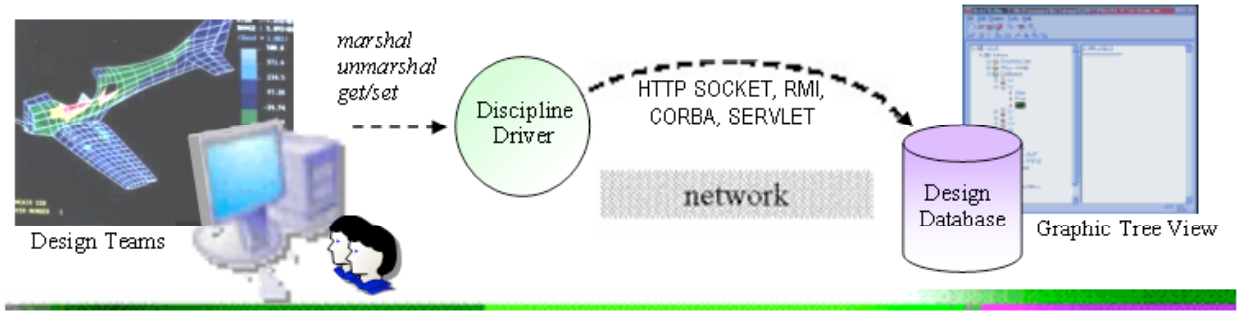


Fig. 8 Design teams use Marshal Framework to access aircraft data remotely

VII. Data Persistence Engine

Aircraft databinding provides an easier way to access sharable disciplinary data. However, it still lacks certain features found in a real database: security, transactions, data integrity, multi-user access, and so on. Databases and XML offer complementary functionalities for storing data. Databases enable efficient transactions between the user interface and the rest of the functional modules within the applications whereas XML offers an easy information exchange that enables interoperability between applications. An underlying database persistence engine is designed to let the user store aircraft data to XML-enabled or XML-native databases.

A. Aircraft Data Store for Persistence

The main goal of an aircraft persistence engine is to have a persistent storage mechanism that is independent of its environment and that is invisible to the simulation users. This is achieved internally by different aircraft *DataAdapters* to connect to disparate data sources, coupled with an internal data transaction and persistence tools for data transfer, data queries, etc. Each *DataAdapter* dynamically integrates diverse data sources and makes them readily available for use within the aircraft design. As mentioned in Section 3, an XML-enabled database is normally used to transfer data-centric aircraft file (such as CFD, FEM, etc.) between a disciplinary document and a database. Transferring a data-centric aircraft document to a relational or object-oriented database requires first creating a mapping between them. This is done by model-driven mapping.[58] In a model-driven mapping, a data model of some sort is imposed on the structure of the XML-based aircraft data file and this is mapped, either

implicitly or explicitly, to the structures in a database and vice versa. Since aircraft data are stored in XML based on the ADO data object models, it is natural to use this model to drive the mapping as well.

Given a set of aircraft schemas that closely reflects ADO models, the database mapping can be automatically generated from the schema by a *MapGenerator*. A *MapGenerator* generates the object view of the element types, attributes, and aircraft data in an XML file and how to map this view to the object/relational database schema. In addition, the *MapGenerator* can specify the database view of all the tables, rows, and columns and how to map them back to elements, attributes, and disciplinary data in an XML file. The map object also maintains the hierarchy relationship between aircraft data specified in ADO model to ensure that all the existing internal relationship will not be broken during transferring from database to XDF and vice versa. As Java Database Connectivity (JDBC)[51] is used in the underlying transfer, this mapping is hidden from the designers during the data transfer, and can work with any object-oriented or hierarchical databases (Fig. 9).

On the other hand, native XML databases are used to store document-centric data, such as test reports, flight maintenance logs and design manuals. Like other databases, native database support features like store, retrieve, transactions, security, multi-user access, programmatic APIs, query languages, and so on. The difference from other databases is that their internal models are based on XML and not something else, such as the relational model. Therefore, a native XML database does not ask the user to worry about how to map XML structures onto some non-XML underlying data model or processing language. Instead, users can continue to use the native XML databases tools that are wrapped by aircraft *DataAdaptors* for data transactions.

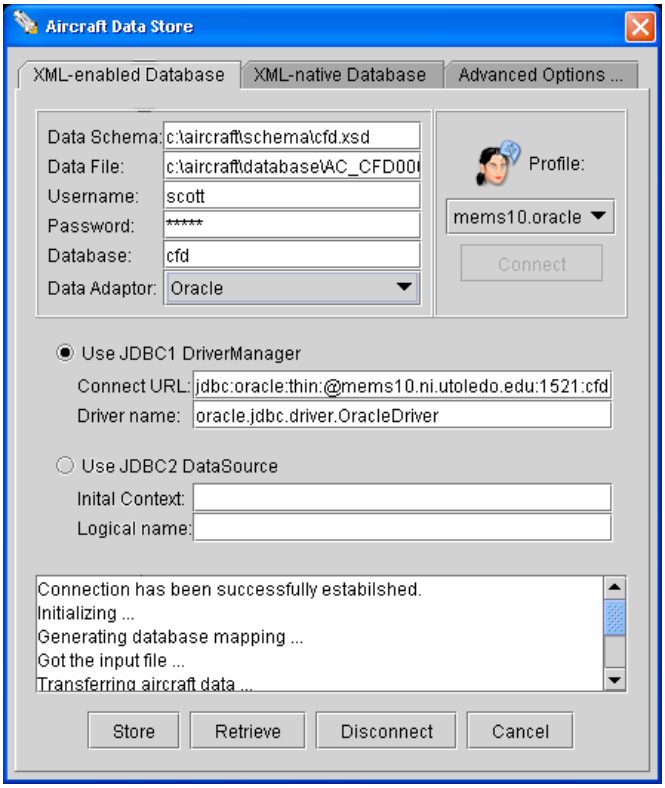


Fig. 9 Aircraft Data Store enables the persistence of XML-based disciplinary data to XML-based or XML-enabled database.

The large pool of data stored by participating aircraft teams improves the capability of analyses to correlate data and identify rare phenomena that may have gone undetected in the past until they caused a serious incident or accident. Because the data collected by aircraft persistence engine comes from many different disciplines, the database systems can also facilitate the exchange of lessons learned among manufacturers. This is important when a problem is relevant to aircraft produced by more than one manufacturer. Also, manufacturers have the equipment to store and analyze these data and have the most detailed understanding of their own products.

B. Opportunities for Electronic Data Interchange

One of the XML/EDI objectives is to make the transfer of documents between organizations as easy as possible to the point that it is transparent to the user. With *DataAdaptors* designed for persistence engine, data objects generated by aircraft databinding process and distributed drivers via the use of URLs, the overall aircraft database architecture aids in interchanging data to legacy applications. Validation, cross-reference look-up tables, and most importantly, application logic can be used during the interchange process. Each designer effectively becomes a trading partner, sending and receiving transactions, internal and external to the organization. Even if design and manufacture clients do not support XML natively, it is not a major hindrance. Industry companies can easily write standard translator codes (using Java codes, XSLT, or other scripting language, etc.) that reflect industry rules, to provide for greater mapping and integration of data exchanged over XML-based database. For example, the aircraft data object APIs provided by aircraft databinding provides a user the tools to access and manipulation design data easily. If these objects and different design companies' electronic commerce (EC) mechanisms are applied to our designed database, we move from data only inferences to apply XML-based agent technology to assist in the design challenges.

VIII. Conclusion

In this work, an XML-based database model for use in multidisciplinary aircraft design has been designed and implemented, which meets design requirements of diverse disciplines. The database consists of aircraft data object models, data schemas, databinding and a database persistence engine. Aircraft Data Object (ADO) model encompasses most 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 AP209 standard is used to describe each component's geometry and FEM data. The ADO model precisely defines the organizational structure supporting aircraft design data and the conventions adopted to standardize the data exchange.

A set of database schemas was designed based on the ADO model in order to store and validate XML-based aircraft data. By using XML Schema to represent aircraft Schema, a set of constraints are established to construct and to further schema-validate the aircraft data. The database schema follows a modular design pattern such that it is extensible for future addition, modification, or 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.

The database persistence engine offers facilities to simultaneously work on independent tasks in a product development process, as well as facilities to collaborate on the design of aircraft components. These include, among other things, an aircraft data store with *DataAdaptors* and *MapGenerator*, and a powerful XML/EDI-enabled collaborative interchange and validation scheme. The designed XML-based aircraft database can exist on the server, the client, or both. Combined with aircraft databinding and XML/EDI technology, the aircraft database persistence engine facilitates the exchange of different types of design data between industrial organizations. Together with the integration of aircraft data modeling, this development means a major step forward in collaborative 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.

References

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

[2]Kroo, I., Altus, S., Braun, R., Gage, P., and Sobieski, I., "Multidisciplinary Optimization Methods for Aircraft Preliminary Design," AIAA Paper 94-4325, *Fifth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Sept. 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]Krishnan, R., **"**Evaluation of Frameworks for HSCT Design and Optimization," NASA/CR-1998-208731, Oct. 1998.

[5]Salas, A. O., and Townsend, J. C. "Framework Requirements for MDO Application Development," *7th AIAA/USAF/NASA/ ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA Paper 98-4740, Sept. 1998.

[6]Jones, K. H., Randall, D. P., and Cronin, C. K., "Information Management for a Large Multidisciplinary Project," AIAA Paper 92-4720, *4th AIAA/AF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Sept. 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. S., Hart, W. E., Bohnhoff, W. J., Romero, V. J., Hutchinson, S. A., and Salinger, A. G., "Utilizing Object-Oriented Design to Build Advanced Optimization Strategies with Generic Implementation," *Proceedings of the 6th AIAA/USAF /NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA Paper 96-4164, 1996.
[9]Monell, D. W., and Piland, W. M., "Aerospace Systems Design in NASA's Collaborative Engineering Environment," *50th International Astronautical Congress*, Oct. 1999.
[10]World Wide Web Consortium, *Extensible Markup Language (XML) 1.0*, 3rd ed., Feb. 2004, available online at http://www.w3.org/TR/2004/REC-xml-20040204/.
[11]"Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)," ISO 8879:1986.
[12]World Wide Web Consortium, "XML Inclusions (XInclude) Version 1.0," Working Draft, Nov. 2003, available online at http://www.w3.org/TR/xinclude/.
[13]World Wide Web Consortium, "XML Linking Language (XLink) Version 1.0," June 2001, available online at http://www.w3.org/TR/xlink/.
[14]World Wide Web Consortium, "XML Schema Part 0: Primer," May 2001, available online at http://www.w3.org/TR/xmlschema-0.
[15]Raymer, D. P., *Aircraft Design: A Conceptual Approach, 3rd Edition,* AIAA Education Series, Washington, DC, 1999.
[16]Lin, R., and 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*, 2002.
[17]Samareh, J. A., "A Survey of Shape Parameterization Techniques," *CEAS/AIAA/ICASE/NASA Langley International Forum on Aeroelasticity and Structural Dynamics,* 1999.
[18]Pratt, M. J., "Introduction to ISO 10303–the STEP Standard for Product Data Exchange," *Journal of Computing and Information Science in Engineering*, Vol. 1, No. 1, 2001, pp. 102-103.
[19]ISO TC 184/SC4, ISO 10303-11: 1994: Industrial Automation Systems and Integration-Product Data Representation and Exchange-Part 11: Description Methods: The EXPRESS Language Reference Manual.
[20]ISO TC 184/SC4, ISO 10303-28: 2003: Industrial Automation Systems and Integration-Product Data Representation and Exchange-Part 28: Implementation methods: XML representation of EXPRESS Schemas and Data.
[21]Arnold, K., and Gosling, J., *The Java Programming Language*, Addison Wesley Publishing Company, Inc., Reading, MA., 1996.
[22]Booch, G., Rumbaugh, J., and Jacobson I., *The Unified Modeling Language User Guide. Object Technology Series*, Addison-Wesley, 1999.
[23]Watson, M., *Creating Java Beans: Components for Distributed Applications*, Morgan Kaufmann Publishers, Sept. 1997.
[24]The National Aviation Safety Data Analysis Center (NASDAC), available online at https://www.nasdac.faa.gov/.
[25]BASIS (British Airways Safety Information System), available online at http://www.winbasis.com/.
[26]Global Analysis and Information Network (GAIN), available online at http://www.gainweb.org/.
[27]Walsh, N., "What is XML?," available online at http://www.xml.com/pub/ a/98/10/guide1.html#AEN58, Oct. 1998.
[28]Muench, S. *Building Oracle XML Applications*, O'Reilly & Associates, Sept. 2000.
[29]Martinsson, T., *Scripting XML and WMI for Microsoft(r) SQL Server 2000: Professional Developer's Guide,* John Wiley & Sons, Jan. 2001.
[30]dbXML database, available online at http://www.dbxmlgroup.com/index.html.
[31]Apache Xindice, available online at http://xml.apache.org/xindice/.
[32]Chaudhri, B. A., *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison-Wesley Publishing Company, March 2003.
[33]Date, C. J., and Darwen, H., *A Guide to The SQL Standard*, *Third Edition*, Addison-Wesley Publishing Company, Inc., 1993.
[34]World Wide Web Consortium, XML Path Language (XPath) Version 1.0 Nov. 1999, available online at http://www.w3.org/ TR/xpath.
[35]World Wide Web Consortium, XQuery 1.0: An XML Query Language, Working Draft 12 Aug. 2003, available online at http://www.w3.org/TR/xquery/.
[36]Autodesk, "Autodesk Unveils Cross-Platform, Cross-Industry Open XML Strategy for Design Data and E-Commerce," Press release, *Directions Magazine*, 13 June 2000, available online at http://www.directionsmag.com/press.releases/index.php?duty= Show&id=1903.
[38]SolidWorks, "3D Instant Website," available online at http://www.solidworks.com/pages/products/solutions/3dinstantwebsite. html.
[39]World Wide Web Consortium, Extensible Stylesheet Language (XSL), Version 1.0, Oct. 2001, available online at http://www.w3.org/TR/xsl/.
[40]Williams, R., "XSIL: Java/XML for Scientific Data," White Paper, available online at http://www.cacr.caltech.edu/ projects/xsil/xsil_spec.pdf.
[41]Begley, E. F., and Sturrock, C. P., "MatML: XML for Material Property Data," ASM International's Advanced Materials & Processes (AM&P), Nov. 2000, available online at xml.coverpages.org/begley-ampmatml.pdf.

[42]Composite Materials and Structures Group, "FemML for Data Exchange between FEA Codes," *ANSYS Users' Group Conference*, Univ. of Maryland, College Park, Oct. 2001, available online at http://femml.sourceforge.net/download/femMLPaperScr12.pdf.
[43]STEPml, sponsored by PDES Inc, available online at http://www.stepml.org/index.html.
[44]World Wide Web Consortium, Namespaces in XML, Jan. 1999, available online at http://www.w3.org/TR/REC-xml-names/.
[45]DAVE-ML Project for Flight Dynamic Model Exchange Using XML, available online at http://dcb.larc.nasa.gov/utils/fltsim/DAVE.
[46]Bruce, J. E., and Hildreth, B. L., "Flight Dynamic Model Exchange using XML," AIAA Paper 2002-4482, Aug. 2002.
[47]Sun Microsystems, "Java Architecture for XML Binding (JAXB) Specification 1.0 - Final Draft," available online at http://java.sun.com/xml/downloads/jaxb.html.
[48]Castor XML Databinding, available online at http://www.castor.org/.
[49]Zeus project, available online at http://zeus.enhydra.org/.
[50]Sun Microsystems, "Java Specification Requests JSR-031: XML Data Binding Specification," available online at http://web1.jcp.org/en/jsr/detail?id=31.
[51]Reese, G., *Database Programming with JDBC and Java*, 2nd ed., O'Reilly & Associates, Aug. 2000.
[52]ASCX12, "What is EDI?" available online at http://www.x12.org/x12org/about/.
[53]Accredited Standards Committee X12 (ASCX12), available online at http://www.x12.org/.
[54]UN/EDIFACT Standard Directories, available online at http://www.unece.org/trade/untdid/.
[55]Kimbrough, O. S., "EDI, XML, and the Transparency Problem in Electronic Commerce," INFORMS Fall 2000 Meeting, Nov. 2000.
[56]Laplante, M. "Making EDI accessible with XML," EC.COM 4, 2 March 1998, pp.23–26.
[57]TC 184/SC 4, ISO 10303-209:2001, Industrial automation systems and integration – Product data representation and exchange – Part 209: Application protocol: Composite and metallic structural analysis and related design.
[58]Williams, K., *Professional XML Databases*, Wrox Press Inc., Dec. 2000.