FemML for Data Exchange between FEA Codes

CMS Group*****

*****Composite Materials and Structures Group, Code 6304
J. Michopoulos, P. Mast, T. Chwastyk, L. Gause, R. Badaliance,
U.S. Naval Research Laboratory
Washington DC, 20375
johnM@cms.nrl.navy.mil

Abstract

The **f**inite **e**lement **m**odeling **M**arkup **L**anguage *(femML)* effort is addressing the problems of data interpretation and application interoperability in the Fninite Element Modeling domain. This is achieved through the development of an extensible markup language (XML) for finite element model data that will permit the storage, transmission, and processing of finite element modeling data distributed via the World Wide Web and related infrastructure technologies. 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 format syntactic specifications. The main aspects of a finite element definition have been used as archetypes for defining the XML element taxonomy definitions. Namely, the geometry, the material, and the loading aspects of a structural component specification are used to define the first level elements of the associated Document Type Definition (DTD). The element list has been amended with a behavior element specification that represents the solution data to be exchanged or visualized. Utilization of the MatML standard for material property data exchange is demonstrated. Various tools have been developed to demonstrate associated concepts along with the ANSYS set of tools.

1. Introduction

1.1 General problems

The main problems associated with all computationally assisted data exchange, interchange and integration activities can be approached from multiple points of view depending on the needs at hand. However, there is a global point of view that is common to all industries in need of data exchange. In the engineering industries it unfolds as a need for *integration* of FEM models encoded in multiple data formats from multiple data sources, with current end-user applications and future data exchange systems between applications. However, data *interpretation* (semantics) varies from data source to data source and therefore introduces semantic correctness uncertainty that destroys robustness of *interoperability* between applications and data receptacles in general.

The CMS group at the NRL experienced this issue from a very close distance when the time came to implement the Data Driven Design Workbench (D^3W) used as a virtual wind tunnel environment for design of composite structures and qualification and certification of composite materials systems [1].

Figure 1 shows the block diagram of D^3W's main components and their relationship in terms of data flow paths. This diagram is intended to represent its abstract architecture in order to expose the major pathways (non-blue lines) of structured data with FEM characteristics.

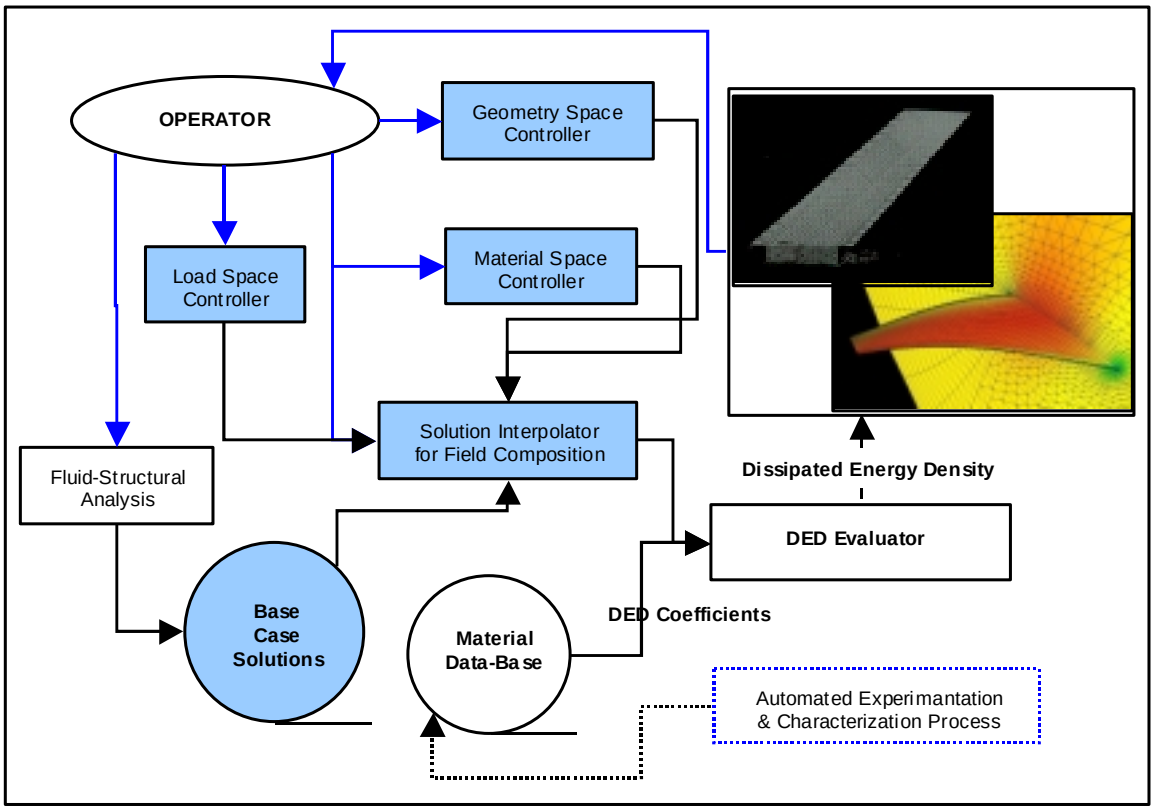

Figure 1. Architecture of Data Driven Design Workbench

This architecture has evolved through time and the whole environment along with the Dissipated Energy Density (DED) methodology have been utilized in various applications including health prediction and sensor optimization of smart structures [2-4].

All the non-blue lines in this block diagram signify data flow that invites the implementation of some highly structured data format for satisfying the data transfer requirements. One of the most dominant issues on the development of this environment was clearly the semantic biasing and preference from each one of the modules as well as the lack of a common representation for the exchanged data. It is exactly for cases like this that eXtensible Markup Language (XML) has been used to develop the finite element Modeling Language (femML).

In addition to this general need, the proliferation of the specific needs of particular domains of application generate a science push for solving the data structure and meaning heterogeneity problems within the pertinent vertical industry context. More specifically, digital content with the WWW as a transport medium is available in many forms i.e. multiple commercial applications, manufacturers data-sheets, materials databases, research and development electronic publications, neutral and custom file formats etc. The need for collaborative dynamic computing through the WWW, strengthens the push for solving the heterogeneity problem by imposing a demand for distributed applications, for problem solving environments, for virtual design and prototyping and for agent-based applications. On the other hand, the multi-industry support and proliferation of XMLware, the Java-Database-XML integration technology, and the XML middleware plethora create a technological pull for the utilization of XML-based solutions.

1.2 Recognition of the problem by the industry

The data exchange, interchange and integration problems have been recognized very early by multiple industries of human activity that entails data transfer.

Industrial automation technology has improved dramatically over the past decades. CAx systems ("Computer-Aided anything", or: CAD, CAM, CAE, CIM, etc.) have provided engineering applications with high-performance solutions. Integration of these technologies is a major issue for industrial competitiveness. From numerical control (NC) in the fifties, through the first design graphics applications and computer controlled production operations in the sixties, Computer Numerical Control (CNC) and Distributed Numerical Control (DNC) in the seventies, and Flexible Manufacturing Systems (FMS) and solid model-based design workstations in the eighties, automation technology has continued to advance and become more sophisticated in order to meet the individual needs of industry. In terms of horizontal integration the CAD industry has responded to geometry data integration and exchange with multiple specific file format specifications. Examples are ACIS, Parasolid, IGES (flavored & standard), STEP, STL, VDAFS, CATIA, CADDS5 etc. [5]

However, as industry moves into the 21st century, a new industrial need is becoming the critical problem to solve: the vertical integration of these diverse automation systems (e.g., CAD, CAM, CIM, CAE) [6].

The complex nature of engineering data may hinder the integration of engineering applications. The major "stumbling blocks" that prevent the effective integration of CAx systems are [7]:

1. Current CAx systems have been designed to input and output data rather than information; and
2. Current CAx tools operate on different levels of abstraction of the mechanical product.

Therefore, information (data with meaning) modeling is a major issue for CAx systems integration. Moreover, data has to be transferred between applications.

An obvious recognition of the importance of XML for information exchange in general, can be evidenced by the plethora of special XML variants developed by and for many non CAx industries.

1.3 Attempts to solve the CAx data exchange problem

1.3.1 Non XML Efforts

The ever present need for data translation to fit the receiving system's data model has been identified as the dominant problem of application integration. To deal with this problem, the International Standards Organization (ISO) launched the STandard for the Exchange of Product model data - STEP (ISO 10303-1 1994) [8], aimed at the representation of all information about a product throughout its entire life cycle.

STEP allows different applications to exchange information using a standard format. All data models in STEP are normalized (i.e., in conformity with the normal forms, described in section 2.1.2) and written in EXPRESS (ISO 10303-11 1994) [9], an "object-flavored information model specification language" [10] allowing for the specification of complex data models with multiple inheritance.

Relative to finite element modeling efforts, the data exchange problem has been traditionally cast under the framework of the product data exchange (PDE) category for most of the historical efforts. Early data exchange specifications focused primarily on geometrical data. Among these were proprietary specifications like Autodesk's DXF, and national standards such as IGES (United States), SET (France), and VDA/FS (Germany). The most significant of these efforts in terms of FEM data representation, is the AP209 ISO/DIS 10303-209 or the STEP 209 Composite and Metallic Structural Analysis and Related Design standard[11]. STEP is a complex standard with huge-sized documents, and was developed as if it was a database itself, adopting the ANSI/SPARC architecture for database systems [12]. Its' most

significant characteristic is that it allows transfer of conceptual information content in addition to raw data. The standard is comprehensive and is made out of a very extensive but well structured document series [8]. Perhaps its massive specifications and custom and proprietary related tool availability are its two greatest disadvantages.

1.3.2 XML Efforts

With the advent of XML and especially since its being adopted as a standard specification by the World Wide Web Consortium (W3C) on 1998 [13], many applications became available very quickly. However, only recently we have seen a utilization of XML technology for the engineering applications industries.

It's indicative of the trend that proprietary products to translate STEP documents to an XML form have already been developed to facilitate STEP document transfer. Some companies have decided to integrate XML technology with their existing product lines.

Router Solutions Inc. has recently announced [14] their custom CAx integration solution strategy.

eXT, is an XML-based 3D MCAD interoperability standard recently introduced by UGS. In short, UGS is wrapping XML around its Parasolid XT format [15,16].

Autodesk Inc.[17] has also recently announced the new XML/Data Extension tool for its AutoCAD 2000i family of products. The XML/Data Extension is part of a broader Autodesk XML initiative to create a common, open standard for delivering design data to the Web, ensuring compatibility between products in different segments of the design industry, and facilitating e-commerce around design specifications. In addition, because of the cross-platform, cross-industry acceptance of XML, the XML/Data Extension will allow developers to create tools that help designers share their design data with other mainstream business functions such as marketing, sales, operations, and customer support.

The aecXML Project was initiated in August 1999 by Bentley Systems, Incorporated with the desire that it be a unifying force for progress in the development of a project communications framework for architecture/engineering/construction (A/E/C). Bentley has developed an initial specification for aecXML, a framework of XML-based schemas to facilitate communications related to designing, specifying, estimating, sourcing, installing and maintaining construction products and materials over the Internet. Building on the success of aecXML, Bentley and Bluestone have entered a three-year agreement to develop engineering software solutions based on Bluestone's Sapphire/Web Application Server, Bluestone XML Suite™ Integration Server, and Bluestone's comprehensive standards-based, e-business solution [18].

In addition to this proprietary efforts there are three public domain efforts very relevant to the engineering data exchange endeavor. These are the Extensible Scientific Interchange Language (XSIL), X3D (the successor of VRML) that was just released the summer of 2001 and the MatML work in progress for material properties exchange applications.

The Extensible Scientific Interchange Language (XSIL) is a flexible, hierarchical, extensible, transport language for scientific data objects. It has been developed at Caltech by Roy Williams [19]. The entire object may be represented in the file, or there may be metadata in the XSIL file, with a powerful, fault-tolerant linking mechanism to external data. The language is based on XML, and is designed not only for parsing and processing by machines, but also for presentation to humans through web browsers and web-database technology. It comes with a Java object model that is designed to be extensible, so that scientific data and metadata represented in XML is available to a Java code. There is also a powerful Swing-based object browser called Xlook that is also designed to be extensible.

The X3D file format was created to substitute VRML for web based 3D geometries by the WEB3D consortium [20] is basically an XML version of VRML [21] in order to enhance functionality, portability

and leverage the Java-XML resources that have been created to support the e-business industry. It can be thought as an XML-interoperable scene graph architecture and encoding standard.

Both of these public formats are very useful to the present effort because they represent an extended body of work capable of dealing with the geometry encapsulation, representation and visualization of FEM geometries.

The MatML effort [22] is being coordinated by the National Institute of Standards and Technology, and is driven by the MatML Working Group, whose members include several ASM International Fellows, and members from various cross industry organizations. The main goal of this effort is the development of the MatML DTD, and associated examples and applications, that will facilitate the transfer, exchange and integration of material properties data related to the needs of most CAx industries.

1.4 Opportunities of Technology from the EDI and XML solutions from e-Business industry.

1.4.1 Electronic Data Interchange

In parallel to the engineering data interchange and integration efforts the business industry has had an opportunity to address the data exchange problem through various Electronic Data Interchange (EDI) initiatives and standards. According to the business industry's perspective, the Data Interchange Standards Association (DISA), the not-for-profit group that oversees the development of EDI standards in the United States, defines EDI as [23] "the computer-to-computer exchange of business data in standard formats. In EDI, information is organized according to a specified format set by both parties, allowing a "hands off" computer transaction that requires no human intervention or rekeying on either end. The information contained in an EDI transaction set is, for the most part, the same as on a conventionally printed document."

Currently there are two main EDI standards that re consistent to this definition of EDI [23]. While North America may have its X12 standard [24], the rest of the world uses the UN/EDIFACT standard[25] for EDI. UN/EDIFACT resembles X12 in many ways but still has many differences that require companies doing business internationally to carry at least two sets of electronic formats for each transaction. X12 has implemented some of the features of the UN/EDIFACT system, but they are still different standards.

However, there some serious impediments associated with these standards:

- Each industry defines its implementation guidelines for the X12 standard differently. In many respects, one cannot avoid this situation since each industry has its own set of business rules and practices.
- In addition to different EDI standards around the world and in different industries, the X12 standards change every year.
- Also, using EDI day-to-day gets pricey. Translator software that takes data from legacy systems and formats them in the X12 syntax and back again, needs to change with the growing and ever changing X12 standard. Therefore it often has a high initial price tag and maintenance costs. Enterprise management software (e.g., SAP, Peoplesoft) often has EDI modules that make implementation a little easier, but they still need to keep up with changes in the EDI standards.

1.4.2 XML solutions for EDI

XML, with its ability to transfer structured data along with meaning over the Web, immediately attracted the attention of people and organizations struggling with these issues of traditional EDI.

XML appears to solve the problem of differing North American vs. international standards found in EDI. The ubiquity of the Web and relative ease of connections worldwide has made development of XML and its various applications international exercises. The XML/EDI Group and CommerceNet, two of the standards groups working on XML/EDI guidelines, have active European and Asian chapters as well as North American participation. IBM, Microsoft, iPlanet (the Sun/Netscape alliance), and most other leading vendors are truly international companies with a stake in preventing national or regional conflicts in standards.

XML also appears to better handle the problem of integrating data from EDI transactions into corporate systems. EDI transactions move door-to-door, that is from the sender's mailbox to the receiver's. When a transaction arrives, EDI translator software strips away the X12 syntax and presents a flat file that the receiver's systems need to parse, check for accuracy, edit, and distribute. Enterprise management software will often include these functions with the package, but most of these packages are priced well beyond the resources of small and medium-sized companies. For smaller organizations, they either need to write custom handlers or print-and-rekey, two less desirable options.

Using XML, enterprises have more options for the display and processing of incoming data. The Extensible Stylesheet Language (XSL) allows for the visual display of incoming data and formatting of those same data for further processing by corporate systems. In addition, more end-user applications packages already or plan to support XML that enables the recipients to capture and process the incoming data directly. Even with legacy systems, industry groups can specify standard scripting language or Java code that reflects industry rules, to provide for greater mapping and integration of data exchanged over the Web with XML.

Many efforts have been initiated to exploit the experience of the EDI lessons with the flexibility of XML. As an example, the XEDI.org site contains information about a simple and complete approach for representing X12 and EDIFACT EDI semantics in XML syntax named XEDI. All of the EDI semantics for transaction sets, segments and elements are stored in a data dictionary that is a collection of XML documents. Users that are familiar with XML can modify and customize the data dictionary to meet their company or industry specific trading requirements.

The EDI related efforts and successes of XML contain many useful lessons for the CAx industries.

2. XML Overview

2.1 XML definition

There are multiple definitions of XML that are based on various perspectives of interest:

- Formally, XML is a World Wide Web Consortium (W3C) standard approved as "Recommendation" in February 1998 [13].
- Functionally, XML is an Extensible Markup Language or a meta-language for developing an unlimited number of special-purpose data languages.
- From an application development point of view, XML is a family of technologies and tools that facilitate data exchange and sharing between applications.
- From an evolutional point of view XML is a simplified form or a subset of Standard Generalized Markup Language (SGML).
- From communication point of view XML is a standard framework for making agreements about communication.

Many people think of XML as HTML on steroids, or "monastic" SGML. However, XML is not an improved HTML but rather a true subset of SGML. It supports all the structural and validation features that are expected from SGML. Valid XML documents are valid SGML documents.

This evolutionary definition of XML arms it with powerful inherited properties. As a subset of the SGML defined in ISO standard 8879:1986 [26], XML is designed to make it easy to interchange structured documents over the Internet or any Local Area Network (LAN). XML files always clearly mark where the start and end of each of the logical parts (called *elements*) of an interchanged document occurs. XML restricts the use of SGML constructs to ensure that fallback options are available when access to certain components of the document is not currently possible over the Internet. It also defines how Internet Uniform Resource Locators can be used to identify component parts of XML data streams. The standard SGML reference is almost 500 pages long, plus about another 100 pages of annexes. The current XML specification is 26 pages, not counting the list of contributors. This fact puts in perspective the significant reduction savings in the XML standards.

By defining the role of each element of text in a formal model, known as a *Document Type Definition* (DTD), users of XML can check that each component of document occurs in a valid place within the interchanged data stream. An XML DTD allows computational validation check, against users' accidental entering of a third-level heading without first having entered a second-level heading, something that cannot be checked using the HyperText Markup Language (HTML) previously used to code documents that form part of the World Wide Web (WWW) of documents accessible through the Internet.

However, unlike SGML, XML does not require the presence of a DTD [27]. If no DTD is available, either because all or part of it is not accessible over the Internet or because the user failed to create it, an XML system can assign a default definition for undeclared components of the markup.

From the utilitarian point of view XML allows users to:

- bring multiple files together to form compound documents
- identify where non text entities (images etc.) are to be incorporated into text files, and the format used to encode each non text entity
- provide processing control information to supporting programs, such as document validators and browsers
- add editorial comments to a file.

It is important to note, however, that XML is not:

- a predefined set of tags, of the type defined for HTML, that can be used to markup documents
- a standardized template for producing particular types of documents.

XML was not designed to be a standardized way of coding text: in fact it is impossible to devise a single coding scheme that would be suit all languages and all applications. Instead XML is formal language that can be used to pass information about the component parts of a document to another computer system. XML is flexible enough to be able to describe any logical text structure, whether it be a form, memo, letter, report, book, encyclopedia, dictionary or database.

2.2 The components of XML family of technologies

As it has already been pointed out XML can be considered as a family of technologies contributing towards alleviating the problems associated with data and document exchange and sharing between software applications. Some of these technologies that are related to our efforts are described below.

XML 1.0, is the specification that defines what "tags" and "attributes" are, but around XML 1.0, there is a growing set of optional modules that provide sets of tags & attributes, or guidelines for specific tasks.

In this context XML itself is based on the concept of *documents* composed of a series of *entities*. (`Entity' is the English spelling of the French word `entité', the Teutonic equivalent of which is `thing'. Those familiar with modern programming techniques will be probably be more comfortable using the word `object'. All these terms are synonymous.) Each entity can contain one or more logical *elements*.

Each of these elements can have certain *attributes* (properties) that describe the way in which it is to be processed. XML provides a formal syntax for describing the relationships between the entities, elements and attributes that make up an XML document, which can be used to tell the computer how it can recognize the component parts of each document.

XML differs from other markup languages in that it does not simply indicate where a change of appearance occurs, or where a new element starts. XML sets out to clearly identify the boundaries of every part of a document, whether it is a new chapter, a piece of boilerplate text, or a block of numerical data associated with coordinated definitions of a node set in a FEM representation.

To allow the computer to check the structure of a document, users must provide it with a document type definition (DTD) that declares each of the permitted entities, elements and attributes, and the relationships between them.

Xlink (still in development as of November 1999), which describes a standard way to add hyperlinks to an XML file. *XPointer* & *XFragments* (also still being developed) are syntaxes for pointing to parts of an XML document. (An XPointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file.)

CSS, the style sheet language, is applicable to XML as it is to HTML.

XSL (autumn 1999) that is the advanced language for expressing style sheets and It consists of three parts:

- XSL Transformations (XSLT): a language for transforming an XML document to another XML document or other format documents such as HTML or text.
- The XML Path Language (XPath): an expression language used by XSLT to access or refer to parts of an XML document. XPath is also used by the XML Linking specification.
- XSL Formatting Objects (XSL-FO): An XML vocabulary for specifying formatting semantics

DOM is a standard set of function calls for manipulating XML (and HTML) files from a programming language and is considered an interface to programming lanuages.

XML Namespaces is a specification that describes how you can associate a URL with every single tag and attribute in an XML document. What that URL is used for is up to the application that reads the URL, though. (RDF, W3C's standard for metadata, uses it to link every piece of metadata to a file defining the type of that data.) *XML Schemas 1* and *2* help developers to precisely define their own XML-based formats. There are several more modules and tools available or under development and they can be found in the W3C web site [28].

2.3 How is XML used?

To use a set of *markup tags* that has been defined by a single user, a trade association, a standards organization or similar body, users need to know how the markup tags are delimited from normal text and in which order the various elements should be used in. Systems that understand XML can provide users with lists of the elements that are valid at each point in the document, and will automatically add the required delimiters to the name to produce a markup tag. Where the data capture system does not understand XML, users can enter the XML tags manually for later validation. Elements and their attributes are entered between matched pairs of angle brackets (<...>) while entity references start with an ampersand and end with a semicolon (&...;).

Because XML tag sets are based on the logical structure of the document they are somewhat easier to understand, and remember, than physically based markup schemes of the type typically provided by word processors. An XML memo might be coded as:

```
<Invitation>
   <to>All members of ASME</to>
   <from>CMS group NRL Washington DC</from>
```

```
   <date>11th September 2001</date>
   <subject>development of femML</subject>
   <text>Please contribute towards solving the FEM data exchange problem.</text>
</Invitation >
```

This file form is ideal for a computational processing. The start and end of each logical element of the file has been clearly identified by entry of a start-tag (e.g. <to>) and an end-tag (e.g. </to>).

Notice that at this point nothing has been said about the format of the final document. From the neutral format provided by XML users can chose to display the memo on a screen whose size can be varied to suit user preferences, to print the text onto a pre-printed form, or to generate a completely new form, positioning each element of the document where is desired.

2.4 Key idea behind XML

Since XML has been developed as an outgrowth of the Internet and WWW evolution of media for presenting static content on browser windows, many people think of it as a technology that allows flexible presentation and reformatting of text documents. Indeed, the SGML heritage of XML clearly contains the publication industry perception. In other words the context of document transformation is that of typesetting and typography industries. According to this context information is contained in documents and reference to their structures means typographical appearance. Later it was extended horizontally for the context of content presentation devices of varying text presentation capabilities like mobile phones and personal digital assistants (PDAs). In fact it is this very idea that has pushed the technology explosion that surrounds it.

However, the key idea behind the power of XML is not related to XML itself but rather to its "markup" property. It is the markup part of each element that carries the meaning and therefore the particular context within which a syntactic construct like a particular string or word can be interpreted. For example the string "apple" can be thought of as typographical entity like a word typeset in bold as it would be expected from an HTML or a text document XML source. Its form would appear like this:

```
<document> ….. <b>apple</b>….</document>
```

On the other hand it could also appear as fruit, or as teacher's present, or as a computer in a room, or as computer company, or as material system, or even as a load-bearing mechanical structure. Examples of the possible XML document appearance of the appropriate element that capturing the specific meaning for each one of these cases for the "apple" can be seen in table 1 where the labels of the columns signify the context and the cells below them contain the appropriate XML fragment.

Cooking	Educational Psychology	Office Assets	Technology Industry	Materials Science	Engineering
<docRecipe> … <fruit> apple </fruit> ….. </docRecipe>	<docFactors> … <teacherPresent> apple </teacherPresent > ….. </docFactors>	<docInventory> … <computer> apple </computer> ….. </docInventory >	<docManuf> … <company> apple </company> ….. </ docManuf >	<docMaterials> … <material> apple </material> ….. </docMaterials >	<docParts> … <component> apple </component > ….. </docParts >

Table 1. XML markup for the word "apple" for various meanings and contexts.

Each one of the columns in this table can be thought as an example document for a custom application of XML specific to the industry or the context that labels each one of these columns. It is evident that the last column labeled "Engineering" would be the one closest to a finite element modeling data representation. This does not mean that XML cannot be used for typesetting and document content

transformations. In fact HTML 4.0 attempts to correct all the non XMLish problems related to document validation, and XHTML is a very close application of XML intended to be taking over HTML's role in the WWW presentation industry.

From a more formal point of view one can express the idea of XML's inherent flexibility to carry meaning along with syntax, by indicating that unlike natural language, XML maintains the non-terminals of the grammar represented by each XML document as the valid carrier of both syntax (tokens within tags) and meaning (entity names within angled brackets or tags). The fact that the meaning included in XML documents is expressed in terms of strings (after all an XML document is a character file) is both a blessing and a curse. The blessing is that it makes the document amenable to computerized processing and automation available trough XML tools. The curse is that it still does not solve the recursive meta-meaning problem of what is the meaning of the strings used to capture meaning and then of those used for that and so on? However, it is the blessing aspect of this XML central idea that has fed the rampant proliferation of XMLware and its successful application to various applications of data and document integration vertically (within a particular industry) and horizontally (across multiple industries).

3. XML and WWW technology Integration for Mission Critical Applications

3.1 The Java Opportunity

It has recently become evident that it is safe to assume our entrance into the era of capitalizing on the Internet and WWW resources both as a transport environment and as a platform for deploying mission critical applications related to engineering applications. The applications that are driving the acceptance of XML are those that cannot be accomplished within the limitations of HTML. Although they all have been motivated from e-business specifications and goals, they are pertinent to the data exchange industry in the CAx communities. These applications can be divided into four broad categories:

1. Applications that require the Web client to mediate between two or more heterogeneous databases that may be stand-alone or be a part of legacy applications.
2. Applications that attempt to distribute a significant proportion of the processing load from the Web server to the Web client.
3. Applications that require the Web client to present different views of the same data to different users (this more of a publication presentation item).
4. Applications in which intelligent Web agents attempt to tailor information discovery to the needs of individual users or/and processes.

The alternative to XML for these applications is proprietary code embedded as "script elements" in HTML documents and delivered in conjunction with proprietary browser plug-ins or Java applets [29]. XML derives from a philosophy that data belongs to its creators and that content providers are best served by a data format that does not bind them to particular script languages, authoring tools, and delivery engines but provides a standardized, vendor-independent, level playing field upon which different authoring and delivery tools may freely compete.

An overhead view of the characteristics of both XML and Java technologies allows the recognition of a strong cross-leveraging capability for both of them. In particular, the most significant features that this recognition is based on the following realizations:

- XML structures can map *homomorphically* to Java Objects
- XML tags map *well* to Java Objects allowing late binding and implementation of hierarchical (OO) data model
- There is Unicode support both in XML and Java that facilitates application localization
- Portability and network friendliness are built-in features of Java

These realizations can be interpreted as advantages that allow the development of applications with endless implementation possibilities such as:

- composition/synthesis of complex models just by simple messaging between dynamic object-ware units automatically produced by XML<->Java toolsets
- light-weight asynchronous processes implementation of distributed, migrating, dynamic and intelligent agents for each one of the femML tag entities

The inherent correspondence between the XML and Java allows freedom of choice for the starting and ending points of potential document transformations between them.

The Java to XML path can be described by the following three steps:

- Start with Java class definitions
- Serialize them - write them to an XML stream
- Deserialize them - read values in from previously serialized file

while the XML to Java path can be described by the following three steps:

- Start with XML document type
- Generate Java classes that correspond to elements
- Classes can read in data, and write in compatible format (shareable)

Availability of resources, personal preference and speed of achieving desired goals are the biggest factors that will determine the direction of transformation for each developer. However, there are some functional arguments that suggest one over the other approach but their description falls well outside the scope of the present paper.

3.2 DOM and SAX

Most applications that accept XML documents require the usage of a parser that validates the submitted XML documents/files against the applicable DTD and by generating an object equivalent representation of the document for the needs of the application. The way parsers achieve this is by implementing the Document Object Model (DOM) recommendation by W3C. Although originally DOM had been implemented to unify the HTML and XML object models of Netscape Navigator 3 and Internet Explorer 3, it clearly reflects the hierarchical (tree-like as a special case of a graph-like structure) that underline XML documents. Each XML element corresponds to a DOM node object to the point that the terms "element" and "node" are being used as interchangeable throughout the WWW and Internet industries. Clearly DOM is not limited to browser applications nor it is limited to Javascript. DOM is a multiplatform, Multilanguage interface. There are versions of it for most languages because W3C adopted the clever strategy of specifying it using Object Management Groups (OMG) [30] Interface Definition Language (IDL) [31], which allows describing which methods and properties an object has, and not what the object does. The implementation that is responsible for what the object does can be encoded in any programming language for which an IDL mapping exists. There are mappings of IDL for Java, C++, Smalltalk, Ada and even Cobol. However, Java is a particularly privileged language for XML development. In fact, most XML tools are written in Java and/or have a Java version. As a matter of fact, there are more Java parsers for XML than parsers written in all other languages combined. Most of these parsers support the DOM interface.

While DOM is an object-based interface and can be used to communicate with an application by explicitly creating a tree of objects in memory that closely match the XML document structure, it may not be very appealing for legacy applications that are "unaware" of the XML industry and are less object-oriented. For these applications sometimes it is more sensible not to build the DOM tree, but to directly load the document in their data structure. To respond to this need the members of the XML-DEV mailing list have developed a standard Application Programming Interface (API) for event-based parsers called

SAX, short for the Simple API for XML. Event-based interfaces tend to be more efficient, require fewer resources, and they allow access of the information in the XML document without waiting for the parsing to finish. For this reasons, Sun has included SAX in ProjectX-an ongoing effort to add an XML parser to the Java platform. ProjectX also supports DOM so the parser offers both event-based and object-based interfaces and can be found at java.sun.com. IBM's and DataChannel's parsers also both support the DOM and SAX interfaces.

It is imperative to emphasize that almost every day there new attempts to facilitate the integration of XML with Java. It is characteristic that a simplified implementation of DOM in Java has already been created (JDOME) [32], while other modular architectures provide tremendous efficiency in code development and usage by extending the idea of Java Beans [33]. One of these is the XBEANS effort [34] that extends the idea of a regular Java Bean to that of a bean that takes XML as input, processes it in some fashion and then passes XML on to the next Xbean. We are seriously considering utilizing both of these technologies for implementing a non-legacy code specific implementation of FEM data transfer and transformation.

3.3 XML and Data Bases

All CAx management applications as well as most FEM codes implement database functionality in order to facilitate efficient transactions between the user interface and the rest of the functional modules of the respective applications. This implicit or explicit relevance of databases with the applications of interest as well as the described morphology of XML raise the reasonable question: "Is XML a database?"

An XML document is a database only in the strictest sense of the term. That is, it is a collection of data. In many ways, this makes it no different from any other file -- after all, all files contain data of some sort. As a "database" format, XML has some advantages. For example, it is self-describing (the markup describes the data), it is portable (Unicode), and it can describe data in tree or graph structures. It also has some disadvantages. For example, it is verbose and access to the data is slow due to parsing and text conversion.

A more useful question to ask is whether XML and its surrounding technologies constitute a "database" in the looser sense of the term -- that is, a database management system (DBMS). The answer to this question is, "Sort of." On the plus side, XML provides many of the things found in databases: storage (XML documents), schemas (DTDs, XML schema languages), query languages (XQuery, XPath, XQL, XML-QL, QUILT, etc.), programming interfaces (SAX, DOM, JDOM), and so on. On the minus side, it lacks many of the things found in real databases: efficient storage, indexes, security, transactions and data integrity, multi-user access, triggers, queries across multiple documents, and so on.

XML documents fall into two broad categories: *data-centric* and *document-centric*. Data-centric documents are those where XML is used as a data transport. They include sales orders, patient records, and scientific data. Their physical structure -- the order of sibling elements, whether data is stored in attributes or PCDATA-only elements, whether entities are used -- is often unimportant. A special case of data-centric documents is dynamic Web pages, such as online catalogs and address lists, which are constructed from known, regular sets of data. Document-centric documents are those in which XML is used for its SGML-like capabilities, such as in user's manuals, static Web pages, and marketing brochures. 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, you will need a database that is tuned for data storage, such as a relational or object-oriented database, and some sort of data transfer software are

needed. This may be built in to the database or might be third-party middleware. Depending on your needs, you may need Web publishing abilities as well.

To store and retrieve document-centric documents, a *native XML database* or *content management system* is needed. Both of these are designed to store content fragments, such as procedures, chapters, and glossary entries, and may include document metadata, such as author names, revision dates, and document numbers. Content management systems generally have additional functionality, such as editors, version control, and workflow control. Although content management systems generally use a native XML database for storage, this is hidden from the user.

The rapidly expanding number of products associated with XML and its association with database applications are divided into eight categories. It's important to realize that the boundaries between some of these categories, especially XML-Enabled Databases, Native XML Databases, XML Servers, and XML Application Servers, are somewhat arbitrary.

- *Middleware*: Software you call from your application to transfer data between XML documents and databases. *For data-centric applications.*
- *XML-Enabled Databases*: Databases with extensions for transferring data between XML documents and themselves. *Primarily for data-centric applications.*
- *Native XML Databases*: Databases that store XML in "native" form, generally either as indexed text or as some variant of the DOM mapped to an underlying data store. *For data- and document-centric applications.*
- *XML Servers*: Platforms that serve data -- in the form of XML documents -- to and from distributed applications, such as e-commerce and business-to-business applications. *Primarily for data-centric applications.*
- *XML Application Servers*: Web application servers that serve XML -- usually built from dynamic Web pages -- to browsers. *For data- and document-centric applications.*
- *Content Management Systems*: Systems for managing fragments of human-readable documents and include support for editing, version control, and building new documents from existing fragments. *Primarily for document-centric applications.*
- *Persistent DOM Implementations*: This category has been merged with native XML databases.
- *XML Query Engines*: Standalone engines that can query XML documents.

In general, code has to be written to integrate Middleware, XML-Enabled Databases, Native XML Databases, XML Servers, and Persistent DOM implementations with applications. XML Application Servers require some scripting, and Content Management Systems need to be configured, which may be a non-trivial task in itself.

4. Current status of femML

4.1 Historical note

As described earlier, femML has been developed as a necessary outgrowth of our group's research efforts, to solve the structured data intensive exchange problem between modules of our custom applications or even between custom stand alone application such as RCfem [35] and existing legacy commercial applications such as ANSYS [36] and ABAQUS [37]. The idea for its creation was naturally generated in the summer of 1999 and has been evolving ever since. FemML's development went from the conceptualization to the implementation phase when we searched the XML repositories and found nothing relevant to this. Special encouragement for the final push was the lack of responses when we posted inquiries about the possible existence of such an XML variant on October 6, 2000 at various

mailing lists like the XANSYS one for ANSYS users. The only relevant XML variants were XSIL, X3D and MatML, all dealing with a partial collection of issues associated with the FEM data exchange, but none of them was directly dealing with the entirety of the main problem.

4.2 What is femML

The finite element modeling Markup Language, is an XML variant designed to facilitate the data transfer, exchange, interchange and integration between finite element modeling applications and their modules. It is work in progress that has accomplished the creation of a DTD, a SCHEMA and certain FEM code specific file generation and parsing tools. It is in a pre-recommendation stage and our focus is to offer it for public discussion, development and distribution. This is not to say that femML cannot eventually evolve to forming the kernel of a set of technologies that will not be solving the data exchange problem, but it will lead to an alternative way of working with FEM data discretizations. A way that would use just three component technologies:

- femML as a transport file format,
- an ordinary Relational Data Base Management System (RDBMS) for dynamic data management,
- and a visualization module.

Such a combination of technologies would allow composition and factoring of FEM discretizations for the needs model synthesis and combination as well as the needs for model decomposition and simplification of the design and prototyping industries.

4.3 femML Objectives

Despite the fact that femML began as a custom effort specific to the data exchange needs within the context of the activities of our group, the objectives employed to motivate the effort of the femML development were very specific and quite general:

- Define a standard for the exchange of FEM data (including product shape, associated FEM models, material properties and analysis results) that will allow a *person* or a *computer application* to interpret and use the data *regardless of its source or target* and *regardless of the taxonomic order of the FEA model*. This effort minimally corresponds to defining: i. A set of XML Tags, ii. Relationships and constraints on these tags, and iii. Document Type Definition (DTD) or/and Schema
- Define and develop a set of examples that follow the standard.
- Define and develop a set of tools for the utilization of this standard from and to other applications.
- Develop examples of using these tools.
- Develop a long-term framework for utilization of legacy RDBMS systems, 3D visualization viewer systems, and light-wait asynchronous processes architectures (i.e. agents), for achieving a truly distributed and transparent capability to utilize FEM techniques in highly functional, economical, and ubiquitous manner.

By the term "regardless of the taxonomic order" we mean the development of an XML dialect for FEM data exchange that can accommodate all, or most of the FE varieties, i.e. structured, unstructured, blocked, hierarchical, spectral, stochastic etc.

4.5 femML Document Type Definition (DTD)

The current state of affairs has been progressed into the development of a DTD that can definitely cover all of the taxonomic categories of FEM data, except the stochastic ones.

The strategy followed for developing femML's vocabulary of terms, relationships and constrains as well as the DTD that encapsulated them was a special application of the process described by the Unified Modeling Language (UML) [38] activity diagram given in figure 2.

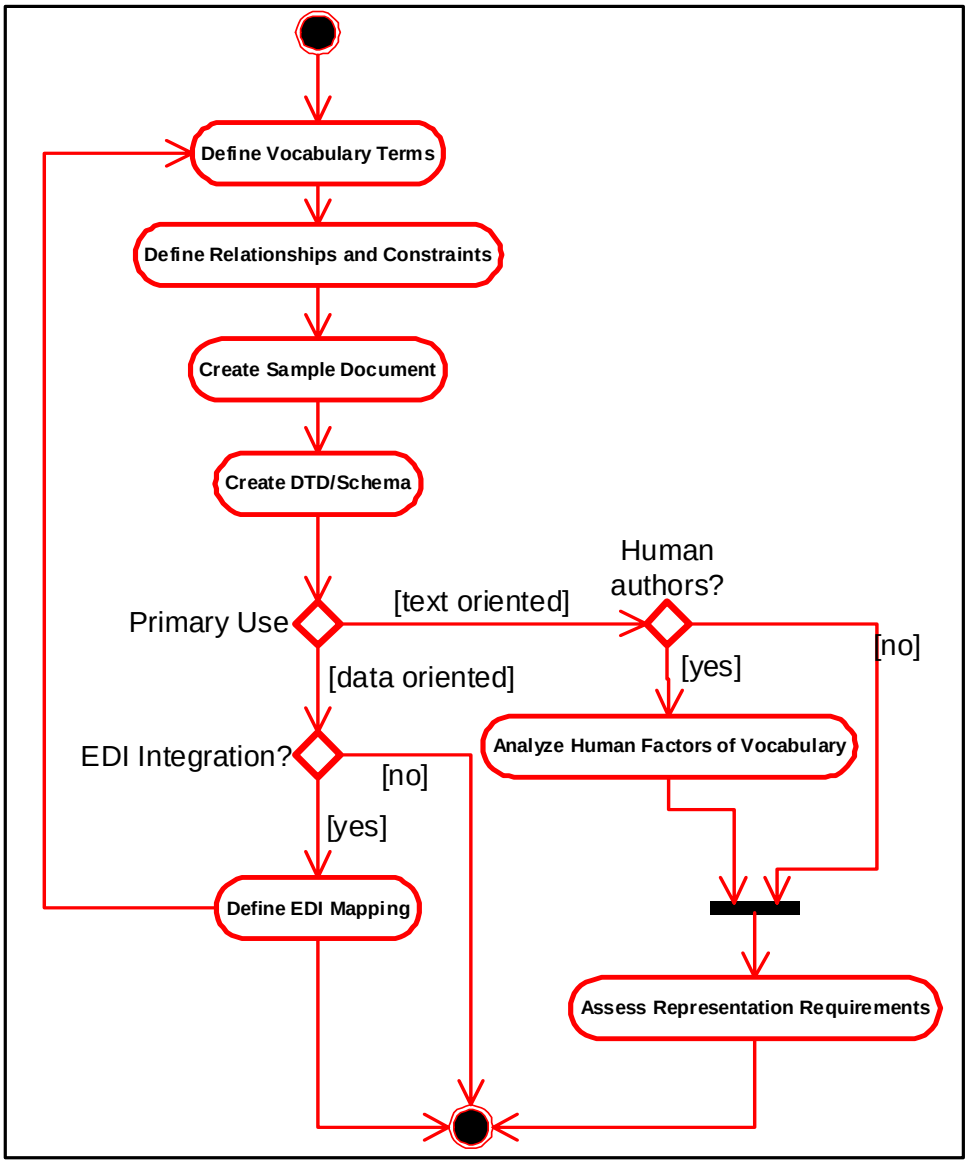

Figure 2. UML activity diagram representation ofor DTD/SCEMA development process

This process in itself is a special case of the general process for developing an XML variant that although obvious, has been formally captured in a UML activity diagram in [39]. Clearly the pertinent path for our case implies engineering electronic data interchange (EEDI), and can only reach its terminal state if successful EEDI has been achieved. Otherwise it cycles through the feedback path on the left of the activity diagram that connects the "Define EEDI Mapping" node with "Define Vocabulary Terms" node.

The EditML [40] application was used to draft the sample document and its built-in capability for automatic DTD/SCHEMA generation was exploited to generate the initial stamp of the particular femML DTD and SCHEMA. The currently implemented DTD for femML appears as a UML class diagram shown in figure 3.

The ***femML*** node can be repeated multiple times within a single document to allow capturing the FEM data specification of multiple parts or domains.

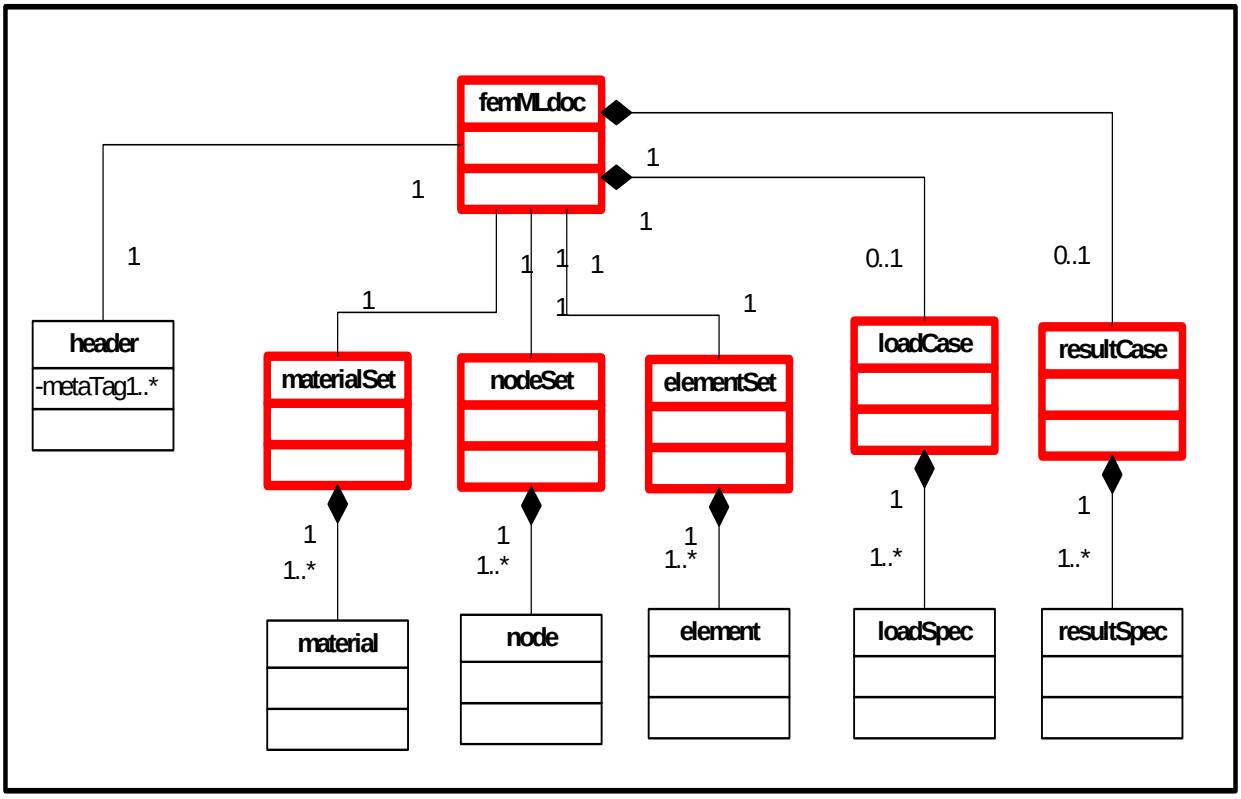

Figure 3. UML representation of the top two levels of femML DTD structure

The ***header*** element is there only to ensure there is a transport mechanism within the document for the meta-data. These meta-data are implemented through meta-tags that encode information about the human author, the application generating the entire document, date of creation, project it belongs to and other non FEM specific data.

The ***materialSet*** element is responsible for carrying the material properties information associated with the FEM representation of the part or component encapsulated in the document at hand. ***NodeSet*** is the element that caries the nodal geometry information of the discretized component, while ***elementSet*** carries the elemental information of the model. These last two elements/nodes are responsible for carrying the appearance information of the model from a 3D geometrical point view.

The ***loadCase*** element is responsible for carrying the loading or/and boundary conditions on a nodal basis. Finally, the first level of femML nodes is completed by the ***resultCase*** node that carries the nodal results data (i.e. displacements, stresses, strains, energies and any other scalar, vector or tensor component quantities associated with the nodes).

Each one of these nodes can have many individual children nodes to carry the specific data associated with each one of them like coordinate elements for each FEM node.

It is anticipated that the XSIL, X3D and MatML efforts will play a pivotal role in the development, testing and integration of femML. XSIL and X3D can serve as target translation languages for exploiting their visualization resources for 3D repreentation of FEM models. When MatML is completed but also even now that is under development, can be integrated through adaptation of the appropriate namespace to define material properties necessary for femML under the materialSet element by borrowing MatML's

capability to describe material properties in place of the existing ***material*** node under ***materialSet***. Our interest in describing material properties for composite materials forms the basis of our current and future cooperation with the MatML working group.

Station To Station Data Exchange via femML

5.1 General case of S2S implementation

Despite the benefits of extending the expressive power of XML with the dynamic data representation and manipulation capability of Java enabled applications, we have decided to focus onto the simplest of the approaches for exchanging FEM model data, by utilizing the Station-to-Station (S2S) approach that is built entirely on XML technology (see figure 4).

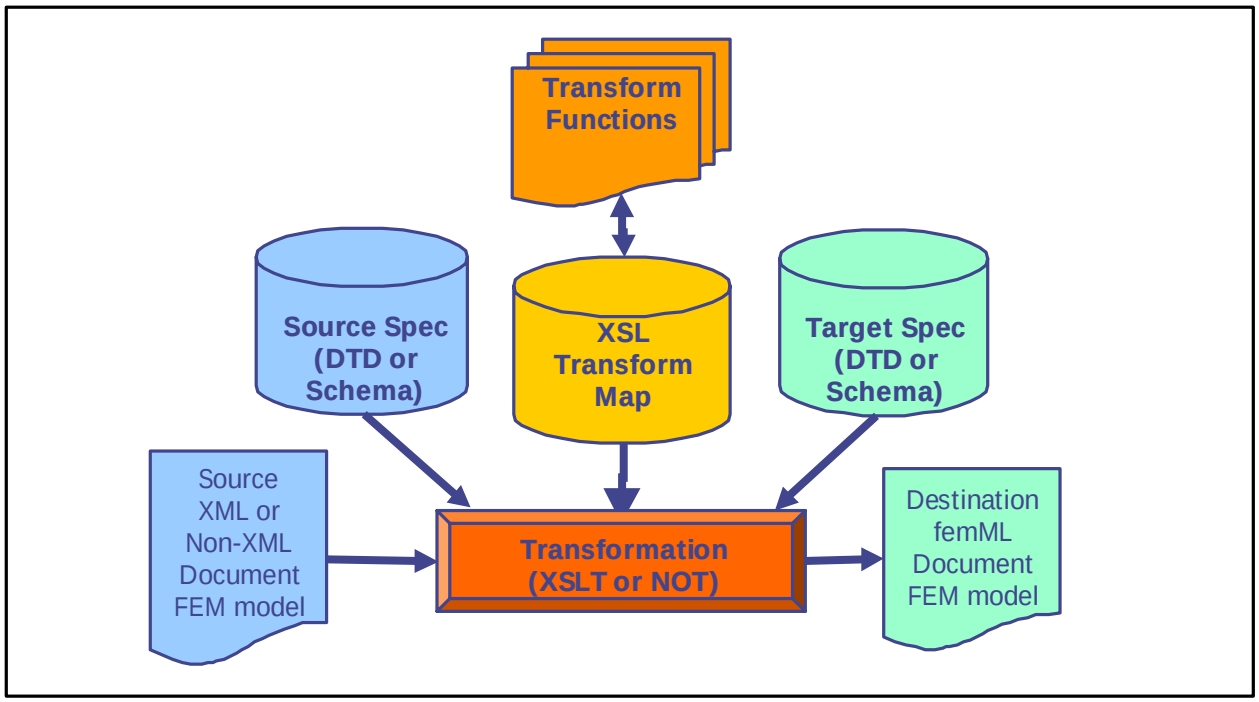

Figure 4. S2S data transformation architecture

This decision does not preclude the future exploitation of the XML-to-Java (or vice versa) cooperative benefits in terms of dynamics, scalability, deployability and economy.

The S2S model assumes the existence of a source and a destination data-document and it is not different than the most generic of the business to business (B2B) models that dominated the Internet during the last few years.

The transformation can be either implemented trough a common XSLT processor or through a Java application that utilizes the parsed DOM equivalent of the source document structure and subsequently rewrite it to a new one that can then be converted to the target document.

In either case, the transformation processor requires a transformation definition defined via a set of transform functions that may or may not implement templates, while at the same time it ensures that both source and target documents/files are valid according to their corresponding DTDs or Schemas. The transformation can be implemented in a multidirectional manner. Users with not extensive XML or/and Java experience can use of the shelve tools to construct the transformation engines as a byproduct of utilizing intuitive tools with simple graphical user interfaces (GUIs). An example of such an application

is BizTalk Mapper that belongs to BizTalk [41] set of tools that is a Microsoft led XML initiative. An aspect of its GUI can be seen in figure 5 where defining the transformation amounts to drawing lines that signify the correspondences between the elements of the source and target elements.

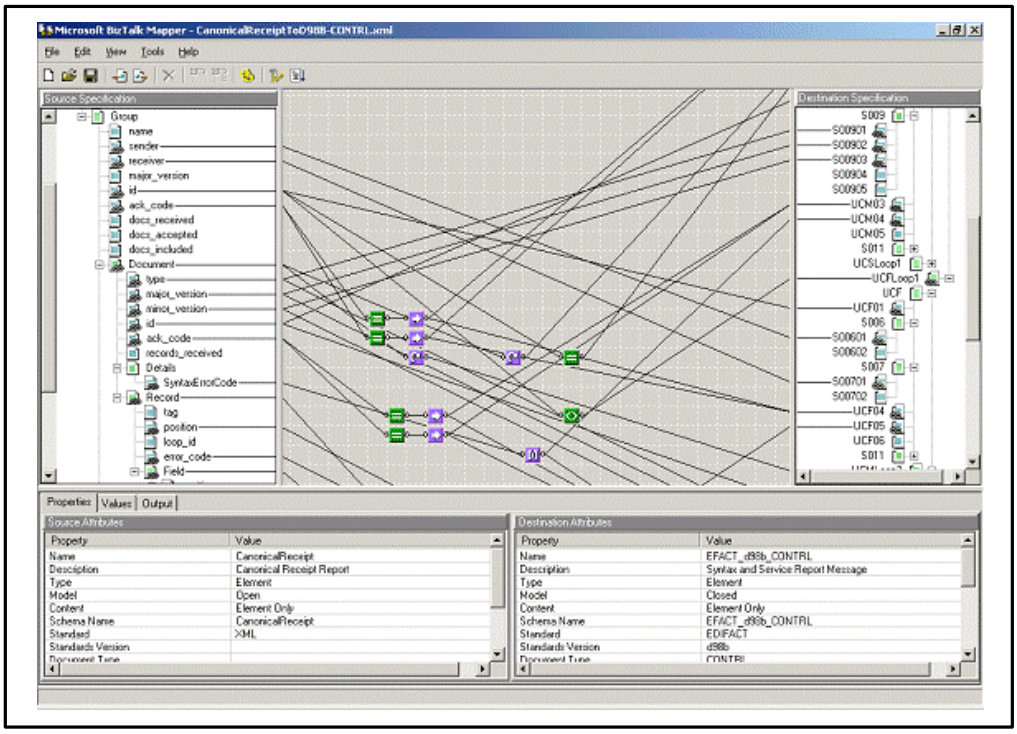

Figure 5. Simplicity of BizTalk Mapper GUI.

There are many other applications that allow the automation of document transformation design and implementation are that can be found either as stand alone applications or as parts of tool suites. Information about such tools can be found at the XML specific portals like xml.org, xml.com, etc.

5.2 ANSYS case of S2S implementation

When it comes to considering ANSYS as being one of the two applications that have to be used for exchanging FEM data, there are several factors to be considered regarding the existence of target and source file formats as well as means for linking to the ANSYS database.

The existence of the very powerful ANSYS Parametric Design Language (APDL) [42], as well as the ANSYS programmable features that allow linking of custom functionality into the main executable [43] through Fortran, C, C++ source code, present a very interesting variety of approaches towards parsing and generating input and output data files. The architecture that has been implemented can be seen in the figure 6. The doted line components have not been implemented yet. Instead of describing the additional components of this architecture to justify the ANSYS specialization of our general S2S approach we will rather describe the followed strategy because it describes the motivation behind the adaptation of this approach.

Thus the strategy followed by our group, can be described by the following steps:

1. Authored the ANS2AGT macro in APDL. This macro can be executed after one has a working model database complete with results. It reads the database and it creates an ASCII file that contains all the FEM pertinent data (geometry information, material definition, loading

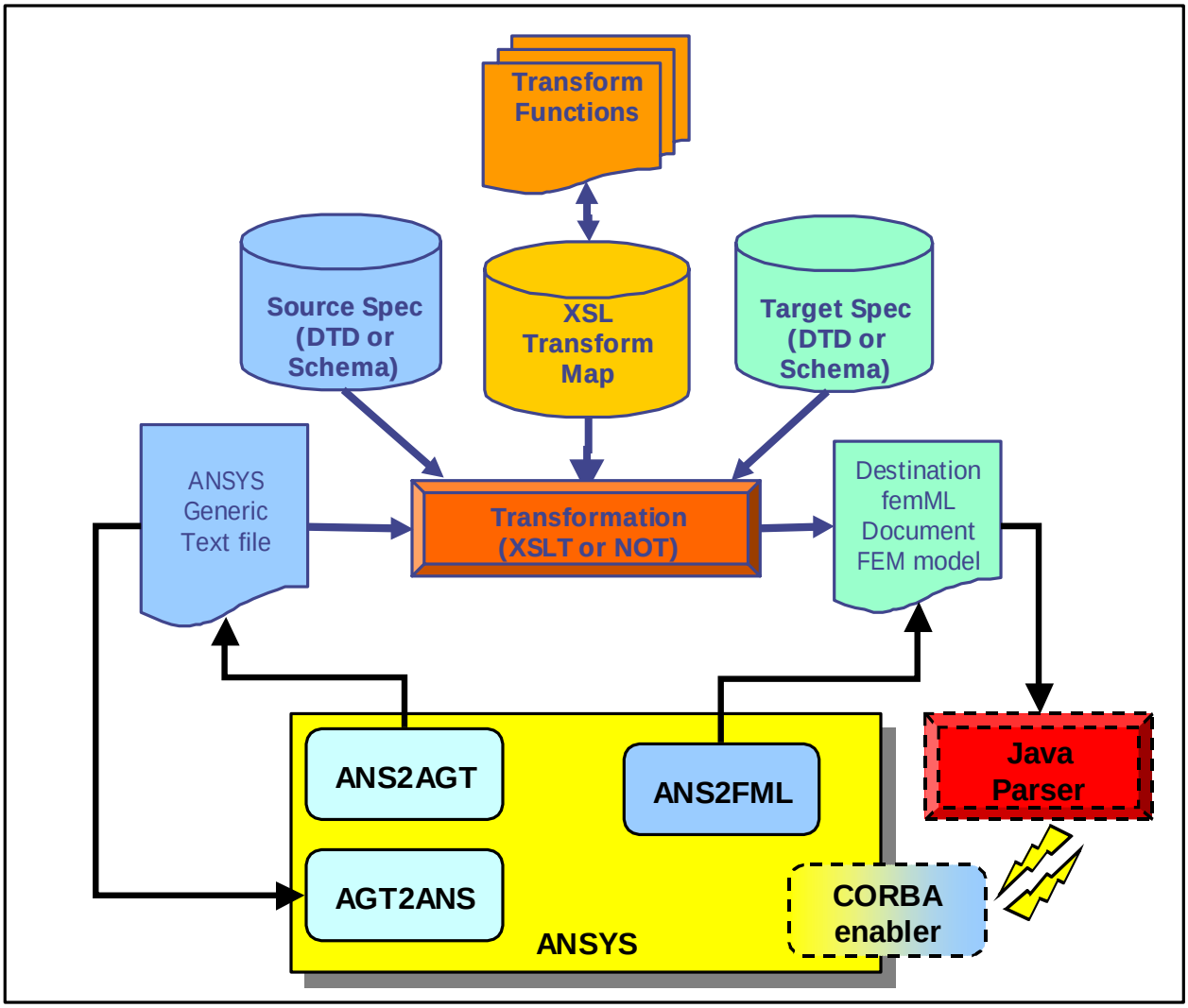

Figure 6. ANSYS based S2S data transformation architecture.

specification and corresponding results). This file has the extension ".AGT" to signify that it is an ANSYS Generic Text file. Other ANSYS file formats could be used (ANF, CBD) as well. The file structure is not important as the particular selection of represented data is. This selection has been designed such as when the data in this file are read back into ANSYS while the database is originally empty, the produced database contains complete information of a bottom-up built model (from nodes and elements), that can be solved and results can be plotted in ANSYS itself.

2. Authored the AGT2ANS macro in APDL. This macro implements the inverse functionality of ANS2AGT. Namely, it reads an .AGT ANSYS Generic Text file into the ANSYS database. The main reason for creating this macro was to establish an independent validation path for the integrity of the data exported by ANS2AGT.
3. Created the ANS2FML macro in APDL. This macro allows exporting the femML file that contains the data for an existing model in the ANSYS database.
4. Considered the strategy for reading femML files into ANSYS. The obvious solution would be to write a parser based on a C++ implementation of the DOM. A second strategy would be to use a Java based parser but since this would require to call Java from C++ that would require to develop

a Java wrapper of ANSYS by utilizing the Java Native Interface (JNI). A third approach could be embedding a Java parser that takes the femML DOM information and passes it to the ANSYS database custom routines available as User Programmable Features (UPFs). This approach would require a mechanism for calling java bytecode from C++ that can certainly be implemented by using the JNI as well. A fourth strategy would be to exploit an application like Mathematica that can both talk to a Java parser of femML and the C++ extensions of ANSYS via the J-Link and Math-Link interfaces. This would make Mathematica a communications arbitrator between the parser and ANSYS. A fifth strategy would be to use a Java parser that can communicate with a custom version of ansys.exe that is enhanced by custom C++ routines for defining database entities. The communication will occur over the TCP/IP layer and will be implemented through one of the object oriented communication technologies like OMG's IDL on CORBA [44] objects. This last approach is something we are seriously considering pursuing given the time and programming resources will become available. None of these approaches could be implemented fast enough to be fully functional by the time the first femML DTD was ready.

5. Finally an indirect approach was used for this purpose. Utilized the S2S approach and initiated authoring of an XSL stylesheet template that would allow using XSLT processor transformation to transform the femML to its AGT equivalent. The AGT2ANS module could be used to load the model into the ANSYS database.
6. An alternative way was to use a proprietary product called XMLjunction [45] that allows implementing bi-directional transformations between flat files and XML documents.

6. The future of femML

6.1 Issues to be resolved

The distance to be traveled for the development of a comprehensive femML standard is very long. Experience from other efforts has shown that on one hand no specification can end up becoming a standard if it does not have the support of industry, government, and academia. On the other hand experience has also shown that no standard is any good if it is not used by end-users in all of these sectors.

In addition to the economic and political factors the play a role in developing and adopting standards there are always technical reasons mostly associated with the derivable utility, that can make or brake a standard. The technical issues that we can foresee that will play an essential role in fem ML's adaptation and usage are the following:

Scale of Generality: Should we be thinking in extending femML to cover finite difference discretizations with all their idiosyncrasies, boundary element discretizations, hybrid discretization or even non-discrete models of continuous systems. For that matter should we be thinking of a femML or discrete model representation Markup Language (dmrML), or even a physical model behavior representation Markup Language (pmbrML)? It appears that the latter would be the most inclusive and general case. However, very ambitious goals may provide all kinds of reasons for not realizing these goals. This is an issue that a decision cannot be taken a priori before considering resources and support.

Separation between Appearance and Behavior: The current DTD implementation follows a strict FEM data file structural architecture. Information holding the geometry information of the 3D model is included along with loading, material and results specification. However, there are reasons for altering this situation. If we consider that there might be cases that the data that need to be exchanged are going to be based only on a particular subset of the original, then we have consider structuring the file in such a way that is easier to access and transfer data subsets of particular nature. The geometry model that is

responsible for the appearance of the model, and the loading, material and results specifications that capture the behavior part of the model are two subsets of this type. The question at issue is under what conditions should we restructure the DTD to attach them under separate elements in order to facilitate transformation? In view of the existence of X3D and XSIL a direct mapping between their elements and the geometry elements of our DTD could be possible. The disadvantage of overextending such an approach may lead to a very verbose data file. However, since files like this are not intended to be human readable (although it actually is) but rather machine-readable. This issue will most likely be resolved from the need to integrate horizontally with other industries (i.e. entertainment industry) that may require CAx models.

Utilization/Leveraging of existing XML dialects: Particular element definitions (i.e. material definition) may already be defined from an already existing XML application (i.e. MatML). The namespace specification allows borrowing such constructs. The question is when we should be doing it when we should not. This issue can be resolved careful consideration of the semantic overlap and proximity between the intension of our application and the extension of the existing dialect's DTD.

Scene graph Structure for geometry: How much should we be aware and should we implement scene-graph internal representation architecture, that follows the lessons for other 3D representation methodologies such as Open Inventor [46], VRML and X3D[20]. What may determine a settlement to this issue may become a mute point when efficient ways to go in and out from the scene-graph representation become available while at the same time do not require users to spend time over steep knowledge curves. However, our group believes in leveraging existing technologies and lessons learned from their usage to make decisions about our contribution to the evolution of femML.

Composition and factoring isomorphism between data of FEM model and their femML expression: In the physical space structures can be thought as aggregates of parts and components. What allows us to synthesize a complex structure out of various parts is called composition of parts. When we decompose the aggregate structure to its parts we perform the operation of factoring. We can obviously think of both of these operations at the data representation level where femML documents can be the result of composing other ones (corresponding to part descriptions), or femML documents corresponding to part representations can be the result of factoring aggregate structures femML representations. The issue here is how much should we strive to establish and maintain a DTD/Schema architectures that preserves a on to one correspondence between the physical and data representation of the abilities to compose and decompose FEM representations.

Distribution of metadata: Should a document created as the composition of other documents that correspond to parts, components or substructures, carry the meta-data of the sources? If yes, should they also be composable or should they exist individually as a part of metadata nodes at lower levels of the DTD graph?

There are other generic issues that are applicable in the development of any XML variant and therefore valid for the case of femML as well.

The most typical of these issues is the dilemma of the choice between implementing a DTD and a Schema based strategy. There are pros and cons for both of them and for the moment we are focusing on the DTD although we have also created a schema for femML. However, we did not need to use the schema-based approach yet. The technical details of what can and cannot be done with each one of them may be of no relevance here. The proliferation of particular tools and their economic impact to the users and developers combined with the usability and learn ability of associated tools, may turn out to be the major factor in the future that will determine which one is more practical for particular applications.

Another generic issue is the historical decomposition capability. Should we be able to factor the component data representations of a femML file based in previous versions of it? Effectively this would

allow for a version control mechanism that is encoded inside the file. This falls well outside the particulars of our intentions regarding femML but it may very well be a seed for a useful debate on an old issue of the application development community brought into the context of the XML document management community.

6.2 Potential evolution of the femML DTD

To address all of the above-mentioned issues except the first one (scale of generality) we have already started modifying the femML DTD structure. A potential evolution of the femML DTD may start from the architecture presented in figure 7, where the top layers of the architecture have been captured as a UML class diagram.

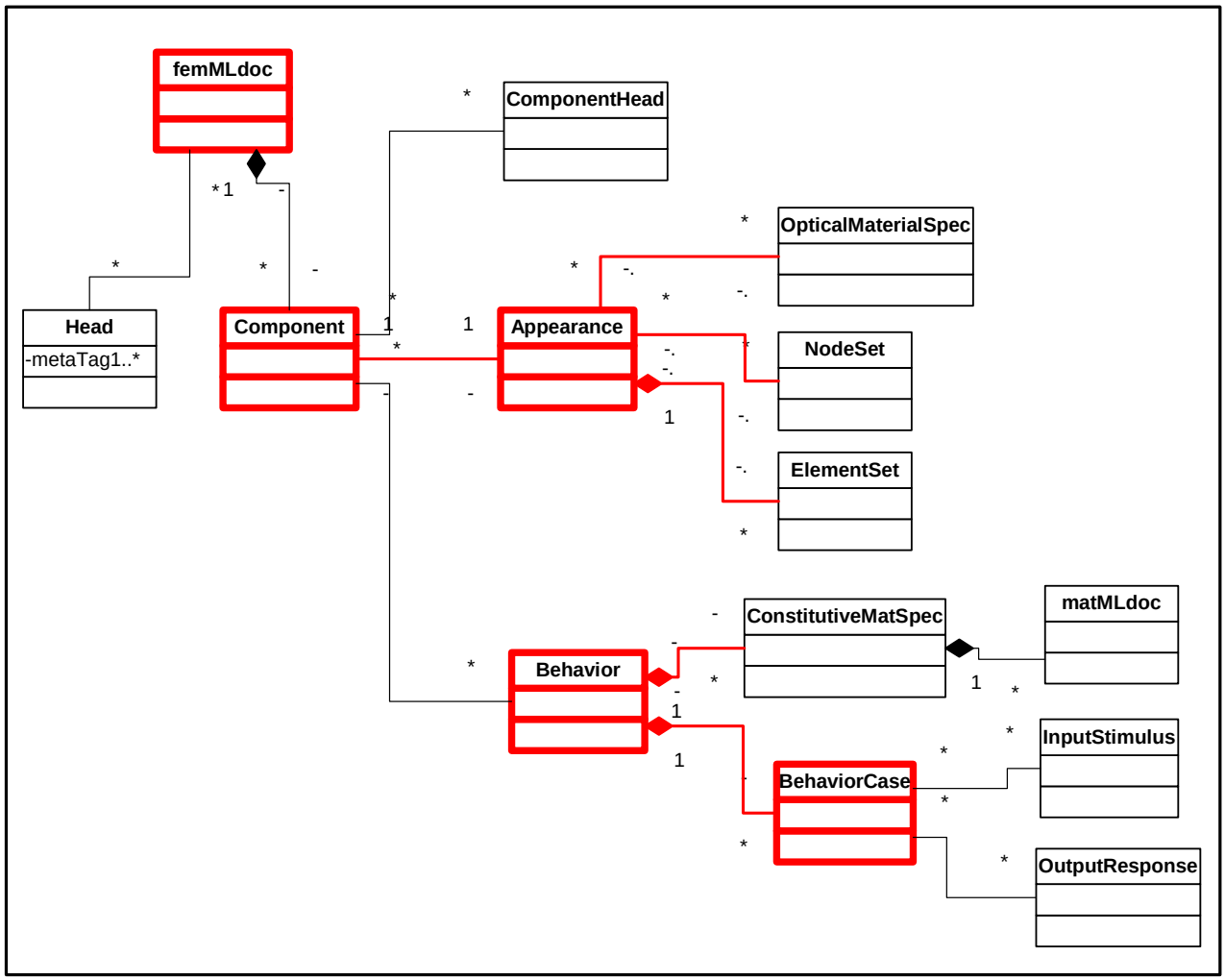

Figure 7. UML representation of the top two levels of an alternative femML DTD

Here the issue of compositionality is addressed immediately by the implementation of the ***Component*** node. Effectively this allows the existence of many components with one or many separate domains in one file. The ***Head*** node at the same level is intended to capture the global metadata for this file with a substructure very similar to the one described before for the current incarnation of femML DTD. What is drastically different from the current architecture here is the fact that each component node has always only three children nodes. The ComponentHead node that contains the metadata associated

with this particular component, and the ***Appearance*** and ***Behavior*** nodes. Effectively, these last two nodes manage to separate the geometry representation of the FEM data, from its behavior. This was done to ensure semantic independence between these two aspects of every discrete model of continuous structural system via FEM.

As expected the Appearance node has the usual children i.e. ***NodeSet*** and ***ElementSet***, but it also has the ***OpticalMaterialSpec*** node. The motivation of this node is based on the need to represent 3D geometries regardless of results data. It allows visualizing the 3D geometry of a model by assigning surface texturing, optical properties of the surface (reflectivity, emissivity etc) and in general parameters mostly known to the 3D rendering industry.

The behavior of a system in the continuous mechanics context depends on two major aspects.

The first aspect relates to the intrinsic material behavior of the structure that is usually captured by the constitutive equations applicable and the constants associated with them. To capture this aspect of the component behavior the node ***ConstitutiveMatSpec*** has been proposed as a child of the Behavior node.

The second aspect relates to the fact that the observed response of the system at the nodal or elemental levels depends on the stimulus on the system that is represented by the loading conditions. One has to have the stimulus-response representation of the system in order to claim an ability to represent behavior. For this reason ***BehaviorCase*** node has been introduced as another child of the Behavior node. Because the response of the system is specific to its stimulus and because we may have to deal with multiple cases of stimulus-response pairs this node can be repeated in the file.

To represent the stimulus for each one of the behavior cases we introduced the ***InputStimulus*** node that contains the loading and boundary conditions, and the ***OutputResponse*** node that contains the corresponding results from the FEM analysis. They are both children of the BehaviorCase node.

There is room for a lot o refinement and restructuring of these ideas and we hope that the interaction from the FEM community will help towards this end.

7. Proposed long term approach and concluding remarks

We feel that the in addition to the efforts on the custom development of femML by our group, a much more inclusive and robust effort would be to follow the approach similar to that used for developing MatML. The strategy of this approach has the following activities:

- Form, maintain and expand a working group with members from Academia, Industry, Government, Professional societies and Standards Organizations
- Identify issues to be resolved and their priority
- Develop and implement strategy for addressing issues
- Develop of the formal MatML document type definition or/and schema
- Development of a catalog of examples
- Application development and acceptance testing
- Utilize Open Source Development Network (OSDN) [47] resources like the "SourceForge" [48] development and deployment repository for DTD, SCHEMA, Examples, XSLTware, and custom format translator components
- Iterate
- DisseminationAt the present moment the working group is limited to NRL's CMS group

representatives and to the International Science and Technology Outreach Society (ISTOS) members [49]. As soon we publish the results of the femML effort at cms.nrl.navy.mil/femML/ we plan to invite people from relevant mailing lists to participate and contribute in this effort.

7. References

[01] J.G. Michopoulos, R. Badaliance, T. Chwastyk, L. Gause, P. Mast, C. Farhat and M. Lessoine, Coupled Multiphysics Simulation Of Composite Material Softening In A Virtual Wind Tunnel Environment, Proceedings of Sixth U.S. National Congress on Computational Mechanics, U.S. Association for Computational Mechanics, Dearborn MI, 1-3 August 2001, pp 521

[02] J.G. Michopoulos, P.W. Mast, R. Badaliance, I. Wolock, Health Monitoring of smart structures by the use of dissipated energy, ASME proc. 93 WAM on *Adaptive structures and material systems*, G.P. Carman/E. Garcia,eds., (ASME, AD-Vol. 35, 1993) 457-462.

[03] P. W. Mast, J.G. Michopoulos, R. Badaliance, H. Chaskelis, Dissipated energy as the means for health monitoring of smart structures, *SPIE Proc. Smart Structures and Materials 1994, Smart Sensing, Processing, and Instrumentation*, J. Sirkis, ed., (SPIE, Vol. 2191, 1994) 199-207.

[04] P. R. Factory, Health error prediction and sensor topology optimization on a smart pressure vessel, *Smart Structures and Materials 1995, Industrial and Commercial Applications of Smart Structures Technologies*, (SPIE, Vol. 2447, 1995), 155-166.

[05] V. M. Kern, Maintaining Semantics In The Integration Of Network Interoperable Product Data Models, Doctoral thesis submitted as a partial fulfillment of the requirements for a degree of 'Doutor em Engenharia de Produção' at the Universidade Federal de Santa Catarina, Brasil, Florianópolis, December of 1997.

[06] Y. Yang. "The STEP Integration Information Architecture". In: Law, K. (ed.): Engineering Data Management: Key to Success in a Global Market. Proceedings of the 1993 ASME International Computers in Engineering Conference and Exposition, San Diego-CA, pp. 39-47, 1993.

[07] P. Wilson, Information And/Or Data?, IEEE Computer Graphics & Applications 7, pp. 58-61, 1987.

[08] ISO 10303-1. Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles. Committee Draft, September 15, 1992.

[09] ISO 10303-11. Product Data Representation and Exchange - Part 11: EXPRESS Language Reference Manual. Document TC184/SC4/WG5 N65(P2). ISO International Standard, November 1, 1994.

[10] D. Schenck and P. Wilson. Information Modeling: The EXPRESS Way. Oxford University Press, New York, 388 pp., 1994.

[11] Recommended Practices for AP 209 – ME007.01.00, PDF document, can be accessed at http://pdesinc.aticorp.org/whatsnew/recprac209v1a.pdf, June 25, 1999.

[12] D. Tsichritzis and A. Klug (eds.) The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, Information Systems 3, pp.173-191, 1978.

[13] T. Bray, Introduction to the Annotated XML Specification, REC-xml-19980210, http://www.xml.com/axml/testaxml.htm

[14] Router Solutions Inc., XML Strategy for CAE/CAD/CAM Integration, Press Release, can be accessed at http://www.rsi-inc.com/Company/Archive/xmlpress.html

[15] Unigraphics Solutions, Unigraphics Goes Interoperable, Press release, can be accessed at http://www.ugs.com/publications/articles/automfg_10/

[16] Unigraphics Solutions, Unigraphics Solutions Introduces Revolutionary Interoperability Standard for *c-Commerce*, Press release, can be accessed at http://www.hoops3d.com/about/press/00_07_18parasolidext.htm

[17] Autodesk, Autodesk Unveils Cross-Platform, Cross-Industry Open XML Strategy for Design Data and E-Commerce, Press release, can be accessed at, http://www3.autodesk.com/adsk/item/0,,279672-123112,00.html
[18] Bentley, Bluestone and Bentley to bring xml-driven applications to the $3.6 trillion a/e/c marketplace, Press release, can be accessed at http://www.bentley.com/news/00q1/bluestone.htm
[19] R. Williams, XSIL: Java/XML for Scientific Data, White Paper, June 27,200, Can be accesses at http://www.cacr.caltech.edu/projects/xsil/xsil_spec.pdf
[20] Web 3D Consortium, X3D specification, Can be accessed at http://www.web3d.org/x3d/
[21] R. Carey and G. Bell, The Annotated VRML 2.0 Reference, Addison-Wesley Pub Co., 1997
[22] E. F. Begley, C.P. Sturrock, "MatML: XML for Material Property Data", ASM International's Advanced Materials & Processes (AM&P), November 2000, can be accessed at http://www.ceramics.nist.gov/matml/matml.htm
[23] ASCX12, What is EDI?, http://www.x12.org/x12org/about/
[24] Accredited Standards Committee X12 (ASCX12), http://www.x12.org/
[25] United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport http://www.unece.org/trade/untdid/
[26] ISO 8879:1986, Information processing -- Text and office systems -- Standard Generalized Markup Language (SGML)
[27] Coombs, James H.; Renear, Allen H.; DeRose, Steven J. "Markup Systems and the Future of Scholarly Text Processing." *Communications of the Association for Computing Machinery* 30/11 (1987) 933-947.
[28] World Wiede Web Consortium (W3C) main web site at http://www.w3.org
[29] P. Chan, R. Lee, and D. Kramer , The Java Class Libraries, 2nd Edition, Addison-Wesley Pub Co, March 1998
[30] Object Management Group (OMG) UML resources, http://www.omg.org/technology/uml
[31] L. E. Gumley, Practical IDL Programming, 1st edition (July 2001) Morgan Kaufmann Publishers
[32] JDOM effort is documented at the site http://www.jdom.org/
[33] Java Beans specification at http://java.sun.com/products/javabeans/docs/spec.html
[34] Xbeans specification at http://www.xbeans.org/
[35] P. Stern, F. Hemez, C. Farhat, RCFem: A Research Code for Finite Element Methodologies, User's Manual, Center of Aerospace Structures, University of Colorado, Boulder, September 1995.
[36] ANSYS, ANSYS Commands Reference, V 5.7 manual, can be accessed at http://www1.ansys.com/customer/content/documentation/57/Hlp_C_CH1.html
[37] HKS Inc., ABAQUS/Standard User's Manual (3 volumes),
[38] M. Fowler, K. Scott. *UML Distilled*, Second Edition. Boston: Addison-Wesley, 2000.
[39] D.Carlson. *Modeling XML Applications with UML: Practical E-Business Applications*. Boston: Addison-Wesley, 2001.
[40] NetBryx Technologies, EDITML XML editor, at http://www.editml.com.
[41] BizTalk.org can be accessed at http://www.biztalk.org
[42] ANSYS Inc., APDL Programmer's Guide, Publ. No. 001261, February 2000
[43] ANSYS Inc., Guide to ANSYS User Programmable Features, Publ. No. 001263 , February 2000.
[44] D. Harkey ,R. Orfali, Client/Server Programming With Java and CORBA , John Wiley & Sons, 1997.
[45] DataJunction Corporation, XMLjunction , can be accessed at http://www.xmljunction.net/
[46] J. Wernecke, The Inventor Mentor : Programming Object-Oriented 3d Graphics With Open Inventor, Release 2, Addison-Wesley Pub Co, (March 1994).

[47] Open Source Development Network, main site can be accessed at http://www.osdn.com/
[48] Open Source Development Network, Source Forge can be accessed at http://sourceforge.net/
[49] International Science and Technology Outreach Society (ISTOS) at http://www.istos.org.