

# XML Hierarchical Database for Missions and Technologies<sup>1,2</sup>

Raphael R. Some, Akos Czikmantory  
Jet Propulsion Laboratory, California Institute of Technology  
4800 Oak Grove Drive  
Pasadena, CA 91109-8099  
818-354-1902, 818-393-7083  
[Raphael.R.Some@jpl.nasa.gov](mailto:Raphael.R.Some@jpl.nasa.gov), [Akos.J.Czikmantory@jpl.nasa.gov](mailto:Akos.J.Czikmantory@jpl.nasa.gov)  
Jon Neff, Matthew Marshall  
The Aerospace Corporation  
2350 E. El Segundo Blvd.  
El Segundo, CA 90245  
626-395-0465, 626-395-0459  
[Jon.M.Neff@aero.org](mailto:Jon.M.Neff@aero.org), [Matthew.F.Marshall@aero.org](mailto:Matthew.F.Marshall@aero.org)

*Abstract*—A hierarchical XML (eXtensible Markup Language) database is being developed by the New Millennium Program to assist in technology Return On Investment (ROI) analysis and technology portfolio optimization. The database contains mission requirements and technology capabilities, which are related by use of an XML dictionary. The XML dictionary codifies a standardized taxonomy and, in effect, begins defining a set of ontological relationships for space missions, systems, subsystems and technologies. The XML dictionary, and the taxonomies defined therein, will be submitted to NASA, DOD and commercial standardization bodies as a proposed standard. In addition to being used for ROI analysis, the database is being examined for use in project planning, tracking and documenting. This paper describes the motivation for the database project, the technical approach, the reasoning behind the selection of an XML based approach, the current state of the project, and future plans for this work.

## TABLE OF CONTENTS

1. INTRODUCTION .....	1
2. DATABASE REQUIREMENTS AND DESIREMENTS .....	2
3. RELATIONAL DATABASE LIMITATIONS .....	4
4. HIERARCHICAL DATABASE CHARACTERISTICS AND ADVANTAGES .....	5
5. ADVANTAGES, CONSEQUENCES AND IMPLICATIONS .....	5
6. ARCHITECTURE AND ORGANIZATION ..	6
7. CURRENT STATUS AND FUTURE PLANS .....	11
8. CONCLUSIONS .....	12
9. ACKNOWLEDGEMENTS .....	12
REFERENCES .....	12
BIOGRAPHY .....	12

## 1. INTRODUCTION

NASA's New Millennium Program (NMP) [3] is chartered with the task of selecting high value, breakthrough technologies for future NASA science missions and maturing these technologies from the TRL 3-4 (breadboard) stage to TRL 7 (successful use in a flight system) [1,2]. In practice, the NMP technologists work with NASA Code S (Space Science Enterprise) and Code Y (Earth Science Enterprise) technologists to define needed capabilities which can only be provided by advanced, (i.e., beyond state of the art) technologies. These Technology Capability Areas (TCAs) are then used as the basis for open solicitations for technologies promising to provide these capabilities. The selection of technology providers is done through the NASA Research Announcement (NRA) process. The TCA identification and prioritization process, as well as the process used in evaluating individual technologies, has been, for the most part, a qualitative one, without a rigorous quantitative analysis by which relative rankings can be formulated, compared and understood.

To assist in the selection of high payoff TCAs and technologies, a means of providing a quantitative, traceable and defensible evaluation of expected Benefit and Return On Investment (ROI) was desired. A team was formed to develop a methodology and tool set for performing these ROI evaluations. The team was given two associated, but separate, tasks. The ROI Evaluation Task entails the development of a methodology for ROI evaluation and of gathering or generating provably valid data with which to populate the database and perform the ROI analyses. The Database Task entails the development of a database for holding the required data (e.g.: NASA goals; the goals, approaches and required capabilities for planned future NASA missions; the capabilities, current TRLs, development costs, and projected schedules of advanced technologies). In this paper, we discuss the database currently under development by the Database Task. A

<sup>1</sup> 0-7803-8155-6/04/\$17.00© 2004 IEEE

<sup>2</sup> IEEEAC paper #1001, Version 3, Updated September 30, 2003

companion paper deals with the tools and methodologies being developed under the ROI Evaluation Task.

The database being developed is XML-based and hierarchical in nature. It contrasts sharply with traditional relational databases currently used to maintain these types of data in that it uses a tree structure, rather than a flat record file, to organize the data. The basis of the tree is a set of taxonomies which describe the NASA organization, NASA space mission functions and structures, and a technology hierarchy. The taxonomies provide a hierarchical decomposition of each section of the database. Each node of the taxonomy tree contains a set of data defining that node's qualitative and quantitative descriptors or metrics. A single schema (data template) is used for all nodes, thus simplifying and unifying the database design. The types of data allowed in any given node, however, are defined by the node type, i.e., its taxonomical identity. The node's taxonomical identity is used, in effect, to personalize the generic schema for that specific node. The taxonomies, including descriptors, are embodied in an XML dictionary. The XML dictionaries rigorously define the data types allowed for each type of node and its associated descriptive metrics, and are the mechanism by which relationships between nodes such as NASA goals, mission requirements and technology capabilities are accessed, identified, and related. They define relationships between structural and functional entities and between organizations, missions and technologies. Thus, we use these XML dictionaries, and the taxonomies defined therein, to specify their conceptual relationships, or ontology. Using these XML tags allows, for instance, qualitative and quantitative comparison of technology capabilities to mission requirements.

In the following section we discuss the desired characteristics of the database, the limitations of relational databases with respect to achieving these goals, the advantages of a hierarchical database organization, and the advantages, consequences and implications of using XML for data definition, identification, and organization. We then discuss the database design approach, including: architecture and organization of the database, the database schemas, the XML dictionary as an implementation mechanism for required taxonomies, and external interfaces to the user community. We conclude with an explanation of current status and future plans.

## 2. DATABASE REQUIREMENTS AND DESIREMENTS

It was decided early in the task that the database would need to contain several different types of data, each in its own segment of the database (Figure 1). These were:

1. A representation of the NASA organization and, for each element of the organization, its scientific goals, priorities and its relationship to other NASA organizational elements.
2. A functional decomposition (aka functional flow or temporal-function diagram) of each NASA science mission of interest with quantitative requirements for each functional element.
3. A structural decomposition of each NASA science mission of interest with quantitative requirements for each structural element.

4. A description of each advanced technology of interest, including quantitative capabilities, as well as a means of relating these technology capabilities to higher-level functions they enable or contribute to.

It was also determined that the following elements would be required:

5. A standard interface for analysis tools, by which the tools can obtain their required data.
6. A standard manual query interface for human users incorporating both pre-defined 'canned' queries, and the ability to construct custom queries.
7. A standard interface for entering data. In this database, it is the users who both enter data and determine the actual structure of the database. While the database provides the ability to define a hierarchical relationship between database elements, it is the user, who determines, at data entry time, the specific local hierarchical structure. The user interface thus requires login authentication and a security model for authorization of viewing and updating different parts of the database. An administrative interface is also required to enable modification of the taxonomy and schemas.
8. A dictionary, rigorously defining the elements of the organizational, functional and structural taxonomies. This dictionary embodies these taxonomies and provides the means by which the database defines both qualitative and quantitative data and the inter-element conceptual relationships or ontology.

During the initial study phase of the task, several existing databases, database engines, tools, and technologies were studied to determine if we could avoid the expense and schedule delay of designing and implementing our own. In the end, it was determined that while many of the existing databases had been built with similar intended or desired characteristics, their implementations did not adequately address our needs. In accessing these databases and investigating their operation, they did, however, serve to point out both strengths and weaknesses in our initial concepts and allow us to refine our understanding of what we needed for our application. From these initial studies, a straightforward set of requirements was established.

1. The database must present the user with hierarchically organized data. The data is, by its nature hierarchical. Users (managers, scientists, and engineers), intuitively tend to view the data as a hierarchy and are most comfortable interacting with the database if the data is thus displayed. The database, therefore, should present a hierarchical interface to the user. This contrasts sharply with the typically flat organization of most relational databases.
2. All "fields" or types of entries in the database must be machine-readable and machine-"understandable". Inasmuch as one of the main purposes of the database was to act as repository of information for a set of automated analysis tools, the automated tools must be able to query the database to obtain their input data.

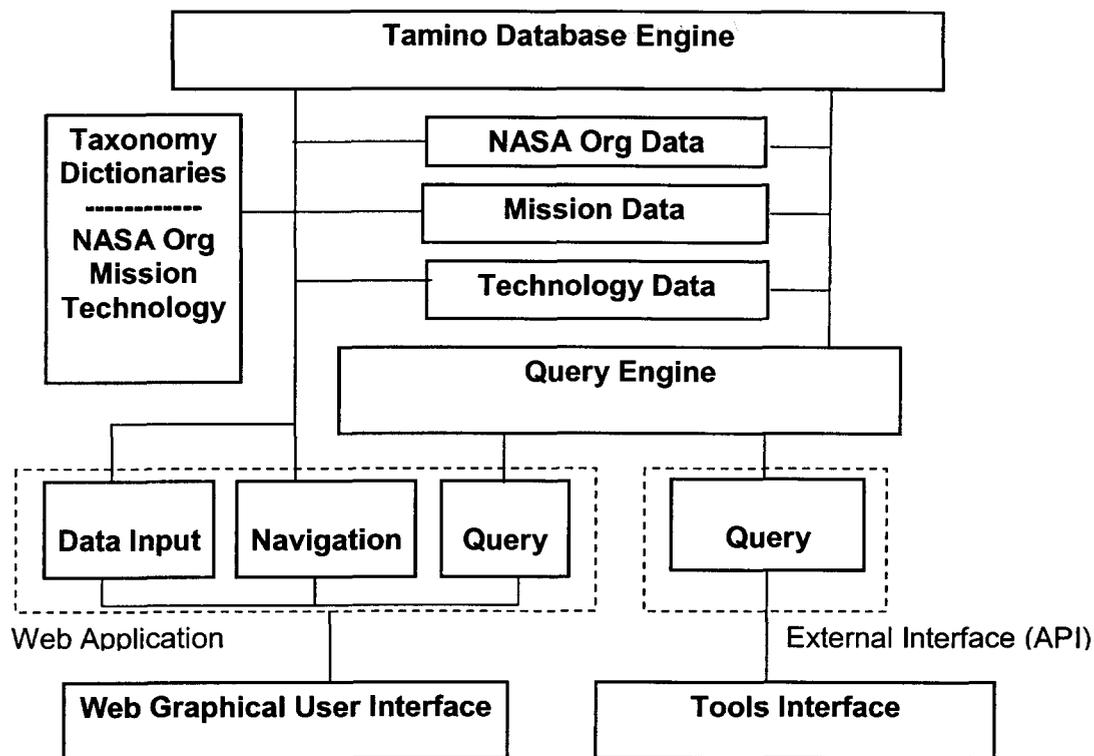


Figure 1 - Database Functional Block Diagram

Thus, the implementation of the database must explicitly define all types of data and the meaning of the data entries, including their conceptual meanings and relationships, in such a way as to allow these automated tools to access and unambiguously interpret the data. The machine readable ontology also provides a means by which the database may assist a user by showing these conceptual linkages and interrelationships, thus handling much of the complexity and offloading the user from having to know and/or manually explore all possible linkages and interrelationships between database elements.

3. The database must be web-accessible through a standard web browser. It was observed that many databases require custom software to be installed in the user's machine. While this is by no means a showstopper, it seemed to us an unnecessary burden on the user. In addition to the initial installation, it requires update every time a new version of the database engine or tool set is released. In some cases, the user is required to download the database itself into his/her local computer. Again, this seemed to us an unnecessary imposition. Finally, it was desired that users be able to access the database from anywhere, i.e., not only from a specific machine or location. This would allow both update and query of the database from arbitrary locations without coordination of client side software, and would eliminate the need for a single central database administrator or owner to enter all data. This last item is significant as it was our intent to not only allow, but to encourage, a wide variety of users across NASA and the US to input data, thus minimizing the data entry work of a single, central, administrator. The source of the data would then become

responsible for its entry, its timeliness, and its validity.

4. The database interface and operation must be, not only "user friendly", but intuitive and obvious. It is our expectation that users will be managers, scientists and technologists in a wide range of fields, and geographically remote. Further, they are expected to be casual and occasional users. To be attractive and useful, the database must be simple and straightforward to operate. Else, our intended customer set will simply avoid using it. As a goal, a technical, but untrained individual should be able to sign on the first time, and, within 5 -10 minutes, be doing useful work. Further, it should take this individual no more than 1-2 minutes the second time around.
5. The database must provide access control and security. It goes without saying that any such system must be secure against hacking and typical web based attacks. In addition, the database must handle multiple concurrent accesses for data input and queries, both manual and automated. Further, since the users, including those inputting data, will not be database experts, there must a simple means for the user to "undo" errors.
6. The database must allow population of arbitrary portions of the overall database structure to different depths of the hierarchy. In many cases, we will not need or want to flesh out the entire data set for a mission or a technology. This must not preclude filling in those portions of the database that are of interest.

7. The database must be efficient and fast. A user will become annoyed and will avoid using the system if access requires multiple minutes. The user cannot tell if he/she is the only user currently logged into the system or if there are 20 others and, quite frankly, doesn't care. The user cares only about his/her experience. Thus, the database must be efficient and fast.
8. The database must provide an historical record. One of the useful outputs of the database will be the historical progression of mission requirements and technology advances. As users continually update their inputs, an historical record must be kept. This history must be retrievable in such a way as to make these progressions clear.

### 3. RELATIONAL DATABASE LIMITATIONS

Currently, most databases are relational. Conceptually, a relational database can be likened to a spreadsheet, and this is often how they are represented. The spreadsheet "rows" are the database "records" and the spreadsheet "columns" are the "fields". If we try to imagine a large spreadsheet (or set of spreadsheets) containing all the data for all missions and technologies we can start to see the limitations of relational databases (See Figure 2).

One problem with using relational databases for this application is that tables quickly grow to be very large. The problem comes down to how one defines technology performance. Engineering requirements almost always

involve performance metrics, which can differ greatly depending on the technology. For example, a structural element may be characterized by the metric "stiffness" while a communications subsystem may be characterized by the metric "data rate." Furthermore, the metrics will have different units of measurement. In a relational database, metrics could be stored in fields, with each record representing a technology requirement. However, as the number of technology types increase, additional fields must be added to the table to accommodate new metrics. Furthermore, most of the fields would be empty.

In relational database models, this problem is addressed by normalization, i.e., breaking down the table into many smaller tables that are related by primary key - foreign key relationships. In principle, any hierarchical data can be decomposed into a normalized form in a relational database. However, as a practical matter, normalizing inherently hierarchical data leads to complex database designs that are time-consuming and inefficient to implement and operate [7]. In our experience, when technology planning databases are implemented, the sheer number of technology types and metrics overwhelms attempts at normalization. As a result, quantitative performance metrics end up buried in text fields.

This is especially true with respect to the user interface. It is possible, with sufficient coding, to implement a relational database that provides a hierarchical view to the user. However, this is not a desirable path given the difficulty of initial implementation and maintenance.

**Technology Requirements Data Entry**

Date Last Modified  Requirement Record #

Acronym  Mission

Science Theme  Mission Environment  Instrument Type

Cognizant Center  Record POC  POC Phone No.

**Product Breakdown Structure (PBS)**

TSG PBS1

TSG PBS2

TSG PBS3

Mission Attributes Affecting This Technology

Technology Item (Record Name)

Performance Metrics

Potential Solution Technologies (funded)

Funding start year:	Yr 1	Yr 2	Yr 3	Yr 4	Yr 5	Yr 6	Additional Cost to Complete	Total Cost
Estimated Funding Required: \$K:	<input type="text" value="240"/>	<input type="text" value="600"/>	<input type="text" value="500"/>	<input type="text" value="1,000"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="2340"/>

Status of Funding Estimate:  TRL Now  TRL Required

Theme Technologist Priority  Mission Priority For Technology

Need date  Need Date Tied to

Comments

Hierarchical structure isn't intuitively displayed or navigable.

Important data is buried in a text field. This does NOT encourage consistency, completeness or easy access to the data.

Technology links are buried in a text field.

Difficult to determine a complete set of common data fields and to modify this set for all entries. No customization for entries.

Figure 2 - Relational Database User Interface Example

#### 4. HIERARCHICAL DATABASE CHARACTERISTICS AND ADVANTAGES

A new type of database is emerging both in the research community and in the commercial market place. This new type of database allows developers to represent hierarchical data in XML form while providing query, transaction and security services similar to commercial relational database software. (In fact, hierarchical databases are not new; some earlier databases used hierarchical data models such as CODASYL. The hierarchical model is enjoying a rebirth with the advent of XML. [8]). While it will most likely not replace all relational databases, it is being aimed at just the sort of problem we have defined in implementing the NMP missions and technology database. The database canonically implements the data hierarchy. A hierarchy, in this case, can be thought of as a tree structure. An example of such a structure that is familiar to most of the technical community is the file system directory, as seen in the Windows or Macintosh stack of folders metaphor. Here, parent or higher-level folders may contain child, or lower-level folders, which, in turn, contain folders of yet a lower level. Each folder may also contain a specific type of data or file. If used as intended (but not enforced), "child folders", i.e., those

contained within their "parent folder" contain data that is a subset of the types of data contained in the parent folder. In addition, pointers (aliases or "shortcuts") are provided to allow linking logically connected, but non-adjacent, folders. In most cases, the folders are displayed, not as a tree, but as an indented list. While the indented list is a convenient and space-efficient format, it does, unfortunately, tend to hide the tree structure. Still it is a familiar construct with which most individuals are familiar, and for which the underlying structure is readily grasped. Figure 3, shows a typical hierarchical tree structure.

The advantages of a hierarchical database are basically the inverse of the disadvantages of the relational database, i.e.:

With a hierarchical database, the hierarchy is the native structure. There is no need to craft custom interfaces to hide the actual database structure or to interpret it for the user. Hierarchical data is stored in a hierarchical format (XML). A simple display interface allows the user direct access to the structure as implemented and the data as stored. System maintenance and debug efforts are much reduced. In this business, surprises are not good, and this ability to view the structures as they are tends to minimize surprises.

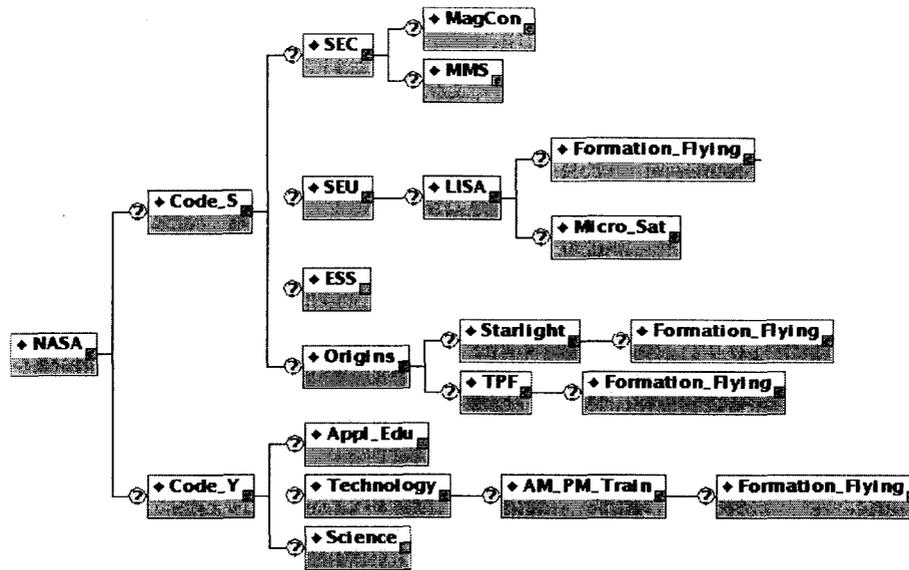


Figure 3 - Hierarchical Tree Structure

#### 5. ADVANTAGES, CONSEQUENCES AND IMPLICATIONS OF XML

##### Advantages

The use of an XML database offers several advantages over relational databases. Unlike relational database tables, XML is inherently hierarchical, which is a good match for space mission requirements and technology data. This data type tends to follow a structured decomposition from fundamental requirements to detailed specifications. Technology planning databases tend to be sparsely populated due to the very nature of research and development; if a technology is well understood and well documented, it probably isn't pushing the state of the art.

XML is also inherently more flexible than relational databases for this application. For example, adding new performance metrics to describe a technology could be painful in a relational database, requiring the addition of new tables, fields and relationships. However, the same modification in an XML database only requires the addition of the new metric element to the document. No change to the schema is necessary. In relational databases for technology planning, quantitative searches on performance metrics can be problematic. Usually, relational database designers end up putting performance metrics into a text field, making quantitative matching difficult. (See Section 3.) In our XML database implementation, the performance metrics are in a data structure by themselves, not buried in heterogeneous text fields. Quantitative matching between mission requirements and technology performance is enabled by the

fact that performance metrics are stored in separate data structures. These data structures are specified in a XML-based hierarchical taxonomy that defines technology capability types and their associated performance metrics. The taxonomy can grow and evolve to meet the needs of the user community for describing technologies and requirements.

### *Consequences*

XML is still a relatively new technology, which poses some problems for implementation. For example, the interface to the database uses XQuery, a query language specifically designed for performing searches on XML documents [4]. XQuery is one of the most complex specifications ever released by the World Wide Web Consortium (W3C) and is still a working draft. Furthermore, at this point, the draft XQuery standard has not been fully implemented by database vendors. This requires that some operations be performed outside of the database server software. Staffing this database project is complicated by the fact that relatively few people are familiar with XML and XQuery. Also, the very flexibility of XML poses some problems as more users begin to work with the taxonomy. If not carefully managed, the taxonomy could grow to unreasonable size and complexity as users request customized structures rather than fitting their data into a generic standard hierarchy.

### *Implications*

Developing and using standard XML taxonomies will allow NMP decision makers to trace the connections between mission requirements and new technologies on a scale that was not feasible with spreadsheets and relational databases alone. Over time, XML-based analysis tools will enable NMP to get more value out of technology funding, which should reduce the technology development cost of future space missions.

Developing taxonomies for NMP use is a first step in creating an industry standard to describe space mission requirements, a "Space Mission Requirements Markup Language." Of course, a community process would be needed to review, validate and support this standard. A successful standard would have profound implications for the space industry, enabling better communication within projects, between projects, and between vendors and customers.

The XML taxonomies being developed for NMP are more than a simple dictionary or set of naming conventions. The taxonomies are rigorous specifications of a conceptual framework for space mission organizations, architectures, structures, functions and technologies. They define relationships between structural and functional entities and between organizations, missions and technologies. Definition or specification of a conceptualization is ontology. By creating these ontological relationships, we begin the process of creating a machine-readable knowledge base through which the database can "understand" and act upon the relationships between the various items in the database. The near term, practical significance of this XML based ontology is that there are no hard links (i.e., explicitly coded linkages) between mission requirements and technologies. The database uses the taxonomical/ontological relationships between data items to understand how they relate to each other, to the questions (queries) being posed, and to new data being entered, by the user. For example, a

technologist need only determine that his or her technology is of a specific type, i.e., find its place in the taxonomy. Once this is known, the database understands how the technology would be used in space missions, the relevant technological parameters and characteristics, and the various subsystems and functions to which the technology is applicable. Further, by examining the qualitative and quantitative characteristics of the technology, the database can associate the technology with specific missions that it might enable or enhance as well as other technologies upon which it might be dependent or with which it may partner to provide a higher level capability. Thus, by encoding the ontological relationships within the taxonomies, we provide the basis for a level of machine intelligence that offloads the users and allows the machine to perform (at least preliminary) analysis and to draw (at least preliminary) conclusions about the data. In the future, this XML based, taxonomically encoded, ontology will enable autonomous software agents to search an electronic virtual space and to acquire and apply relevant meaningful data. The implications of these simple statements on how we work on and interact with computers are profound and far reaching.

## **6. ARCHITECTURE AND ORGANIZATION**

### *System Overview*

Figure 4 shows a high-level system overview. This information system is implemented in client-server architecture. The server side consists of the Tamino (TM) XML database server [7] and Tomcat, an open-source Java-based web application server. Tamino stores the XML documents and processes XQueries from the client side. Tomcat serves as a container for the Java Server Pages (JSPs), which generate the user interface, and utilizes the Java API objects (NmpDB API) to connect to the database. Tomcat version 5.0 is used to take advantage of the JSP 2.0 specification, especially the Expression Language (EL) [5]. The JSP Standard Tag Library (JSTL 1.0) is used extensively [6]. The client side includes a graphical user interface (GUI) accessed through a web browser and optionally an Excel based analysis tool for computing technology valuations.

### *Data Presentation in the User Interface*

The GUI requirements and analysis tool requirements drove the internal database organization. A brief review of the GUI is presented here to introduce the database structure.

The basic purpose of the GUI is to enable the user to browse the hierarchical data, query the database, display results, and edit data entries. There are two fundamental types of data stored in the database: mission capabilities (requirements) and technologies. Figure 5 shows the user interface when browsing capabilities. In the left frame is a tree that represents the NASA organizational hierarchy integrated with the required capabilities for a given mission. Each folder in the tree represents a data node. The capability taxonomy defines the metrics and allowable children for every node. The right frame presents a view of the data stored in the node selected from the tree. Capability nodes have a name, description, and capability type. Capability nodes contain metrics, which are figures of merit that quantify performance. Each metric instance has a name, a metric type (defined by the taxonomy), operator (=, >, <, >=, <=), units, and numerical value.

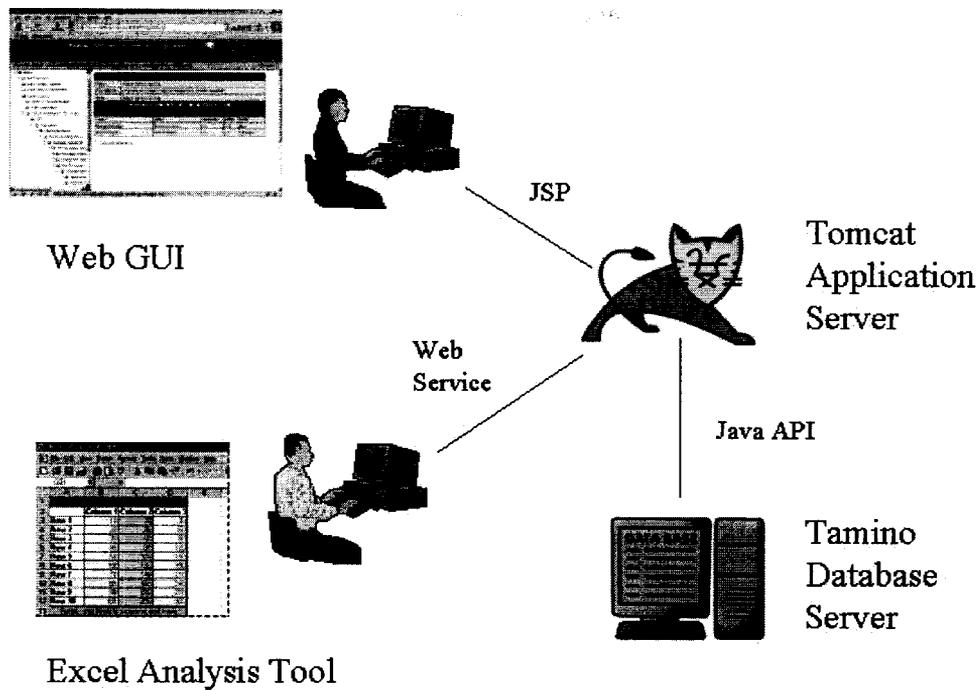


Figure 4 - System Overview

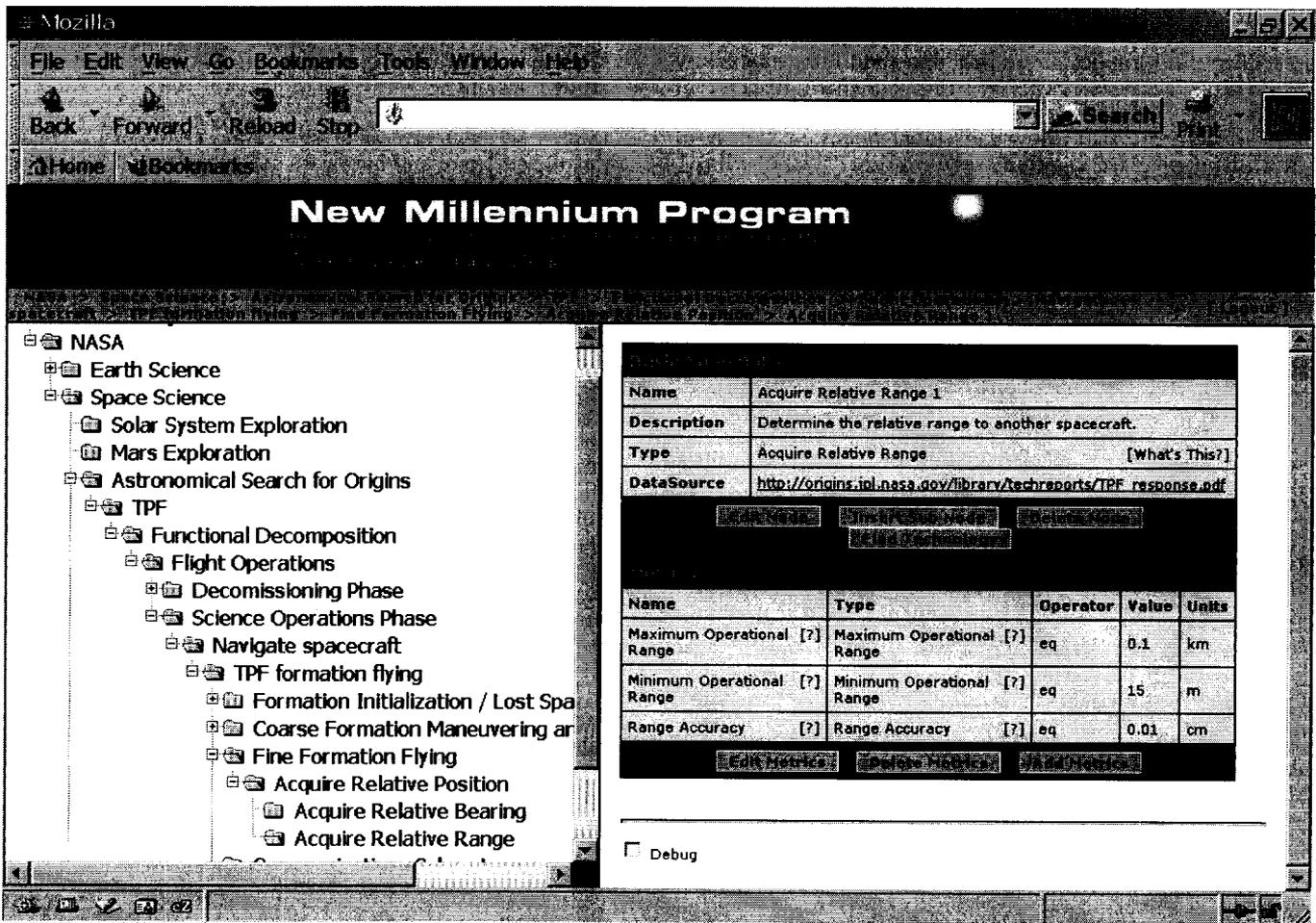


Figure 5 - Web GUI

Technology nodes are similar to capability nodes, although they are organized according to a separate technology taxonomy. In addition to name, description, and technology type, technology nodes have "related capability types." Rather than having their own metrics, technology nodes "inherit" metrics from the capability taxonomy by way of their relationship with capability types. Users specify the related capability types for a technology node. Related capability types enable users to search the database for technologies that satisfy required capabilities in the form of quantitative metrics. No one has to manually point a specific technology to a specific mission. This feature will become increasingly valuable, as the possible combinations of missions and technologies will grow rapidly.

In the future, the GUI will also display budgets, schedule milestones, and data sources.

#### *Database Architecture*

The database was designed, from the beginning, for flexibility. Creating additional "fields" for a node can be accomplished by adding optional elements and attributes to a schema without invalidating existing data. Data are not stored in the database as large monolithic documents. Rather, the data are stored in the form of individual nodes of different types. Allowable relationships between nodes are defined by the taxonomy documents. Thus, the data nodes form a "virtual hierarchy" connected by IDs that serve as pointers. The general rule is that the parent node contains the IDs of child nodes. In this way, the taxonomy can be updated with minimal impact on existing data. (Note that this flexibility applies to adding and modifying taxonomy elements; deleting elements is still problematic because of the potential for null pointers.) In the future, this internal virtual hierarchy will allow individual nodes to appear in more than one place in a tree using a sort of "symbolic link." A data warehouse, containing pre-composed hierarchies formed from data nodes and taxonomies, is used to cache data for presentation in the web GUI.

#### *Database Collections and Schemas*

The NMP Database is divided into three high level logical collections. They are the Data collection, Taxonomy collection and the Warehouse collection. Within each collection are defined schemas representing XML document types. Each instance of a schema (an XML document) represents a set of data elements at a particular level in a document hierarchy.

The **Data** collection defines the following schemas: *CapabilityNode*, *TechnologyNode*, *Metric*, *DataSource*, *Budget* and *Milestone*. (See Figure 7.) The Data collection is a grouping of XML documents that hold Capability/Technology related instance data.

The **Taxonomy** Collection defines the following schemas: *TaxonomyElement*, *MetricType*, and *Descriptor*. The Taxonomy collection holds exclusively taxonomy related data. The *Descriptor* schema defines a structure that holds a single instance of a Descriptor XML document. While not yet implemented in the GUI display, descriptors provide a

standardized way of describing non-quantitative characteristics of a technology. For example, a *TelescopeType* descriptor could indicate whether an electro-optical instrument is a *reflector* or *refractor*.

The **Warehouse** Collection defines the following schemas: *OrgCapability*, *MissionCapability*, *Taxonomy* and *Technology*. (See Figure 6.) The Warehouse collection holds the composed hierarchies of Data and Taxonomy Collection documents.

#### *Taxonomies*

Taxonomy is defined as: "the science of classification according to a pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval."<sup>3</sup> For the NMP XML database, four taxonomies have been developed: organizational, functional, structural, and technology. These taxonomies are a critical component of the NMP XML database philosophy.

*Organizational Taxonomy*—The organizational taxonomy decomposes the NASA organization into its various science themes (e.g., earth science, earth system science, space science) and sub-themes (e.g., solar system exploration, mars exploration).

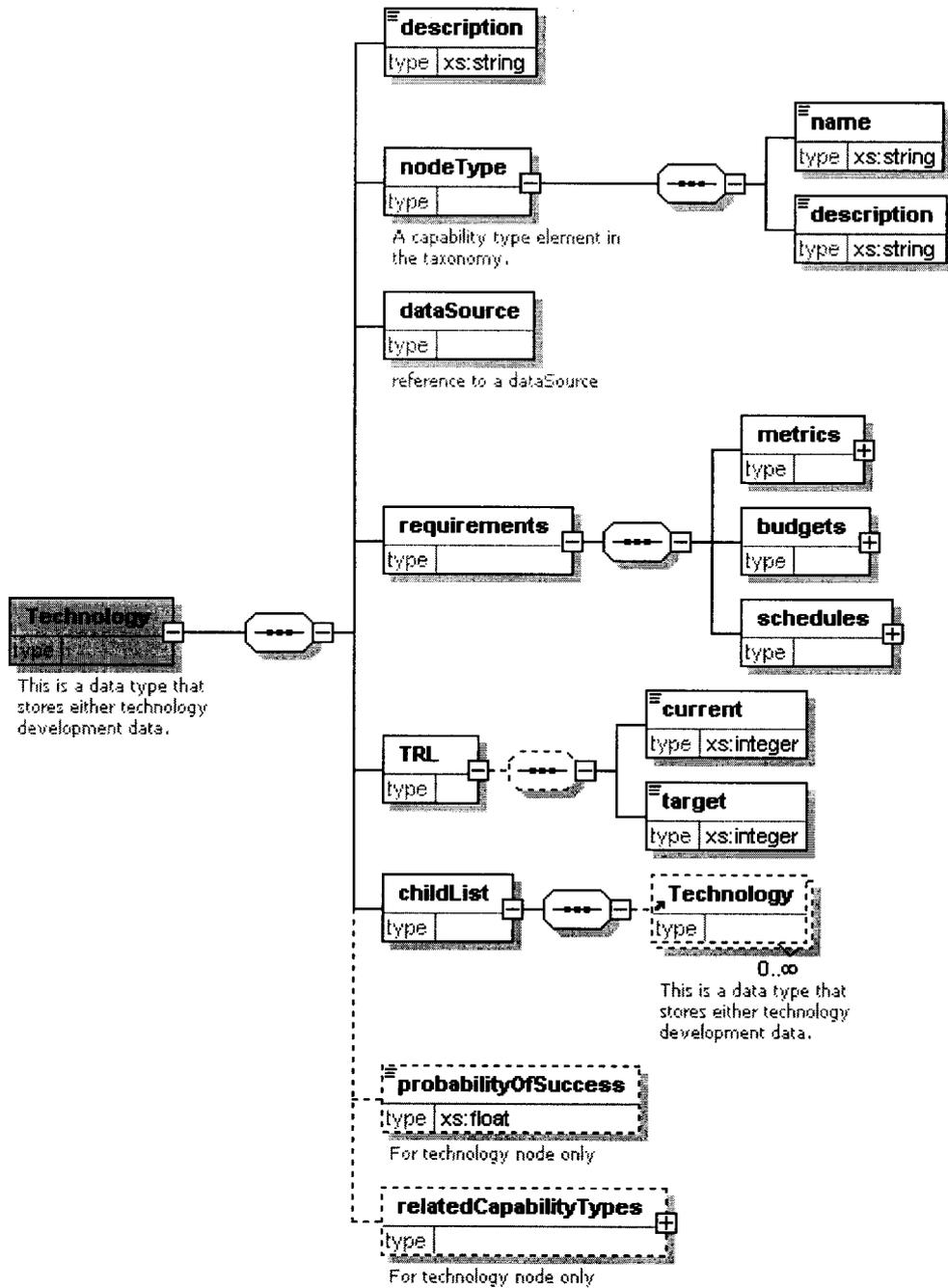
*Functional Taxonomy*—The functional taxonomy organizes all of the functions or operations that a space mission encompasses. Typical categories of functions include: launch and early orbit operations, cruise, science operations, and entry and descent operations (in the case of planetary landing missions).

*Structural Taxonomy*—The structural taxonomy organizes all of the software and hardware components of a spacecraft mission including the ground system, launch vehicle, spacecraft bus and payload. Figure 1 shows a portion of the hierarchy for the structural taxonomy, expanding on some of the detail of the spacecraft bus thermal control system. A portion of the structural taxonomy hierarchy is shown in Figure 8.

Both the structural and functional taxonomies include metrics or figures of merit for each of their levels. An example of the metrics in one of the nodes in the structural taxonomy (MEMS Louvers) can be seen in Figure 9. These metrics contain key performance parameters that are appropriate to their node of the taxonomy. The XML in Figure 9 contains tags for the metric name, description and units.

*Technology Taxonomy*—The technology taxonomy is a way to organize the various technologies, which NMP may consider for flight validation. Note that unlike the structural and functional taxonomies, the technology taxonomy does not itself contain any metrics. However when the XML database is populated, technology entries will be linked to the appropriate application of the technology on either the structural or functional taxonomies. The technology entry then inherits the metrics from its linked structural or functional node.

<sup>3</sup> [http://whatis.techtarget.com/definition/0,,sid9\\_gci331416,00.html](http://whatis.techtarget.com/definition/0,,sid9_gci331416,00.html)



**Figure 6 - Warehouse Collection – Technology Schema**

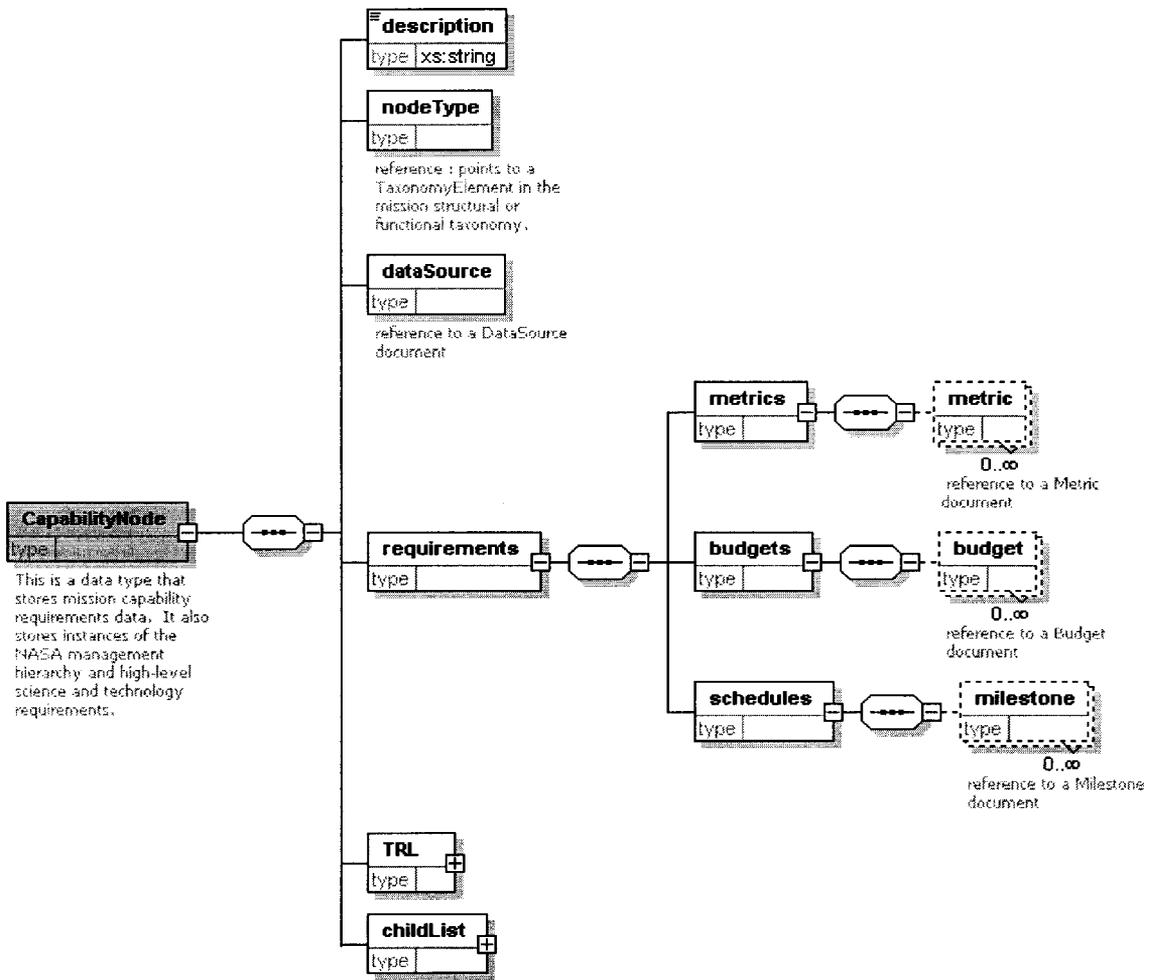


Figure 7 - Data Collection – Capability Schema

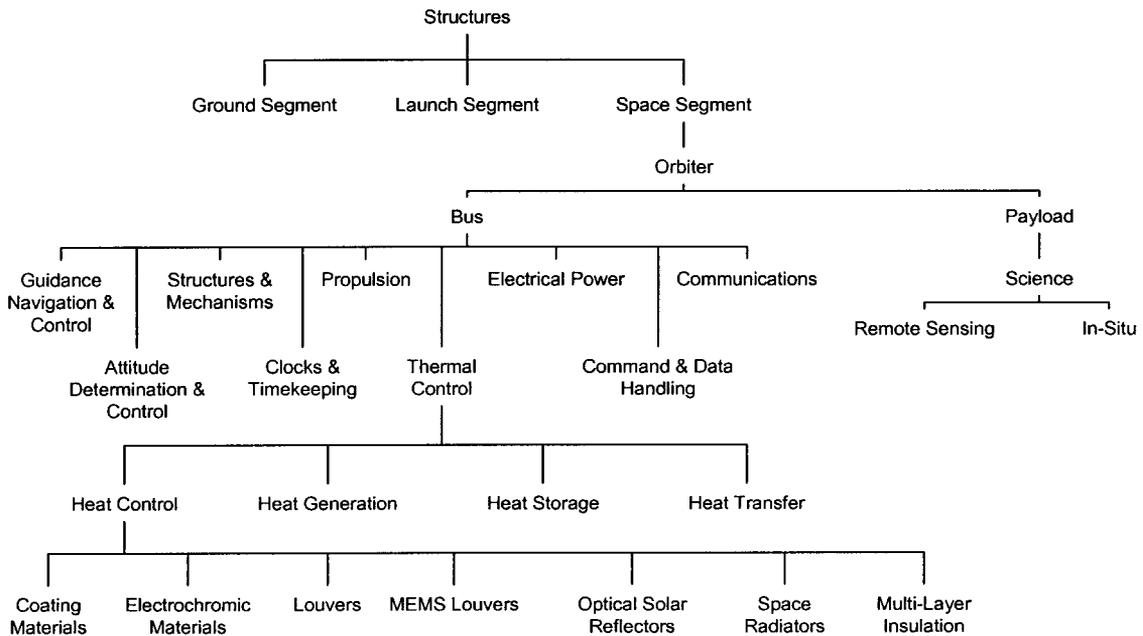


Figure 8 - Portion of Structural Taxonomy Hierarchy

```

    <component name="MEMS Louvers">
<description>MEMS Louvers perform nearly identically to conventional louvers except that they
    are
    fabricated on a microscale. Small actuators on a silicon wafer move small blades to
    change surface properties.</description>
    <metrics>
    <metric name="Mass per Unit Area">
    <description>Mass per unit area for given thickness.</description>
    <unit>kg/m^2</unit>
    </metric>
    <metric name="Operating Temperature Range">
    <description>Range of temperatures that component can operate within.</description>
    <unit>deg-C</unit>
    </metric>
    <metric name="Survival Temperature Range">
    <description>Range of temperatures that component can survive.</description>
    <unit>deg-C</unit>
    </metric>
    <metric name="Solar Absorptivity">
    <description>Ratio of solar energy absorbed compared to total incident energy.</description>
    <unit>>null</unit>
    </metric>
    <metric name="Infrared Emissivity">
    <description>Ratio of energy emitted by surface at a given temperature to the energy emitted by
    an ideal blackbody at the same temperature.</description>
    <unit>>null</unit>
    </metric>
    <metric name="Emissivity Range">
    <description>Range of emissivity values that can be attained by activating
    component.</description>
    <unit>>null</unit>
    </metric>
    <metric name="Time for State Change">
    <description>Time required for material to switch emissivity.</description>
    <unit>sec</unit>
    </metric>
    <metric name="Areal Power Consumption">
    <description>Power consumed by component per unit area.</description>
    <unit>W/m^2</unit>
    </metric>
    <metric name="Cycle Lifetime">
    <description>Number of cycles before component failure.</description>
    <unit>cycles</unit>
    </metric>
    </metrics>
    </component>

```

**Figure 9 - Sample Component Entry in Structural Taxonomy**

### *External Interfaces*

A Java Application Programming Interface (API) has been developed to provide an external interface to the NMP XML Database. There is a one-to-one mapping between API classes and the database schemas. The Java API can be utilized from a standalone application or a JSP-enabled web application. The API can also be published as a Web Service, enabling non-Java applications, such as the Excel-based analysis tool, to access the database. The API also provides some utility classes that aid in the development of the web GUI.

## **7. CURRENT STATUS AND FUTURE PLANS**

The database and user interface are currently in alpha testing. Taxonomies have been defined for the NASA

organization, structural capabilities (bus and electro-optical instruments), functional capabilities (including formation flying), and technologies. Schemas have been created to define the internal representation of the data as a virtual hierarchy. The user interface can browse the database and do limited capability-technology matching through a predefined XQuery. Basic login authentication has been implemented.

In the near future, we plan to add database update functions and more advanced capability-technology matching queries.

We will also add data source, data definition display, and related capability type navigation functionality to complete the basic user interface. We intend to modify the database to include a more advanced security model including user groups and authorization. An administrator's interface will allow easy modification of taxonomies as well as user and

group management. The taxonomies will be updated to include representations of active electro-optical, passive microwave and active radar instruments, as well as other technology areas of interest to NMP such as precision GNC (Guidance, Navigation and Control) and high-speed laser crosslinks. This set of functionality will allow the project to proceed into beta testing with NMP users in early 2004.

During beta testing, we plan to work with NMP to develop a more complete functional taxonomy, possibly with standard "blocks" of mission functions that can be assembled to describe many different missions.

Eventually, the database will serve as a common repository from which several different types of analysis tools, covering areas such as schedule and budget planning, as well as space mission trade studies, can retrieve data. Several interesting challenges have been identified, such as handling correlated metrics and technology dependencies, technology-dependent decompositions, and providing smarter software for relating technologies to capabilities.

## 8. CONCLUSIONS

A set of XML based taxonomies is being defined and submitted for acceptance by various government, academic and industry standardization bodies. The XML taxonomies form the core of a new type of database with broad application across the aerospace sector. The initial use of the database will be as an aid in technology ROI evaluation. The database is currently in alpha test with a first production version expected to be in use by end of CY'04. The XML dictionaries, if accepted and ratified, will, for the first time, provide a common language, allowing free and unambiguous exchange of information, across all sectors of the aerospace community.

## 9. ACKNOWLEDGEMENTS

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## REFERENCES

- [1] R. Mackey, R. Some, A. Aljabri, "Readiness Levels for Spacecraft Information Technologies" *Proceedings of the 2003 IEEE Aerospace Conference*, March 2003
- [2] C. Minning, P Moynihan, J. Stocky, "Technology Levels For The New Millennium Program", *Proceedings of the 2003 IEEE Aerospace Conference*, March 2003
- [3] C. Minning, D. Crisp "NASA's New Millennium Program: Flight Validation of Advanced Technologies for Space Applications", *Journal Of Space Mission Architecture*, Vol. 3, Fall 2003
- [4] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, J. Siméon, "XQuery 1.0: An XML Query Language," W3C Working Draft 02 May 2003. <http://www.w3.org/TR/2003/WD-xquery-20030502/>
- [5] Q. Mahmoud, "Developing Web Applications With Java Server Pages 2.0," July 2003. <http://developer.java.sun.com/developer/technicalArticles/java/vaserverpages/JSP20/>

[6] Q. Mahmoud, "Faster Development with Java Server Pages Standard Tag Library (JSTL 1.0)," October 2002. <http://developer.java.sun.com/developer/technicalArticles/java/vaserverpages/faster/>

[7] M. Champion and F. Jung, "Tamino XML Server," Software AG White Paper, August 2003. [http://www.softwareag.com/tamino/download/e-WP\\_XMLserver\\_INS-WP17E0803.pdf](http://www.softwareag.com/tamino/download/e-WP_XMLserver_INS-WP17E0803.pdf)

[8] S. Hong, "Introduction to Database Management Systems," <http://www.cis.gsu.edu/~shong/teaching/cis814/slides/intro1.pdf>

## BIOGRAPHY

**Raphael Some** has over 27 years experience in technology planning, development and commercialization. He is a Level 1 Manager at the Jet Propulsion Laboratory, where he currently serves as avionics and microelectronics technologist in NASA's New Millennium Program. Prior positions at JPL include Chief Engineer and Systems Engineering Manager of the REE project and initiator of JPL's Smart Sensor Web. Prior to joining JPL, Raphael held a variety of positions in the aerospace industry including VP of Engineering at 3D Micro, Chief Engineer and Technical Director at Irvine Sensors Corporation., and Technical Director at Raytheon Corporation.

**Jon M. Neff** is a Project Leader in the Civil and Commercial Division of The Aerospace Corporation. Previously he worked at the Jet Propulsion Laboratory and as an engineering manager at small software companies. His areas of interest include software engineering, strategic planning for technology development, and remote sensing. Jon has taught risk management and decision theory in The Aerospace Institute's System Architecture/System Engineering Certificate Program. Jon has received several awards from NASA and The Aerospace Corporation for his work in software engineering, technology planning, and project management. He has BS, MS, and PhD degrees in Aerospace Engineering from the University of Texas at Austin, and an MBA from Pepperdine University.

**Akos J. Czikmantory** is a Software Engineer at the Jet Propulsion Laboratory in Pasadena, California. He has been with JPL for the past six years. His interests include database development, Java programming and infusing new technologies such as XML into current JPL projects. He has a BS degree in Computer Science from the California Polytechnic University at Pomona.

**Matthew Marshall** is a Senior Project Engineer for The Aerospace Corporation supporting the Civil and Commercial Division. Matthew has a broad background in systems engineering, integration and architecture on both civil and military space programs. He has been involved in a number of activities focusing on satellite communications and the military's use of commercial space systems. In 2001 he led the first ever commercial cell at an Air Force space war game. He was also involved in market research activities for the Wideband Gapfiller Program. Currently he provides support to NASA's New Millennium Program and conducts studies of future Mars robotic outpost architectures. Matthew holds a Masters Degree in Systems Architecture & Engineering from the University of Southern California.