

# Lean Middleware

David A. Maluf  
NASA Ames Research Center  
MS 269/3  
Moffett Field, CA 94035  
(001)-650-604-0611  
[David.A.Maluf@nasa.gov](mailto:David.A.Maluf@nasa.gov)

David G. Bell  
USRA RIACS  
NASA Ames Research Center  
MS 269/3, Moffett Field CA 94035  
(001)-650-604-0771  
[Dbell@email.arc.nasa.gov](mailto:Dbell@email.arc.nasa.gov)

Naveen Ashish  
USRA RIACS  
NASA Ames Research Center  
MS 269/3, Moffett Field CA 94035  
(001)-650-604-2822  
[Ashish@email.arc.nasa.gov](mailto:Ashish@email.arc.nasa.gov)

## ABSTRACT

This paper describes an approach to achieving data integration across multiple sources in an enterprise, in a manner that does not require heavy investment in database and middleware maintenance. This “lean” approach to integration leads to cost-effectiveness and scalability of data integration in the enterprise.

## 1 INTRODUCTION

Current EII technology requires significant investment in ‘heavy-weight’ middleware for an application of any scale or requirements. For each integration application, we need to define schemas or views for each source, and reconcile the schemas or form integrated global schemas or views to facilitate the integration. This approach, unfortunately, causes the IT cost for integration applications to increase linearly with the application size as shown in Fig 1.

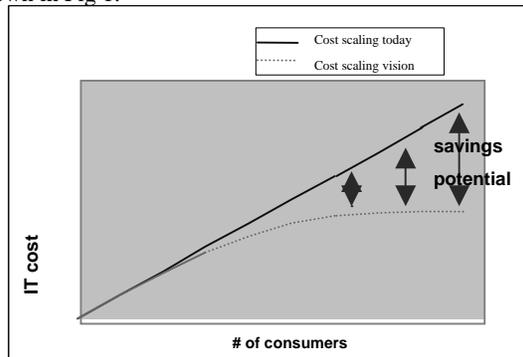


Fig 1. Costs of data integration

We present an approach to data integration that is *cost-effective* and scalable. The approach is based on key insights that permit an integration approach that is more flexible and *adaptable* to different integration applications.

The majority of integration applications at NASA (and enterprises in general) are built over data in documents, spreadsheets, reports

Copyright 2002 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by a contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA  
Copyright 2005 ACM 1-59593-060-4/05/06 \$5.00

and presentations as input. Also, the query processing (and result composition) requirements mostly require extracting particular sections from documents, composing new documents with sections from other multiple documents, or doing keyword based searches on documents (we use the term ‘documents’ to include documents in formats such as Word, PDF, HTML, XML or others, Excel spreadsheets, presentations in Powerpoint, reports, etc., henceforth). Our approach has thus been to develop a data management and integration system focused on the above capabilities, and that does not require investing in functionality that is unnecessary for common applications.

The following sections contain a description of this approach. A fundamentally new approach to data integration is presented, including technical details for data storage, query processing etc., with this approach, followed by a brief overview of enterprise applications built and a conclusion.

## 2 A COST-EFFECTIVE and SCALABLE INTEGRATION APPROACH

A cost-effective approach to integration is achieved by keeping the middleware “lean”. Specifically, this is based on the following design principles:

- The database will be nothing more than an intelligent storage component that stores the data, but does not impose a formal structure or semantics in the form of schemas on the data. In other words it is “schema-less”.
- Any imposition of schema on the data will be done at the client side.
- Any required integration across multiple sources will be done at the client and on the fly.

Arguments questioning the tacit acceptance of formal schemas and subsequent necessity for schema maintenance are outlined in [1]. The following sections describe an approach and system for integration that does away with schemas and associated schema-maintenance.

### 2.1 NETMARK: Schema-Less Information Management and Integration

NETMARK[2] is an information management and integration system developed based on the above design principles. Fig 2 provides a high-level architectural view of NETMARK. In most basic terms, NETMARK can manage and integrate information from both databases as well as raw information in business documents. Further, aggregated and integrated data can be

accessed through commonly used business documents (such as MS Word and Excel) in the format desired. The following sections discuss aspects such as data storage and management, query processing, and integration in NETMARK.

### 2.1.1 Data Storage and Management

NETMARK is designed to effectively store and manage structured data as well as semi-structured data found in documents, web-pages and spreadsheets. For managing semi-structured data, we have seen a significant amount of activity in building XML data management systems in the last several years. The XML data management systems fall into two broad categories. One is based on an approach of building an XML data management system over a relational data management system [3]. Any XML documents to be stored are “shredded” into relational tables and stored as relational data. The other approach, called the native XML approach, is based on storing XML documents and structures in underlying tree structures corresponding to the XML documents [4]. Note that both approaches are “schema-centric” and “schema-dependant” in that the structure of the data stored in the database system depends on the structure of the XML document being stored.

In NETMARK, we first convert any documents to be stored, originally in any format such as MS Word, PDF, HTML, Excel spreadsheets and others to XML. This conversion is done automatically using a library of parsers for various document formats. These parsers essentially take hints from the formatting information in a document to extract what are called *Contexts* in that document. Contexts can be thought of as sections or subsections in a document. For instance, for this paper, the “Abstract”, “Introduction”, or “Data Storage and Management” sections can be thought of as contexts. The data within a section (or context) is referred to as *Content*, for that context. So the text in the abstract section is the content for the “Abstract” context. This context and content information for a document is captured in XML, for instance for this paper, the XML representation would look as shown in Fig 3.

We will discuss query processing with context and content shortly after, and continue with describing document storage. Each document is converted to XML with context and content information as illustrated above and then stored in the NETMARK XML Store. In NETMARK we store the XML documents as relational tables in an underlying ORDBMS. Approaches such as [3] define different relations for different XML element types. The NETMARK storage scheme however uses the *same* relational tables to represent and store *any* XML element type. The NETMARK ‘SGML parser’ (Fig 2.) decomposes the XML (or even HTML) documents into its constituent nodes and dynamically inserts them into two primary database tables—namely, XML and DOC—within a NETMARK generated schema. The descriptions of the XML and DOC tables along with their respective relationships are listed in Fig 4.

Object-relational mapping from XML to relational database schema models the data within the XML documents as a tree of objects that are specific to the data in the document [14]. In this model, element type with attributes, content, or complex element types are generally modeled as classes. Element types with parsed character data (PCDATA) and attributes are modeled as scalar types. This model is then mapped to the relational database using

traditional object-relational mapping techniques or via SQL3 object views. Therefore, classes are mapped to tables, scalar types are mapped to columns, and object-valued properties are mapped to key pairs (both primary and foreign). This mapping model is limited since the object tree structure is different for each set of XML documents. On the other hand, the NETMARK SGML parser models the document itself (similar to the DOM), and its object tree structure is the *same* for all XML documents. Thus, NETMARK is designed to be *independent* of any particular XML document schemas and is termed to be “*schema-less*”.

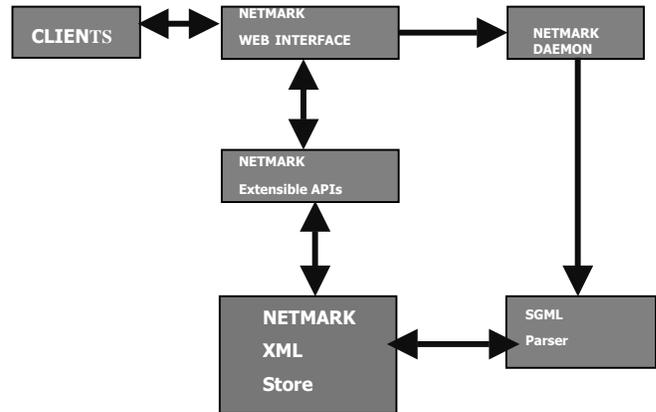


Fig 2. NETMARK system architecture

```

    ....
    <Context>Abstract</Context>
      <Content> This paper describes an ... </Content>
    <Context>The NETMARK Data .... </Context>
      <Content> We now describe NETMARK .. </Content>
    <Context>Data Storage and Management</Context>
      <Content>NETMARK is designed to .... </Content>
    .....
  
```

Fig 3. Document in XML

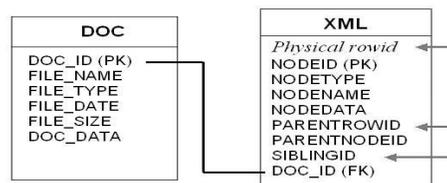


Fig 4. NETMARK Generated Schema

Note that we have now provided a means to generically store any XML or HTML document without requiring a new schema for a new document (type). We have also captured the context and content information in each document. Relationships between various XML elements (such as parent, child, sibling etc.) are

captured using ROWID pointers between nodes. As described shortly after, we end up retrieving nodes related to a node repeatedly during query processing. We have exploited the feature of physical ROWIDs in Oracle [5] that provide the fastest access to any record within a table with a single block read access, for very fast traversal between nodes that are related

### 2.1.2 NETMARK Data Input and Access

We outlined the NETMARK system architecture and process flow in Fig 2. above. Users insert new documents (in any format such as Word, PDF, HTML, XML or others) into NETMARK by simply dragging the documents into a (NETMARK) desktop folder. The 'NETMARK DAEMON' periodically picks up these documents passes them onto the 'SGML Parser', which converts the documents into XML. The XML documents are then stored in the 'NETMARK XML Store' in a schema-less manner, as described above. Communication between the user folders and the NETMARK server is done using Web DAV [6] which is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers.

Clients and applications can access and query data through the 'NETMARK Extensible APIs' using a variety of protocols based on J2EE, RMI, and ODBC. Users can access NETMARK documents by simple HTTP requests, in fact HTTP provides an extremely simple yet powerful mechanism for users and clients to access NETMARK

### 2.1.3 Querying Data in NETMARK

We now look at data querying capabilities in NETMARK, which are centered around the notions of context and content. A key capability is that of *context search*. A context search query, such as Context=Introduction (query syntax illustrated informally) will return the content portion in the 'Introduction' sections (the text in the Introduction section) in *all* the documents in a document collection. NETMARK also provides for result composition and formatting where we can use XSLT [7] to specify the format of the query results before presenting to the user. For instance we could specify a context search for "Technology Gap" and specify that the integrated results be presented in a new document. This is illustrated in Fig 5. Users can also specifying *content searches* which are essentially keyword searches that return all documents containing the specified search terms. For instance, a content query such as Content=Shuttle will return all documents that contain the term 'Shuttle' anywhere in the document. One can also combine context and content searches, for instance a query such as Context=Technology Gap & Content=Shrinking returns the "Technology Gap" contexts (sections) of all documents where the term 'Shrinking' occurs *within the Technology Gap context (section)*.

We must mention that there could be different interpretations of what is the content within a particular context. For instance, in this paper, the content associated with the context "2.1 The NETMARK Storage and Integration Approach" could be the text and figures in the *entire* section, i.e., including all sub-sections, or it could be the text (and figures) up to the next context which is the sub-section "2.1.1 Data Storage and Management". We can configure exactly what interpretation is to be made and will not mention the details here.

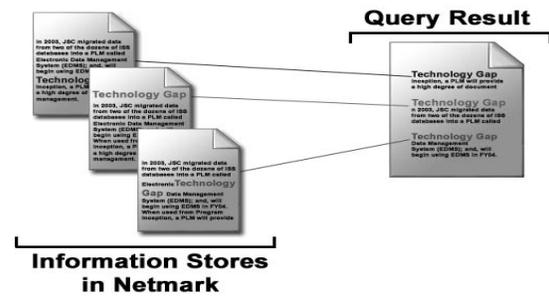


Fig 5. Context Search

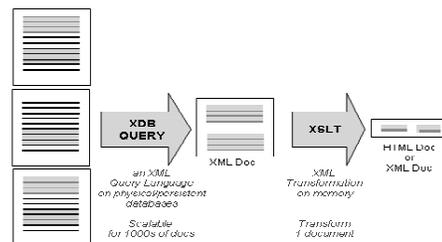


Fig 6. XDB Query Search and Transformation

The Netmark query language is a language called XDB Query [8]. XDB Query allows for posing the context and content kinds of queries over XML documents, as illustrated above. We will not go into the query syntax details here but the key features are that context and content search specifications are appended to a URL that is sent to NETMARK. In this URL we may also specify an XSLT stylesheet which specifies how the results are to be formatted and composed into a new document. Fig 6. provides an illustration of using XDB Query to query the data in NETMARK and then using XSLT to format the results. XSLT transformation is done using the Xalan XSLT processor [9].

Query processing itself involves taking a context and/or content query and doing an index search for nodes that match the keywords specified. Also the ROWID pointers are used for fast traversal between nodes (which is required as associated context and content nodes have parent-child or sibling relationships to one another). The reader is referred to [2] for more details.

### 2.1.4 Accessing Multiple Data Sources

In the above sections we have elaborated on query processing over data in the NETMARK XML Store. NETMARK can also provide integrated query access to multiple information sources that may be distributed at other locations. This is done through a simple declarative process where an administrator creates a 'Databank' for an application. The databank specifies what sources are to be queried when a user fires a query to that application (databank). A source that is queried need not necessarily have XML querying or even Context+Content searching capabilities. However NETMARK 'augments' the query capability in that it uses whatever query and search

capabilities are available at the source and then does further processing required.

Having outlined the key features and functionality of NETMARK we summarize the distinguishing characteristics of this system and approach:

- We are able to provide sophisticated query facilities, based on context and content, over any information repository (that may otherwise have limited or no query capabilities) with NETMARK.
- Integration can be specified (and executed) at the *client* side by specifying databanks. Thus integration can be done on-the-fly.
- Middleware requirements are reduced to needing just a thin router capability across the various information sources.
- The approach is highly scalable and flexible in that we can take arbitrary numbers of sources and compose applications that access one or more sources amongst these as and when required.

### 3 NASA APPLICATION EXAMPLES

NETMARK has proven to be a highly flexible, nimble, and easy to assemble application framework for several integration applications that we have built using this framework at NASA. Table 1 contains a list of some of these applications along with the time that was taken to assemble them with NETMARK. We cannot describe all the above mentioned applications here, but all of them are centered around having to extract and integrate data from several heterogeneous and distributed documents to form either an integrated information system or an integrated document or report. For instance, the Proposal Financial Management application is an information system for tracking proposal financial information for outgoing (NASA) proposals in response to a call for proposals. This allows querying of aggregated and statistical information about the proposals such as proposal numbers by NASA division type, dollar amounts requested etc. The application takes as input all the proposals (typically in formats such as Word or PDF) that have been submitted in response to a particular call. The Integrated Budget Performance Document (IBPD) is an integrated budget document which unifies previously disconnected budget documents. While manual assembly of the IBPD can take several weeks, NETMARK was

used to extract and integrate information from thousands of NASA task plans containing the required budget information and compose an integrated IBPD document. Finally, Anomaly Tracking is an application that allows integrated querying of two NASA (web accessible) data sources that are essentially anomaly tracking databases. The application facilitates more sophisticated querying than provided by either original source and also facilitates simultaneous querying of both sources.

**Table 1. NASA integration applications**

Application Assembly Time	NASA Application
1 hour	<u>Proposal Financial Management</u> <u>Risk Assessment</u>
1 day	<u>Integrated Budget Performance Document</u>
1 week	<u>Anomaly Tracking</u>

Clearly NETMARK has proven to be a scalable, fast, and flexible integration framework for all of the above NASA integration applications.

### 4 CONCLUSIONS

The above integration approach does not claim to replace traditional schema-centric mediation approaches completely. Mediation based approaches such as [10, 11] certainly provide a more formal and expressive power for the “integration-glue” across different sources, which may well be needed in some applications. Our focus is on the requirements of common integration applications in the enterprise. Formal schema imposition on any data is there only to the extent that it needs to be. All integration functionality is pushed to the client. No mandatory heavy-weight integration middleware is required, rather the desired integration capabilities can be specified on the application side and on-the fly. The integration framework has been very successfully used to develop several NASA enterprise applications in a very cost-effective manner and within short time-frames.

### 5 REFERENCES

[1] A. Halevy *et al.*, Enterprise Information Integration: Successes, Challenges and Controversies *ACM SIGMOD International Conference on Management of Data*, Baltimore MD 2005.

[2] D. Maluf, P. Tran, NETMARK: A Schema-Less Extension for Relational Databases for Managing Semi-structured Data Dynamically *ISMIS* 2003.

[3] J. Shanmugasundaram *et al.*, *SIGMOD Record* **30**, 20-26 (2001).

[4] H.V.Jagadish *et al.*, *VLDB Journal* **11**, 274-291 (2002).

[5] Oracle 9i Database Release 9.0.1 Developer Guide.

[6] J. Whitehead, Y. Goland, WebDAV: A network protocol for remote collaborative authoring on the Web *CSCW* 1999.

[7] XSLT, <http://www.w3.org/TR/xslt>.

[8] D. Maluf, P. Tran, T. La, "An Extensible 'Schema-less' Database Framework for Managing High-Throughput Semi-Structured Documents *IASTED, Applied Informatics* 2003.

[9] Xalan, <http://xml.apache.org/xalan-j/>.

[10] D. Draper, A. Y. Halevy, D. S. Weld:, The Nimble XML Data Integration System *ICDE* 2001.

[11] C. A. Knoblock *et al.*, *International Journal of Cooperative Information Systems (IJCIS) Special Issue on Intelligent Information Agents: Theory and Applications* **10**, 145-169 (2001).