View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NLR Reports Repository

Nationaal Lucht- en Ruimtevaartlaboratorium

National Aerospace Laboratory NLR

Executive summary

Storing heterogeneous helicopter signal data: Advantages of using an XML Database

Problem area
The Netherlands Armed Forces operate several types of helicopters, such as Chinook, Apache, Cougar, Lynx, and in the future NH90 helicopters. To increase reliability and availability, and to decrease maintenance costs, insight on the impact of helicopter missions on the life of the helicopter's airframe, helicopter's engine, and other sub systems is required.

Description of work
To support gaining insight on the impact of the missions on the life of the helicopters' sub systems, the Netherlands Armed Forces awarded NLR a contract for the development of a Helicopter Life and Usage Monitoring system (HELIUM) for collecting, storing and analyzing large amounts of flight data, health data and usage data of the helicopters. An essential part of HELIUM is the data storage environment for storing flight administrative data as well as data available from Flight Data Recorders, Spectrapots, ACRA boxes, and Health and Usage Monitoring Systems. This paper investigates the advantages of using Extensible Markup Language (XML) and XML Database technology to implement this data storage environment.

Results and conclusions
In general, using structured documents as input for the data storage and output from the data storage provides many advantages such as easy conversions, well defined syntax, excellent support in for example the Java programming language, and a human readable format.

Using XML database technology for storing the measured flight and administrative documents provides a data storage in which the logical structure – the structured documents – is separated from the physical storage.

Report no.
NLR-TP-2007-536

Author(s)
B.C. Schultheiss
A.M. Vollebregt
C. Hummelink

Report classification
UNCLASSIFIED

Date
September 2007

Knowledge area(s)
Aerospace Collaborative Engineering & Design
Health Monitoring & Maintenance of Aircraft

Descriptor(s)
XML Database
XQuery
XPath
Signal data
Life and Usage monitoring

This report is based on a presentation held at the IEEE AUTOTESTCON 2007 conference, Baltimore, U.S.A., 17-20 September 2007.

Using XML database technology, heterogeneous datasets can be accessed without having to know how the information is internally stored in the database. For example, the query does not need to know that an XML document is internally stored in multiple tables. New XML document types can be added without having to alter the queries in use.

Applicability

This paper addresses the rationale behind using an XML Database for storing heterogeneous helicopter signal data. The data is heterogeneous because a helicopter may have multiple data sources, such as Flight Data Recorders and Spectrapots, and also because data of multiple helicopter types must be stored and analyzed. Due to the generic set-up, the results are applicable to storing and analyzing the flight administrative data and measured flight data for the Netherlands Armed Forces used helicopter types Chinook, Apache, Cougar and NH90.

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2007-536

Storing heterogeneous helicopter signal data: Advantages of using an XML Database

B.C. Schultheiss, A.M. Vollebregt and C. Hummelink[1]

[1] Defence Materiel Organisation

This report is based on a presentation held at the IEEE AUTOTESTCON 2007 conference, Baltimore, U.S.A., 17-20 September 2007.
The contents of this report may be cited on condition that full credit is given to NLR and the authors.
This publication has been refereed by the Advisory Committee AEROSPACE VEHICLES.

Customer	NLR
Contract number	----
Owner	NLR
Division	Aerospace Vehicles
Distribution	Unlimited
Classification of title	Unclassified
	January 2010

Approved by:

Author	Reviewer	Managing department
[illegible]	EB 1/3'10	[illegible] 07-03 2010

Summary

The Netherlands Armed Forces operate several types of helicopters, such as Chinook, Apache, Cougar, Lynx, and in the future NH90 helicopters. To gain insight on the impact of helicopter missions on the life of the helicopter’s airframe, helicopter’s engine, and other vehicle systems, large amounts of measured flight data, and flight administrative data of the helicopters have to be collected, stored and analyzed. This paper presents the advantages of using an XML database for storing and querying the collected heterogeneous helicopter signal data.

Contents

1 Introduction

The Netherlands Armed Forces operate several types of helicopters: Chinook (Figure 1), Apache, Cougar, Lynx and in the future NH90 (which will replace the Lynx).

Figure 1- Boeing CH47-D Chinook transport helicopter

The Netherlands Armed Forces need insight in how the helicopters have performed and what the impact is on the life of the helicopter, and in particular the airframe, the engine and the other vehicle systems. This insight provides valuable information for sustaining the fleet and helicopter readiness for deployments. To get this insight, large amounts of measured flight and flight administrative data of the helicopters have to be collected, stored and analyzed.

The Netherlands Armed Forces awarded NLR a contract for the development of a Helicopter Life and Usage Monitoring system (HELIUM) to support gaining this insight. This paper focuses on the data storage environment within HELIUM. First, the context will be explained: helicopter usage monitoring. Next, the advantages of using XML documents and using an XML database will be shown. After that, querying the stored XML data is explained. Conclusions and references can be found in the last two sections.

2 Helicopter usage monitoring

NLR derives data using several analysis tools from the measured flight data and the flight administrative data, and uses this derived data for reporting the status (for example damage information) of the helicopter fleet to the Netherlands Armed Forces. The purpose of these status reports is not only to support the aim for increased operational availability of the helicopters, but also Performance Based Logistics.
The combination of using the Helicopter Life and Usage Monitoring system (HELIUM) and the expected use of more advanced data recorders in the near future will provide the Netherlands Armed Forces with increased insight in the status of their helicopter fleet and thus can optimize the operational availability of the helicopters.
Performance Based Logistics is a logistics model that puts in place specific goals and metrics to evaluate success. When an operator employs this model and outsources the maintenance and part supply to commercial parties it may happen that the operator looses its insight into its own key performance figures, and thus looses its grip on the flight safety, usage and availability of aircraft. This makes it necessary for the operator to have control and assessment tools to monitor the numbers involved in Performance Based Logistics. The HELIUM system provides these figures. Examples of information directly generated from the data storage environment include periodic reporting such as: “the damage accumulated during the last month” and ad hoc queries such as: “How many Chinook helicopters have flown above a certain weight limit?”.

The remainder of this section will explain the measured flight data, the need for centrally storing this data, and the process of applying analysis tools to obtain the required derived data.

With respect to sustaining its helicopter fleet, the Netherlands Armed Forces have several sources of measured flight data and flight administrative data. The measured flight data is collected using various Flight Data Recorders (FDRs), such as L-3 voice and data recorders, spectrapots, and ACRA boxes. Also data from Health and Usage Monitoring Systems (HUMS) will be used.
The size of raw data captured during one hour of a single flight of for example a Chinook helicopter equipped with a CVFDR (Combined Voice and Flight Data Recorder) in combination with the MDAU (Modular Data Acquisition Unit), a SPECTRAPOT-4C (a four channel Structural Data Recorder) and an ACRA-box is in the magnitude order of tens of Megabytes. Data must be stored for numerous flights of several types of helicopters - Chinooks, Apaches, Cougars, NH90s - during years, thus requiring a database capable of storing a large amount of data.

The input data for the data storage environment will change over the years due to new types of data recorders, additional signals, and even new types of helicopters. The data storage environment will also change due to improvements of the database: for example by adding additional constraints on data. So flexibility is an important requirement.

NLR applies several analysis tools to the measured flight data, such as calculation of damage indices for the airframe and engine, derivation of flight regimes, and re-creation of flights. NLR uses this derived data for reporting the status (for example damage information) of the helicopter fleet to the Netherlands Armed Forces. This process is shown in *Figure 2*.

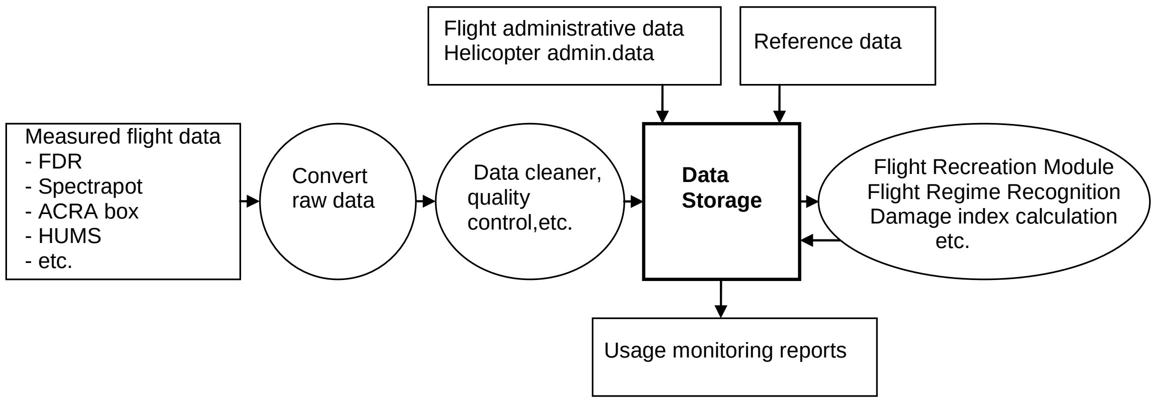


Figure 2 - Measured flight data and flight administrative data for helicopter usage monitoring

To be able to report on for example specific flights, on specific helicopters, or on specific periods, the measured flight data must be retrieved from the helicopters and be stored in a central data storage environment. Reports generated from the measured flight data and flight administrative data contain information that is generated by querying the data storage environment, and contain the already mentioned derived data.
However, before the data is ready for report generation processing, the yet raw data must be converted to a format that is accepted by the data storage environment, and the quality of the data must be checked and cleaned (check for spikes, flat liners, or remove or mark blocks of anomalous data).

3 Storing measured flight data and flight administrative data in an XML Database

The previous chapter showed that Data Cleaner tools, Data Quality Control tools, and Analysis tools must be able to access the stored data. Section 3.1 will analyse the advantages of using the Extensible Markup Language (XML) for the interfaces between these tools and the data storage.

The previous chapter also explained the need for a flexible data storage capable of storing data from new and yet unknown types of data recorders, additional signals, and even new types of helicopters. Section 3.2 discusses database technology capable of providing this flexible storage. When using XML Database technology to implement this flexible storage, XML Schema Revolution is an important subject, see section 3.3.

3.1 Using XML documents

Using XML as input for, and as output from the data storage environment has a number of advantages:

- The XML format can be specified unambiguously using an XML-schema. So, input for the data storage environment can be checked before submitting it to the data storage environment. Components that provide data to the database can be developed independently because the interface is defined using the XML schema.
- XML is readable text which is an advantage in an engineering environment using all kinds of analysis tools. Using a browser such as Internet Explorer will show the XML nicely formatted.
- XML can be easily translated into Excel, (X)HTML, PDF, MSWord, and other formats using the Extensible Stylesheet Language Transformations (XSLT). XSLT is an XML-based language used for the transformation of XML documents.
- Parsing and using XML is well supported by many platforms, including the Java programming language.

For each of the helicopter types, the measured flight signals are stored as XML data. The format is similar to the format shown in Figure 3.
If for the same time steps a number of signals are measured simultaneously, then these values are combined in a single line specification (*<line>...</line>*).

XML Schemas specify the syntax including restrictions on the data of the XML documents. The XML schemas used in HELIUM for the different types of helicopters are similar, but not the same. The similarity is that each schema will define a DataSet containing lines line; that each line contains the signal values for a certain time step, and that each signal value is optional thus allowing for maximum flexibility. However, the actually measured signals and their node names are different for the different types of helicopters.

```
<?xml version="1.0" encoding="UTF-8" ?>
<chinookmeasureddata>
        <DataSet>
                <line>
                        <time>6711.11</time>
                        <magneticHeading>225.791</magneticHeading>
                        <calibratedAirspeed>69.1875</calibratedAirspeed>
                        <yawad>0.0</yawad>
                        ...
                </line>
                <line>
                        <time>6711.23</time>
                        ...
                </line>
                ...
        </DataSet>
</chinookmeasureddata>
```

Figure 3 - A Chinook example: combine signals that are measured at the same time into one line

The examples in this paper use long XML node names such as “magneticHeading”. For the storage of the XML documents, using long names does not affect the database size and it improves the readability. However, for retrieving XML documents from the database, uploading XML documents to the database, the pre-processing steps (data cleaning and quality control), and the post-processing steps (e.g. damage index calculation), smaller XML node names will result in smaller files that are easier to handle. For example use “t” rather than “time”, and “mhdg” rather than “magneticHeading”.

Input for the XML data storage environment is for example “IRIG-106 chapter 10” data, see [2]. Output of the XML data storage environment will be used by other HELIUM tools, such as analysis tools, report generators, and a flight recreation module (i.e. flight visualization software).
So, interoperability is an important issue. XML standards for storing time series data with metadata are being developed in for example the Statistical Data and Metadata Exchange

standard [3], and the Automatic Test Markup Language [4] initiatives. However, we want to exploit the knowledge about signal data parameters to be stored to optimize the storage. Also, our XML schemas may define restrictions on signal values for specific signals. Interoperability issues can be solved by including a translator that translates the internally used XML data format from and to the requested interoperable format.

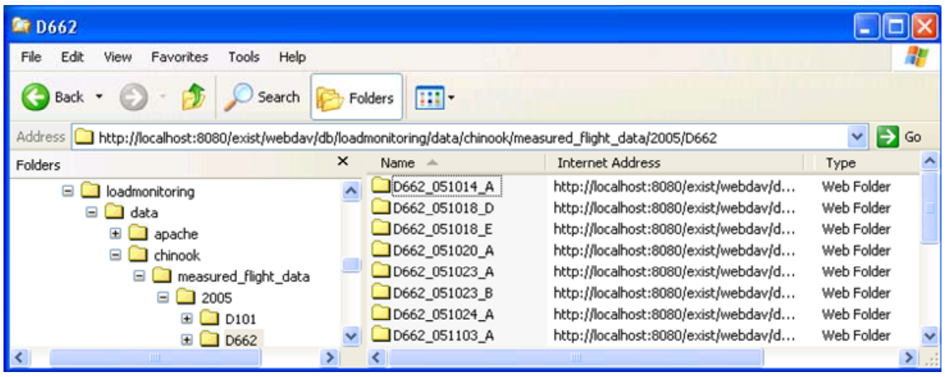
3.2 XML database or relational database

In previous (still operational) NLR projects, Oracle relational databases have been used to store F-16 [1] and C-130 Hercules measured flight data. For the Hercules database, Oracle specific technology has been applied: VARRAYs, a data type that allows for efficient storage of a variable number of data elements. Typically, the measured flight data may not always contain the same amount of parameters, and using VARRAYs is therefore an efficient solution. A disadvantage of using VARRAYs, but also of using a relational database in general is that the queries depend on the relational table structure; on how XML documents are mapped on the tables.

Since we want to support data storage for multiple helicopter weapon systems, and we want to be prepared for new measured signals or even additional helicopter types, separation of the logical model (the XML documents) from the physical model is important. XML databases together with the XQuery[6] language provide this advantage.

Several commercial and open source XML Databases are available.
eXist [9] is an Open Source XML Database featuring index-based XQuery processing, automatic indexing, extensions for full-text search, XUpdate support, XQuery update extensions, and support for a RESTful [14] interface. eXist was surprisingly easy to install and to use, and very usable for gaining first experiences. We have not investigated the application of eXist with a really large dataset: eXist was only used with a limited dataset of large (up to 100MB) XML documents.
Oracle XML DB [7][8] provides capabilities for storage and access to XML data and extends the well-known Oracle database. Our experience with the Oracle 10g XML DB is that it responds less friendly to small mistakes. However, first tests with Oracle 11g XML DB are promising. Also, Oracle supports versioning of XML documents. In the NLR environment with preprocessing tools that provide input XML data, and analysis tools that use data from the XML Database to calculate derived data, the version of the data as well as the version of the software that has been used to calculate derived data is important. Therefore, XML document version support may proof to be useful.
Also IBM’s DB2 VIPER product supports XML database with XQueries [10].

Another advantage of using an XML Database instead of a relational database is that storing and retrieving the original XML documents without any loss of fidelity can be guaranteed. Moreover, XML databases such as Oracle and eXist provide a WebDAV folder hierarchy view on the stored XML documents. WebDAV - Web-based Distributed Authoring and Versioning - is an extension of the HTTP protocol and allows for adding and retrieving documents from a remote server. Although the XML Databases will store the XML documents in an internal format, XML documents as submitted to the database can be retrieved without any loss of fidelity. Figure 4 shows that this folder/file hierarchy can be viewed via Microsoft Explorer. As a result, XML documents can be added or retrieved using simple drag and drop operations.



Address: http://localhost:8080/exist/webdav/db/loadmonitoring/data/chinook/measured_flight_data/2005/D662

```
loadmonitoring
  data
    apache
    chinook
      measured_flight_data
        2005
          D101
          D662
```

Name	Internet Address	Type
D662_051014_A	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051018_D	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051018_E	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051020_A	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051023_A	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051023_B	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051024_A	http://localhost:8080/exist/webdav/d...	Web Folder
D662_051103_A	http://localhost:8080/exist/webdav/d...	Web Folder

Figure 4 - An XML Database with WebDAV support can be viewed in a folder/file hierarchy using Microsoft Explorer

Since preprocessing tools providing the input XML data and the analysis tools providing the derived XML data may change in time, the traceability from a usage monitoring report to its input data and applied analysis tools and preprocessing tools (including their version number) is important. An XML database such as Oracle provides support for versioning of XML documents.

3.3 XML Schema revolution and database creation

Oracle supports both structured and unstructured storage [8]. A key decision to make when using Oracle XML DB for persisting XML documents is the choice when to use structured storage and when to use unstructured storage.
Unstructured storage provides high throughput when inserting and retrieving entire XML documents. It also provides the greatest degree of flexibility in terms of the structure of the XML that can be stored. However, there is little the (Oracle) database can do to optimize queries and updates.

When using structured storage, the XML Schemas are used to generate the underlying database structure. Since we need to query the database on for example maximum, minimum, average values of signals, we need to use Oracle's structured storage. Then, creating the database consists of using Oracle's "dbms_xmlschema.registerSchema"; see [8] for more details.

Using structured storage implies that we need to take care of changing XML Schemas. XML Schemas will change in time due to improved typing of, and constraints on elements (for example to improve the quality of the stored data, constraints on allowed values might be added later on), missing subcomponents (for example a new measurement tool is added providing additional signals) and additional helicopter system types with new signals. Oracle supports XML Schema revolution.

For eXist [9], the separation between structured and unstructured storage is not made. Structural indexes that keep track of the elements (tags), attributes, and nodal structure for all XML documents are created and maintained automatically. Range indexes are indexes based on the data type of specific node values in the document. These indexes provide a shortcut for the database to directly select nodes based on these type values. Unlike structural indexes, range indexes can be created and configured directly by the user, and in this sense, they are similar to indexes used by relational databases. When starting the eXist database, a database instantiation is readily available; creating a database for a specific user is not available.

4 XPATH and XQuery access to the XML Database

The previous chapter analysed the advantages of using an XML Database to store the measured flight data and the flight administrative data. The query language to access an XML Database is XQuery[6]. This section will show that when using XQueries on an XML Database, a flexible infrastructure is provided in which queries or additional helicopter systems can be easily added.

The XQuery examples in this section show that although the measured flight data XML Schemas are different for the different types of helicopters, obtaining the “DataSet” is exactly the same. This is achieved by using an XPath expression to address the “DataSet” portion of the XML document only and skipping the surrounding XML code. For an example, see the usage of the “//” XPath operator in Figure 5.
Figure 5 also shows that the XML Database contents can be seen as a hierarchy of folders with XML documents. The XQuery in Figure 5 searches in XML documents that are stored in folder “/loadmonitoring/data/chinook/measured_flight_data/2005/D101/D101_050406_C”:

```
/loadmonitoring
        /data
                /chinook
                        /measured_flight_data
                                /year=2005
                                        /tail=D101
                                                /flightId=D101_050406_C.
```

XQuery:

```
for $dataset in collection(
        "/loadmonitoring/data/chinook/measured_flight_data/2005/D101/D101_050406_C") // DataSet
return $dataset
```

Answer:

```
<DataSet>
      <line>
            <time>14252.1</time>
            <magneticHeading>315.879</magneticHeading>
            <cruiseGuide>194.296</cruiseGuide>
            <groundSpeed>130.25</groundSpeed>
```

```
                <latAcceleration>-0.0080974</latAcceleration>
                <longAcceleration>-0.090067</longAcceleration>
                <weightOnWheels1>1.0/weightOnWheels1>
                ...
        </line>
        <line>
                <time>14252.2</time>
                ...
        </line>
        ...
</DataSet>
```

Figure 5 - XQuery and answer for obtaining a dataset for a specific flight. "//DataSet" ensures that independent of XML schemas, the DataSet will be returned.

Input for, for example, a flight recreation module, or input for a damage index calculation requires in general only the time stamp with a specific signal value. Figure 6 shows the XQuery for requesting a *DataSet* that will only contain the time step and a specified signal.

XQuery:

```
<DataSet>{
        for $line in collection(
                "/loadmonitoring/data/chinook/measured_flight_data/2005/D101/D101_050406_C")//DataSet/line
        let $t:=$line/time
        let $signal:=$line/rotorSpeed
        where $t and $signal
        return <line>{$t}{$signal}</line>
}</DataSet>
```

Answer:

```
<DataSet>
        <line>
                <time>27246.6</time>
                <rotorSpeed>100.406</rotorSpeed>
        </line>
        <line>
                <time>27247.1</time>
                <rotorSpeed>100.469</rotorSpeed>
        </line>
        ...
</DataSet>
```

Figure 6 - XQuery for obtaining one DataSet containing only time time *and signal* rotorSpeed

Our experience is that requesting large XML documents may take a lot of time due to its memory consumption, e.g. during parsing. An approach to avoid large XML files when obtaining signal data is shown in Figure 7. Instead of requesting one large "DataSet" document, a large set of single "line" XML documents is returned.

XQuery:

```
for $line in collection(
        "/loadmonitoring/data/apache/measured_flight_data/2005/PQ16/PQ16_041115_B")//DataSet/line
let $t:=$line/time
let $signal:=$line/totalTrueAirspeed
where $t and $signal
return <line>{$t}{$signal}</line>
```

Answer: 1339 items

```
1    <line>
         <time>69758.7</time>
         <totalTrueAirspeed>0.0</totalTrueAirspeed>
     </line>
2    <line>
         <time>69767.7</time>
         <totalTrueAirspeed>3.92</totalTrueAirspeed>
     </line>
3    ...
```

Figure 7 - XQuery for obtaining multiple lines line answers instead of one large DataSet

In the above examples, we have not shown the usage of namespaces. An XML namespace provides uniquely named elements and attributes in an XML instance. XML namespaces have also been adopted by XQueries. When using Oracle XML DB in combination with multiple XML Schemas, namespaces must be used. This complicates the reuse of the same XQueries for different helicopter types. However, in our environment we are not using queries that span for example Apache and Chinook XML data at the same time. In HELIUM, first a helicopter type is selected. Knowing the helicopter type, the XQuery can be easily prefixed with a namespace declaration, and the name space prefixes can be inserted in an XQuery. An example of such an XQuery is shown in Figure 8.

```
declare namespace ns= "http://www.nlr.nl/apachemeasureddata.xsd" (: :)
for $dataset in collection(
            "/loadmonitoring/data/apache/measured_flight_data/2005/PQ16/PQ16_041115_B")//ns:DataSet
where $dataset/ns:totalTrueAirspeed
return $dataset
```

Figure 8 - XQuery with namespace declaration

An example of a more complicated XQuery is searching for the names of the flight data signals that have been stored in the measured flight data. Another more complicated example is an XQuery that returns DataSets depending on for example conditions on signals such as on the maximum weight, the external load, or climb speed. And a third example of a more complicated XQuery is an XQuery that derives for each signal of each flight its characteristics such as maxima and minima, and that stores this derived data in derived data XML documents. The goal of storing this derived data instead of recalculating it each time it is required is to be able to provide quick answers when working interactively with the database. Also versioning of XML documents may further complicate XQueries.

The conclusion of this section is that using XQueries (that include XPath expressions) provides a flexible infrastructure in which queries on new signals (for example due to new flight data recorders), or additional helicopter systems can be easily added, and in which existing queries can be re-used without any change.

5 Conclusions

In general, using XML documents as input for the data storage and output from the data storage provides many advantages such as easy converting to and from XML documents, well defined syntax using XML Schemas, excellent support in e.g. Java, and a human readable format, see section 3.1.

Using an XML database for storing the measured flight and administrative XML documents provides a data storage in which the logical structure – the XML documents – is separated from the physical storage, see section 3.2. Chapter 4 showed that using XQueries and XPath expressions, heterogeneous datasets can be accessed without having to know how the information is internally stored in the database. For example, the XQuery does not need to know that an XML document is internally stored in multiple tables. New XML Schemas can be added without having to alter the XQueries in use.

Care must be taken to be able handle large XML documents. When designing the XML Schemas, using short node names may be important to reduce the size of XML documents during pre-processing and the post-processing steps. When retrieving results from a database, our experience is that using XQueries that result in a large set of small XML document results have better performance than using XQueries that result in a single large XML document.

Section 3.2 also showed that the XML database WebDAV interface provides a useful folder/file view on the database, allowing for easy adding and retrieving XML documents using drag and drop as today’s computer user is familiar with.

Concluding, storing the measured flight data and the flight administrative data of the several types of helicopter weapon systems in an XML Database provides a flexible database infrastructure for storing and accessing these heterogeneous data, and supports adding new types of data recorders, additional signals, and even new types of helicopters.

References

[1] F.C. te Winkel and D.J. Spiekhout, RNLAF/F-16 Loads and usage monitoring/management program, NLR report NLR-TP-2002-309, can be obtained via http://www.nlr.nl

[2] J.M. Klijn, Development of flight test instrumentation: an evolutionary approach, NLR report NLR-TP-2006-407, can be obtained via http://www.nlr.nl

[3] Statistical Data And Metadata Exchange Initiative SDMX, SDMX standards version 2.0, http://www.sdmx.org/

[4] Automatic Test Markup Language (ATML), http://grouper.ieee.org/groups/scc20/tii/

[5] H.H. Ottens, and R.J.H., Wanhill, Review of aeronautical fatigue investigations in the Netherlands during the period March 2001 - March 2003, prepared for The 28th ICAF Conference in Lucerne, Switzerland, 5-9 May 2003, NLR-TP-2003-251, can be obtained via http://www.nlr.nl

[6] XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/

[7] Oracle® XML DB, http://www.oracle.com/ technology/tech/xml/xmldb/index.html

[8] Oracle® XML DB Developer's Guide 10g Release 2 (10.2) Part Number B14259-02; can be found using http://www.oracle.com/ pls/db102/portal.all_books#index-XML

[9] eXist, an Open Source native XML Database, http://exist.sourceforge.net/

[10] K Beyer et al, DB2 goes hybrid: Integrating native XML and XQuery with relational data and SQL, IBM Systems Journal, Volume 45, Number 2, 2006, http://www.research.ibm. com/journal/sj/452/beyer.html

[11] A. Balmin et al, On the Path to Efficient XML Queries, 2006, http://www.vldb.org/conf/2006/ p1117-balmin.pdf

[12] Eric Sadler, Managing Structure in Bits & Pieces:The Killer Use Case for XML, 2005, http://delivery.acm.org/10.1145/1070000/1066256/p818sedlar.pdf

[13] Application programming interface for XML Databases, http://xmldb-org.sourceforge.net/ xapi (eXist supports an XML:DB interface)

[14] R.T. Fielding, Chapter 5: Representational State Transfer (REST), http://www.ics.uci.edu/ ~fielding/pubs/dissertation/rest_arch_style.htm (eXist supports RESTful interfaces)