DOI 10.5162/etc2014/3.1

History and Evolution of Metadata Standards for the FTI Community

Alan Cooke[1]
[1] Principal Software Architect, Curtiss-Wright, Dublin, Ireland,
acooke@curtisswright.com

Abstract

The paper discusses the history and background of metadata standards for the FTI community over the last 20 years and speculates on how they may develop in the future. It starts by highlighting the deficiencies of proprietary formats and the resulting problems. It then discusses the characteristics and features of specific industry standard metadata descriptions such as TMATS, iHAL, MDL and XidML in addition to their levels of maturity. The attributes of what constitutes a fully mature FTI metadata standard is then discussed. It is suggested that any standard must serve at least two functions, Configuration and Validation, and outlines what exactly each means. Finally, it is argued that there is now a significant level of convergence and consensus in both the scope and application of metadata, and in the associated concept of operations (ConOps). The details of this Concept of Operations are then discussed along with suggestions as to how this may evolve in the coming years.

Key words: XML, TMATS, iHAL, MDL, iNET, XidML, XdefML, Metadata, Meta-metadata, Metadata validation, Fight Test Instrumentation, Concept of Operations (ConOps).

Introduction

The history of metadata in the Flight Test Instrumentation (FTI) community has a long history and can trace its origins right back to the beginnings of flight test. This paper is an attempt to illuminate this history and to describe its current state. The paper will further describe ongoing efforts to produce new metadata standards.

The discussion begins by defining exactly what is meant by metadata. Proprietary metadata formats will then be discussed. Specifically, the paper will describe what formats proprietary metadata typically take and what the strengths and disadvantages to these formats are.

The paper will also suggest what the desired attributes of a mature metadata standard should be and will discuss briefly a *Concept of Operations*.

What is Metadata?

In brief, metadata is data describing other data. There are two classes of metadata, *structural* metadata, which is information pertaining to the structural aspects of the data and *descriptive* metadata, which describes the actual data.

Typically, it is the descriptive metadata that is of most concern to FTI users as this is the data that is actually used in the definition and configuration of flight test equipment. On the other hand, structural metadata is typically of most use to developers of tools that process files containing the descriptive metadata.

Metadata descriptions generally take three forms. The first is to hardcode the structural rules for processing the metadata in code, the second way to is to describe the metadata format using a formal grammar, while the third way is to use an XML schema.

The first method is not ideal as the rules for processing the metadata are typically difficult for those who are not software engineers to determine. Typically, there is also an extra overhead in maintaining this type of software compared to other methods as metadata processing code is typically constructed using one-off components. Furthermore, it can also be very hard to extend and update the software if it is not designed with extensibility in mind.

The second method is to define the structure of the metadata in a formal grammar using a notation such as Backus-Naur Form (BCF). In brief, a grammar is a mechanism that is used to describe the allowable “strings”[1] and the sequencing of the “strings” that an instance document can contain. Typically, software

[1] A string is essentially a collection of alpha numeric characters

developers would use libraries such as YACC or ANTRL to generate what are called “parsers” that in turn can be used to process and extract information from instance documents. This method has the benefit of allowing the authors of metadata standards to rigorously define the structure and allowable content. It can also result in software that can process instance documents extremely efficiently. However, using specialized grammars can be complex, especially for non-specialist. Obtaining descriptions for grammars from individual vendors, if they exist at all, can also be troublesome.

The third method is to use an XML schema to define a metadata standard. XML, or the extensible markup language, is a mechanism for encoding documents that is similar in structure to an HTML file and consists of opening and closing *elements* in addition to *attributes*. It has been described as the “Lingua Franca” of data exchange and has been used to describe the format of many hundreds of documents’ formats[2]. The structure and allowed content of an XML file can be defined by an XML schema[3]. The main benefits of using an XML schema to define a metadata standard is that it is ubiquitous, with many hundreds of off-the-shelf (and often free to use) tools and libraries for both processing and validating XML files against their associated schemas. XML is also very well understood with a large user base and the widespread availability of expertise.

Proprietary Formats

Most of the early proprietary metadata file formats were typically ASCII based, and varied widely in their structure and in the type of data that they contained.

The most basic of these is the CSV (Comma Separated Variable) format. In this format the metadata usually consists of lines of data separated by commas, white space or tab characters. The advantage of this format is that it is relatively easy to parse but suffers from the drawback of being vendor-specific in nature and is not self-describing and is therefore not very human readable.

Another common format that was used is the INI file format. This format essentially consists of labelled “sections” that are marked up using the “[“ and “]” characters. These sections in turn contain collections of “properties”. These properties were essentially name-value pairs marked up using the “=” character. This format has the advantage of being reasonably self-describing, and therefore human readable, but is not “structurally rich” and therefore cannot be used to describe complex data constructs (e.g. an IRIG-106 Chapter 4 Frame definition) in a natural way.

For more complex applications, such as when the configuration of an entire FTI network needed to be described, vendors typically developed their own custom file formats. Sometimes they would be described using a formal grammar, but more often than not the rules of their “language” would be either opaque to the user, or consist of a set of informal rules that were known only to the software developers that created and maintained the proprietary format.

The biggest disadvantage in having to work with vendor formats, however, is that users have to relearn a new “language” every time they purchase hardware from a new supplier. This overhead is compounded even further when they have to integrate FTI equipment from more than one vendor.

Attributes of a Mature Metadata Standard

It is suggested that, at a minimum, a mature metadata standard for the FTI community should meet the following criteria:

- **Vendor neutral**: The standard should be designed is such a way that it is capable of describing equipment from any vendor.
- **Easily processed**: The metadata format should be easily processed, preferably using freely available off-the-shelf tools.
- **Models the FTI domain**: The data model behind the metadata standard should effectively model the FTI domain. This is to ensure that the standard is both easy to understand by the user community and is capable of being used for the full range of future needs.
- **Mature and widely used**: While this is difficult to achieve in a new standard, a metadata format that has evolved and been adapted over a long period of time to meet unanticipated user needs is almost certainly better suited than any putative new standards.
- **Flexible**: Not all FTI applications are the same. Some involve large aircraft and some small; some are completely Ethernet based while others rely on older protocols

[2] XML Applications and Inititives http://xml.coverpages.org/xmlApplications.html.

[3] W3C.org http://www.w3.org/standards/xml/schema

  and so on. Any mature standard needs to be able to cope with this diversity.

- **Extensible**: The requirements of the FTI community are not static and will change over time as new technologies and protocols emerge and the number of measurements increases. Any mature metadata standard should be designed to allow it to grow and adapt over time.
- **Self-describing**: Ideally, a metadata standard should be easy to interpret and unambiguous. This applies both to the mechanism used to markup the data and the data model used as the basis of the standard.
- **Concept of Operations**: Any metadata standard should also be able support an FTI engineers typical "Concept of Operations" or ConOps. This ConOps generally involves metadata validation, device discovery, device configuration and requests for descriptions of device capabilities.

Early Standard Formats

TMATS

Dating back to 1989, this is perhaps the oldest attempt at a metadata standard for the FTI community. It is an ASCII-based format consisting of twelve sections, each of which defines a different aspect of the acquisition system that is being configured. These sections include a General Information, PCM Format Attributes, PCM Measurement Attributes, Recorder and Reproducer Attributes and Bus Data Attributes.

The main strengths of the TMATS standard are that it is a reasonably well accepted standard and it was designed predominantly by those involved in FTI for FTI engineers. Given its origins it is also unsurprisingly very strong in describing the IRIG-106 Chapter 4 PCM standard and Recorders.

Its weaknesses arguably include that it is too PCM focused and does not handle modern Ethernet based protocols and infrastructure very well. Furthermore, it only handles two bus protocols (MIL-STD-1553 and ARINC-429) explicitly. The standard also lacks a way of describing and *validating* vendor-specific instrument data in a general way.

Additionally, although several tools exist for reading and validating TMATS files, in comparison to XML based standards, there are very few of the shelf tools and libraries. Furthermore, no formal grammar exists for TMATS which sometimes results in tools and APIs disagreeing on whether or not given instance files are actually valid

TMATS XML

In an attempt to address the disadvantages of using a custom ASCII format the TMATS committee developed an XML version of the standard. This has meant that TMATS users no longer have to rely on vendor supplied or other specialist software to process and validate a TMATS XML file. However, it is still not possible to validate vendor-specific information in the file.

Modern Open Metadata Formats

XidML

XidML is an XML based metadata standard that was first proposed in 2004 [1]. Since XidML 2.0 was released, it has gone though numerous iterations and the latest version is now XidML 3.0[4]. From the beginning the schema was designed to be vendor neutral, and was based on a generic data model of the FTI domain. It was also designed to be extensible and future proofed.

The standard took a fundamental shift in the way it described instrumentation setup between versions 2.41 and 3.0. It moved from a schema per class of instrument (40+ types) approach to a single highly generic instrument schema.

The shift in methodology came as it became clear that the standard would always be chasing the technology. The continuous changes to the schema and maintenance overhead resulting from these changes were not sustainable. The solution to this problem was to the let the technology shape the standard.

The generic instrument schema introduced in version 3.0 is extremely flexible and extensible, and gives vendors the power to model their hardware whichever way they want. However, this highly generic approach has its drawbacks, it lacks rigor in setting names (structural Metadata) and validation information for the descriptive Metadata. The XIdML dictionary and the XdefML schema were introduced to solve these problems respectively.

Today XidML consists of two parts; the XidML schema itself and the optional XdefML schema. A XidML instance file contains the data used to actually configure the data acquisition network and is built around the five key concepts of Instruments, Parameters, Packages, Links and

[4] The schema can be found at www.XidML.org

Algorithms. An XdefML instance file contains data that is used by software to validate the user specified data contained within a XidML file. An XdefML file is provided for each type of instrument defined in a XidML file.

XidML strengths include its simplicity (e.g. the *Instrument* element can be used to describe any device from any vendor), the ability to describe data constraints for any vendor device (using XdefML) and has been used to configure hundreds of devices from scores of manufacturers [2]. XidMLs main drawback is that even though it is an open standard it's perceived as not being open due to its origins with a particular FTI vendor.

Future Metadata Efforts

MDL

The Measurement Definition Language (MDL) [3] is one the most recent initiatives to formulate a metadata standard for the FTI community and has now been incorporated into the iNET initiative. It is an XML based standard with an associated schema consisting of six top-level constructs. Its underlying approach is fundamentally measurement-centric and aims to side-step vendor issues by avoiding the use of vendor-specific information completely in the MDL file. However, another of its main goals is to fully describe network based data acquisition systems.

In the MDL philosophy, users specify the required characteristics of measurements, such as accuracy, uncertainty and so on, and pass this information on to vendor software for processing. The vendor software then returns a new MDL file which indicates what is actually achievable by the vendor hardware.

The MDL schema also features very comprehensive and structurally rich mechanisms for describing networks and networks of networks, in addition to the flow traffic between these networks. MDL also has specific provision for Quality of Service (QoS) using Differentiated Services (DiffServ).

MDLs strongest features are undoubtedly its comprehensive description and approach to measurements. MDL facilitates the detailed description of the digital filter characteristics to be applied to measurements via its *AnalogAttributes* section.

Since MDL is network focused, one of its weaknesses it that older, well-established technologies are not natively supported by the standard (e.g. PCM or CAIS etc.). The standard does acknowledge that these technologies may be required to interact with a network based system and recommends creating a hardware proxy to convert data from these technologies into network data and then model that hardware proxy in MDL. Perhaps its biggest weakness lies in the complexity of the schema. This is particularly so in the *NetworkNodes* section of the schema that contains over 20 *ManagebleAPPS*, each dedicated to a specific function. The use of XML schema "ID" construct to uniquely identify key entities can sometimes make relationships opaque to the user. There is also very little support for the configuration and validation of vendor hardware, although it has improved as the schema has evolved from the original version.

iHAL

iHAL (Instrumentation Hardware Abstraction Language) [3] is the most recent attempt to develop an open standard for the FTI community. In its original form, iHAL was exclusively focused on the configuration of vendor hardware. It also provided mechanisms for validating vendor-specific information on top of the basic schema level validation.

Specifically, iHAL instance documents are broken up into two sections. The iHAL "Use" section contains the actual data used to configure vendor hardware, while the iHAL "Pool" section is used to define vendor-specific constraints for the instrumentation in an acquisition system. Software can use the constraints contained in the "Pool" section to validate user data contained within an iHAL instance file.

In later iterations of the schema support for measurements and measurement units, IRIG-106 Chapter 4 PCM, and network based systems was added by including parts of the MDL, TMATS XML, and XidML schemas respectively.

Another interesting aspect of the iHAL project is the iHAL API. This is the only standard with an associated API. The API, which is a RESTful API, has the following functionality

- Users can validate iHAL files using the API
- Users can pass the API an iHAL file to program hardware
- The API can return an iHAL file describing all hardware, and how they are configured
- The API can return an iHAL file describing the capabilities (i.e. using the "Pool" section of the iHAL file) of vendor hardware.

This API therefore covers all the basic ConOps scenarios described above.

Arguably the best aspect of the iHAL standard is the mechanism used to separate configuration data from the data used to describe vendor-specific hardware constraints. The decision to follow the RESTful paradigm with the API also appears to be a good choice for a number of reasons including its inherent scalability and its ubiquitous use in APIs on the internet. The approach also allows vendors the choice of either implementing the API on hardware itself or by using a software proxy.

Undoubtedly iHAL's biggest weakness is its immaturity, especially relative to TMATS, TMATS XM and XidML. As with MDL, the explicit inclusion of other schemas can make it difficult for users to understand and process. Additionally, the relationship between the various constructs in each of the constituent schemas is immature and needs to be developed further.

Conclusion

This paper outlined a brief history of FTI metadata standards. It started with a discussion on what metadata is and outlined some of the common formats used in these standards. The paper then discussed FTI metadata history starting with TMATS and then describing the main metadata initiatives that followed on from this standard. The paper then suggested some of the characteristics that a mature and functional metadata should have. Table 1 compares each metadata standard against these characteristics.

The industry needs a universal metadata standard and most recent initiative, MDL, has the potential and the momentum behind it to become that standard, but it must learn from mistakes made and problems solved by all of its predecessors in order to succeed.

	TMATS XML	XidML	MDL	iHAL
Vendor Neutral	✓	✓	✓	✓
Easily processed	✓	✓	✓	✓[5]
Models the FTI domain	×	✓	✓[6]	×
Widely used	✓	✓	×	×
Flexible	×	✓	✓	✓
Extensible	×	✓	✓	✓
Self-describing	✓	✓	✓	✓
ConOps Modelling	×	✓	×	×

Table 1 Standards Maturity Comparison

[5] XML namespaces are heavily used, can make parsing complex

[6] Only supports network based models

References

[1] "*XML: A Global Standard for the Flight Test Community*", Alan Cooke and Diarmuid Corry, *ETTC Proceeding 2004*

[2] "I*ntroduction to XidML 3.0 an Open XML Standard for Flight Test Instrumentation Description*", Alan Cooke and Christian Herbepin, *ITC 2010*

[3] "*A Metadata Language for Describing Telemetry Systems*", Michael S. Moore, Jeremy C. Price, Andrew R. Cormier, William A. Malatesta. *ETC 2009*

[4] "*IHAL and Web Service Interfaces to Vendor Configuration Engines*", John Hamilton, Timothy Darr, and Ronald Fernandes, Knowledge Based Systems, Inc. Joe Sulewski, L3 Communications - Telemetry East; and Charles Jones, Edwards AFB, *ITC 2010*