WHERE NEXT FOR XidML

Alan Cooke
ACRA CONTROL

ABSTRACT:
XidML is an open, vendor neutral, XML based standard for the FTI community used to capture the metadata associated with flight test instrumentation and data acquisition networks. This paper discusses the difference between metadata and meta-metadata, how these concepts apply to XidML and an optional schema, XdefML. The paper also describes how Settings and their validation may be applied to a Generic package definition. Some examples are given of how the current version XdefML can be leveraged to auto-generate graphical user interfaces that are both feature rich and incorporate sophisticated error checking and data validation.

KEYWORDS:
XML, XidML, XdefML, Metadata, Meta-metadata, Metadata validation, Flight Test Instrumentation

1 XidML - A REMINDER

XidML stands for the Extensible Instrumentation Metadata exchange Mark-up Language. It was first released in 2005. It is an XML schema [1] designed to meet the needs of the FTI community [2] [3] [4]. It is an open, vendor-neutral metadata standard whose scope is everything from "Sensor to Screen". Extensibility, flexibility and the ability to meet future needs of the FTI community were key design goals. Since its first release, it has gone through several refinements, each with the aim of both simplifying the schema and of increasing its expressive power and utility.

1.1 THE FIVE MAIN COMPONENTS IN XIDML

Conceptually, XidML is built upon a data model that consists of five key components, Instruments, Parameters, Packages, Links and Algorithms.

1.1.1 INSTRUMENTS

Instruments are the physical devices that make up an FTI system. Examples include data acquisition units, A/D modules, sensors, RF transmitters and data storage devices. The Instrument schema is used to define how each device in an FTI system is configured. This configuration information is specified using Settings. Instruments can also have optional inputs and outputs called channels. Examples of channels include an input to an A/D device, an egress port from a switch and the input to an IRIG-106-Chp4 PCM decoder. All instruments have a globally unique name.

1.1.2 PARAMETERS

A Parameter is used to describe the data that is being sampled or measured as part of a flight test program. In XidML, the Parameter schema describes all the properties that allow a flight test or ground-station engineer to transmit, visualise and analyse the data that is being sampled. These properties include, the number bits used to encode the sampled data, encoding format (e.g. offset binary), engineering unit etc. All parameters have a globally unique name.

1.1.3 PACKAGES

A Package is a description of how data is transmitted and how data is stored to a disk or some other medium. It includes any header or trailer information that is needed to specify a transmission or storage format and how the data (i.e. Parameters) are transmitted or stored in the Package. Examples include IRIG-106 Chapter 4 PCM frame definitions and Ethernet transmission formats (e.g. AFDX). All packages have a globally unique name.

1.1.4 LINKS

A Link is the entity that connects two physical devices together. A Link can be used for programming, providing power or transmitting data. It can exist physically (e.g. an Ethernet cable) or be wireless (e.g. an RF link) in nature. Links are also integral to discovering the topology of a data acquisition system. In XidML, users can use Link-wide Settings to specify configuration options that are common across both ends of a Link. All Links have a globally unique name.

1.1.5 ALGORITHMS

Algorithms can be used to describe how data is generated or processed. Examples include algorithms that run on CPUs and look-up tables used to linearize data. In XidML, Algorithms can have zero or more input parameters and zero or output Parameters. All algorithms have a globally unique name.

2 XdefML

XdefML was released in version 3.0 of XidML. This schema is optional and complementary to the XidML schema and allows vendors to describe both the physical and functional characteristics of their devices. It is a meta-metadata language in that it is *not* used to define or affect the behaviour of an FTI data acquisition system. Instead, it provides a mechanism that allows users of vendor equipment to *validate* data that is used to configure or affect the behaviour of an FTI system. The same data can also be used to auto-generate user interfaces for software that are XidML aware [5]. An XdefML file contains three main sections, *InstrumentIdentification*, *InstrumentSpecifications* and *InstrumentConfiguration*.

2.1 INSTRUMENT IDENTIFICATION

This section is used to uniquely identify a particular type of device such as the manufacturer name and a manufacturer part number.

2.2 INSTRUMENT SPECIFICATIONS

This section contains some general device specifications such as

- **Mechanical class**: This is a vendor specified string that essentially identifies the physical properties of a device.
- **Functional classes**: A functional class identifies a logical group to which a device belongs. Examples include Bus Monitor, Analog, Digital and so on. A device can belong to one or more functional classes.
- **Slots/Locations occupied**: This specifies how many slots or locations a device takes up in a parent/platform instrument.
- **Datasheet**: A URI of a datasheet for the device.

2.3 INSTRUMENT CONFIGURATION

This section is used to describe the general capabilities of an instrument and contains the following elements:

- **Settings**: Defines constraints, default values and the visibility of settings that are used in XidML to define how a device is configured. This element can also be included as a child of the channel element to specify the constraints for settings used on channels, and to define "Link Wide" settings that are common to the two sides of a link.
- **Parameters**: This section is used to describe what data (i.e. parameters) can be read from a device. The properties and constraints of each Parameter are also described. These properties include, the number of bits used to encode the data, the data format used (e.g. Offset Binary, Twos Complement etc.), Engineering unit and so on. The Parameters section can also be included as a child of the channel element to define what Parameters can be read from a channel.
- **Channels**: Specifies the number of channels on a device, the directionality of each channel and what type of link can be connected to a channel. Manufacturers can also use this section to specify what processes are associated with a channel.
- **Specifications**: A specification is data about a device that does not directly affect the functioning of that device. An example of a specification would be "Storage Capacity" on a memory card. This is a read-only value that may be useful, for example, to validation software.

One of the big advantages of this section, and of XdefML in general, is that it allows vendors and customers that develop XidML-aware software, to use XdefML to both auto-generate user interfaces and to automatically validate user input [5]. Figure 1 gives an example of how DAS Studio 3 uses XdefML to construct a user interface to instrument settings.

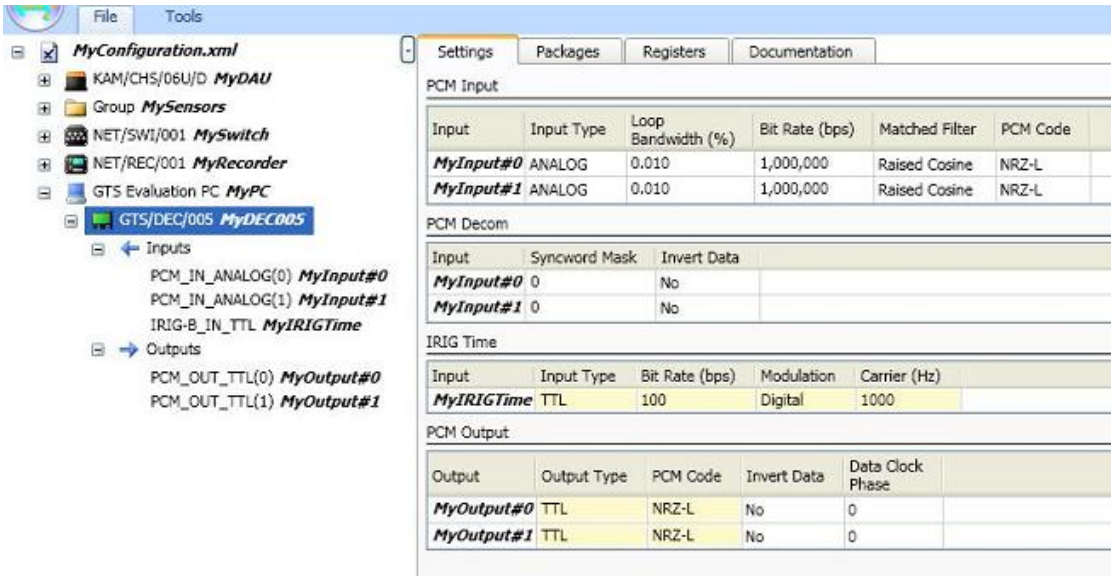

Figure 1: A screenshot of the Settings tab in DAS Studio 3

3 EXTENDING XIDML

3.1 GENERIC PACKAGE DEFINITIONS

There are currently nineteen package schemas defined in XidML. These package schemas include ones for IRIG-106-Chp4 PCM, MIL-STD-1553, ARIN-429, IENA, iNET and so on. These nineteen package types cover the vast majority of packages in use in the FTI community. However, there will always be proprietary, and other little used and niche package types that will not be of interest to the general FTI community.

A solution to this problem would to introduce a new Generic package schema. Any generic package schema should be sufficiently flexible and generic to allow for the definition of potentially any package type of interest to any FTI project. Specifically, a generic package schema should be able to describe a package's Properties and Content in a generic way. The schema should also be able to define how a package is identified in a generic way.

A possible way to achieve such a generic package schema is to use the concept of Settings, as is currently defined under the instrument schema, in the Properties section of a Generic package schema.

- **Properties**: This section would use the same Settings section used in the Instrument schema, each Setting consisting of a Name/Value pair. The constraints, default values and visibility of each of these Settings would also be defined in XdefML, providing vendors the same scope for developing auto-generated user interfaces and data validation that has been leveraged in the case of instrument settings.

- **Content**: For the Generic package the Content section would effectively be a *superset* of the Content sections of all the other package types defined in XidML. There are essentially three types of location identifiers

    - **Content ID**: This is where an ID or tag identifies a location in a package. An example of this would be AFDX packages.
    - **First Word/First Minor Frame**: Two coordinates identify the first location in a package. An example of this would be IRIG-106-Chp4 PCM.
    - **First Offset**: This can be specified in Bits, Bytes or Words. An example of this would be an RS-232/433 message definition.

In XdefML, depending on the package type being described, Vendors or users would specify a specific offset type that is applicable to the package type being modelled.

- **Identifiers**: An identifier would be used to specify how a package is identified. An identifier would contain the offset (relative to the start of the package) where the identifier is located, the number of bits used in the identifier and a name for the identifier.

3.2 XdefML

3.2.1 SNMP

Some devices require complex configuration. Other devices such as switches may require very simple, SNMP based setup. It may be useful to be able to indicate when a device can be configured using SNMP.

One way of doing this is to indicate in a device's XdefML file that it can be configured using SNMP. Currently in XdefML there are four "Classes" or "Value Types" associated with a Setting value:

- **Fixed**: Indicates the setting value must be fixed to a certain value
- **Selections**: Used when a value must be selected from a discrete set of values
- **Range**: The Range type is used to specify that the setting value must be within a specific range
- **Reference**: Specifies that the value of the setting refers to another XidML component such as a package or algorithm
- **VariableString**: This type indicates that the value is a string but the format of the string can also be specified (e.g. its length)

For each of these value types users can also specify a default value and its native data type (e.g. string, float, integer).

It is proposed to treat an SNMP variable as a normal setting. This SNMP Setting would be defined like any other Setting with each of the above "Value Types" used in the same way.

However, for SNMP Settings, an OID (Object Identifier) would also be specified in addition to describing whether it is a Get, Get|Set, Set or Trap variable.

Furthermore, in XidML-aware software [5] that use XidML and XdefML to auto-generate user interfaces, these "SNMP" settings would be treated in the same way as normal settings, and read-only "Get" SNMP settings would be treated as Specifications.

3.2.2 INSTRUMENT SPECIFICATIONS

The *InstrumentSpecification* section of the XdefML schema contains general information of use to FTI Vendors and users such as Mechanical class, Functional class and so on. The original intention of this section was that it would grow over time to include even more information. It is suggested that the following details could be added:

- **Link characteristics**: This type of data would include, Pin Numbers, Input impedance, Connector type (D-Type, Mighty Mouse Circular – Male or Female) and so on.
- **Device dimensions**: The width, height and depth of a device
- **Device Weight**: The weight of a device.

4 CONCLUSIONS

Since it was first released in 2005, XidML has become increasingly popular. With the release of version 3.0 of XidML, and the introduction of the concept of a generic instrument schema, describing device setup has become particularly easy. XdefML has also facilitated software in the auto-generation of user interfaces and automatic data validation. Its simplicity and scope should be further enhanced if the proposals on SNMP and the extension of the same generic settings mechanism to the concept of a generic package schema are adopted.

5 REFERENCES

[1] XidML website, http://www.xidml.org/
[2] Cooke, Alan and Corry, Diarmuid, "XidML: A Global Standard for the Flight Test Community", ETTC Proceedings, 2004
[3] Corry, Diarmuid: "Unleashing the Power of XidML", ITC Proceedings, 2007
[4] Cooke, Alan: "Using XML in an FTI Environment", ETC Proceedings, 2008
[5] Herbepin, Christian and Cooke, Alan: "Introduction To XidML-3.0 – An Open XML Standard for Flight Test Instrumentation Description", ITC Proceedings, 2010
[6] World Wide Web Consortium, http://www.w3.org/