DOI 10.5162/etc2016/3.1

An Adaptable Constraints-based Metadata Description Language (MDL) System for Flight Test Instrumentation Configuration

Michael Neumann[1], Jessica Moore[1], Satyaraj Pantham[2],
Myron Moodie[3], Patrick Noonan[3], Austin Whittington[3]
[1] The Boeing Company, Seattle, Washington, USA
michael.neumann@boeing.com, jessica.d.moore@boeing.com
[2] Consultant, Seattle, Washington, USA
srpantham@gmail.com
[3]Southwest Research Institute®, San Antonio, Texas, USA
mmoodie@swri.org, pnoonan@swri.org, austin.whittington@swri.org

Abstract:
The current telemetry device provider landscape is diverse in approach and capability, and choosing the "best" devices for a particular solution inevitably involves a mix of devices from different vendors. Devices have differing capabilities and limitations. Each vendor offers their own proprietary configuration toolset. In the past it has been possible for the flight test community to build system-wide configurations at very high cost by reverse engineering business rules for existing flight test configuration systems. In addition, technical proficiency was required by the users for every tool in this widely varying set or risk being locked into a single vendor. Historically, the Boeing Flight Test Computing System (FTCS) and similar systems have device rules tightly coupled into the software, and so as device constraints change or new devices are added, significant work is required to update and maintain the software. To solve this problem, Boeing and Southwest Research Institute (SwRI) developed a constraints-based and MDL-centric configuration management approach. Based on this approach, the Boeing Company has developed its own MDL based Modular Instrumentation Setup Tool (MIST) to manage and simplify the configuration of new devices for their users. This paper highlights the success of this approach on the 737 MAX program and discusses how constraints were implemented, how validation occurs, and shows how the system can be rapidly updated with new constraints based on device changes or user insight.

Key words: Flight Test Instrumentation, Constraints, MDL, XForms, XPath.

Introduction

Data acquisition devices used in onboard flight test networks are currently supplied by providers with their respective de facto methods for configuration. In addition, the capabilities and limitations of the devices is widely variable across the spectrum of telemetry device vendors. This often results in a complex and time consuming effort for instrumentation setup as users have to learn and deal with different configuration software and processes.

Recent standardization efforts in flight test networks provide hope for managing this complexity. One example is the Metadata Description Language (MDL) created by the US DOD integrated Network-Enhanced Telemetry project as an interoperable flight test device configuration language and growing in use for both commercial and military flight test applications. An application providing a standardized data transfer medium enables it to interact with multiple applications serving similar business objectives.

In this paper, we will discuss the benefits of using MDL for configuration augmented by multiple levels of constraints in XForms as part of a new system that removes large portions of the complexity and cost for adding new devices and maintaining existing devices. The goal has been to create a system that is vendor agnostic and provides reusability and extensibility. MDL and the constraints representation external to

MDL serve as key components for creating valid device configurations.

We will describe the constraints and MDL backgrounds, followed by description of the chosen constraints format. The final section will explain the implemented Modular Instrumentation Setup Tool (MIST) and its use of constraints and MDL.

Constraints

A constraint is, by definition, a limit or restriction placed on a person or thing, or on an action or behavior. In context of a device within a Flight Test Instrumentation system, a set of constraints defines the limits of how the device may be programmed to perform a specific task. Since each vendor offers their own proprietary configuration toolset and capabilities, the set of constraints for a particular vendor's device must be known in order to build a valid programming file for the device.

A user of this device requires some sort of user interface in which they can define all the necessary inputs to build the programming file. Constraints on these inputs may be simple field validations, such as a set of allowable values or upper and lower limits. There may also be more complex constraints in which values entered in one or more fields affects the constraints on one or more other fields.

In addition to the user interface constraints, there are typically some parameters which the device requires but which the user does not want to explicitly specify. Constraints may be used to limit these parameters to a single value which can be entered into the file with no user interaction. Again these may be simple constraints which limit the parameter to a single value, or they may determine the single value based on the value of other parameters.

For any given system or device, there can be multiple sets of constraints from different sources. The vendor will provide a set of constraints that describe the capabilities and limitations of their device settings. The user of the device may want to add additional constraints based on their own preferences. For a system of devices, there may be additional vendor and/or user constraints describing the capabilities and limitations of the system.

A constraints validation system, then, must be able to describe all of the possible constraints of the device or system, and must be able to validate a configuration against multiple sources of constraints.

Similar to typical flight test systems, Boeing's Flight Test Configuration System has historically embedded all of these constraints in code. Any change or addition to the constraints was costly due to the significant work required to update and maintain the software.

Our many years of building Flight Test Configuration Systems has shown that a *correct-by-construction* approach is needed. The *correct-by-construction* approach prevents invalid configurations being created at any level. Whether by using constraints or hard-coded business rules, you bypass the need to have continual communication with the device being configured as you negotiate and validate the programming file incrementally.

Since input is validated in real-time as the user builds up their configuration, any invalid pieces or new cascading requirements are immediately made known to the user. Additionally, by using *correct-by-construction*, the passing of the completed file to the device for final proofing becomes just a formality. This could otherwise be a step which, stemming from some small value change since the last pass, requires a total rework of the programming file.

By externalizing the constraints into modular XForms files, constraints can be easily modified without changing the code of the configuration system. This paper describes a method by which these constraints files may be used to validate a user-defined MDL configuration for a Flight Test Instrumentation device.

Metadata Description Language (MDL)

Metadata Description Language (MDL) is a common configuration language that describes requirements, design choices, and configuration information for Telemetry Network Systems (TmNS) [1]. MDL encapsulates the setup data of the network nodes and measurement devices, along with their units of measurement. In a typical flight testing computing system, the analog and digital data acquisition units (DAUs) are represented by the network nodes, and various transducers and sensors are represented by devices.

The setup information in MDL is represented in a hierarchical style and is highly readable through any standard XML editor, text editor or even in a browser. Readers can easily walk through the data tree, its nodes and associated data. The data items are defined as elements in terms of tags and attributes. The attributes can lead to utilizing highly efficient search engines or intelligent data mining agents.

Along with simplicity, MDL also comes with all the great advantages of XML which include a wide variety of data types. MDL can also serve

as the single-document view for dispersed data across multiple devices, and an MDL instance document supports localization and internationalization.

Except for some trivial cases, an instrumentation setup process for flight testing is cumbersome and may require multiple sessions of interaction with setup systems. A smart client is a preferable choice since it can support the setup data to be saved temporarily into some local data storage. MDL can serve as an XML based data repository for holding device configuration data in one or more offline sessions. When the setup system is online, the MDL configuration can be stored in the flight test database.

MDL has another significant advantage: flight test setup data becomes reusable. A particular flight test setup stored in an MDL instance document contains the content relevant to that test process such as the instrumentation setup and device data. This data may either be utilized in other testing scenarios for the same airplane, or for similar test scenarios of other airplanes, thereby leading to a considerable saving on time and effort.

Flight test systems contain numerous devices that read and format data, and multiplexers that combine and transmit data to other onboard systems. The task of setting up these devices becomes unwieldy if vendors supplying the instrumentation do not conform to a common standard. The flight test instrumentation setup engineers and technicians have to produce multiple data files for each of the vendor device categories. MDL was created and standardized by the integrated Network Enhanced Telemetry (iNET) program to provide a single vendor-neutral standard which promotes interoperability between these systems, devices, and applications which may have been developed by different organizations and vendors [2].

An MDL instance document represents a comprehensive description of a given flight test setup. This kind of a standard configuration language enables flight testing processes to be executed with better portability of setup data among the flight testing systems as well as other enterprise systems that include those that are geographically distributed for other lines of business. The standardization enables reuse of instrumentation setup in other testing scenarios with significantly reduced effort. The standardization of instrumentation setup data can lead to other advantages like reusability of application tools. In the next sections to follow, we will discuss the many advantages in using MDL and the challenges that arise in the development of flight test setup applications.

Constraints Combined with MDL

XML in general uses the common language of XPath to address parts of an XML document [3], and perform calculations and checks upon target elements. This language is used for defining element relationships and schema constraints in XML Schemas, node tests and matching in XSLT, and many other applications along the breadth of the XML ecosystem. For MDL specifically, XPath is used within the schema to define uniqueness on fields and referential constraints checking that elements refer to the correct targets.

For the purposes of vendor and user constraints, XPath is again used. These new layers of constraints go beyond merely checking that the document is valid as an MDL file, which forces their presence external of the MDL schema. For the application of constraints described in this paper, these XPath constraints were built up in XForms, another XML-based technology made for gathering and processing XML data [4]. XForms was chosen for its clear separation of the validation required to check the constraints and the presentation which gives the result of that validation to the user, as well as its direct use of XPath to simplify the application of the constraints to the MDL documents and the availability of a variety of existing tools for processing XForms.

Vendor constraints can exist in many forms, from logic buried deep within compilers to information contained in user manuals to a spreadsheet of requirements. All of these constraints are candidates to be written in XPath and used in a system such as MIST. Due to the set of circumstances present at the beginning of this project, the constraints were not available in XForms directly from the vendor. Consequently, the vendor constraints files were developed by Boeing and SwRI using knowledge of the device capabilities. We were first provided with a list of compiler error messages from the vendor of the device. By consulting with the vendor and through knowledge of the instrumentation field, we were able to translate these error messages into English-language descriptions of the target constraints. From these descriptions, we could create the XPath which checks the description's conditions in a straightforward manner.

There are many industry-standard XML tools that we used for editing, testing, and validating our XML instance documents. However, a custom tool was needed for the XForms and constraints specific functionality we required. As

such, we made use of the SwRI-developed XFORGE toolkit to generate several XForms from the MDL schema, each of which contained the presentation layer for the desired elements necessary to constrain the user's input. We then added bindings for the XPath constraints to the XForms model and added descriptive error messages to inform the user of any MDL elements and fields which did not meet the device's requirements discovered by the constraints. These completed XForms were then used by MIST to provide the constraint validation capabilities.

A sample constraint follows in Figure 1. The constraint encodes the English-language sentence "A SignalRange must have two ConditionParameter bounds whose values are not inverted." The constraint therefore checks that the lower bound (the ConditionParameter with a greater-than or greater-than-or-equal sign) is less than the upper bound (the ConditionParameter with a less-than or less-than-or-equal sign).

```
<xf:bind
    ref="mdl:SignalRange"
    constraint="count(mdl:ConditionParameter) ge 2 and
        number(mdl:ConditionParameter[mdl:ConditionOperation
        = ('&gt;=', '&gt;')]/mdl:ConditionValueFloat) lt
        number(mdl:ConditionParameter[mdl:ConditionOperation
        = ('&lt;=', '&lt;')]/mdl:ConditionValueFloat)">
```

Fig. 1. Constraint Example

Modular Instrumentation Setup Tool (MIST)

As part of the 737-MAX flight test program, the Boeing Company has implemented a Modular Instrumentation Setup Tool (MIST) for the configuration of new flight test devices that can be programmed using MDL setup files. The tool works on multiple platforms (Windows, Linux), is scalable for additional modules and provides a vendor agnostic interface where changes to the tool can be limited by having vendors provide business rules in a constraints format using XForms and XPath expressions. Configuration was successfully provided for all new devices for the 737-MAX test airplanes.

MDL was chosen as a vendor interface because the first devices for which MIST provides the programming files are able to accept MDL files. In addition, the Boeing team wanted to conform to the emerging iNET and MDL standards. The tool is capable of interfacing with vendor hardware that can accept different XML schemas, which can also be validated by using the XForms/XPath constraints mechanism.

MIST uses vendor and Boeing specific constraints to validate user input and provide immediate feedback for any values that are not within the constraint specified limitations. Boeing instrumentation users are able to configure a stack of modules through instant feedback for the data they have entered, and save complete or incomplete configurations for later work. If a configuration has been completed with no errors, and successfully compiled by a vendor provided compiler, the resulting MDL file can be sent to the actual devices using the Boeing Flight Test Computing System (FTCS).

Future enhancements will include an onboard version of MIST that will allow users to dynamically configure MDL devices, and a standalone version that will provide users the capability to work on configurations offline and import changes back into the system.

The MIST MDL interface connects the tool to FTCS or any other flight test system that can ingest MDL data. For vendor devices, the tool interfaces using programming files in MDL format based on the vendor and Boeing specific constraints that have been provided and are being used during the validation stage in the process.

DOI 10.5162/etc2016/3.1

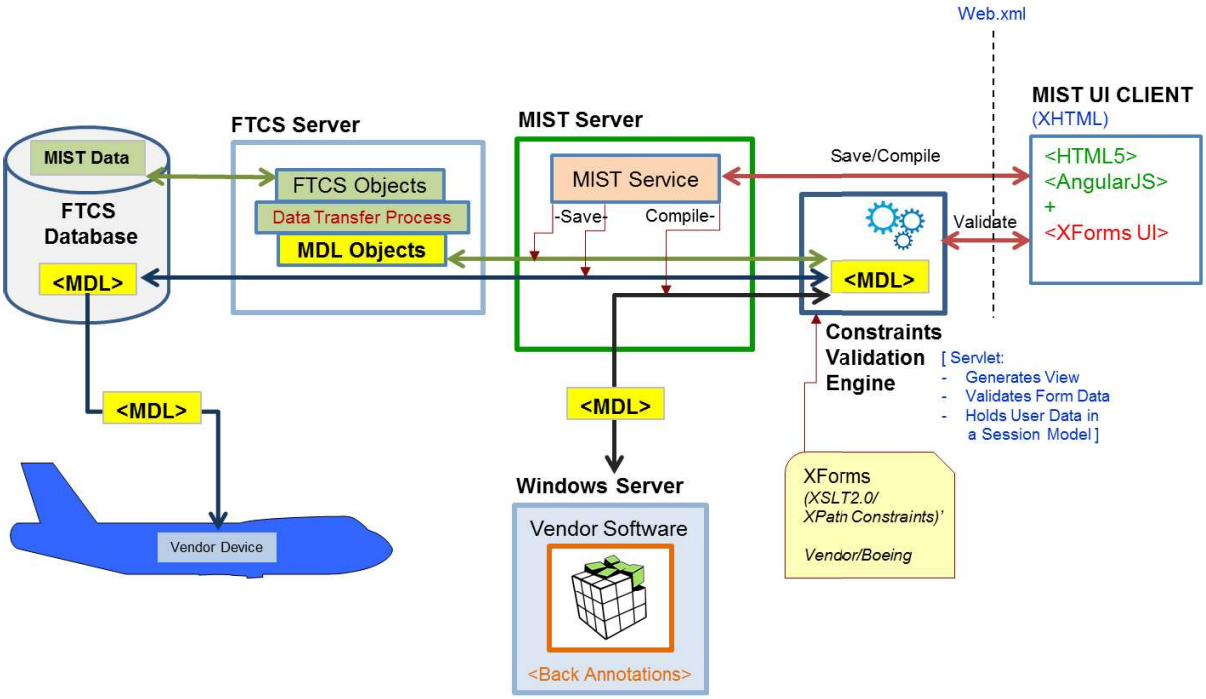

Fig.2. Modular Instrumentation Setup Tool (MIST) Logical Architecture.

Constraints based validation is at the center of the MIST architecture and provides the means for not having to hard-code vendor specific business rules and to reverse engineer vendor provided software that creates the final configuration files for their devices.

MIST accepts constraints in an XForms/XPath standardized format. The constraints are loaded into a third party validation software that will instantly verify user input which can also include additional non-editable vendor constraints and Boeing user and system constraints. An MDL object model for storing the configuration data is internally maintained by the third party validation software, which allows it to instantly verify user input. During the compile and save actions from the MIST tool, the MDL object model will be exported and sent to the vendor software or the FTCS database as a MDL data stream or physical file. Updating any existing constraints will be performed by the vendor or Boeing user in the appropriate constraints file. Only the addition of new constraints that include an update to the MIST user interface will involve new coding.

MIST opens in a Web browser and configures stacks with varying numbers of modules. The user interface contains XForms segments that allow the third party validation software to display components that are constrained in addition to non-constrained components that display information from the FTCS database. Users interactively validate their data using the vendor and Boeing provided constraints and receive instant constraint validation errors in the user interface for their selected input (Figure 3).

Fig. 3. Instant Constraint Validation

Boeing instrumentation users configure a stack by airplane and test number. Measurements can be added to channels for each module, and measurement properties can be modified on different panels with the application providing instant constraint validation. Figure 4 shows an example user interface with proprietary information having been replaced with generic data.

DOI 10.5162/etc2016/3.1

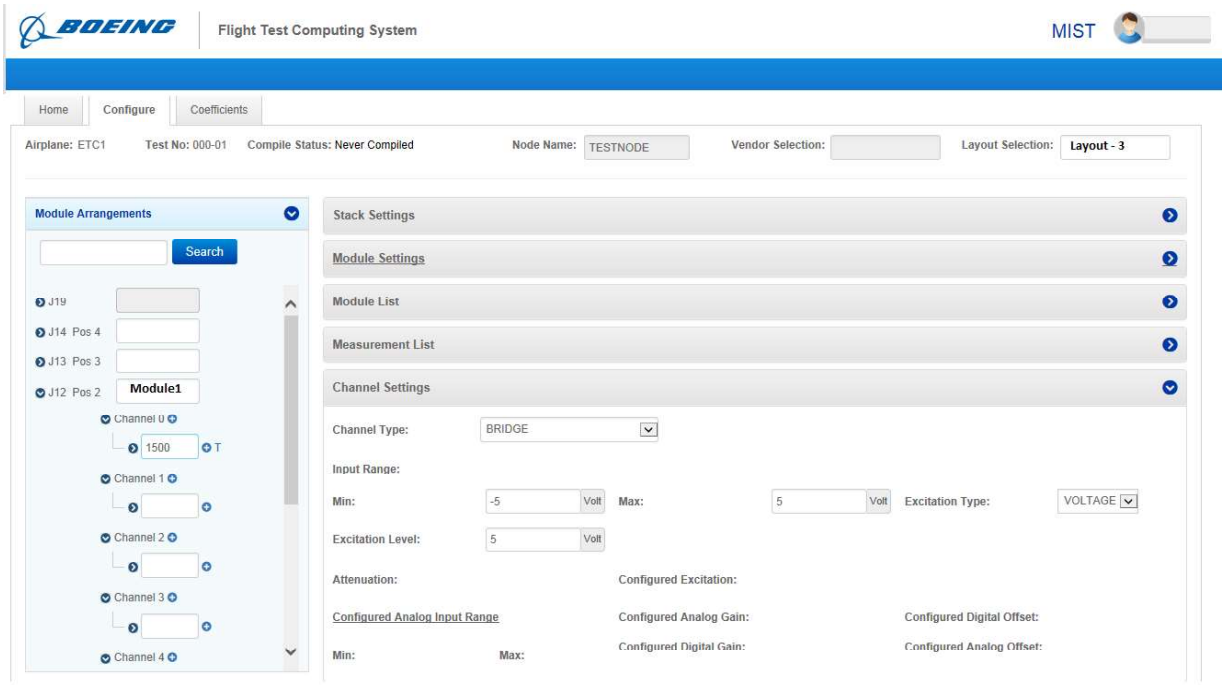

Fig.4. Modular Instrumentation Setup Tool (MIST) Screenshot.

The vendor software is called during the compile process and back-annotates additional vendor specific data to the MDL file that gets returned to MIST and stored in the MDL object model inside the third party validation software. At any time, the current configuration can be saved to the FTCS database as MIST data and the complete MDL file for further loading on a vendor device once the validation process returns no configuration errors.

Conclusion

An adaptable constraints-based MDL system for flight test instrumentation configuration has been successfully implemented by Boeing for the 737-MAX flight test program. Constraints allow for faster integration of new hardware devices since business rules are no longer hard-coded and can be directly provided by the vendor. The end user experiences operational efficiencies through early validation and a process that can guarantee a valid configuration file for the devices in use. The responsive system avoids mistakes and provides an easier learning curve for new instrumentation engineers.

Constraints provide maintenance benefits by allowing engineers and vendors to only modify a constraints file without developers having to write additional code, except in situations where there are new user interface changes required. This also allows Boeing engineers to work their own user constraints and can lead to a future system where engineers can be allowed to directly create programming files for new or modified constraints

The use of XForms to capture the constraints has been shown to provide the flexibility necessary to describe the constraints of complex network flight test instrumentation. When combined with MDL this provided a capable and vendor independent device configuration approach that should scale to a wide variety of future devices.

References

[1] iNET system Management and Configuration, ETC 2010

[2] Metadata Description Language: The iNET Metadata Standard Language ITC 2009

[3] https://www.w3.org/TR/xpath/

[4] https://www.w3.org/TR/xforms/