RULE-BASED CONSTRAINTS FOR METADATA VALIDATION AND VERIFICATION IN A MULTI-VENDOR ENVIRONMENT

John Hamilton, Timothy Darr, Ronald Fernandes, Dave Jones
Knowledge Based Systems, Inc.

Jon Morgan
JT3, LLC
Edwards AFB, CA

ABSTRACT

This paper describes a method in which users realize the benefits of a standards-based method for capturing and evaluating verification and validation (V&V) rules within and across metadata instance documents. The method uses a natural language based syntax for the T&E metadata V&V rule set in order to abstract the highly technical rule languages to a domain-specific syntax. As a result, the domain expert can easily specify, validate and manage the specification and validation of the rules themselves. Our approach is very flexible in that under the hood, the method automatically translates rules to a host of target rule languages. We validated our method in a multi-vendor scenario involving Metadata Description Language (MDL) and Instrumentation Hardware Abstraction Language (IHAL) instance documents, user constraints, and domain constraints. The rules are captured in natural language, and used to perform V&V within a single metadata instance document and across multiple metadata instance documents.

KEYWORDS

Instrumentation configuration, XML, rule-based validation, metadata, controlled natural language, IHAL, MDL, TMATS, verification and validation (V&V)

INTRODUCTION

Several efforts to define eXtensible Markup Language (XML) standards for the representation of test and evaluation (T&E) metadata have reached maturity. The Range Commander's Council (RCC) data sciences group (RCC-DSG) adopted the Data Display Markup Language (DDML) XML schema standard for describing data display metadata in IRIG 106[1]. The iNET program promoted a family of standards for capturing network telemetry descriptions, including the Metadata Description Language (MDL). The RCC telemetry group (RCC-TG) adopted multiple

1 Specific versions of IRIG 106 (i.e. IRIG 106-05, IRIG 106-07, IRIG 106-09, IRIG 106-11, IRIG 106-13, etc.) are omitted.

standards for telemetry metadata: the TMATS ASCII standard was transliterated into XML, and the Instrumentation Hardware Abstraction Language (IHAL) XML standard for describing the details of telemetry hardware was adopted.

The objective of these efforts was to promote multi-vendor, multi-range standards for describing T&E metadata and facilitating interoperability. The adoption of these standards looks very promising. However, these standards only capture the syntax of the respective metadata and support minimal, if any, verification and validation (V&V). Existing XML libraries can support very primitive syntactic validation. The next level of interoperability involves *semantic* V&V *within* XML metadata standards as well as V&V *across* XML standards. A solution that achieves the next level of interoperability must include a way to easily capture the rules, a standards-based format for storing the rules in a machine interpretable way, and open source inference engines for evaluating the rules.

RULE REPRESENTATION AND CAPTURE

There currently exist generic standards for representing rules for V&V (some of which are XML-based); for example, the W3C Rule Interchange Format (RIF) [1] and the de facto Drools [2] standard for business logic. There is no reason to introduce yet another custom standard for the T&E community that must be approved by a standards committee. The role of the standards committee(s) will be to help identify the requirements for rule categories that must be implemented in an existing standard. The available standards have varying degrees of inferential support, so care must be taken to define the requirements for a rule representation standard and select the standard that best meets the T&E community requirements.

However, rules written in these standards are not accessible to the typical instrumentation engineer--the rules are often described only in XML. Few authors and users in the T&E community have the time, inclination or interest in learning XML. To address this, a controlled natural language (CNL) will be used to capture the rules in an intuitive, domain specific way. This will allow those that know the rules most intimately--the engineers--to define the rules without having to learn or even know about the underlying standard.

Controlled natural languages (CNLs) are subsets of natural languages that are obtained by restricting the grammar and vocabulary in order to reduce or eliminate ambiguity and complexity. Since the language has a restricted vocabulary and defined grammar, it would be very difficult to write a statement without Intellisense[2] editor support. In essence it is a normative process to write a specification in CNL because it prescribes how to effectively communicate using a language to achieve a particular goal. An intelligent interface can make this writing process unobtrusive and effortless. Use of error messages, domain term highlighting, predictive feedback and conceptual authoring were some of the techniques that were found to support the CNL writing process.

2 http://en.wikipedia.org/wiki/Intelligent_code_completion

Figure 1 shows a screen capture of a tool for intelligently capturing V&V rules using CNL. As the user begins typing, the tool suggests possible terms and phrases to complete the rule. These suggestions are context-aware; that is, they take into account what has already been captured.

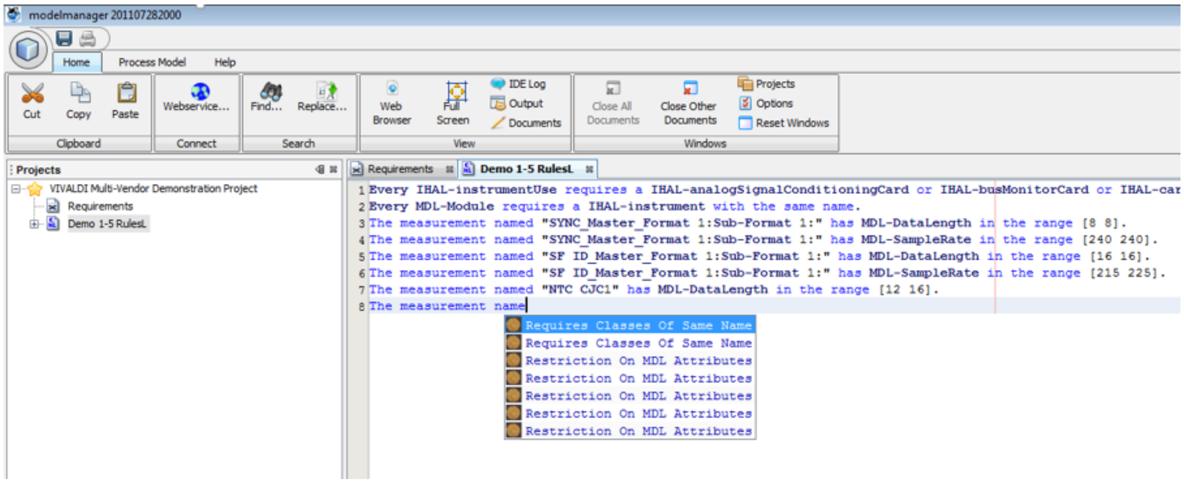

Figure 1 – Intelligent CNL Rule Editor

Some sample rules captured in CNL for this demonstration include the following:

- Every IHAL-instrumentUse requires a IHAL-analogSignalConditioningCard or IHAL-busMonitorCard or IHAL-card with the same name.
- The measurement named "SYNC_Master_Format 1:Sub-Format 1:" has MDL-DataLength in the range [8 8].
- The measurement named "SYNC_Master_Format 1:Sub-Format 1:" has MDL-SampleRate in the range [240 240].
- The total IHAL-baseWeight for IHAL-instrumentationGraph identified by "iNETValidation" is no more than 20 pounds.
- Every R-Group-Characteristics R-Group-Manufacturer required-when R-Group-Type value is "Analog."
- Every R-Group-Characteristics R-Group-ExternalRMMBusSpeed required-when R-Group-Type value is "Analog."

Once the CNL is captured, it is transformed into a target rule representation language that can be used to perform V&V against an XML instance document.

RULE DEVELOPMENT AND MAINTENANCE ARCHITECTURE

Figure 2 shows the layered architecture for the rule maintenance capability. The top layer shows the capabilities that will be provided to the user as a dashboard. The dashboard provides the following capabilities to the user:

- A T&E Metadata Standards Browser for browsing the supported schemas. Using this capability, the user will be able to browse a schema for context in creating rules.
- A CNL rule editor for creating and editing rules.

- A rule browser for finding existing rules to apply to a given validation task.
- A V&V analysis results capability for reviewing the results of applying rules to an instance document.
- A rule browser for finding existing rules to apply to a given validation task.
- A rule consistency editor for identifying and resolving inconsistent rules (rules that are in conflict with one another).
- A de-confliction / optimization capability for optimizing a schema and performing rule de-confliction.

The middle layer shows the data / knowledge base information utilized by the architecture and consists of a repository or database for supported schemas and instance documents and a repository that stores CNL rules.

The lowest layer shows the analytics used to support the validation and includes rule-based inference engines, rule consistency engines, constraint satisfaction engines, optimization engines, semantic reasoning and workflow engines

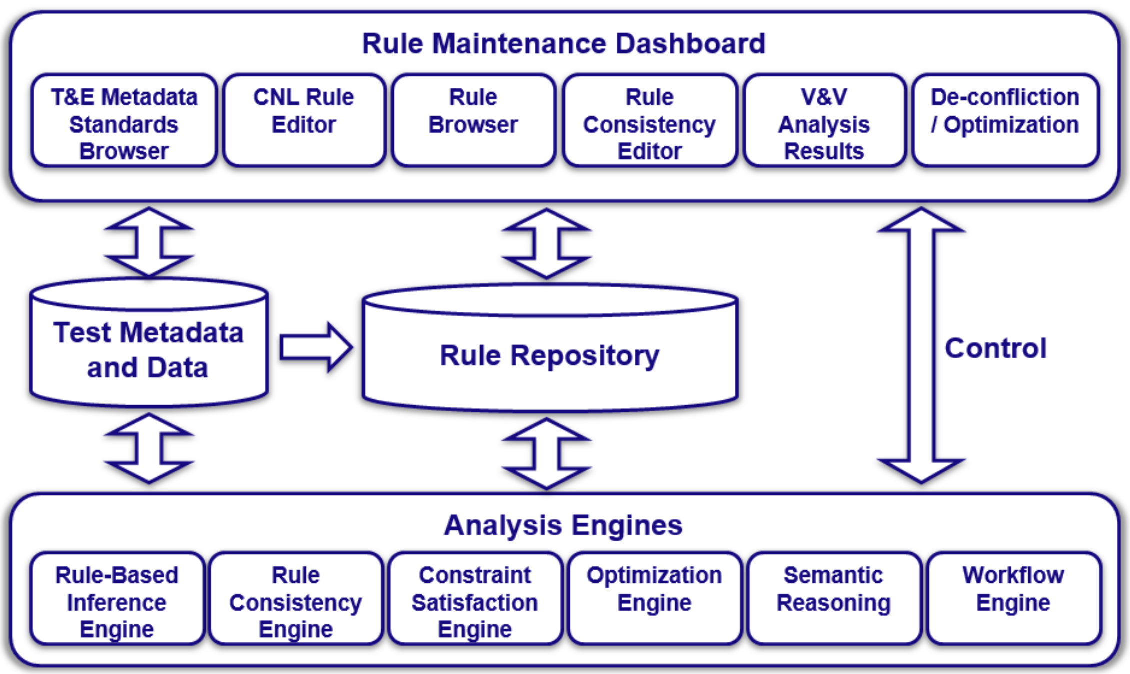

Figure 2 – Rule Maintenance Capability Layered Architecture

Figure 3 shows the preliminary component architecture that highlights the information flows among the components.

The capabilities are accessible via a *User application (rule capture)* installed on the user desktop or available on the web. In this application, the user defines the rules in CNL format and selects one or more XML instance files to validate against the rules. The captured CNL and selected XML instance files are information flows to the *Schematron Translator* and *Drools Translators* components. The *Intellisense* editor of the user application is smart enough to understand the approved structures of the CNL and able to look-ahead within the grammar defined. The author gets immediate feedback while writing the text and the editor prevents the author from entering sentences not allowed in the CNL grammar. While writing, the analyst can request suggestions from the editor. The suggestion provided to the user is context aware: i.e., the editor provides suggestions based on cursor position and the applicable grammar rule.

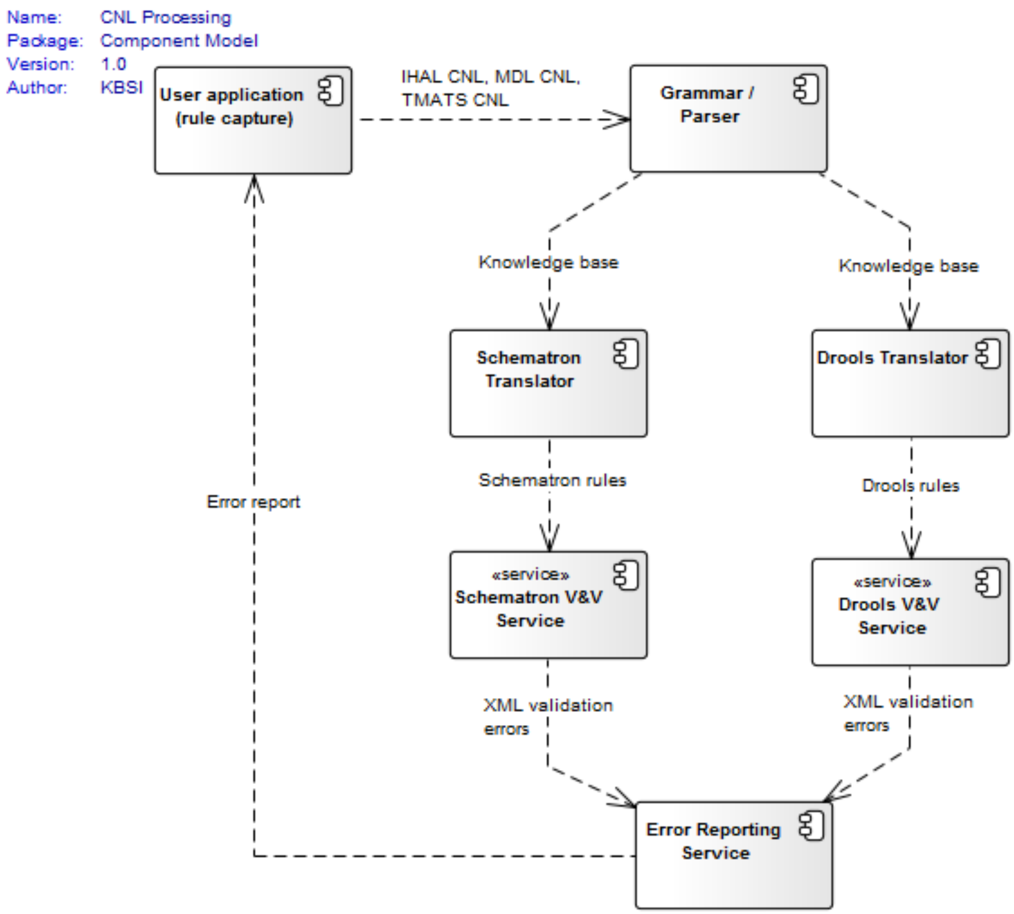

Figure 3 – CNL Processing

One of the most obvious ways to support writing in restricted languages like CNL is error highlighting. Initially users are trained in the syntax of the CNL and then asked to formulate specifications in the learned language. As the users type, the CNL interface parses the text and validates the text against the CNL grammar rules. If the validation fails, the interface will try to identify the cause of the error and provide suggestions for fixing the error.

The *Grammar / Parser* component transforms the captured CNL into a target-rule representation (i.e., Schematron or Drools). This component receives the CNL information flows from the *User application (rule capture)* component and sends the rules to the respective validation services.

The *Schematron Translator* and *Drools Translator* components translate the parsed CNL into the Schematron and Drools target rule representation. These components receive the CNL information flow from the *Grammar / Parser* component and send the rules information flow to the validation service components.

The *Schematron Validation Service* and *Drools Validation Service* components validate an XML instance document against a collection of rules originally captured in CNL. These components receive the rules information flow from the translator components and send the XML validation error information flows to the *Error Reporting Service* component.

The *Error Reporting Service* component collects the error reports from each validation service and presents them in a format for the end user. This component receives the XML validation errors information flow from the validation components and sends the report information flow to the user client components.

DEMONSTRATION ARCHITECTURE

Figure 4 shows the demonstration architecture with L3-Com and Curtiss-Wright as the equipment vendors[3]. The key components of this demonstration are:

- User Validation Tool is a software application for capturing V&V rules in CNL and transforming the captured rules into a target rule execution language; currently supported target rule languages include Drools [2] and Schematron [3] (XPath expression for rule logic).
- L3 Com Configuration Service is a service provided by L3-Com that implements the IHAL API and a similar API for accessing MDL data and capabilities.
- Curtiss-Wright Configuration Service is a service provided by Curtiss-Wright that implements the IHAL API and a similar API for accessing MDL data and capabilities.

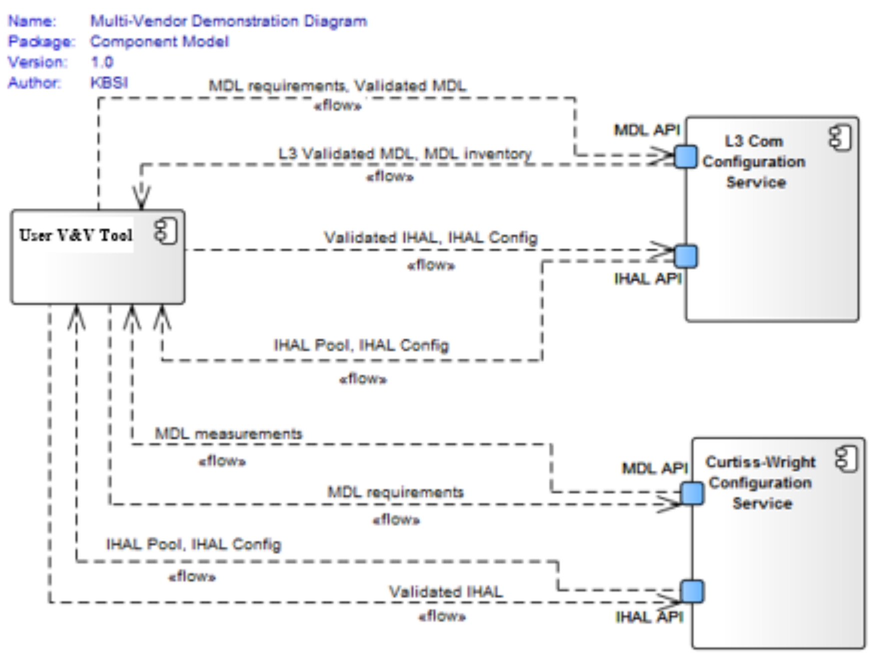

Figure 4 – Demonstration Architecture

3 There is no public MDL API, so L3-Com and Curtiss-Wright implemented an API to provide the capabilities required for this demonstration scenario.

MDL / IHAL MULTI-VENDOR DEMONSTRATION SCENARIO

Figure 5 shows an outline of a demonstration sequence diagram. Starting with an MDL instance file that contains only requirements, the user validation tool calls the vendor validation service (L3-Com service or Curtiss-Wright service). These services return the MDL inventory, the IHAL Pool and the IHAL Configuration. The user validation tool then performs cross-validation on the MDL and IHAL instance documents. Next, the user manually details the MDL configuration and uses the vendor validation services to validate the MDL requirements against the detailed MDL file using internal vendor validation logic. The vendor validation services return a validated MDL file that includes updated configuration settings. The user then uses the user validation tool to validate the original MDL requirements against the newly-validated MDL and also validates the validated MDL against the IHAL configuration.

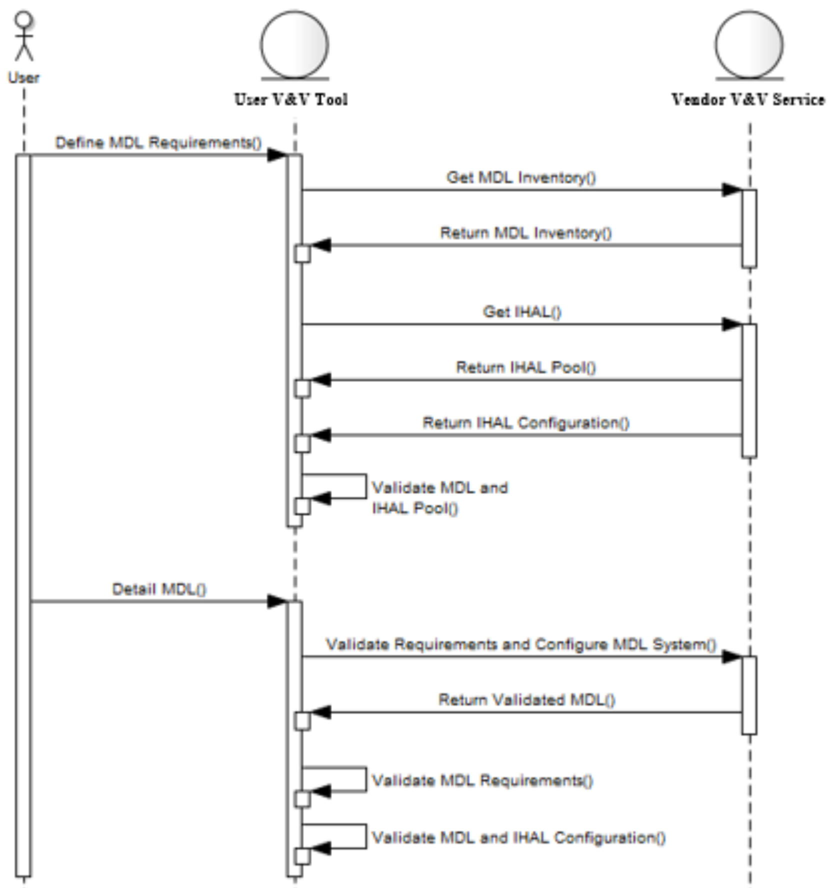

Figure 5 – Demonstration Scenario Outline

TMATS VALIDATION DEMONSTRATION SCENARIO

The final demonstration shows the use of the *User application (rule capture)* for validating the semantic TMATS rules as documented in IRIG 106-15 Chapter 9. Figure 6 show the User application. The user selects a TMATS XML file and a rule file containing G Group rules from the left panel. The XML file is validated against the G Group rules by selecting the rule file and a

TMATS XML file, right-clicking and selecting “Run Rules.” The rules in CNL format are shown in the right panel. Figure 7 shows the results of the G Group validation rules. The program name length rule is violated as shown.

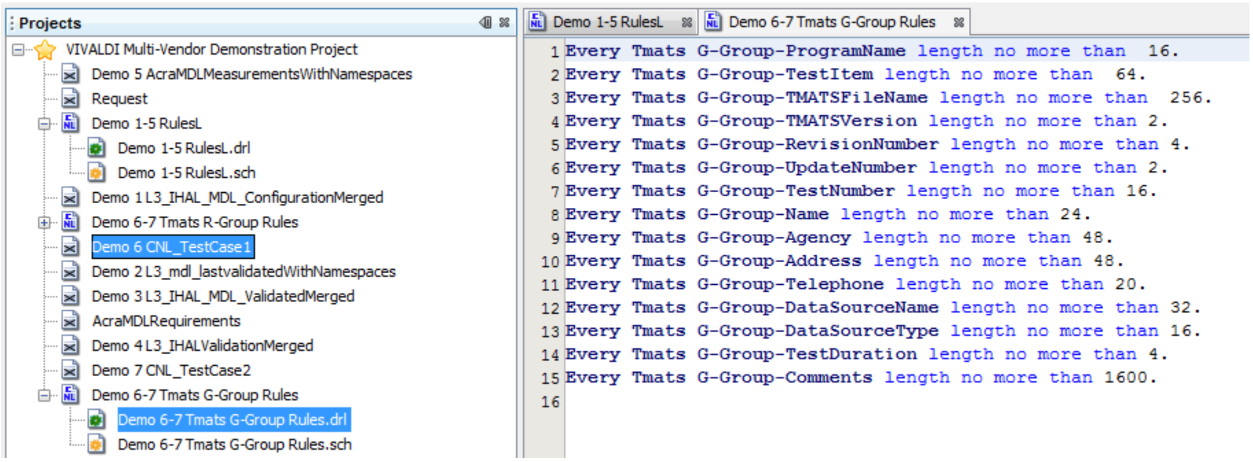

Figure 6 –TMATS G Group Validation

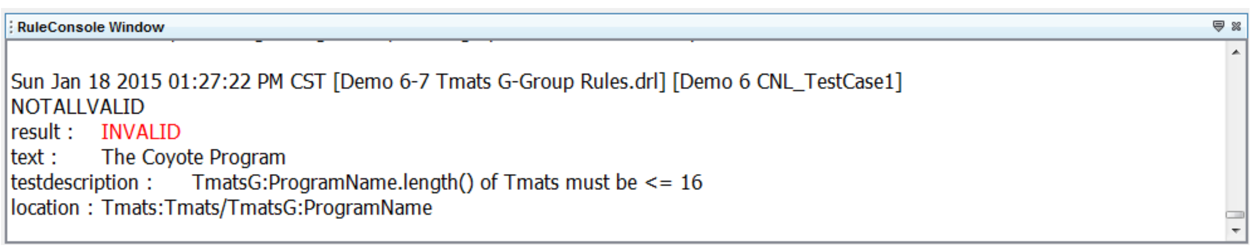

Figure 7 –TMATS G Group Validation Results

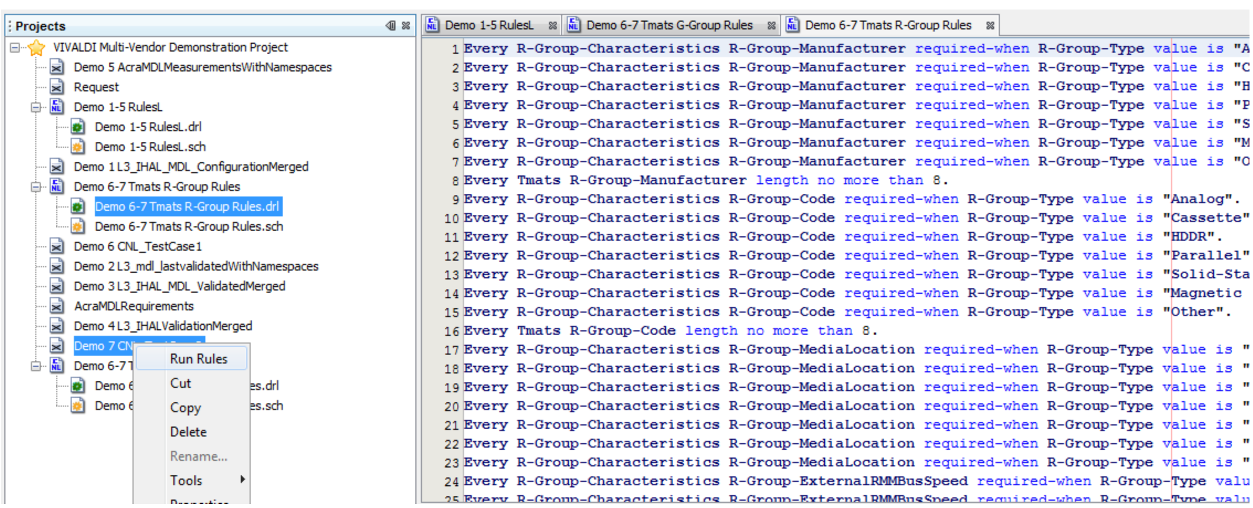

Figure 8 – TMATS R Group Validation

Figure 8, shows a similar scenario for validating a TMATS XML file against the R Group rules. The user selects the rule file and an XML file, right clicks and selects “Run Rules.”

BENEFITS / CONCLUSION

The rule-based validation and verification approach presented in this paper has the primary benefit that it uses a natural language based syntax for the T&E metadata V&V rule set in order to abstract the "computer science"-heavy rule languages to a domain-specific CNL-based syntax. As a result, the domain expert can easily specify, validate and manage the specification and validation of the rules themselves. Our approach is very flexible in that under the hood, the framework automatically translates CNL rules to a host of target rule languages, including Schematron, Drools, and semantic web rule languages (SPARQL, SWRL).

Additional benefits anticipated from this approach include:

- No need for committee approval of a new standard, minimizing the time to adopt the approach;
- A catalog of commonly-used rules provides a way for new users to boot strap into the framework without having to write new rules from scratch;
- An architecture that protects the intellectual property (IP) and proprietary nature of vendor configuration logic; and
- Distributing rules along with metadata reduces errors and inconsistencies in the shared T&E package.

For T&E ranges, our approach supports:

- Correctness and coverage of required measurements;
- Correctness at the instance level;
- Instrumentation vendor neutrality; and
- Best practices (e.g., safety, range operation directives, etc.).

REFERENCES

[1] http://www.w3.org/standards/techs/rif#w3c_all, last accessed June 20, 2015.
[2] http://www.jboss.org/drools/, last accessed June 20, 2015.
[3] http://www.schematron.com, last accessed June 20, 2015.