THE VIVALDI PORTAL – AN APPLICATION FOR METADATA VALIDATION AND VERIFICATION

Timothy Darr, Dan Corlette, Atul Jain, Ryan Bollweg
Knowledge-Based Systems, Inc.
College Station, TX, 77840
tdarr@kbsi.com

Jon Morgan
JT3, LLC
Edwards Air Force Base, CA
jon.morgan.2.ctr@us.af.mil

ABSTRACT

This paper describes the VIVALDI (Verification and Validation of Metadata for Test & Evaluation) portal, a web-based application that gives test and instrumentation engineers and T&E stakeholders the ability to define rules and apply these rules to support the verification and validation (V&V) of metadata for multi-vendor test and instrumentation systems.

INTRODUCTION

There currently exist generic standards for representing rules for verification and validation (V&V). However, rules written in these standards are not accessible to the typical instrumentation engineer-the rules are often described in a computer programming language or a language meant to be processed by a machine. Few authors and users in the T&E community have the time, inclination or interest in learning these languages. To address this, the VIVALDI (Verification and Validation of Metadata for Test & Evaluation) portal captures rules in an intuitive, domain specific, unambiguous controlled natural language (CNL). A CNL allows those that know the rules most intimately-the engineers-to define the rules without having to learn or even know about the underlying rule standard.

The VIVALDI portal is an implementation of a CNL-based method in which users capture and evaluate V&V rules within and across metadata instance documents. The method uses a CNL syntax for the T&E metadata V&V rule set in order to abstract the highly technical rule languages to a more natural T&E form. As a result, the users 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. 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.

Our paper outlines provides an overview of the VIVALDI portal. Key capabilities include: an extensible web-based architecture for capturing and executing rules; catalogs of TMATS [1], MDL [2] and IHAL [3] [4] rules captured in CNL; translators from CNL to target rule languages; and rule execution on open source rule engines.

BRIEF TOUR OF THE PORTAL

This section provides a brief tour of the VIVALDI portal.

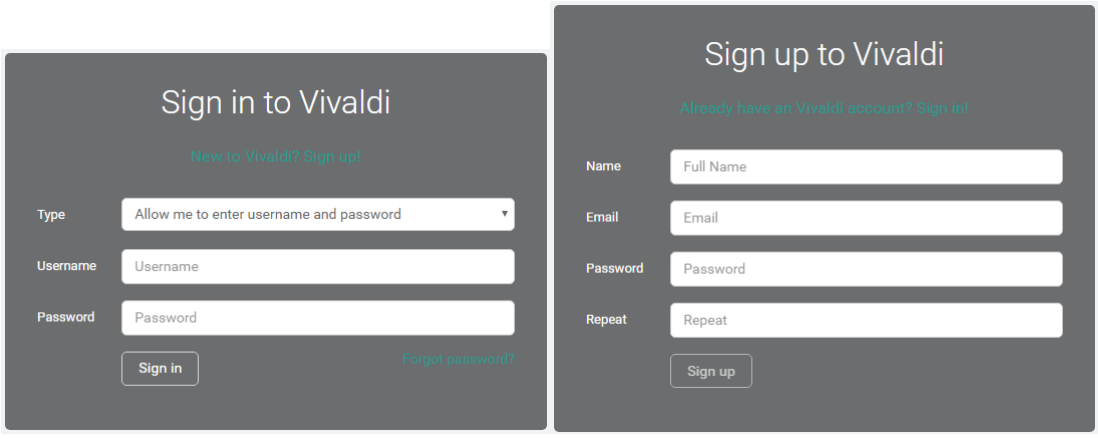

Figure 1 – Authentication

User Authentication

Users of the VIVALDI portal will need to have a login. Figure 1 shows the screens for authentication. At the left of the figure is the login screen where the user enters an email and

password. At the right of the figure is the registration screen for users that do not have a login. To get authentication credentials, the user must provide their name, email and a password.

Dashboard

The VIVALDI portal dashboard is the landing page for the VIVALDI portal. Figure 2 shows the dashboard. After login, the user will be redirected to this page. The central area of the dashboard will include a summary of activity in the VIVALDI portal, to include: number of rules, number of instance documents, active users, jobs processed, supported standards, etc.

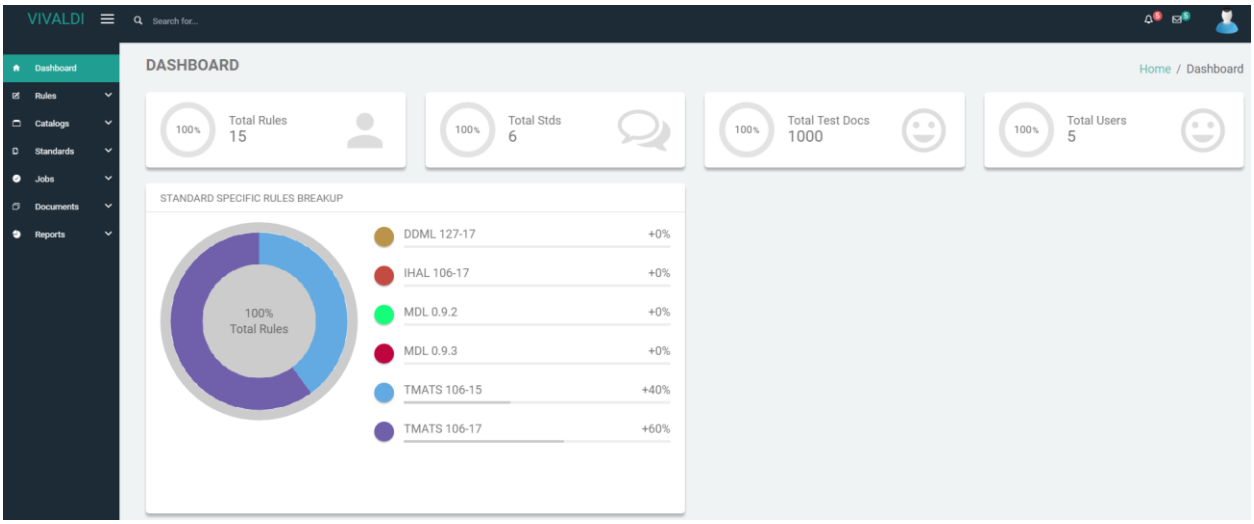

Figure 2 – VIVALDI Portal Dashboard

The left side of the dashboard is where the user accesses the VIVALDI portal capabilities.

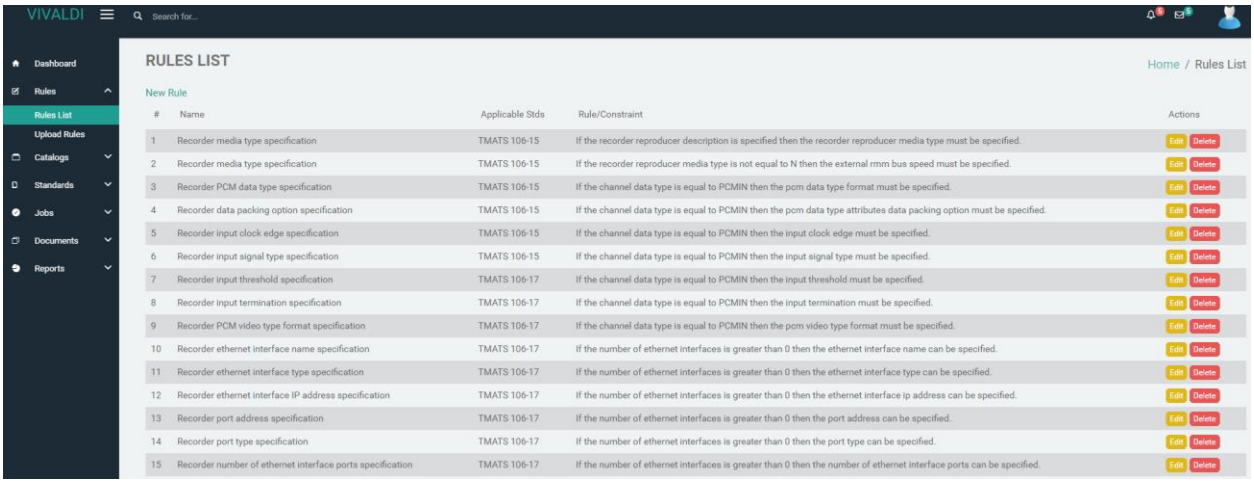

Figure 3 – Registered Rules

Rule Management

At the heart of the VIVALDI portal capability is the management (create, read, update, delete) of rules. The notion of a “registered rule” is a rule that can be used by any authenticated user of the VIVALDI portal system. The VIVALDI portal will allow the user to view all the registered rules. Figure 3 shows a screen of the registered rules. Each row includes metadata about the rule, including a name, description, the standard(s) that the rule applies to and the machine-

understandable representation of the rule (this can be hidden based on user profile). Users can edit or delete a rule by selecting the appropriate button.

Figure 4 shows a screen for creating a rule. The rule author provides a name, and the standard(s) to which the rule applies. The user authors the rule using the VIVALDI portal predictive editor. This editor ensures that the rule can be unambiguously understood by both a human and machine. The editor suggests possible rule completions.

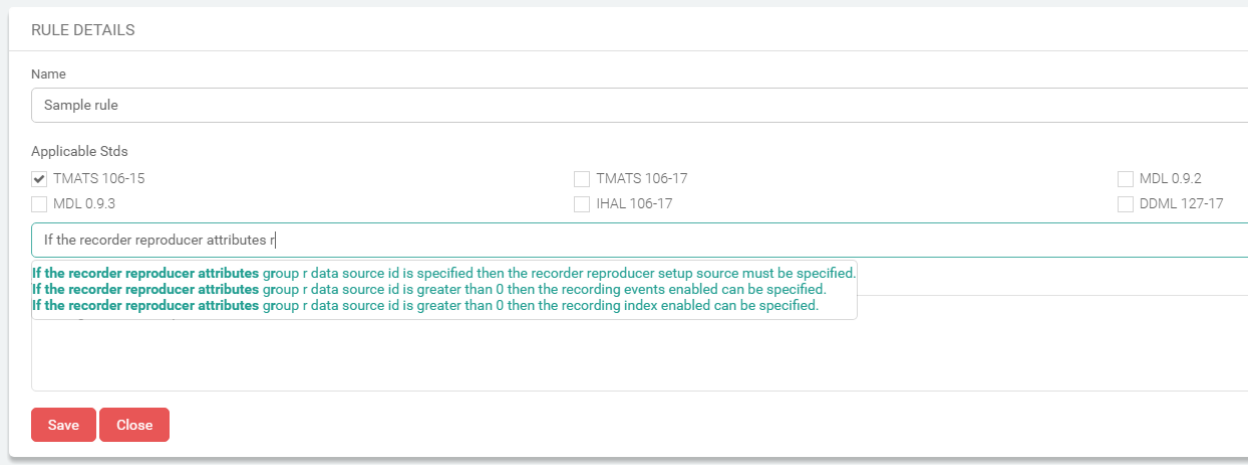

Figure 4 – Rule Creation

Rule Catalog Management

Rules are organized into catalogs so that they can be sent to a validation engine to validate an instance document. Figure 5 shows a screen that lists the rule catalogs. Each row includes metadata about the rule catalog, including a name and description. Users can edit or delete a rule catalog by selecting the appropriate button.

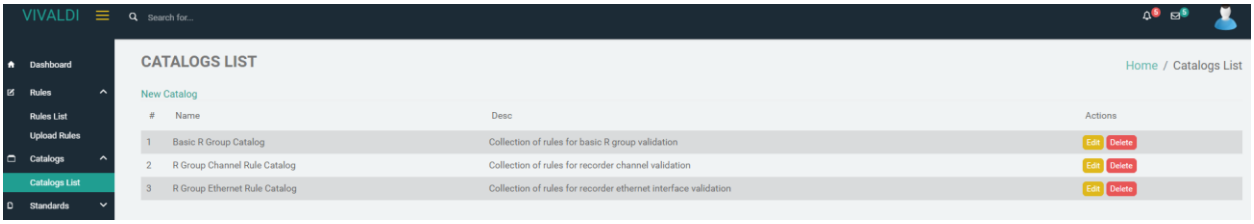

Figure 5 – Rule Catalogs

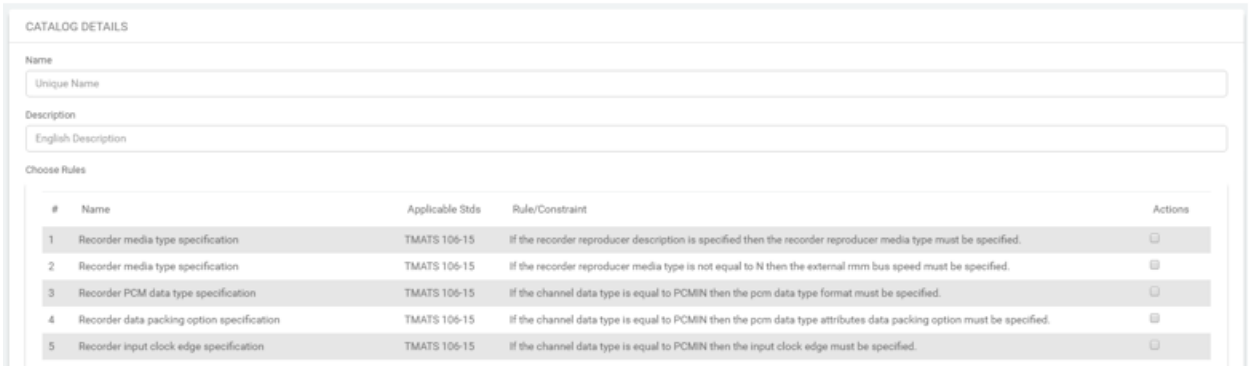

Figure 6 – Create Rule Catalog

Figure 6 shows the screen to create a rule catalog. The rule catalog author provides a name and description (creator, creation / modification dates are automatically assigned) and selects the rules to be included in the catalog.

VIVALDI PORTAL ARCHITECTURE

Figure 7 shows the current VIVALDI portal architecture. The VIVALDI portal is implemented using the Node.js® framework [5]. Basic authentication is provided using a datastore managed by the VIVALDI portal application. If necessary in the future, we will implement domain authentication using the appropriate framework. The connection from the user to the Node.js® server and the authentication databases will be encrypted. The VIVALDI portal on the Node.js® server connects to the supporting capabilities via REST APIs over encrypted channels. The REST APIs will be standardized as described later in this paper.

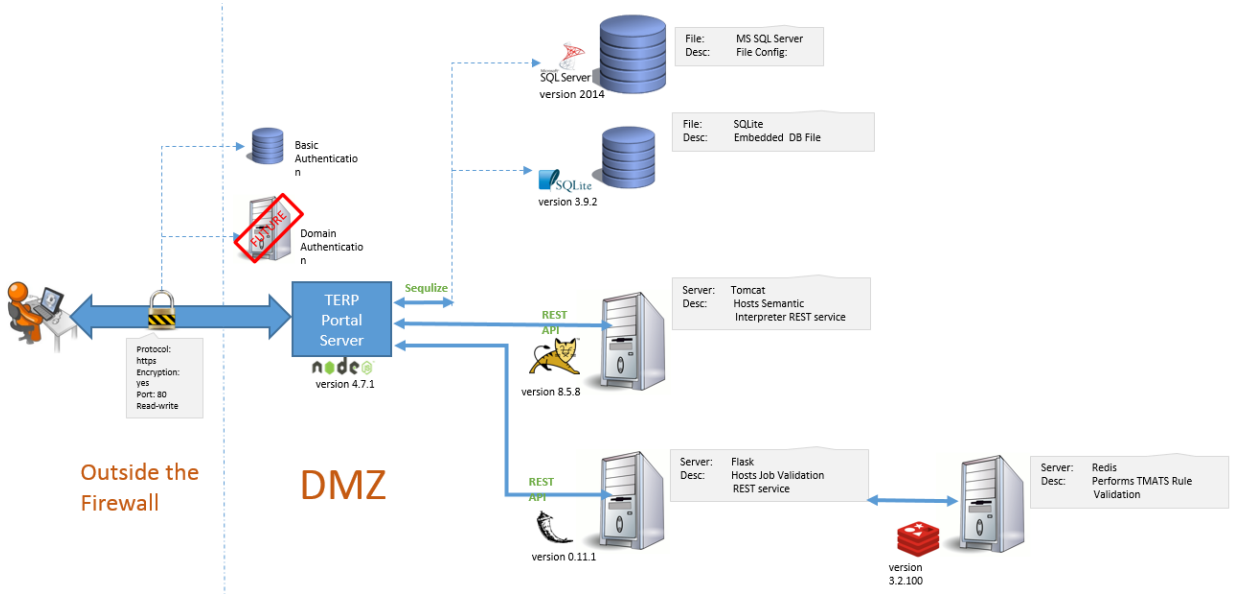

Figure 7 – VIVALDI Portal Architecture

The backend capabilities include a SQL database for storing rules, rule catalogs, jobs, etc., a Tomcat server [6] that hosts the language processing capabilities, a Flask server for processing jobs and a Redis server [7] for performing TMATS validation.

VIVALDI PORTAL LANGUAGE TECHNOLOGY

In this section, we outline the major components for the VIVALDI portal rule processing.

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 very difficult to write a statement without auto-completion or predictive editor support, making the rule authoring 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.

One of the most obvious ways to support writing restricted languages like CNL is error highlighting. While they author rules, the interface parses the text and validates the text against the CNL rules. If the validation fails, the interface will try to identify the cause of the error and provide suggestions for fixing the error.

VIVALDI Portal Grammar: The VIVALDI Portal Grammar is a controlled natural language (CNL) grammar that is an extension of the ACE "Attempto Control English" controlled natural language grammar [8]. These extensions construct a new language specific to the problem domain that ensures that the underlying grammar does not contain syntactic ambiguities. The greatest challenge in processing natural language with computing systems is the ambiguity that exists in both the grammar and semantics. The grammar makes the resulting system scalable in the sense that as new rules are added that conform to the CNL they do not collide with existing statements.

CNL Rules: These rules represent the input statements the system must support and are representable within the CNL grammar.

Predictive Editor: The predictive editor is the GUI-based input device that is used for users to author rules and store them in the system. The predictive editor recognizes the approved structures of the CNL and is able to look-ahead with in the defined grammar. The author gets immediate feedback while the text is being written and helps prevent the author from entering sentences that are not allowed in the CNL grammar. While writing, the author can request rule suggestions. The suggestions provided to the user are context aware: i.e., the editor provides suggestions based on the cursor positions and the applicable grammar rule(s).

Free-Text Input: The free-text input capability takes a rule typed in by a user and maps that statement to the best controlled natural language statement based on how the system was configured. This allows for the system to take in ambiguous statements and map them to the "best" known rule that has been vetted by the system designers. The user has the ability to accept or reject the mapping. Once accepted, the user has confidence that the accepted sentence is a CNL sentence that can be processed by the system.

Semantic Interpreter: The semantic interpreter takes a CNL rule as input and creates a semantic representation with the help of a small CNL ontology. The semantic interpretation process incorporates a small ontology that seeks to make minimal commitments to external ontologies. The goal of the semantic representation is to provide a representation that is well suited to augmentation with external ontologies while at the same time carrying enough meaning to be immediately useful. Users of the system can take the semantic interpretation directly and drive other computing systems or components.

CNL Ontology: The CNL ontology is a light-weight ontology that contains some basic semantics that supports pulling semantic information directly from a CNL syntactic structure. The goal is to not tie the representation to external ontologies so that the final semantic interpretation is flexible and easily mapped or used to drive richer semantic representations.

Semantic Interpretation: This is the final output of the VIVALDI portal language components. The current form of this interpretation is a set of triples in an SVO (subject-verb-object) format. The intent behind the semantic interpretation is to provide a representation that contains as much semantics as possible without the use of external ontologies.

The following statements are characteristic of rules that VIVALDI portal can support:

- The measurement identified 0x0100 has a data rate of 480 pps.
- The measurement identified by 0x0200 has a sample rate of 48 Hz.
- The measurement identified by 0x0300 has data length of 2 bytes.
- The measurement identified by 0x0600 has resolution of 5 percent.

Figure 8 shows some controlled natural language representations and semantic representations for the rules above. The web interface supports entities; which are selectable from a dropdown or other control. The semantic interpretation is a representation of the rule in a triple format.

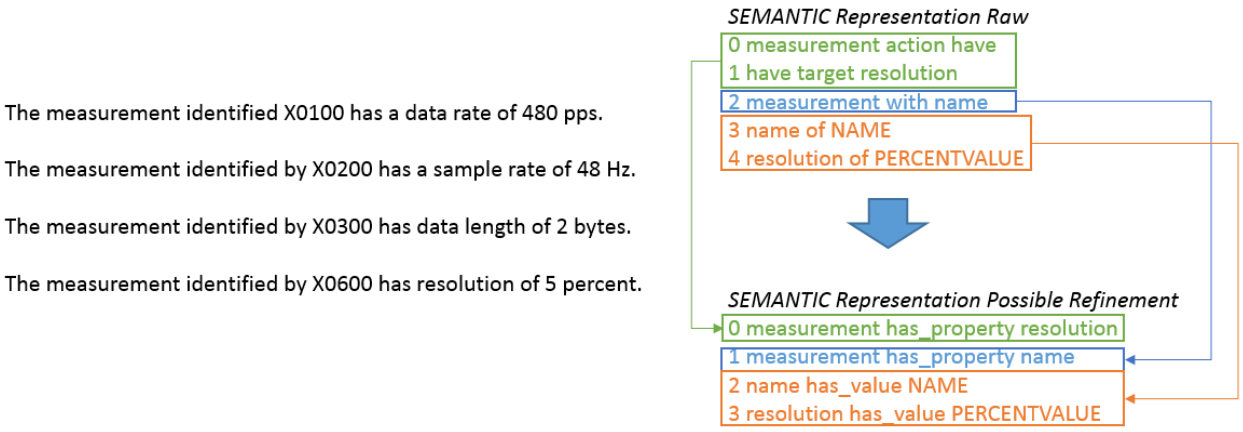

Figure 8 - Semantic Representations for Statements

STANDARDIZATION

KBSI proposes to support the standardization of VIVALDI portal capabilities with RCC standardization committees so that the T&E community end users can take advantage of its features. Figure 9 overviews the expected VIVALDI portal standards (shown in green). The VIVALDI portal and services are shown as well as a vendor configuration service.

We expect at least four standards:

- The rule catalog contains a collection of rules that are useful to the T&E community and can be applied across T&E missions.
- The CNL standard as the CNL grammar to support rule capture.
- The API as an interface standard for utilizing the portal capabilities within a broader computer program infrastructure.
- A method for capturing rules in CNL, translating the rules into a target language, and executing the rules against XML instance documents.

Initial thoughts of an API would include the following:

- The validate() method takes a metadata instance document and a collection of CNL rules as input and returns XML validation report as output; the validation report documents the rules that are satisfied and those that are violated.
- The translate() method takes a CNL rule set and target rule language as input and returns the target rule instances translated from the CNL as output.

- The optimize() method takes a metadata instance document and a CNL rule set as input and returns an optimized metadata instance document as output; the CNL grammar would be extended to include optimization criteria to support this method.
- The deconflictRules() method takes a CNL rule set as input and returns a de-conflicted[1] CNL rule set as output.
- The deleteRule() method takes a CNL rule and a rule catalog as input and removes the rule from the catalog.
- The addRule() method takes a CNL rule and a rule catalog as input and adds the rule to the catalog.
- The updateRule() method takes a CNL rule and a rule catalog as input and updates the rule in the catalog.

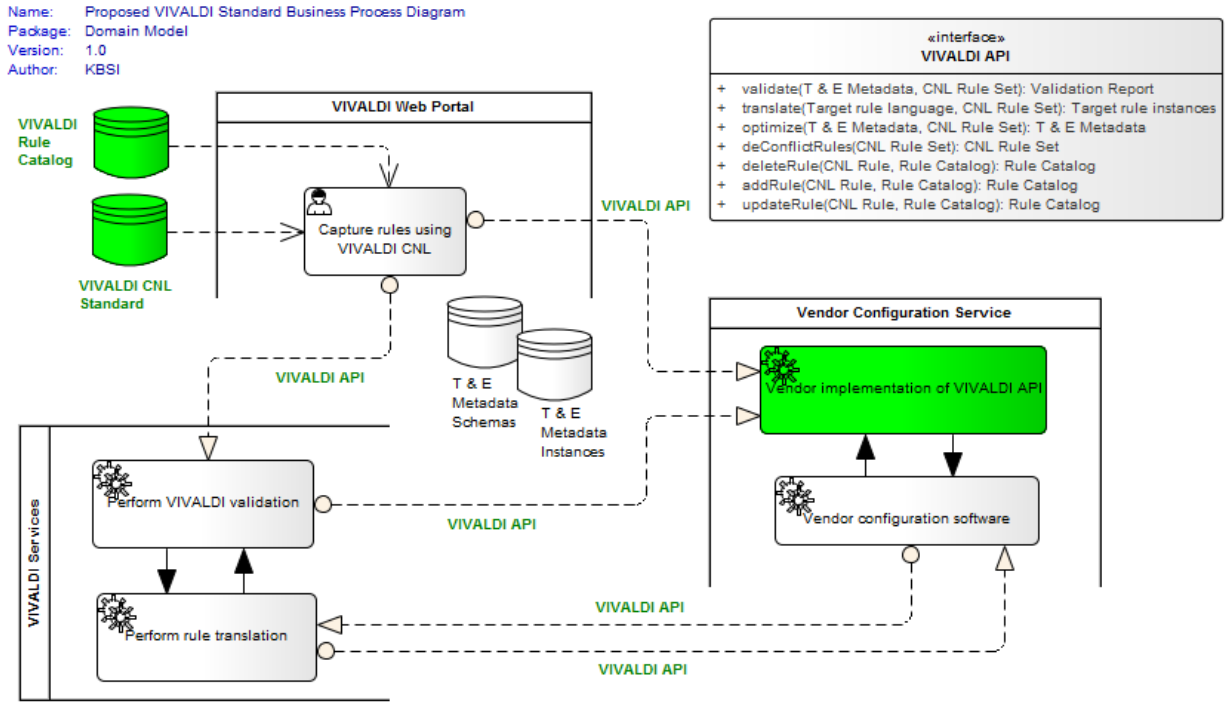

Figure 9 – Overview of the Expected Standardization

As shown in Figure 9, the standards are used as follows:

- During the rules capture step, the user selects rules from the standard rule catalog or authors new rules using the standard CNL language.
- Once the CNL rules have been captured, there are two possible scenarios:
  - The user validates a metadata instance document against the rules using the VIVALDI portal services.
  - The user validates a metadata instance document against the rules using a third-party vendor implementation of the API. In this scenario, the vendor software can

[1] Two rules conflict if there is no way that both rules can be satisfied.

access VIVALDI portal capability for CNL translation, or validation using the API.

CONCLUSIONS

Verification and validation (V&V) of metadata and systems design are critical functions in systems engineering because they help ensure that requirements are realizable, that designs are correct and consistent, that designs satisfy all requirements, and help reduce costs by optimizing over design variables. In short, V&V reduces lifecycle costs by ensuring correctness in design phase. The VIVALDI portal includes: (i) 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, (ii) CNL rules can be transformed into any target rule language, (iii) there is no translation from XML to another format for rule execution, and (iv) the approach is based on open sources and established standards.

These benefits will apply directly to instrumentation design for T&E ranges and to the entire DoD T&E community by standardization of various technology components. The proposed standards enable interoperability among T&E users and vendor configuration software at the level of English-language statements of the intended rule. Unambiguous statements of the rule semantics mean that the users can clearly communicate rules in nothing more than a text editor. Translators from the CNL to a target rule language enable V&V of the rules at the machine level using the right target language and rule engine.

REFERENCES

[1] Range Commander's Council Telemetry Group, "Chapter 9 Telemetry Attributes Transfer Standard," in *Telemetry Standards, IRIG Standard 106-13 (Part 1)*, US Army White Sands Missile Range, NM, Secretariat, Range Commander's Council, 2013.

[2] M. Moore, "Metadata Description Language: The iNET Metadata Standard Language," in *Proc. International Telemetering Conf.*, Las Vegas, NV, 2009.

[3] J. Hamilton, R. Fernandes, P. Koola and C. H. Jones, "An Instrumentation Hardware Abstraction Language," in *42nd International Telemetering Conference*, San Diego, CA, 2006.

[4] J. Hamilton, R. Fernandes, M. Graul and C. Jones, "Extensions to the Instrumentation Hardware Abstraction Language (IHAL)," in *44th International Telemetering Conference*, San Diego, CA, 2008.

[5] "Node JS," Node.js Foundation, [Online]. Available: https://nodejs.org/en/. [Accessed 12 May 2017].

[6] "Apache Tomcat," The Apache Software Foundation, [Online]. Available: http://tomcat.apache.org/. [Accessed 12 May 2017].

[7] "redis," redislabs, [Online]. Available: https://redis.io/. [Accessed 12 May 2017].

[8] N. E. Fuchs and R. Schwitter, "Attempto Controlled English (ACE)," Katholieke Universiteit Leuven, 1996.