Building Multi-Vendor T&E Systems in iNET

JAKUB MOSKAL
MIECZYSŁAW “MITCH” KOKAR

BEN ABBOTT
AUSTIN WHITTINGTON

JON MORGAN

MAY 15, 2019

[DISTRIBUTION STATEMENT A. Approved for public release; Distribution is unlimited 412TW-PA-19229]
Context: Multi-Vendor Systems

- Benefits:
  - Engineers can choose the best device for the task, regardless of the vendor
  - Vendors can compete in terms of performance and cost
A single vendor alone cannot anticipate:

- Dependencies on the settings from devices from other vendors, for instance:
  - Timing constraints imposed by one device can affect the timing of another one
- Systemic constraints that are specific to a particular customer, for instance:
  - Limitations on the total bandwidth, weight, or power usage
  - Globally unique properties, e.g. Role ID
  - Values that can be reused only within a specific scope, e.g. RF MAC cannot be reused across different Radio Access Networks

Access to device constraints is needed to effectively build multi-vendor systems.

However, today, device constraints are “hidden”, hard-coded in the vendor configuration software (e.g. in Java).
Multi-Vendor Systems Today

- Vendor-specific configuration software
  - “Hides” the constraints
  - Requires training for each software suite
  - Keeping up with standard updates is a burden on the vendors
- System-level validation requires a dedicated software
- Many iterations required to make the configuration valid
  - Last minute changes are hindered
Multi-vendor systems are expensive, impractical
In practice, they are often avoided via vendor lock-in
  - Capability, performance, and cost of T&E devices may need to be compromised
Multi-Vendor Systems: Tomorrow

- Standard constraint language
- Standard validation protocol
- Vendor-neutral, third-party configuration software and constraint engines
- Single validation process for device and system constraints
- Users only need to use one software suite
- Vendors no longer need to maintain configuration software
- Multi-vendor systems become practical
  - New market opportunities
- New use cases possible:
  - Deconfliction
  - Automatic configuration & optimization based on measurement requirements
iNET System Manager

- Instrumentation Engineer
  - Provides User/System Constraints
  - Configures Devices
- Standard Constraints
- Constraint Validation Engine
  - Provides Device Constraints
  - Validation Protocol
- iNET System Manager
  - RCC Standards
  - Configures Devices
- Multi-vendor T&E System
  - Vendor 1 Device A
  - Vendor 1 Device B
  - Vendor 2 Device C
  - Vendor N Device
- Device A/B/C/.../N Vendor
  - Manufactures Devices
iNET System Manager
TACL: A Proposal for a T&E Constraints Language
W3C SHACL – Shapes Constraint Language

- A language for validating semantic graphs against a set of conditions, called **Shapes**

**Diagram:**
- Defines a **TARGET** subset of the graph
- Shape (Constraint)
- Defines **FILTERS** to determine if the targets meet specific conditions

**Process:**
- **Input** Semantic Graph
- **Target** determines the Focus Nodes
- Invalid Nodes are filtered out

**Note:**
- **TARGETS** and **FILTERS** can use **FUNCTIONS** in their definitions
Processing Shapes with a SHACL Engine

- Shape (Constraint)
- Semantic Graph
- SHACL Engine
- Validation Report
- Semantic Lifter
- Other...
- { JSON }
- XML Tree
Shape Example

Names of the Measurements in the TMATS C Group

must not contain the symbol ‘@’

ShapeExample

TARGET

Measurements in TMATS C Group

FILTER

Name must not contain ‘@’

ShapeExample (SHACL)

TARGET

sh:targetClass C:Measurement

FILTER

sh:property[
  sh:path Common:hasName
  sh:pattern "^[^@]*$"
]
Proposal – standardize **TACL**:
- MDL/TMATS Ontology (semantic graph)
- Common T&E Extensions for SHACL:
  - Targets
  - Filters
  - Functions

ITC’18 Best Standards Paper Award
- “Introducing TACL – A Proposal for a New Standard T&E Constraint Language”
For all devices in a network with a TmNS Master Clock, the excitation source for each module **must** be external.

**Constraint Example (TACL)**

```
sh:target [ 
  a tacl:DeviceOnNetworkWith;
  tacl:appKind mdl:TmNSMasterClock ]
```

```
tacl:allModulesWithExcitationSource "External".
```
TACL – Benefits

- TACL hides the nitty-gritty details under the covers
- It introduces a layer of separation from the data model
- Constraints written with TACL are succinct and resilient to schema changes
- It can be processed with SHACL engines available today
- As an extension of a W3C standard, TACL benefits from:
  - W3C Endorsement
    - Long-lasting support
    - Guarantee of quality
  - Access to a growing user group
    - Discussion groups
    - Forums
    - Active development efforts
  - Access to open-source tools
    - Validators
    - GUI’s
Constraint Validation Engine

- Standard Constraints
  - Provides Device Constraints
  - Provides User/System Constraints

- Device A/B/C/.../N Vendor
  - Manufactures Devices

- Constraint Validation Engine
  - Validation Protocol
  - RCC Standards

- Multi-vendor T&E System
  - Vendor 1 Device A
  - Vendor 1 Device B
  - Vendor 2 Device C
  - Vendor N Device

- Instrumentation Engineer
  - Configures Devices
xVISor is a reference implementation of a TACL Engine
- Supports TACL validation of MDL 1.0 and TMATS/XML 106-17
- Developed in Java
Architectural Challenge

• Premise:
  o The users expect almost-instant feedback if any constraint is violated
  o Constraints are developed against MDL graph (ontology)
  o System Manager uses proprietary data model

• Challenge:
  o How to avoid the “Validate now” button?
Integration with iNET System Manager

Solution:
- System Manager pushes configuration state changes via notifications
- Validation Engine performs the same changes to its internal data model and executes the validation

This protocol needs to be standardized!
ITC’18 Demo: iNET System Manager and xVİSor
Tool Support: Constraint Manager

**Rationale:**
- In our opinion, TACL should be the standard constraint language
- Even though TACL expressions are rather succinct and readable, even for complex constraints, they still require a form of coding
- We developed a **Constraint Manager** – proof of concept tool that allows creating TACL constraints using widgets, and without coding
Constraints Manager (1/2)

Constraint: **Excitation Source Restriction (MDL)**

For all devices in a network with a TmNS Master Clock, the excitation source for each module must be external.

**TARGET** Which data is constrained?

- All instances of `mdl:NetworkNode`
  - Where: 
    - Add Characteristic
      - It is on a network with a Master Clock

**FILTERS** How is it constrained?

- Add Filter
  - The excitation source for all modules must be External
  - Violation message: specific to this filter
Constraint Manager (2/2)

Select a property path

```
mdl:hasNetwork \ mdl:partOf \ mdl:hasRANConfiguration
```

Diagram:

- Network Domain
- Test Mission
- RAN Configuration
- Network

Properties:
- has network domain
- has test mission
- has center frequency (Hz)
- is TmNS complete
- has network node
- has RAN Configuration
- has name
- has network node

Message: GIPN cannot be longer than 16 characters
Standards Create New Opportunities

• To be standardized:
  - TACL, the constraint language
  - Constraint Validation Protocol

• Once standardized (or in the process of), new opportunities open up:
  - Automatic deconfliction: conflict between constraints is detected, a tool suggests how to mitigate it without causing a ripple effect with other constraints
    - E.g. if you lower the gain to X, the configuration will be valid
  - Generation of a configuration based on:
    - Measurement Requirements
    - Device inventory
    - Database of past and present configurations
Thank You!

VISTology, Inc.
http://www.vistology.com
+1 (508) 788-5088

Southwest Research Institute
http://www.swri.org
+1 (210) 684-5111

JT4, LLC
Edwards AFB
Backup Slides
About VISTology

- Past and present R&D DoD contracts

- Professional collaborations
  - BAE, BBN, JHU APL, Lockheed Martin, Northeastern University, Raytheon, Referentia Systems, Southwest Research Institute, Vulcan, W3C, Wireless Innovation Forum

- Areas of expertise
  - Information Integration and Fusion,
  - Situation Awareness,
  - Formal Reasoning Systems,
  - Artificial Intelligence,
  - Cognitive Radio,
  - Cognitive EW,
  - System Engineering,
  - Ontology Engineering

- Products and Services
  - BaseVISor – highly efficient inference engine
  - ConsVISor – consistency checker for ontologies
  - DeVISor – future-proof matchmaking middleware
  - HADRian – next generation COP system for HA/DR
VIStology (lead) and SwRI (sub) are funded via a Phase II SBIR (AF141-227)

- Full description can be found here (look for “AF141-227”):

**Objective:**

- Develop methods for capturing semantic rules of Test & Evaluation metadata and automate validation of related XML instances.

**Phase II objectives:**

- Refine and expand the rule structures and the mechanisms for capturing the rules developed in Phase I.
- Work towards standardizing these structures and the non-software based mechanisms (as a non-proprietary standard).
- Develop software to automate the validation process and, to the extent possible, the deconfliction and optimization processes.

**Period of performance:**

- Phase I: 07/2014 – 02/2015
- Phase II: 09/2016 – 12/2018, no-cost extension until 12/2019
SHACL Core

- Sufficient for simple constraints (similar to XSD)
- Supported by every SHACL-compliant engine
- Examples:
  - Checking cardinality, e.g. measurements must have a name
  - Checking value range, e.g. modules cannot occupy more than 2 slots
  - String length, regex, e.g. network node names cannot use the ‘@’ character
  - Comparison of values of different properties, e.g. TmNSApp’s last validation date/time must be greater than its last configuration date/time
From T&E XML to a Semantic Graph

XML/XSD

Semantic Graph (Ontology)
Extending SHACL

- Arguably most powerful feature of SHACL is its extensibility
- SHACL Core can be extended to a specific domain by defining custom:
  - Targets
  - Filters
  - Functions
- Why extend?
  - Encapsulate complex constraints in simple expressions
  - Operate using familiar domain terms, not generic language expressions
  - Increase reusability → improve efficiency and maintenance of the constraints
  - Build more resilience from XML schema changes

*Any SHACL constraints are validated with a standard W3C SHACL engine. They do not require changes in the engine.*
• SHACL Core primarily supports two kinds of targets:
  o Single node, e.g. a concrete Device Module
  o A class of nodes, e.g. all TmNSApps

• TACL targets are more specific:
  o Devices produced by vendor X
  o NetworkNodes with a TmNSRadio whose power level is greater than X dBm
  o TmNSApps on NetworkNodes that have a module with a network interface whose DHCP is disabled
TACL Filters

- **SHACL Core contains filters that are entirely domain-independent (~ XSD):**
  - Property X must have a value that is >= Y
  - The length of property X must be less than Y

- **TACL filters are more specific:**
  - The total weight of all sub-elements may not exceed X
  - A coefficient order cannot be higher than X
  - A value of a given property must be unique in the entire configuration
Redis-based integration with partial change notifications
- Very minimal effort to expose the changes to xVISor
- Good performance (according to the audience)
- TACL constraints can be changed at runtime
- Validation can be disabled (useful when building from scratch)