CURRENT AND FUTURE DEVELOPMENTS IN FLIGHT TEST CONFIGURATION TECHNIQUES

Austin Whittington[1], Hakima Ibaroudene[1], Ben Abbott[1], Di Yao[2], Joseph Hite[2], Theodore Bapty[2], Jakub Moskal[3], and Michael Neumann[4]

[1]Southwest Research Institute, San Antonio, TX
[2]Vanderbilt University, Nashville, TN
[3]VIStology, Inc., Framingham, MA
[4]The Boeing Company, Seattle, WA

ABSTRACT

As technologies like network-based telemetry and standardized configuration languages begin to see wider adoption within the flight test community, new techniques exploring the new possibilities they provide are also developed. This paper reviews a subset of these techniques, including successful use in commercial flight test, focusing on the concepts of constraints and their application in the field, specifically their use in helping users to create correct-by-construction configurations. We then explore ongoing efforts with the Air Force and DARPA to extend these techniques into constraint satisfaction and real-time adaptation, providing the ability to create and adapt configurations to match (possibly changing) test requirements.

INTRODUCTION

New technologies including network-based telemetry and standardized configuration languages are slowly trickling into use in the flight test community. The IRIG 106-17 standards[1] have been been released, formalizing MDL v1.0.0 for use by both government and commercial flight test programs. The Boeing Company's Modular Instrumentation Setup Tool (MIST) has been developed, making use of a dialect of MDL to provide a vendor-agnostic solution for their use[2]. Techniques for semantic validation of Test and Evaluation XML data are being developed, building higher level semantic rules on top of the basic XML syntax of the language[3].

In parallel, and originally unrelated to these standardization and expansion efforts, DARPA has been running its Building Resource Adaptive Software Systems (BRASS) program, whose goal is "to realize foundational advances in the design and implementation of long-lived, survivable and complex software systems"[4]. Teams of performers tackle varied domains with specialized, cutting-edge adaptation techniques with a goal of "making software that will last for a hundred years".

During phase 2 of the program, SwRI (funded by DARPA and AFRL) was chosen to present a third-party challenge problem of military significance: flight test. SwRI's role is to develop and frame a series of scenarios, complete with modeling tools and accessibility considerations, to allow these non-domain experts to apply their specific techniques to flight test, under the umbrella of the "Adaptive Constraint Satisfaction in Flight Test Configuration" (SIFT) team. This brings the research advances in adaptive software to this new domain, benefiting the program by providing a useful transition direction, and opening new possibilities for adaptable systems in flight test.

This paper first looks at the successful commercial use of these configuration techniques, in Boeing's MIST. Near-future adaptation techniques using constraint satisfaction and offline reasoning are discussed, with the language advances that make those possible. Finally, this paper provides a look into the future of how flight test is being integrated with the BRASS program, and the new capabilities this may provide to prospective programs.

SUCCESSFUL COMMERCIAL USE

The Modular Instrumentation Setup Tool (MIST) was originally implemented by the Boeing Company to configure data acquisition devices from Zodiac Data Systems for the 737-MAX flight test program. MIST resides on a server and accesses data from a Boeing database. A new version has since been made available to allow the 777-X flight test program to use MIST onboard test airplanes, in addition to providing additional configuration capabilities on the ground. A new version to be released later in 2018 will allow users to work completely offline, onboard and on their laptops, modifying MDL files and dynamically configuring MDL devices without being connected to the larger Boeing database. These capabilities will also provide a path for MIST to be integrated into the Next Generation Flight Test System from Boeing Test and Evaluation.

Architecturally, MIST was designed to use MDL as a standard interface with vendor hardware and software, breaking the previous dependence on proprietary formats. MIST also utilizes externalized Constraints written in XPath, eliminating the need to hard-code configuration rules for each specific device type on a test airplane. When any of these XPath Constraints are violated, errors are dynamically displayed to users using XForms, HTML5 and Angular JS.

During the 737-MAX flight test program, MIST provided its greatest benefits to the user community through the use of Constraints and Instant Validation. User input and configuration dependencies are instantly evaluated by the Orbeon (Open Source) Constraints Engine using Vendor, System and User Constraints. Boeing Engineers provided their own set of User Constraints to complement or additionally restrain Vendor and System Constraints. These Constraints facilitated quick changes and adjustments during the flight test program, as they reside outside the code in XML formatted files that can be modified independently of the source code for the tool.

Moving forward with MIST as an integral part of the Boeing Next Generation Flight Test System will present opportunities for integrating new MDL devices and adopting the official MDL 1.0 standard (MIST currently uses an older version to support vendor devices). In addition, a more generic architecture can be created that will allow new tools to be developed on an MDL and

Constraints based framework, where business rules will reside outside the application code and can be added and modified by Flight Test Engineers using a potential new set of tools available from SwRI, VIStology and other research companies.

CONSTRAINT SATISFACTION

Standardization of new configuration languages such as MDL opened up new opportunities that were not even thought of before. Configuration management software can now access all parameters: not only those specified by the standard, but also those that are vendor-specific, which was made possible due to flexible extensibility mechanisms embedded in the language. With all parameters uniformly expressed in one language, it is now possible to specify constraints that pertain to all stakeholders of the configuration process in one place: the instrumentation engineers building the configuration, the vendors supplying the hardware, and the standardization organizations, such as RCC. It is now feasible to significantly reduce the engineering time required to build a configuration by performing a constraints validation that considers all parameters and constraints at once. In addition, new types of tools can now be developed to perform some form of constraint satisfaction in order to (semi-)automatically build configurations that meet the test engineer's measurement requirements and satisfy all of the constraints.

While the configuration languages are sufficient to represent all parameters, they are not equipped with rich enough semantics to express all constraints. In particular, the widely used XML schema is not expressive enough to capture inter-element, or inter-attribute constraints, which are actually quite common. Without the standard language to express all constraints, proprietary vendor-specific solutions are developed. Thus, to build a configuration with n different vendor devices, n different configuration tools must be used to validate each setup. Furthermore, additional software is needed to validate system-level constraints that do not pertain to any specific device. This is arguably the main reason why cross-vendor configurations are rather avoided in practice.

The T&E community is well-aware of the lack of a standard T&E constraint language and has been actively searching for a solution to this problem for the past several years. Given the fact the community is relatively small, and the problem of putting constraints on XML-based documents is domain-independent, there has been a lot of interest in reusing existing technologies to expedite the standardization process. The main candidate that has been considered so far is W3C XPath, which is a *query* language for XML documents. While it has much richer semantics than XML, it is closely-coupled with the XML trees, which leads to major challenges with maintenance required to stay up to date with the frequent standard updates.

VIStology and SwRI are proposing TACL, a T&E extension to the W3C Shape Constraints Language (SHACL) as the new standard T&E constraint language. SHACL is a product of a long effort led by a large community of enterprise stakeholders, which faced a similar problem to the T&E community: lack of a standard constraint language to validate data originating at disparate sources. SHACL targets Resource Description Framework (RDF) [5] documents, widely used to represent Linked Data [6] on the Web, but it may as well be applied to other data sources after a straightforward conversion. The key features that lead to development of TACL are 1) the fact that

SHACL is an official W3C standard and will soon be followed by an ecosystem of software tools that support it, 2) all schema constraints and those exposed publicly by vendors and users can be expressed in SHACL, 3) it can be integrated with OWL [7] ontologies and utilize the power of OWL inference to introduce a high-level of abstraction in constraint definitions.

TACL bears a similar relationship to SHACL as MDL does to XML — a customization of a standard language to a specific domain that can be processed with the tools developed for the core language. Just like MDL or TMATS (Telemetry Attributes Transfer Standard) XML documents can be processed with domain-agnostic XML tools (editors, schema validators, query processors), TACL can be processed with those developed for SHACL. It can be used for formulating constraints on configurations represented in MDL and TMATS, independently of any configuration software, introduces high-level components that help to form constraints close to the user's intent, and are less concerned with the low-level syntax details. It exhibits much better resilience to changes in the XML schemas than the languages that refer directly to the XML trees.

VIStology's xVISor, a proof of concept TACL engine, has been successfully developed and applied to MDL/TMATS configurations. The next step is to develop solutions that leverage this technology further and automatically build configurations that satisfy all constraints.

In the BRASS realm, there are several classes of problems that can be solved or addressed by the constraint satisfaction type of approach. More specifically, these are problems for which all the parameters and metrics are known and static, and answers can be verified without the information of a particular circumstance.

One class of problems matching these criteria is that of configuration requirements analysis. An engineer may have a set of requirements for her acquisition system, defining the measurements needed, data rates, and other information. In the theoretical world where a BRASS-powered system has all the capabilities that are foreseen as possible outcomes of the program, there are a few options open to her.

The engineer can validate that the acquisition system she assembled from available parts is capable of satisfying the requirements, based on the capabilities of the individual components and the system topology. She can get a measure of the redundancies of the system in the face of failures (e.g. how many sources there are of a particular piece of safety-critical information). She can opt to have the BRASS-powered system build configuration possibilities for her, drawing from the available devices and equipment available for use at that facility, with a goal of the minimum hardware required to satisfy the requirements.

One particular problem that has been explored is the question of what to do when hardware is in need of replacement. A scenario may present where, in a pre-flight situation, a Data Acquisition Unit (DAU) is broken and needs to be replaced. Inventory stock for that DAU is low, so the risk of depriving other concurrent programs which rely on that part is high. The decision could be made to order a new DAU from the vendor, which adds a purchase cost and a significant lead time during which no more tests can be run. Alternatively, there may be another type of DAU, even from a different vendor, which has similar capabilities and is in stock at the facility.

Using requirements analysis techniques, it is possible that the BRASS-powered system can prove

that the secondary DAU will satisfy the original requirements that the DAU being replaced was responsible for (regardless of the actual sampling rates and other parameters used in operation). This would save money and time for the program, and would avoid putting other programs at risk. These BRASS adaptations can explore beyond what an engineer would be tasked to do, because there isn't a minimum feasibility that would have to be met. In other words, it doesn't have to seem likely for BRASS to explore the possibility. These types of exploratory optimizations and adaptations are generally only done by humans with significant domain experience and "intuition" to help guide them, but a BRASS-powered system can explore further and faster with more formal satisfaction of the requirements.

REAL-TIME ADAPTATION

Dynamic, real-time adaptation of telemetry systems involves reasoning about, and making decisions on a range of system factors, from a number of potential automation suppliers. These reasoning techniques, and the software tools used for their implementation may span many technologies. Attempting to integrate diverse software technologies into a single, tightly coupled system both compounds the difficulty in construction and may limit the available tools. In addition, the evolution of software solutions would be hampered, as technology lock-in prohibits adoption of the current best-in-class tools. Consequently, the SIFT team opted for a loosely coupled integration of the telemetry systems configuration.

Another goal of the system is to simplify the process of integrating tools with the system. While the native XML format is compact and expressive for specifying a configuration, it requires substantial domain knowledge and an XML interpretation interface in a tool.

Finally, since the file is monolithic, incremental editing to the file can be problematic, and consistency issues can arise if multiple readers/writers are attempting to modify a configuration.

The approach to allow for loose coupling, semantic isolation, and multi-client access is to maintain the instrumentation system state in a NoSQL (Not only SQL) database. The advantages of this approach are:

- Native Graph Support: The test configuration involves many entities, often related to multiple other entities in the system. To reason about the test configuration, these relationships must be evaluated. Graphs are a natural format for capturing this information. Physical, software, and information entities are captured as attributed nodes. Relationships (dependencies, assignment, sequencing, etc) are captured as graph edges. The NoSQL databases have powerful queries that assist a tool in finding and traversing these entities and relationships.
- Multi-User Access: Core to the database technology is the ability to provide access to a large number of client systems. This technology has powerful and high performance mechanisms for locking and synchronizing updates, including monolithic updates for consistency.
- History Maintenance: Databases have mechanisms to maintain versions, to support post-test

analysis of the system state as it progresses through a test campaign.

- Multiple Formats: In addition to graphs, the NoSQL databases support key-value stores, document stores, and standard relational database approaches to permit matching of the data to the best/highest performance native format. While the graph format is the best match for MDL, documents and ontology can capture high volume data and semantic mappings to help external parties understand the semantic meanings of a configuration.

This combination of technologies serves to maximize accessibility of the system to the performers, who, while they have considerable technical expertise in their specializations, are not familiar with much of the flight test field, much less the intricacies of MDL. By abstracting these into general reasoning structures, framing the data into graphs backed by ontologies that describe both the relationships between fields and the constraints placed upon them by both the physics of the test and the limitations of the devices under use, the performers can apply their solutions without needing the decades of flight text experience that our team supplies.

This then leads back into our theoretical world where the BRASS-powered systems are available and performing as predicted. In addition to the configuration-time adaptations discussed above, there are many points in a flight test program where real-time adaptations are desired (though they may not be actually carried out, due to the relatively static nature of running tests and the complex interdependencies between simultaneous flights). A BRASS-powered system could take into account the environment of the whole test range, and make decisions that either avoided impact to other tests while adapting a single test, or trigger adaptations to maximize the value (or minimize the cost) of the entire range, depending on the scope of the operation the system is responsible for managing.

Recent evolutions in flight test telemetry mean that some scenarios are actually adaptable without a BRASS system. An example of this is a situation with a test article (TA), which, after encountering strange vibration, requires more safety of flight data to continue the test. In a traditional PCM system, this TA would be configured with a fixed transmission schedule, and would just be forced to land. Using an iNET TmNS system, but still configured with a fixed transmission schedule (an unlikely scenario), a BRASS system would be able to dynamically reconfigure the radios, taking into account other transmissions scheduled on that frequency. A more typical iNET system would be able to use the Link Manager to adjust the scheduled bandwidth for that TA, and grant the bandwidth rights to be able to send down the necessary data without any reconfiguration.

A more interesting situation is a TA performing a test when another parallel test is grounded, leaving that TA with significantly more available bandwidth for a period of time. The TmNS system can make the adjustments to grant the TA that bandwidth, but there are likely no guidelines for how to adapt for beneficial situations that operators can follow. As such, a traditional system would just continue as if no change had occurred, and the range would lose the cost and value associated with the grounded test. A BRASS-powered system, however, could have knowledge of many relevant parameters to decide on a course of action: the current configuration of the TA's instrumentation system, the progress in the testing program (which tests have and have not been done), and even environmental conditions. This system could advise anything from simply transmitting more data for the existing test to suspending it to do a more "lucrative" test, in order

to maximize program value. Obviously, these types of complex adaptations are far away, but it is important to highlight the potential of these types of approaches.

CONCLUSIONS

Configuration techniques have made rapid strides in the past decade, with the introduction of MDL as a standard configuration language and the rest of the TmNS standards playing a crucial role. Standards that normalize behavior between vendors mean vendor-agnostic tools can be created, including Boeing's MIST tool. These same standards can be built upon, in turn, by technologies and languages such as TACL and XVisor, which add further reasoning capabilities and ease the definition of rules and interactions within the system. Going further, the BRASS program, as applied to flight test, can use these same configuration languages, abstracted into general reasoning problems, to adapt to changing environments and situations across complex systems. As these advances are slowly adopted by the community, programs will be able to set up devices quicker, intelligently making decisions that increase program value and reduce cost across all areas of flight test.

REFERENCES

[1] Range Commanders Council Telemetry Group, Range Commanders Council, White Sands Missile Range, New Mexico, *IRIG Standard 106-17: Telemetry Standards*, 2017. (Available on-line at http://www.wsmr.army.mil/RCCsite/Pages/Publications.aspx).

[2] M. Neumann, J. Moore, S. Pantham, M. Moodie, P. Noonan, and A. Whittington, "An adaptable constraints-based metadata description language (MDL) system for flight test instrumentation configuration," in *Proceedings of the European Telemetry and Test Conference, Nuremberg, Germany*, 2016.

[3] J. Moskal, M. Kokar, and J. Morgan, "Semantic validation of T&E XML data," in *International Telemetering Conference Proceedings*, International Foundation for Telemetering, 2015.

[4] DARPA, "Building resource adaptive software systems." `https://www.darpa.mil/program/building-resource-adaptive-software-systems`, 2015. Accessed: 2018-07-06.

[5] R. W. Group, *Resource Description Format (RDF) Primer*. W3C Recommendation, February 10, 2004. Available at `http://www.w3.org/TR/rdf-primer/`.

[6] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International journal on semantic web and information systems*, vol. 5, no. 3, pp. 1–22, 2009.

[7] W3C, "Owl 2 web ontology language document overview," 2009. Retrieved from `http://www.w3.org/TR/owl2-overview/`.