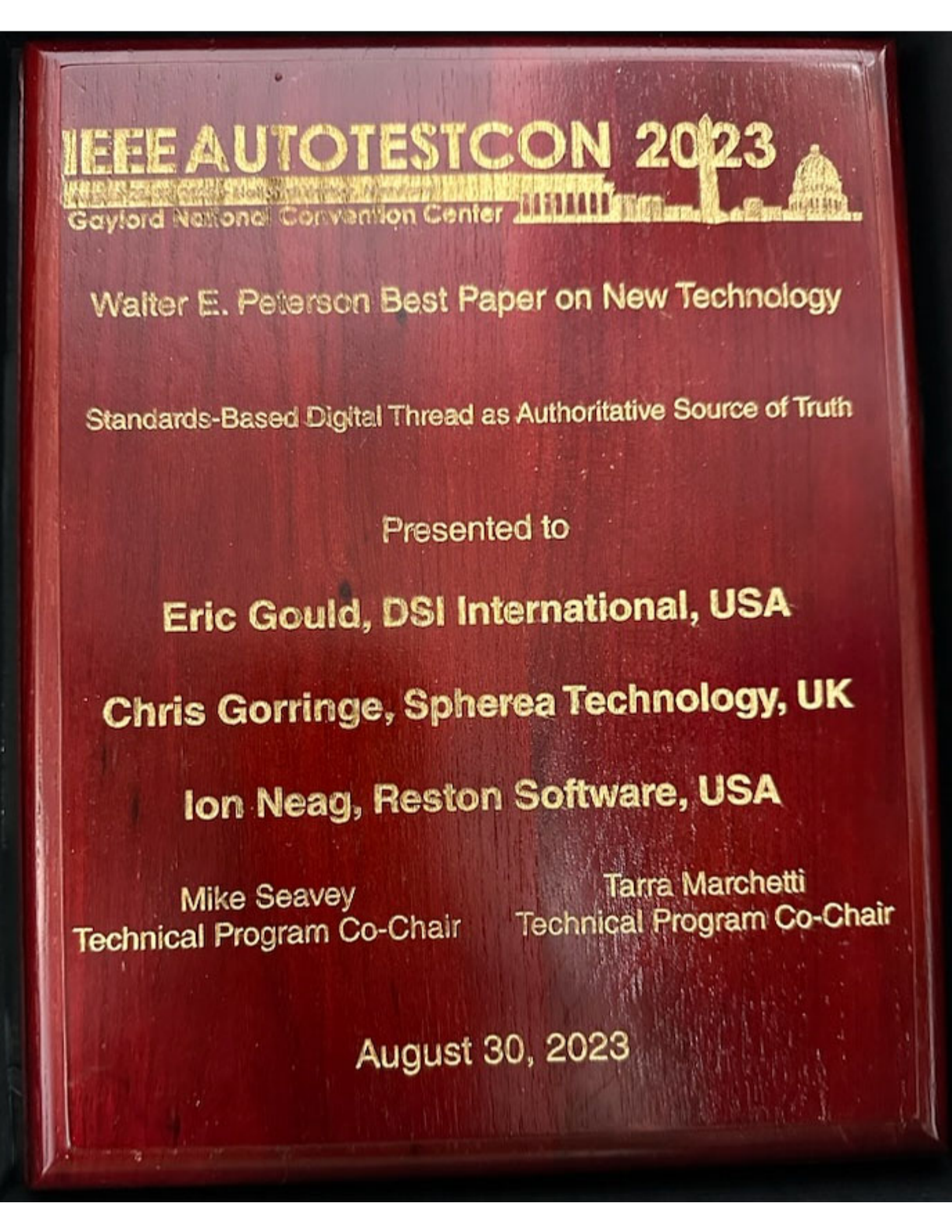
IEEE AUTOTESTCON 2023
Gaylord National Convention Center
Walter E. Peterson Best Paper on New Technology
Standards-Based Digital Thread as Authoritative Source of Truth
Presented to
Eric Gould, DSI International, USA
Chris Gorringe, Spherea Technology, UK
Ion Neag, Reston Software, USA
Mike Seavey
Technical Program Co-Chair
Tarra Marchetti
Technical Program Co-Chair
August 30, 2023

Standards-Based Digital Thread as Authoritative Source of Truth

Chris Gorringe
Spherea Technology
Christchurch, UK
chris.gorringe@spherea.co.uk

Eric Gould
DSI International, Inc.
Orange, CA, USA
egould@dsiintl.com

Ion Neag
Reston Software
Reston, VA, USA
ion.neag@restonsoftware.com

***Abstract*—This paper discusses the integration of product, test, diagnostic, and sustainment engineering disciplines using a standards-based digital thread. After reviewing some of the relative strengths and challenges of this approach (as compared to other trends in digital engineering), we present a case study in which a standards-based digital thread is used to flow detailed design data into diagnostic and test engineering activities. This integration not only creates opportunities for seamless automation during product development, but also provides full traceability from engineering databases to the test programs performed on Automatic Test Systems and subsequent test results and diagnostic reasoning. Notably, because all essential model data is stored unambiguously in the thread, the thread itself becomes the authoritative source of truth for a given project or enterprise.**

Keywords—digital engineering, digital thread, model-based engineering, diagnostic engineering, test program set development

I. Introduction

Recent trends in the digital transformation of engineering practices have largely been characterized by two opposing, though not necessarily incompatible, approaches. One of these approaches attempts to incorporate as much project data as is feasible into a master model that acts as the authoritative source of truth for a given project or enterprise. This approach—which is often based on Model-Based Systems Engineering (MBSE) practices [1] and involves the extension of models used by MBSE to include data from other disciplines—can be characterized as a homogeneous, or assimilative approach to digital engineering.

Another approach is to consider the various activities that make up the engineering effort to be individual centers of expertise, each the master of its own data. Adopting practices similar to those used by Product Lifecycle Management (PLM), digital transformation is achieved through the use of translators and/or Application Programming Interfaces (APIs) that allow data to be shared as needed between different activities and their respective software applications (tools). Here, the authoritative source of truth consists of the aggregate of these separate data sets. This approach could be thought of as a heterogeneous, or distributive approach to digital engineering.

Although each of these approaches has its strengths and weaknesses, they both face the same challenges in semantics when data is shared between disciplines with incompatible ontologies—challenges that must be addressed repeatedly for each new domain or data set that is incorporated into an enterprise's digital framework.

This paper explores a third approach to digital integration—the use of a standards-based digital thread—where the domain-specific formats that are used to transfer model data between engineering activities collectively make up the authoritative source of truth for that project or enterprise.

The use of standardized formats not only reduces the risk of semantic disconnects; it also helps ensure that the enduring record of a project (or, at a minimum, key activities within the project) can be more easily resuscitated at a later time, when engineers who had originally worked on that project—and within whose minds much of the design knowledge would (for better or worse) traditionally reside—are no longer available.

II. Three Approaches To Digital Integration

We shall now discuss the relative strengths and weaknesses of these three approaches to digital integration, bearing in mind that these are not mutually-exclusive alternatives. At least for the foreseeable future, the digital integration for any large-scale project may well be a *bricolage* of different approaches—some of which will have been strategically planned, with others taking a more *ad hoc* approach, exploiting opportunities to leverage engineering data as they arise.

A. Centralized (Master) Data Repository

a) Description: With this approach, data is stored in a master format that constitutes the enduring source of truth for a given project or enterprise. This data is translated, as needed, into the formats used by the individual tools or activities that are involved in the project (and translated back when the central repository is updated). This approach offers the most obvious compliance with the 2018 U.S. Department of Defense Digital Engineering Strategy [2], which depicts Authoritative Source of Truth as a central nucleus around which specialized models radiate.

b) Strengths: The key advantage of this approach is its singularity of data. Because master data is stored in a single format/repository, configuration control is singularly focused. The "master model" approach also promotes multi-purposing, since project data is (presumably) exposed in an easily-accessed location. Finally, for the reasons mentioned above, this approach should—at least in the long term—produce a good Return On Investment (ROI).

c) Challenges: Unfortunately, this ROI will likely not be immediate; in fact, considerable resources may be required to develop the infrastructure—including the underlying data formats and translators—used by the central repository.

Moreover, the enforcement of configuration control will inevitably result in internal political battles over who can and cannot update the master models. If agreed-upon policies are too restrictive, this may stunt the sharing of design data. If, on the other hand, access is too liberal, there is the added risk that one activity's modifications to the master model might impede the progress of activities in other disciplines (due either to semantic incompatibilities, or inconvenient timing).

This exposes the fundamental challenge of the centralized, "master model" approach to digital engineering—the fact that different disciplines or activities often define data in different ways and that different elements in the design may utilize the same terminology, even though they are entirely different things to their respective stakeholders. As a case in point, consider the various (and incompatible) meanings of the word "function" in a typical engineering project. Unless these semantic disconnects are exposed using disciplinary ontologies, projects will run the risk of data being multi-purposed in ways that hinder project throughput, thereby casting a long shadow over the project or enterprise's digital transformation efforts.

B. Interconnected Centers of Expertise

a) Description: This approach to digital transformation treats individual engineering activities as arbiters of their own domain-specific data. Integration is achieved not by storing data in a centralized master format, but rather by developing interfaces and translators that allow data to be shared between tools. With this approach, the authoritative source of truth for a project or enterprise is essentially the aggregate of the models used by the various activities and tools.

b) Strengths: By introducing integration only where it is most useful, the "interconnected centers of expertise" approach has a lower cost of entry than the implementation of a central "master model" (which integrates activities regardless of whether they benefit directly from that integratation). Also, because digital transformation is performed on a smaller scale, this approach typically produces a more immediate ROI. It is also less likely to result in internal political battles, since activities maintain control over their own data.

c) Challenges: Unfortunately, even though integration is developed on a smaller scale, this strategy suffers from the same semantic deficiencies as the "master model" approach. Activities that intend to share design knowledge must clearly define what the shared data items actually *mean* within their disciplines; this is essential not only when developing a new interface or translator, but also each time data exposed using an existing mechanism is consumed by a new activity. These discussions may not add much overhead; they are nevertheless non-trival and their omission can introduce significant risk to a project's ability to meet its development milestones.

An additional challenge, unique to this approach, is posed by structural incompatibilities between models. Resolving these incompatibilities adds complexity to translation algorithms, especially when bidirectional translation is required.

As data is passed between different engineering activities (with duplicated data residing in the models used by each), the risk still remains that different activities will end up working with different versions of a design. Ad hoc digital integration does little to ameliorate issues caused by the non-singularity of data, issues that have plagued large-scale engineering projects for decades. Because of this, configuration control for a distributed source of truth involves not only the aggregation of multiple models, but also sufficient knowledge of the overall process to ensure that design updates are successfully pushed through to all activities that rely on that data.

C. Standards-Based Digital Thread

a) Description: Within the MBSE domain, "digital thread" is a data-driven architecture that links together information generated from across the product lifecycle [3]. Like interconnected centers of expertise, a standards-based digital thread stores data in multiple, domain-specific models. The formats of these models, however, are non-proprietary, well-defined and (if available and suitable) specified in widely supported and consensus-based standards, set by recognized standards organizations or the marketplace [4]. Individual engineering activities both populate and consume the data that is stored in these standardized formats. Collectively, these models not only constitute the authoritative source of truth for a given project, but also establish a digital thread that allows design knowledge to flow between different engineering activities.

b) Strengths: This approach has many of the strengths exhibited by the other two approaches—in particular, low initial investment, relatively quick ROI and the ease with which design knowledge can be multi-purposed. More importantly, because individual models are based on published standards (and are therefore documented in detail), a standards-based digital thread is much less likely to exhibit semantical inconsistencies than the other approaches we've discussed. Not only will different engineering activities be in agreement, but the enduring source of truth will truly endure through years and possibly decases. Design knowledge stored in the thread can be unambiguously retrieved and used for future projects, since the standards define the meaning of the stored data.

c) Challenges: Of course, this semantic clarity assumes the existence of standards that are both unambiguous and extendable—unambiguous in that they provide precise data definitions (both in terms of syntax and semantics); extendable in that they allow bespoke project or enterprise-specific data to be stored in the models. Any extensions to the standardized data in the digital thread must be (1) limited to data items for which a usable standard not not exist and (2) documented as thoroughly as the data items defined within the standards being extended [5]. Disregarding these rules, opens the door—at least in the short term—for some of the same political battles that we described for the "master model" approach to digital integration, with different groups vying for ownership of model extensions and the information stored in the thread.

Moreover, with design knowledge distributed—with non-singularity—across the digital thread, configuration control will face the same challenges that we discussed for interconnected centers of expertise (both in terms of model aggregation and data

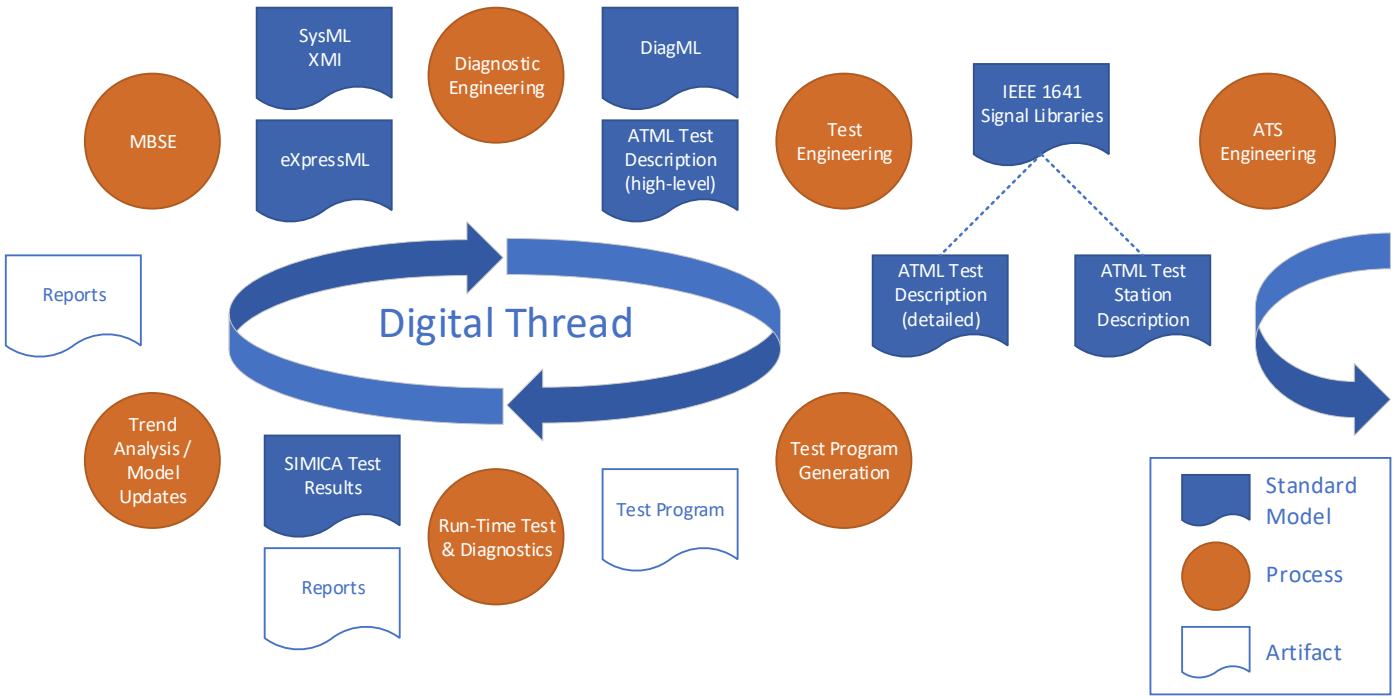

Figure 1 – Case Study - Process and Digital Thread

consistency). However, strong configuration control should be regarded as beneficial to the design process, rather than a nuisance. Each update is backed up by a transaction history, offering a full record of the design's evolution. Standardized interfaces facilitate the definition of enterprise rules that automate verification of data consistency (because the interconnecting of the models via standards is known).

These challenges notwithstanding, standards-based digital threads appear to offer the most promising and sustainable approach to large-scale digital integration. The benefits of universally understandable models—to platform owners, prime integrators and their sub-contractors and suppliers—cannot be underestimated. As existing standards mature over time—and new standards appear extending the scope of standardized engineering models—it would not be surprising if standards-based digital threads become the dominant form of digital integration for large-scale engineering efforts.

III. Case Study

The following case study—in which a standards-based digital thread is used to integrate design, test and diagnostic-related activities performed by tools already in use within industry—was demonstrated virtually in June 2021 and then physically both at AUTOTESTCON 2022 and in a subsequent industry-wide web demo. All in all, more than a dozen different activities/software tools are linked by this thread. In the physical demo, faults were inserted on a circuit card that was hooked up to an automatic test station; diagnostic results were displayed on a workstation that allowed the technician to visually identify the components to replace or initiate additional troubleshooting.

The foundation of this case study is the series of standard data formats that make up the digital thread that would represent the authoritative source of truth for any project that utilizes this process.

Although this process, shown in Figure 1, is actually a closed loop, one could say that the digital thread "begins" with system engineering's model of the target circuit card. This data is then used to create the model used by diagnostic engineering to generate test sequences that can achieve the desired diagnostic results. These sequences not only inform run-time diagnostics, but also list the tests to be implemented by test engineering. The resulting test definitions are mapped to signals, targeted to the specific test equipment and then used to automatically generate the test program code to be run on the test station. For each inserted fault, test program results are processed by the run-time diagnostics to identify repair items and—as needed—trigger guided troubleshooting. Results from multiple test and diagnostic sessions are used to enhance the diagnostics with empirical data; this history is also analyzed for trends to identify areas where updates to engineering models can improve product design, manufacturing, and sustainment (thereby closing the optimization loop).

Bear in mind that this digital thread represents an indicative example; different projects may require different process flows using these tools and formats. Moreover, although the specific tools used in this study will be mentioned below, one of the advantages of a standards-based digital thread is that other tools can be plugged in and used in their place—provided they can read from and write to the appropriate formats.

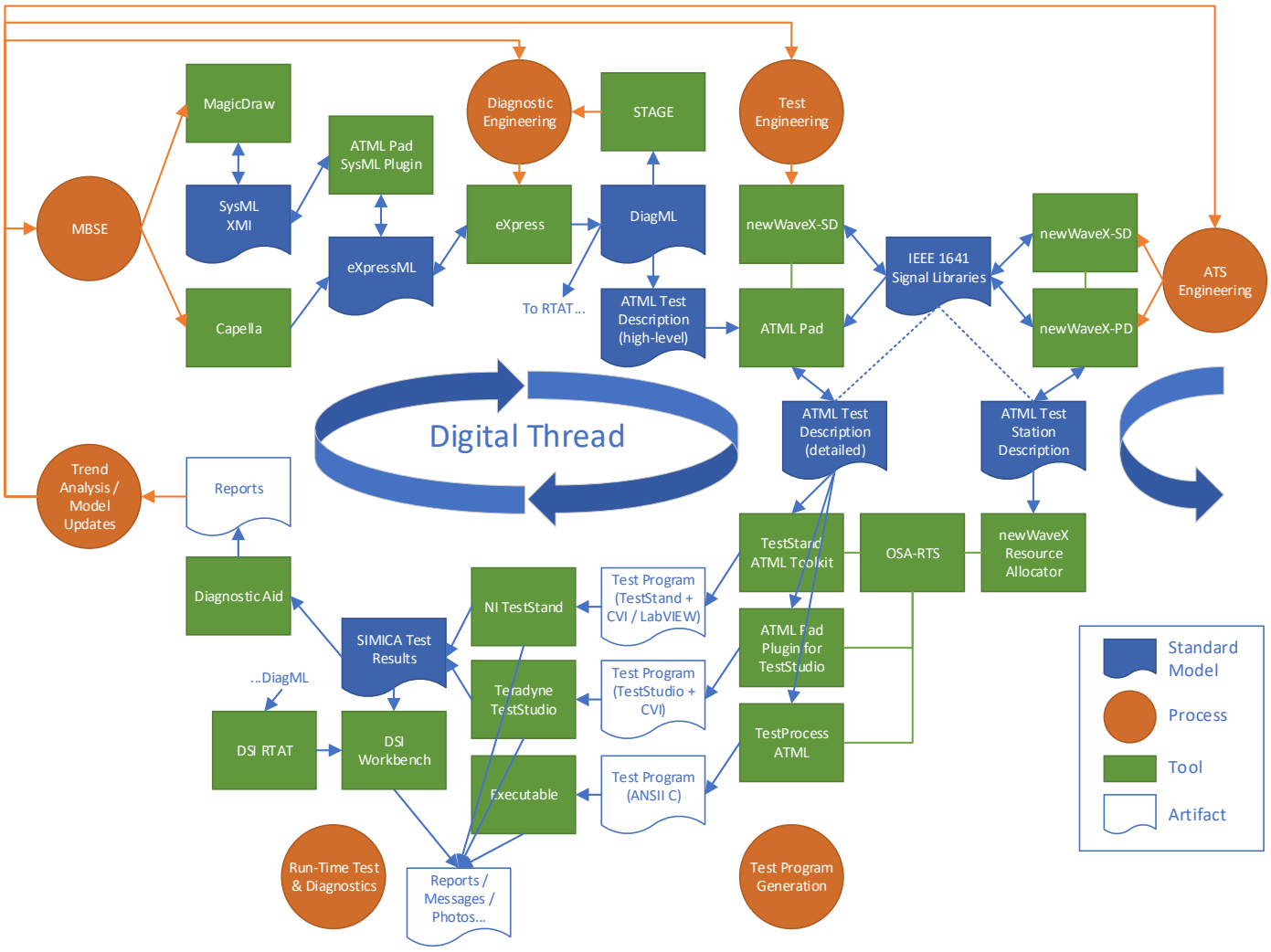

Figure 2 – Case Study – Software Tools Implementing the Digital Thread

Figure 2 adds detail to the generic thread diagram from Figure 1, showing the software tools used in the case study and their use of standard model data. These will be described in detail in the following sub-sections.

A. Model-Based Systems Engineering

The case study uses a purpose-built electronic board with low-frequency analog, Radio Frequency (RF), and digital circuits. Switches located throughout the circuit allow physical fault insertion.

The first step in the process was to develop a SysML model for the board, using CATIA MagicDraw. The model defines the external interface of the board, its components, and the interfaces of these components. A SysML Internal Block Diagram (IBD) defines component interconnections.

In addition to the functionality provided out-of-the-box by the SysML language, we used a specialized profile (extension) to describe the functional dependencies between the input and output ports of components. This information, typically available in the design phase, is used in the next step of the case study.

We should point out that the use of SysML for modeling an electronics board at the component level was chosen for expediency. SysML models work best at higher levels of assembly, while board-level design data is typically exchanged in EDIF or similar data formats.

The SysML model is exported from MagicDraw into an XMI document, which becomes part of the digital thread (Figure 2). The XML Metadata Interchange (XMI®) format is an Object Management Group (OMG) standard for exchanging metadata information via XML. It is commonly used in to exchange with UML and SysML models [6].

In a parallel MBSE prototyping effort, a model of the same board was developed in the open-source tool Capella [7].

B. Diagnostic Engineering

The next step in this case study is diagnostic engineering—where ***eXpress***™ (DSI's flagship diagnostic engineering tool [7]) is used to develop and validate diagnostic procedures, as well as perform diagnostics-informed assessments of the design. In the process represented within this case study, diagnostic engineering is performed prior to test engineering. This not only

helps ensure that the set of implemented tests achieves the diagnostic goals of the given project, but also reduces the burden on test development (since effort will not be spent implementing redundant or otherwise unnecessary tests).

Design data from the XMI document is first translated into the eXpress Markup Language (eXpressML) format. eXpressML documents the information that comprises the model(s) used by diagnostic engineering—not only to develop diagnostic test sequences, but also to perform diagnostic-informed analyses of a system or device. The key information that can be represented in eXpressML includes model topology (components, ports, connectivity, functions and object states), functional dependencies, reliability data (failure rates and failure modes) and test definitions (including the rules used to generate the coverage of each test).

Translation from SysML to eXpressML is performed by a specialized plugin of the ATML Pad software. ATML Pad™ is a development and integration environment for IEEE 1671 Automatic Test Markup Language (ATML) and related XML formats [9]

The Capella model data is exported directly into eXpressML using a specialized translator developed by Spherea [10].

The eXpressML data is then imported into ***eXpress***, where it is enhanced with reliability information (failure modes, failure rates) and test definitions. The updated model data is then be written back to eXpressML to provide an enduring, non-proprietary representation of the work performed during diagnostic engineering (Figure 2).

In addition to preliminary test descriptions (which will ultimately be converted into test program code during later stages of the process), ***eXpress*** also provides the diagnostic test sequences that will be deployed in the run-time diagnostics—not only to interpret test program results, but also to provide additional troubleshooting where automatic testing does not suffice. This test and diagnostic information is exported into a DiagML document, which becomes part of the digital thread (Figure 2). DiagML is a marketplace standard for representing diagnostic procedures (with supporting data) [10].

It is important to note that, in this case study, the diagnostic procedures deployed in the run-time diagnostics are the same as those that are used to analyze, optimize & validate the diagnostic capability of the system. This exemplifies one of the key benefits of digital integration—that the different design and analytical disciplines involved in a project utilize the same source data. In this case, diagnostic-related assessments (not only Testability analysis, but also diagnostic-informed Reliability, Risk & Safety analyses) are not disconnected from the development of the actual diagnostics to be performed in the field.

Furthermore, in this case study, these same test sequences, exported to DiagML, are also used by STAGE™ (DSI's platform for performing turnkey sustainment simulations [12]) to provide predicted behavior based on different maintenance "cocktails"—that is, different combinations of predictive, preventative and corrective maintenance. Simulation results from STAGE are subsequently compared with the trend analyses performed later in the process.

C. Test Engineering

During the test engineering phase of this process, DiagML data is imported into ATML Pad, where it is translated into a high-level ATML Test Description [13] for precisely the set of tests that will be used by the diagnostics. These high-level test descriptions are then augmented with sufficient behavioral detail to allow the fully automatic generation of test program code. The behavior information includes: stimuli sent to the Unit Under Test (UUT), measurements of UUT's responses, and comparisons with the expected (nominal) UUT responses. The comparisons produce the Pass/Fail outcomes that will drive run-time diagnostics.

The stimuli and measurements are specified in terms of signals applied, measured, or monitored at the interface of the UUT. The signals are defined in Test Signal Framework (TSF) libraries, using the format specified in the IEEE 1641 standard [14].

The development of test descriptions was done in ATML Pad, while the signal definitions were edited in newWaveX-SD [15]. The complete test description model is exported to an ATML Test Description, which becomes part of the digital thread (Figure 2).

Next, the ATML Test Description model is translated into test program code, in a target test programming language. In this case study, we used the TestStand ATML Toolkit [16] to generate an NI TestStand sequence and LabWindows/CVI code modules [17] [18]. A custom Test Description translator from the OSA-RTS toolkit [19] translates signal definitions into CVI code inserts, in the format specified by the Test Procedure Language (TPL) defined in the IEEE 1641 standard [14].

To produce executable test code, the TPL signal definitions must be first allocated to the target Automatic Test Equipment (ATE) platform. In this process, the requirements specified for each signal are compared with the capabilities of the instruments in the target ATE, with the goal of selecting an instrument resource and a capability for each signal. The allocation is performed by a component of the OSA-RTS toolkit, working in conjunction with the newWaveX Resource Allocator.

Alternate development paths in our case use study the same ATML Test Description model to generate test programs in different languages:

- A custom Test Description translator from the OSA-RTS toolkit translates signal definitions into LabVIEW code modules for the NI TestStand sequence file.
- A specialized plugin for ATML Pad generates a Teradyne TestStudio project file and LabWindows/CVI code modules.
- The "TestProcessATML" application (part of the newWaveX distribution) generates ANSII C test code, which can be built directly into executables.

The encoding of test information in a standard format allows the integration, in the same digital thread, of code generators for any other test executives and programming languages.

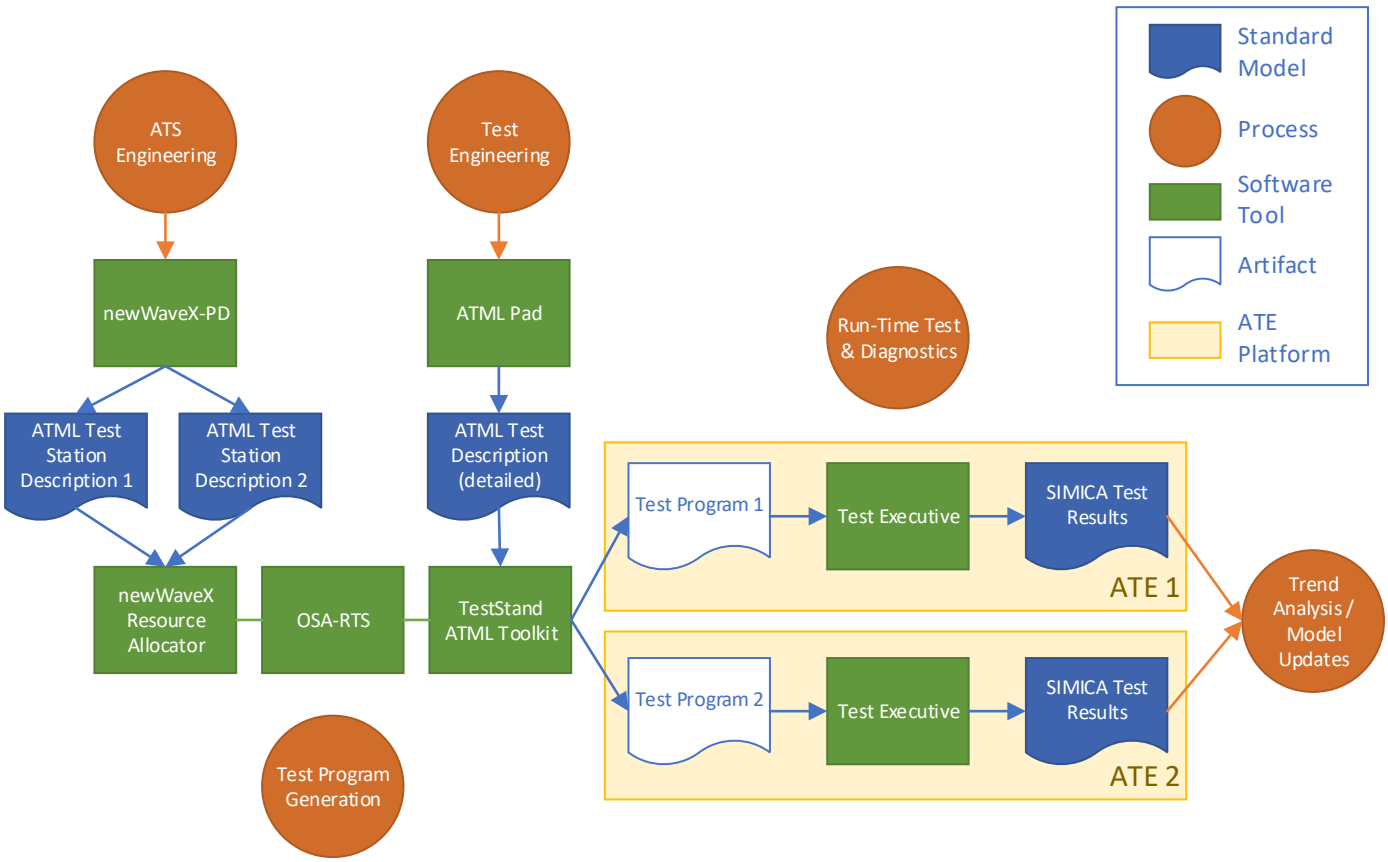

Figure 3 – Case Study – Support for Multiple ATE Platforms through Resource Allocation using Standard Signal and ATE Models

Note that allocation relies on a signal-oriented description of the target ATE, in the ATML Test Station description specified by IEEE 1671.6 [21]. This description is typically produced by a separate engineering process, depicted on the right-hand side of Figure 2, which is outside the scope of this paper [22]. In the case study, the ATML Test Station description was developed using newWaveX-PD [23]. The ATML Test Description and the ATML Test Station Description use common 1641 Signal Libraries.

In the last step of this process, the allocated signal definitions are converted into source code that implements the appropriate instrument control operations. This step was again performed by a component of the OSA-RTS toolkit, using a template-based approach.

The flexible allocation algorithm based on standard signal and ATE models allows the use of the same ATML Test Description model to produce test programs targeted to multiple ATEs (Figure 3).

It is important to note that the Test Program files don't need to be a part of the digital thread. All the information needed to regenerate the Test Program is contained in the ATML Test Description and the Signal Libraries, which are part of the tread.

D. Run-Time Test & Diagnostics (with History & Feedback)

When the test program is run, the test executive saves the test results of individual executions using the standard SIMICA Test Results format [24]. The Test Results documents become part of the digital thread (Figure 2).

The NI TestStand test executive uses the standard test results documents, in conjunction with an XSL Stylesheet, to generate reports in HTML formats. These reports contain measurement results, limits, and the pass/fail results of each executed Test. Note that the Reports files don't need to be a part of the digital thread. All the information needed to regenerate the Reports is contained in the SIMICA Test Results, which are part of the tread.

Operator messages and Test Reports are commonly used to identify the faulty components at the end of test program execution. To highlight the flexibility gained through the use of standard data formats, in this case study we also implemented an alternative solution that uses run-time diagnostics.

Run-time diagnostics are developed using the test sequences and isolated fault groups that were stored using DiagML in the digital thread. This information is imported into a run-time authoring tool (DSI's RTAT), where the diagnostics are supplemented with alternative views of the circuit card, pop-up messages/labels, test & repair procedures, reference documents, and any other information that will be useful to the technician during production or maintenance. The authored project file is then loaded into DSI Workbench™, DSI's guided troubleshooting and embedded diagnostics tool, which provides run-time diagnostics for this case study [25].

The directory where SIMICA Test Results documents are saved by the test executive is monitored continuously by DSI Workbench. When a new file is added to this directory, DSI Workbench automatically loads the test results and displays the status of the system on the technician's monitor. Suspected failures are both listed on the screen and color-coded in the appropriate graphic view of the design. If the diagnostics contains additional tests that can be used to further resolve a set of suspected components, a troubleshooting session can then be launched, guiding the technician through the additional manual testing.

Once the technician has "corrected" the inserted fault, the final resolution of the issue—regardless of whether it was the action suggested by the diagnostics—is written to a database so that it can be listed during future sessions as one of the historical resolutions associated with that fault signature (i.e., the set of failed tests). This provides the technician with an empirical resource that addresses situations not covered by the engineering diagnostics (such as unanticipated manufacturing defects, out-of-date firmware or test station deficiencies).

E. Trend Analysis / Model Updates (closing the loop)

The Test Results produced over time by the test executive can be used for trend analysis, to identify areas where the design, tests or diagnostics can be further optimized to improve product sustainment over time.

The Diagnostic Aide developed by the UK MOD [26] uses historical SIMICA Test Results in conjunction with ATML UUT and Test Descriptions (when available) to generate trend analysis reports and improved ATML UUT and Test Descriptions with added "learned" information. The improvements can be, for example:

1. Changing the limits in a test or adding new tests to the end of a sequence, so that the correct component or failure is identified within an existing fault tree
2. Optimizing test flow to identify frequent faults quickly, reducing overall test times
3. Adding tests to reduce ambiguity groups

IV. Case Study - Lessons Learned

A. Model Linking

As data items such as Components, Ports, Tests and Failure Modes appear in multiple data models, stored within the digital thread in documents with different formats, it is vital to be able to "link" these items across models.

Ideally, each data item should be identified by a unique identifier that gets recorded in all the applicable thread documents and is recognized by all the software tools. Unfortunately, this is not always possible because different data formats and tools use different identification methods. In this situation, the translation algorithms must ensure that identification data is propagated and converted as needed.

For example, in our case study the identifiers for Tests are recorded as follows:

- Attribute "Test Code", in the ***eXpress*** tool, eXpressML, and DiagML
- td:Test attribute "ID", in ATML Test Description, ATML Pad, and the TestStand TD Translator
- tr:Test attribute "testReferenceID", in SIMICA Test Results
- Attribute "Test Code", in the DSI Workbench tool

The propagation of Test identifiers through the digital tread is what allows DSI Workbench to recognize the Tests defined in ***eXpress*** (earlier in the thread) and correctly interpret test outcomes during run-time diagnostics.

As a second example, the signal requirements specified in ATML Test Description and the signal capabilities from ATML Test Station Description reference signal definitions from TSF Libraries. These signal definitions are identified through the unique combination of (XML namespace + complex type name).

- When the Test Descriptions use the same TSF Libraries as the Test Station Description, the XML namespaces are common and identification by (XML namespace + complex type name) is trivial.
- In TPS rehosting scenarios, the TSF Libraries will be different, which means that the namespaces will be different. The resource allocation algorithm cannot rely on names and will have to identify signal definitions that are functionally equivalent through their signal models

B. Model Incompatibilities

The data models used in different engineering disciplines exhibit various incompatibilities. Here are some examples:

Structural/content differences:

- The SysML models support the concepts of "class" (SysML Block) and "instance" (SysML Part); in the functional models of ***eXpress***, all objects are "instances".
- The ***eXpress*** models and the ATML Test Description models can accommodate arbitrary outcomes, to describe Tests that can fail in multiple ways. On the other hand, test executives usually only consider test outcomes as Pass/Fail, possibly with High/Low qualifies, potentially missing the more subtle diagnostic outcomes available in the design.

Semantic differences:

- "Function" has a very specific meaning in eXpress, different from its more generic meaning in a SysML <<functionalRequirement>>.
- "Signal" has a restrictive meaning in SysML (i.e., asynchronous message) and a more general meaning in IEEE 1641 (i.e., the output of a SignalFunction, which could be an asynchronous message, an analog signal, a digital signal etc.).

The model translation algorithms can take into account and compensate for most of these differences. Occasionally, models have to be designed in a manner that facilitates data translation to or from other models.

C. Incremental Updates

As new information gets added in each phase of the iterative process, it is important that model updates can be propagated to other models without losing any of the data that was added directly to the target models since the previous import.

For example, the TestStand TD Translator has an incremental update mode where small changes to the ATML Test Description (e.g., a limit change) can be incorporated in test sequences that were generated earlier. Furthermore, the code generation solution that incorporates the OSA-RTS custom TD Translator eliminates this problem by supporting fully-automatic code generation.

D. Customization and Extension Capabilities

In the implementation of the case study, we were able to achieve model linking and compensate for model incompatibilities through a combination of customizable translation and translator plug-ins. For example, the model linking issue for Test identifiers, described earlier in the paper, was resolved as follows:

1. eXpress allows the specification of user-defined *custom attributes*. One of these attributes is used for Test Codes.
2. The DiagML exporter of ***eXpress*** can be *customized* to export specific custom attributes. The "Test Code" attribute was added.
3. The DiagML importer of ATML Pad can be *customized* to map specific custom attributes from DiagML to ATML Test IDs. The "Test Code" attribute was selected.
4. NI TestStand supports the use of *plug-in components* to customize Test Results generation. A simple plug-in was developed, to assign the value of attribute td:Test.ID to attribute tr:Test@testReferenceID.
5. DSI Workbench was *customized* to interpret the attribute tr:Test@testReferenceID as the Test Code.

V. Steps Forward

As we implemented the digital thread for the case study described above, we identified a number of areas where the process can be improved—not only for this specific case study, but to support standards-based digital threads in general.

First of all, the process should be extended "to the left", to encompass requirements capture and the storage of requirements information in a standard format such as the OMG Requirements Interchange Format (ReqIF) [27]. Requirement IDs should be traceable through all the standard documents in the digital thread, and possibly through the artifacts produced by the tools.

Furthermore, there are additional opportunities for digital integration of the different activities that make up this particular case study.

- In the case study, tests are defined through the Diagnostic Engineering process. In practice, this is often achieved through dialog between diagnostic modelers and product engineers. To formalize this dialog, the thread should allow for an initial set of tests and test-related data items to be defined in the System Engineering process [28], with subsequent refinement through iterative data exchanges between Diagnostic Engineering and System Engineering. Integrating this approach in a standards-based digital thread is currently impossible, due to the lack of a standard SysML representation for hardware tests. The adaptation of the OMG-defined UML Testing Profile [29], designed for software testing, is a possible solution to this limitation.
- The run-time diagnostic executive currently stores individual diagnostic sessions and their resulting maintenance actions in its own history & feedback database. Although this relational database can be easily parsed by other tools, the diagnostic reasoner should also be updated to store this same data using the SIMICA Maintenance Action Information format (IEEE 1636.2) [30]. This would open up opportunities for extending this standards-based digital thread into maintenance-related activities.
- Simulation-based maintenance predictions are exported to a simple, non-standard document. It would be useful if a suitable standard can be identified for storing these predictions so that the results of maintenance trade studies can be more rigidly documented—and more seamlessly compared with data from the field.

Another area in which this process can be improved would be to replace commercial data interchange formats that originate from the marketplace with formats that are defined by industry standards.

- Unfortunately, because diagnostic engineering is not performed in a consistent manner across industry, it is unlikely that a universally acceptable standard will emerge for the models used by this discipline (as a standard substitute from eXpressML).
- On the other hand, the diagnostic procedures that are generated by diagnostic engineering—which, for this study, are represented using the marketplace standard DiagML—can likely be stored using an industry standard format such as IEEE 1232 AI-ESTATE [31].

Finally, during the implementation of this case study, a limitation was identified that was shared by every one of the formats that comprise this digital thread. Although the data formats always have an area where version information can be recorded, there is no standard way of representing—within the individual data files of the thread—the history of the data stored within that file. There is no way, for instance to identify if a given Test Description has been updated to address design changes introduced in a new version of the MBSE model (which is several steps earlier in the thread). Each stage of the thread may contain date and time stamps that indicate when the file was updated—as well as the version of the data in the tool that most recently updated the thread—however there is currently no accepted (let alone standardized) format for documenting the

version of the source data and the different tools through which it has passed before reaching a given stage of the thread. The development of a standard practice and standard data structures will become essential as configuration control procedures evolve to address data consistency across a distributed model of a design, including standards-based digital threads like those described in this paper.

VI. Conclusions

The case study described in the paper demonstrates that "digital thread" solutions for product, diagnostic, test, and sustainment engineering can be developed using existing industry standards and marketplace-defined open data formats. While this end-to-end integrated solution is an innovative prototype, the individual software tools, data formats, and the processes they support are already integrated in applications with TRL levels 8 … 9.

Standards-based digital threads offer semantic cohesiveness that is missing from many other methods of digital integration. When the knowledge stored in each stage of a digital thread is unambiguously defined in an associated standard, software interfaces and translators can be developed with fewer time-consuming incompatibilities and misconceptions. This not only facilitates the sharing of data along expected avenues, but also fosters the discovery of new and unanticipated opportunities for multi-purposing design knowledge.

Moreover, when a standards-based digital thread is used as a primary method for documenting design knowledge (that is, as an authoritative source of truth), data can be more easily resuscitated and leveraged by future endeavors—not only within updates to the current design, but also as a basis for the development of new designs.

The use of standardized formats within a digital thread does not, of course, eliminate all areas where disconnects can occur. Resolving these disconnects may require extensions to standard formats, customization of software tools, or changes in model design.

Configuration control is of primary importance, especially given that the reduction in duplicate efforts leads to greater interdependence between different activities. When changes are made to a design by one discipline, the updated data must flow through the thread to all activities that are impacted by those changes. Although this scenario is endemic to engineering in general, a well-defined digital thread should ultimately help projects manage the data flow, resulting in fewer version-related disconnects.

All in all, the effective use of a standards-based digital thread will result in more cohesive product development and sustainment. Moreover, if the thread also serves as a project's enduring, authoritative source of truth; design knowledge will be represented unambiguously, thereby providing a permanent, reusable record of the design. These benefits should be welcomed by all stakeholders in a given project or enterprise—developers, integrators, maintainers, and end users.

References

[1] INCOSE, "INCOSE MBSE Initiative". [Online]. Available: https://www.incose.org/incose-member-resources/working-groups/transformational/mbse-initiative

[2] Office of the Deputy Assistant Secretary of Defense for Systems Engineering, "Department of Defense Digital Engineering Strategy (June 2018)". [Online]. Available: https://sercuarc.org/wp-content/uploads/2018/06/Digital-Engineering-Strategy_Approved.pdf

[3] V. Singh, and K. E. Willcox, "Engineering Design with Digital Thread", *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Kissimmee, FL, January 2018

[4] Office of the Under Secretary of Defense for Research and Engineering, "Modular Open Systems Approach". [Online]. Available: https://ac.cto.mil/mosa/

[5] T. P. Lopes, I. A. Neag and J. E. Ralph, "The role of extensibility in software standards for automatic test systems," *IEEE Autotestcon, 2005*, Orlando, FL, USA, 2005

[6] Object Management Group, "XML Metadata Interchange". [Online]. Available: https://www.omg.org/spec/XMI

[7] Capella, Open Source Solution for Model-Based Systems Engineering. [Online]. Available: https://www.eclipse.org/capella/

[8] eXpress, System Modeling for Diagnostic Design and Analysis. DSI International. [Online]. Available: https://www.dsiintl.com/products/express/

[9] ATML Pad, Development and Integration Environment for Automatic Test Markup Language. [Online]. Available: https://www.atmlpad.com/

[10] B. Bossa, "Model based approach to embrace digital thread", *Test and Diagnostics in the Digital Thread Panel*, AUTOTESTCON 2022

[11] E. Gould, D. Hartop, E. Lee, I. Neag, and M. Wilson, "DiagML – An Interoperability Platform for Test and Diagnostics Software," *2002 AUTOTESTCON Proceedings*, pp. 597-607, IEEE 2002.

[12] STAGE, Health Management and Operational Support Simulation. DSI International. [Online]. Available: https://www.dsiintl.com/products/stage-diagnostic-simulation/

[13] IEEE Std 1671.1-2017 – IEEE Standard for Automatic Test Markup Language (ATML) Test Descriptions. IEEE, 2017.

[14] IEEE Std 1641-2010 – IEEE Standard for Signal and Test Definition. IEEE, 2010.

[15] newWaveX–SD. Spherea Technology. [Online]. Available: https://www.spherea-technology.co.uk/Products/newWaveX/newWaveXSD/newWaveXSD.htm

[16] TestStand ATML Toolkit. NI. [Online]. Available: https://www.ni.com/en-us/shop/software/products/teststand-atml-toolkit.html

[17] A. Jain and S. Delgado, "Automatic ATML test description translation to a COTS test executive," *2009 IEEE AUTOTESTCON*, Anaheim, CA, USA, 2009, pp. 190-194

[18] L. Lindstrom and I. Neag, "Reducing test program costs through ATML-based requirements conversion and code generation," *2013 IEEE AUTOTESTCON*, Schaumburg, IL, USA, 2013

[19] M. Cornish, A. Jain, M. Brown and T. Lopes, "An open source software framework for the implementation of an open systems architecture, run-time system," *2012 IEEE AUTOTESTCON Proceedings*, Anaheim, CA, USA, 2012, pp. 209-214

[20] IEEE Std 1671.2-2012 – IEEE Standard for Automatic Test Markup Language (ATML) Instrument Description. IEEE, 2012.

[21] IEEE Std 1671.6-2015 – IEEE Standard for Automatic Test Markup Language (ATML) Test Station Description. IEEE, 2012.

[22] C. Gorringe, T. Lopes and D. Pleasant, "ATML capabilities explained," *2007 IEEE Autotestcon*, Baltimore, MD, USA, 2007

[23] newWaveX–PD. Spherea Technology. [Online]. Available: https://www.spherea-technology.co.uk/Products/newWaveX/newWaveXPD.htm

[24] IEEE Std 1636.1-2018 – IEEE Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA):

Exchanging Test Results and Session Information via the eXtensible Markup Language (XML). IEEE, 2018.

[25] DSI Workbench, Guided Troubleshooting and Embedded Diagnostics for the Technician. [Online]. Available: https://www.dsiintl.com/products/dsi-workbench/

[26] C. Gorringe and M. Brown, "Maximising diagnostic performance: By integrating COTS solutions via ATML standards to create smart diagnostic," 2017 *IEEE AUTOTESTCON*, Schaumburg, IL, USA, 2017

[27] Object Management Group, "Requirements Interchange Format (ReqIF)". [Online]. Available: https://www.omg.org/reqif/

[28] C. A. Lansdowne, C. Gorringe and P. McCartney, "Experimental applications of Automatic Test Markup Language (ATML)," *2012 IEEE AUTOTESTCON Proceedings*, Anaheim, CA, USA, 2012

[29] Object Management Group, "UML Testing Profile 2 (UTP2)". [Online]. Available: https://www.omg.org/spec/UTP2

[30] IEEE Std 1636.2-2018 – IEEE Standard for Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Maintenance Action Information via the eXtensible Markup Language (XML). IEEE, 2018.

[31] IEEE Std 1232-2010 – IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments. IEEE, 2010.