2018

Toward an Interoperability and Integration Framework to Enable Digital Thread

Mary Bone
Stevens Institute of Technology

Mark Blackburn
Stevens Institute of Technology

Benjamin Kruse
Stevens Institute of Technology

John Dzielski
Stevens Institute of Technology

Thomas Hagedorn
University of Massachusetts Amherst

See next page for additional authors

Recommended Citation
Bone, Mary; Blackburn, Mark; Kruse, Benjamin; Dzielski, John; Hagedorn, Thomas; and Grosse, Ian, "Toward an Interoperability and Integration Framework to Enable Digital Thread" (2018). *Systems*. 624.
https://doi.org/10.3390/systems6040046

systems

MDPI

Article

Toward an Interoperability and Integration Framework to Enable Digital Thread

Mary Bone [1,*], Mark Blackburn [1], Benjamin Kruse [1], John Dzielski [1], Thomas Hagedorn [2] and Ian Grosse [2]

[1] Systems Engineering Research Center, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA; mblackbu@stevens.edu (M.B.); bkruse@stevens.edu (B.K.); jdzielski@stevens.edu (J.D.)

[2] Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Amherst, MA 01003, USA; thagedorn@umass.edu (T.H.); grosse@ecs.umass.edu (I.G.)

* Correspondence: mbone@stevens.edu; Tel.: +512-630-6558

Received: 1 November 2018; Accepted: 12 December 2018; Published: 18 December 2018

Abstract: This article discusses ongoing research investigating the feasibility of supporting an interoperability and integration framework to enable the digital thread, or an authoritative source of truth with current technology. The question that initiated this exploratory research was, "Is there current technology that can enable cross-domain digital artifact data sharing needed for the digital thread?" A thorough review and investigation of current state-of-the-art model-based systems engineering was performed by reviewing literature and performing multiple site visits and interviews with organizations at the forefront of digital engineering. After this initial investigation and review, a Semantic Web-enabled framework that would allow data in the thread to be captured, stored, transferred, checked for completeness and consistency, and changed under revision change control management began to be formed. This framework has gone through revisions. This paper reflects the most current demonstration of the framework and its capability of acquiring digital data, and parsing and querying the data using Semantic Web technology to generate a decision table that allows the decision data to be visualized. The article concludes with future demonstrations of the framework to further advance toward a framework that can enable a digital thread in practice.

Keywords: Semantic Web; digital thread; authoritative source of truth; systems engineering; MBSE; SysML; cross-domain integration

1. Introduction

The ability to manage digital media from multiple cross-domain sources during the systems engineering life cycle process has been identified as a key enabler of the digital thread [1]. To enable the digital thread discussed in [2], the digital media data in the thread must be captured, stored, transferred, checked for completeness and consistency, and changed under change control management. The question that initiated this exploratory research was, "Is there current technology that can enable cross-domain digital artifact data sharing needed for the digital thread?"

The first step to answer this question was to do a current review of cross-domain tool integration tools and research both in the literature and applied in practice. In Reference [3], an overview of the state-of-the-art techniques used for cross-domain model-centric engineering is reported. This was done by having informal discussions with leaders at many of the known top model-centric engineering adopters like the NASA Jet Propulsion Laboratory (JPL), DARPA, and Sandia. During these discussions and information sharing sessions, it was discovered that while there are tools that help with cross-domain data (e.g., ModelCenter), they were still application programming interface (API)-based

and considered rigid and brittle since changes to an interface can disable capability or cause issues in the data interchange. Authors in [4] elaborate on this point and discuss how tool integration is not static due to changing tool combinations along with execution sequences that vary from case to case. During these discussions, an emerging technology called Semantic Web technology (SWT) [5] was mentioned as a potential solution to these rigid and brittle interfaces. Semantic Web technology can allow data to be exchanged, reconciled, and checked for consistency and completeness at the data level using a standard language, therefore avoiding many of the issues that arise with current APIs [6].

In exploring SWT, it was found that the technology may aid in the effort to create a digital thread or authoritative source of truth (AST) for digital engineering [7]. The concept of AST for this research is to provide access to the most authoritative data at the element level known in the design of a system at a given point in time. More information about AST in the context of this research can be found in [8]. Research conducted with Industry 4.0 indicated that SWT could be a key enabler of the digital thread [9].

Ontologies are a key technology within SWT. During this research it was discovered that ontologies should be engineered and designed in such a way to they can be built upon and reused [10]. The concepts for ontology development for this research were based on the most extensive and successful use of ontologies found, which exists in biology with the Open Biological and Biomedical Ontology (OBO) Foundry [11–13]. OBO is a collective of ontology developers that are committed to collaboration and adherence to shared principles, which has allowed them to create successful suites of ontologies [14]. As an example of how powerful these suites of ontologies are, Gene Ontology (GO) was used to annotate 50% of the Drosophila genome (i.e., a genus of small flies) to the molecular function and biological process ontologies using the GO method [15]. This annotation was done with little human intervention. This type of automated data relationship identification and annotation is an enabler of the digital thread with cross-domain data.

With successful research results in biology [14,15] and promising systems engineering examples [9], the goal of this research is to build upon and expand the current research. To meet this goal, the research is working toward developing an architecture that implements SWT in such a way that it allows for a tool-independent solution for sharing data between different digital artifacts. In this demonstration of the interoperability and integration framework (IoIF), the goal is specifically to exchange data for decision making regarding an unmanned aircraft system (UAS) [16]. This particular demonstration of the IoIF was chosen because decision making was of interest to our research sponsors, an initial decision ontology was available [17], and a UAS case study with decision-making data was provided to the research team [18]. The IoIF has the potential to handle other data if the corresponding ontologies, rules, and queries are provided or developed. Future demonstrations of the IoIF plan to show this ability.

The IoIF's current state is shown in Figure 1. It is a high-level SWT-enabled architecture solution for the interoperability and integration of data in the development and management of engineered systems with multiple digital artifacts. One goal of the IoIF is to be able to retrieve data from any source and convert the data into a World Wide Web Consortium (W3C) linked data standard format such as JSON-LD or Resource Description Framework (RDF) [19]. Once the data are in a W3C linked data standard format, then the data can be aligned with developed ontologies. After the aligning of the data with an ontology, queries in SPARQL [20] or another W3C query standard [6] can be developed and executed against the data. The queries can check for completeness and consistency of the data from multiple digital sources. The final goal of the IoIF is to notify users when data do not pass queries with regards to completeness and consistency. One example of a basic check would be for correct and consistent units for data from multiple sources. The IoIF could flag data and/or be given the authority to determine the digital artifact that is the AST for the specific data element to then use the correct AST data to propagate to other artifacts. The IoIF has two main functions: (1) data acquisition and aggregation; and (2) semantic query and reasoning, which allows for consistency and completeness checking of the data.

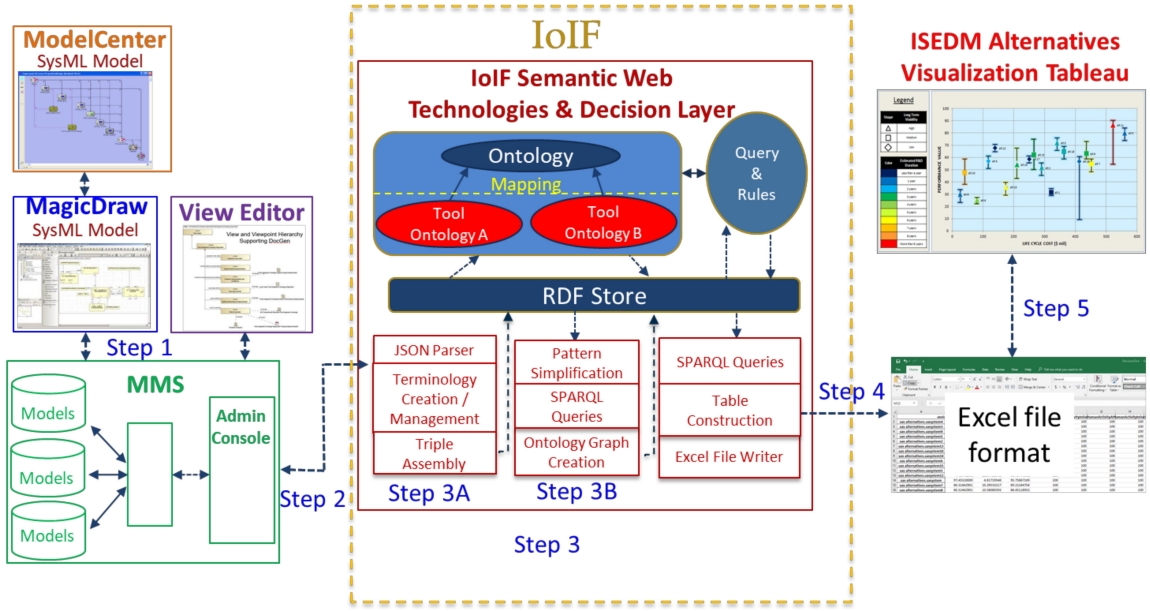

Figure 1. Demonstration flow diagram with the Interoperability and Integration Framework (IoIF).

2. Results

This section will discuss the demonstration setup and results. The demonstration is broken up into five steps (see Figure 1). The discussion will focus on these five steps in terms of the inputs, outputs, and processing required for the next step. The demonstration flows from the left to right in Figure 1, but there will be some discussion at the end of how the demonstration could flow from right to left, which allows for changes or updates to flow in both directions through the IoIF. All of the demonstration's steps will be discussed for context, but the focus of this paper is the IoIF. The IoIF is shown by the dotted box in the middle of Figure 1 including the input and output. In Figure 1 the content on the right and left side of the IoIF (i.e. outside of the dotted box in the middle) are only used for demonstration purposes. The goal of the IoIF is to be digital artifact agnostic so that other tools can be utilized in future demonstrations of the IoIF. The IoIF's focus is at the data element level.

2.1. Systems Modeling Language (SysML) Model to IoIF (Step 1)

The left side of the demonstration in Figure 1 includes ModelCenter, MagicDraw, View Editor, and the model management system (MMS). ModelCenter is a commercial software tool for the integration of simulation models for system design and optimization by Phoenix Integration [21]. The View Editor and MMS are part of the Open Model-Based Engineering Environment (OpenMBEE) [22]. OpenMBEE is a platform that can be utilized by SysML tools like MagicDraw to provide infrastructure for versioning, workflow management, and access control. OpenMBEE facilitates multi-tool and multi-repository integration across engineering, computing, and management disciplines. In this IoIF demonstration, there were multiple goals of the larger research scope [8,22] encompassed on the left side of Figure 1: (1) Develop a SysML model that could use a tool like ModelCenter to create multiple instances/variations of a system; (2) Develop a SysML model that represents all the data needed for decision analysis using the Integrated Systems Engineering Decision Management (ISEDM) [23]; (3) Edit a SysML model through View Editor; and (4) utilize MMS to export SysML model data in JSON data format.

2.1.1. SysML Model and Multi-Disciplinary Analysis and Optimization (MDAO)

The SysML model was generated with the goal of being able to populate it with UAS design variables and attributes. This population was done through defined parametrics of the UAS used to

automatically generate alternative instances (see Figure 2) through the multi-disciplinary analysis and optimization (MDAO) tool ModelCenter [21]. Being able to generate a full set of design alternatives with a set of variables and attributes is important in being able to perform decision and/or trade-off analysis with ISEDM [23].

In this demonstration, 15 instances of the UAS were generated and each instance had 54 factors. To generate the instances of the UAV, a trade study was developed in SysML [8]. The first step to generating the instances was providing the definition of a set of alternatives for the Payload (i.e., weight, field-of-view, number of pixels, and pixel size) and the Air Vehicle (i.e., wingspan, altitude, and engine type). Next, the assigned values were read in from a formatted spreadsheet. Finally, an activity combined the instances of these alternatives in all possible pairings and created an ordered list of UAS alternatives along with associated parametric data. More detail on this process can be found in [8], and more information regarding the UAS parameters can be found in [18,24]. The number of UAS alternative instances for the live demonstration were limited to 15 for the sake of processing time.

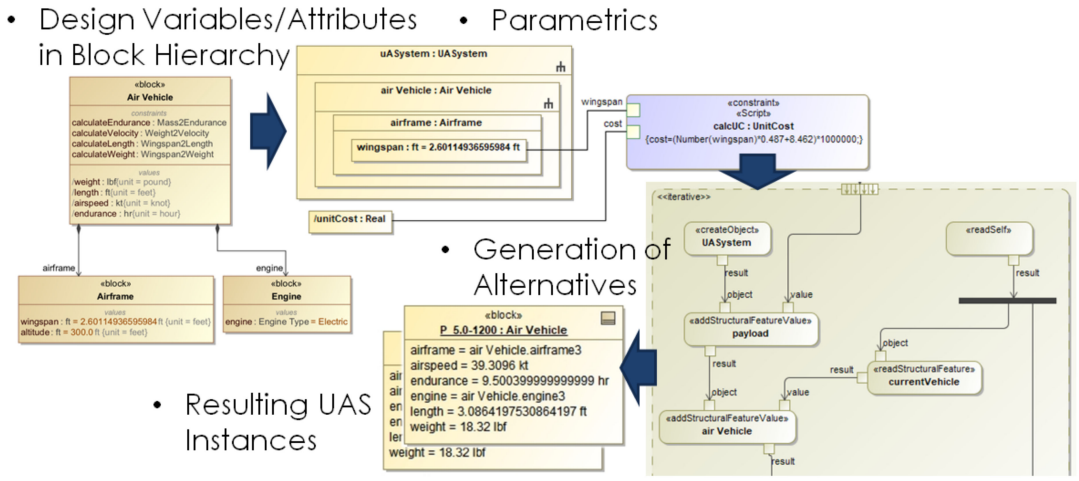

Figure 2. SysML model generation of unmanned aircraft system (UAS) instances. Blocks define design variables and attributes, which are interlinked on parametrics diagrams, which are executed using activity diagrams and ModelCenter (not shown) to generate alternative instances.

2.1.2. OpenMBEE

View Editor and MMS shown in Figure 1 are parts of OpenMBEE [22]. In the demonstration, the goal of using the View Editor was to provide the ability to edit a SysML model outside of its native tool and in a more textual format, similar to MS Word, that should be familiar to most stakeholders. These changes are then populated back into SysML through the MMS utilizing the Model Development Kit (MDK) plugin for MagicDraw. MMS captures all model elements of a project (e.g., classes, instances, properties, values, and relations, but not diagram layout), including their change history and view instances for View Editor. MMS maintains a copy of the SysML model as a JSON file [6]. This JSON file is retrieved by the IoIF to begin Step 2, as shown in Figure 1. In this demonstration, the data needed by the IoIF was in this JSON format, but the IoIF could be updated to parse XML, RDF, or other Semantic Web technology data instead. A key point is that the IoIF only needs the JSON file as input.

2.2. IoIF Semantic Web Processing for Visualization (Steps 2, 3, and 4)

In Step 2 (see Figure 1), the IoIF retrieves the JSON file that contains the set of UAS alternatives generated in Step 1 from the MMS. Once the IoIF receives the JSON file, Step 3 (see Figure 1) begins. Step 3 can be broken into three main processes: Data Parsing, Data Mapping, and Data Acquisition. The detailed process flow and interaction of these three processes is shown in Figure 3. The Data Parsing process (Step 3A) uses Python scripts to parse the JSON file into named individuals and triples,

while managing their terminology. Owlready2, a package for manipulating OWL 2.0 ontologies [25], is used next to store the terms, individuals, and triples parsed from the JSON file in an RDF graph. The next main process, Data Mapping (Step 3B), begins with simplifying the RDF data by shortening the triple patterns. Then, an ontology-mapped graph is created based on SPARQL queries, which are executed using RDFLib [26]. For this demonstration, a set of ontologies aligned with Basic Formal Ontology [10] were used. They can be found at [27]. The final process of Data Acquisition (Step 4) uses queries of the ontology-mapped graph to retrieve the specific data needed for the ISEDM table construction. The table is then written into a Microsoft Excel format, which is needed input for Tableau, using the Python Pandas package. The following sections will go into more detail on each of the main process steps.

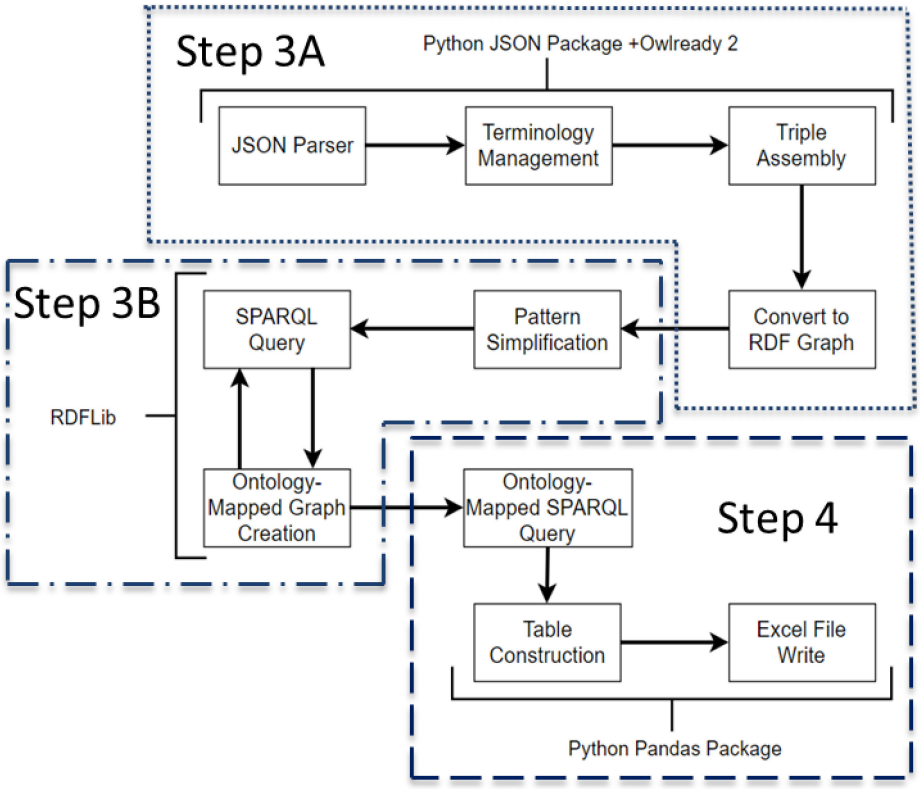

Figure 3. IoIF data process flow diagram.

2.2.1. Data Parsing–JSON Parser, Terminology Management, Triple Assembly, and Convert to RDF Graph

The Data Parsing process is shown in Figure 4 and encompasses the JSON Parser, Terminology Management, Triple Assembly, and Convert to RDF Graph steps from Figure 3. The demonstration utilized Python to perform the data parsing flow represented in Figure 4. Python was chosen since it is a language that is easy to learn; was familiar to the researchers; and allowed for all the manipulation, reasoning, etc. of the data that was required for the IoIF demonstration. The process in Python starts by importing the JSON file from the MMS, along with a SysML ontology that defines tool-specific keywords. The JSON file is then parsed using a built in Python function to convert it into a dictionary, together with a custom-made recursive algorithm that unpacks the dictionary into a list of unique instances and triples using field titles as predicates and element IDs to identify the subject instances. A provided tool namespace and a dictionary of existing terminology is used to define new classes and properties during subsequent creation of RDF statements. In cases where terminology is not previously defined, it is created using Python's Owlready2 package along with the tool namespace, and subsequently saved to a source ontology file. Instances corresponding to

element IDs are generated using a similar approach, with the process repeated every time a JSON file is imported. Once all terminology and instances are populated, triples are first constructed with their subject, object, and predicate, and then stored in an RDF graph. The triples at this point are not aligned with an ontology, and are instead only generated from the JSON input data.

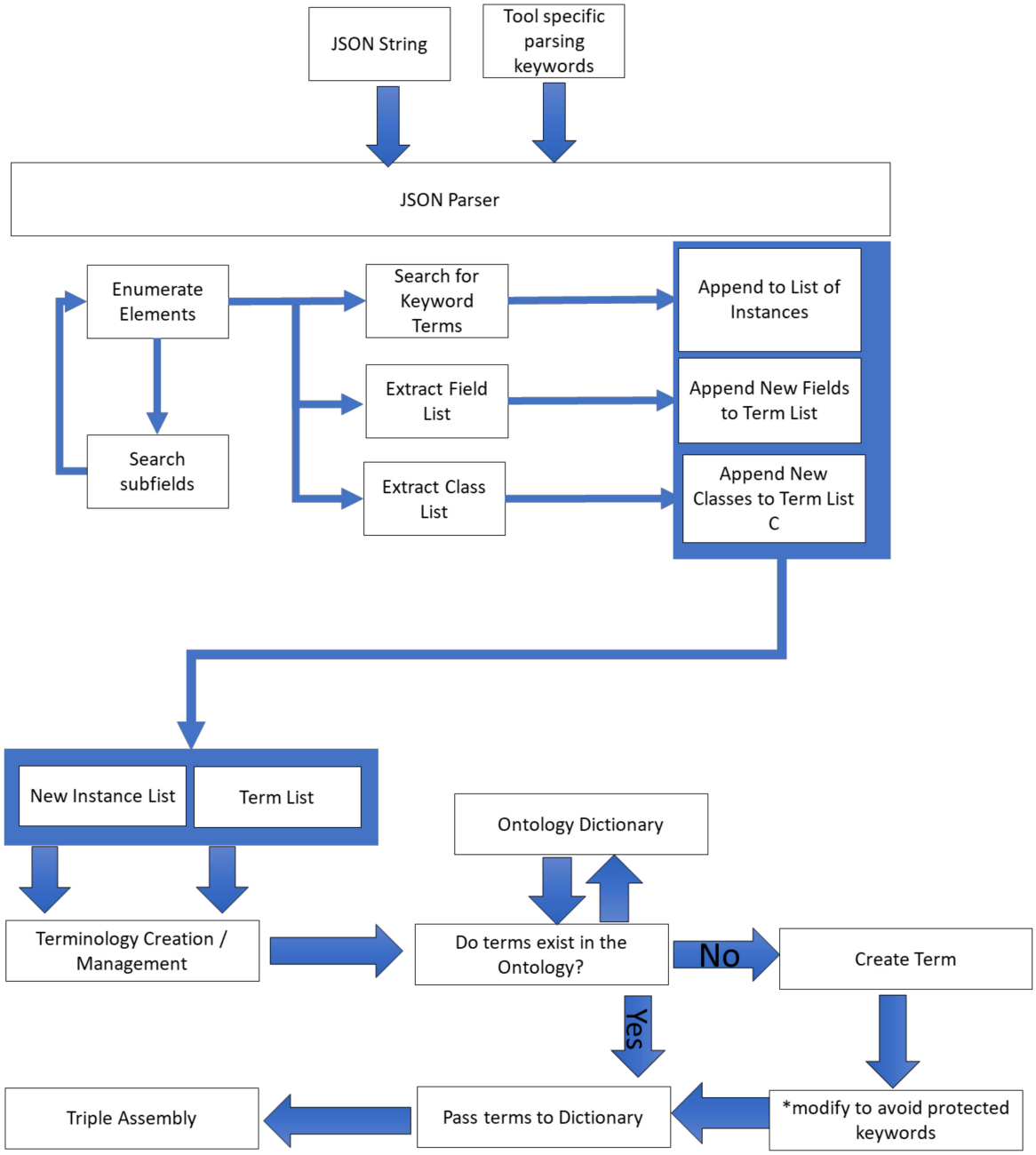

Figure 4. Data parsing.

2.2.2. Data Mapping—Pattern Simplification, SPARQL Query, and Ontology-Mapped Graph Creation

In the previous steps, the incoming raw JSON data was parsed and stored in an RDF graph. At this point the data needs to be mapped to ontologies so that the appropriate queries can be run against it. The first step is to simplify the patterns found in the SysML data in the RDF graph by searching the graph and creating new triples. These new triples allow for commonly recurring chains of relations to be simplified into a single equivalent statement. This shortens the run time of subsequent queries by decreasing the number of triples needed to determine a relationship. When this whole process is

executed for the first time, a SysML ontology has to be generated, which is done by inspecting the RDF graph generated and identifying patterns in the graph. These patterns are then used to create SPARQL queries that retrieve all the tool-specific term patterns found in the graph. The tool specific ontology and SPARQL queries only need to be generated once. They replace the current tool API schema. A Python code then uses the results of these queries, along with a set of mapping instructions to generate an ontology-mapped version of the data. This mapping only needs to be generated once for the SysML decision model and can then be used for all subsequent work with data from that or similar models. Once this instance data is mapped to ontologies, predefined SPARQL queries can be utilized to manipulate the data as desired, since the ontology-mapped data is tool-agnostic. At this point, the data is mapped against all the ontologies that have been imported (Figure 5). All the ontologies shown are aligned with the top-level Basic Formal Ontology (BFO). This means that the lower-level ontology classes can be traced back BFO classes. This allows for interoperability between the ontologies. The research was able to easily utilize the available ontologies without having to modify them, because they were already aligned with BFO.

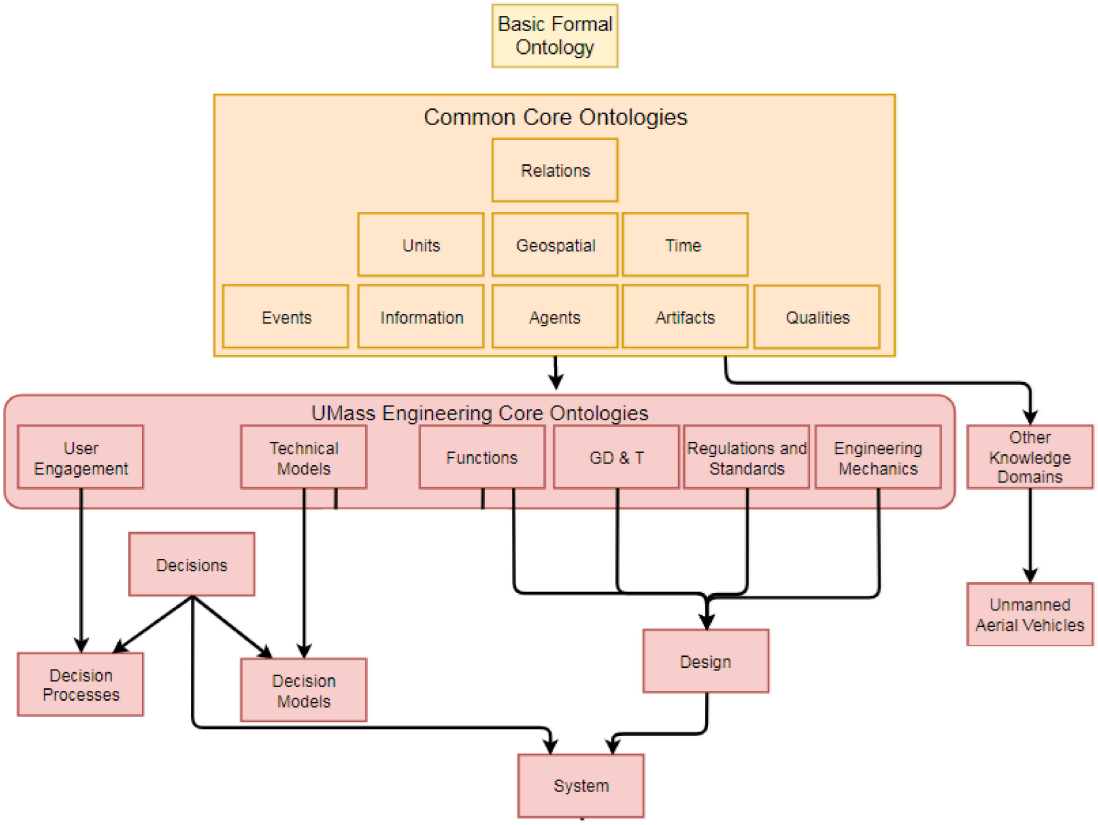

Figure 5. Hierarchical structure of the decision ontology and supporting domain ontologies used in IoIF. Arrows indicate direct dependencies in the form of shared terms. Geometric dimensioning and tolerancing (GD&T).

2.2.3. Data Acquisition—Ontology Mapped SPARQL Query, Table Construction, and Excel File Write

Once the data is mapped to ontologies, the process is straightforward. Predefined SPARQL queries are executed to retrieve the desired ISEDM output data (instances). The ISEDM data is then saved in an Excel file format using the ExcelWriter Python function. For this demonstration, the desired output data was predefined, but in the future this may or may not be the case. Future demonstrations can show how a tool could "ask" for data from the IoIF and how the IoIF would handle querying the data on the fly to retrieve the desired output data.

2.3. Visualization Automatic Updates (Step 5)

The final step, Step 5 in Figure 1, provides the required Excel file to the visualization tool, Tableau, to generate the graphics based on ISEDM. The Tableau visualizations can be pre-scripted and therefore auto-generated once the Excel data are available. This ability is a key contribution of the research. The researchers did not find any other published method that can auto-populate the complex ISEDM type of data into Excel or similar format from only JSON data.

The visualization is important because it allows decision makers to quickly and easily see how changes, such as those entered through the View Editor, affect different relationships among the data. One example is the compiled chart in Figure 6, which shows how different engine types change the average performance (a unitless measure) and unit cost (US dollars). During the live demonstration, the audience was able to visually witness the dynamic change of the graph due to the change of the engine type. For this paper, the before and after graphs were merged to show the change. This type of dynamic change visualization can be shown for any of the combinations of UAS factors, and Figure 6 is just one example.

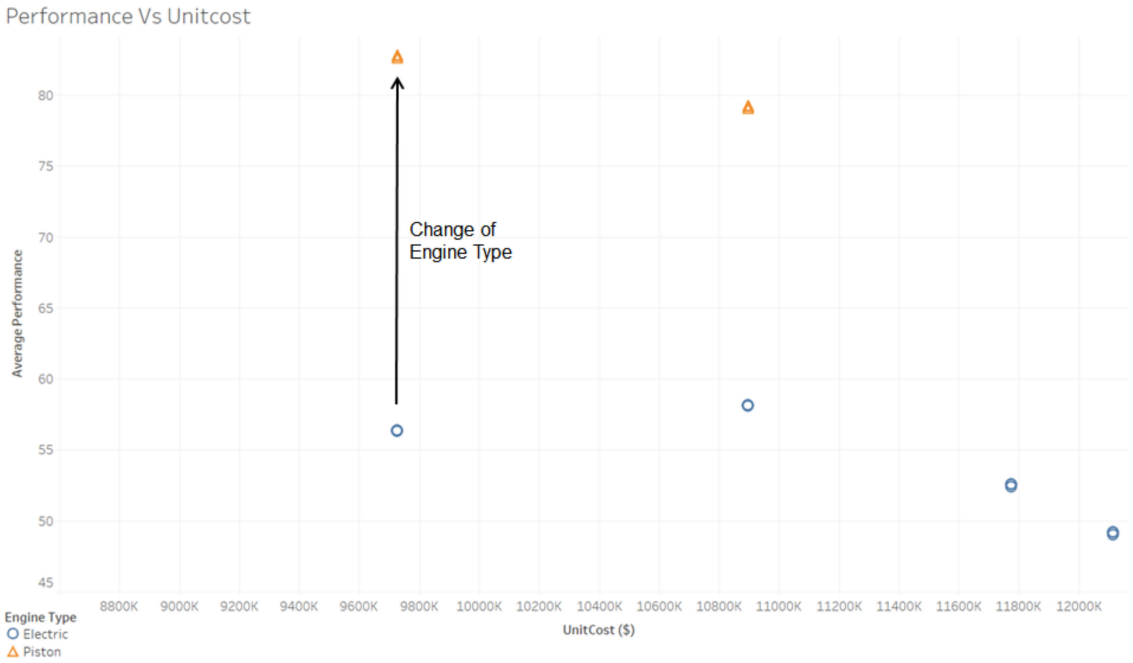

Figure 6. UAS performance change electric vs. piston engine type.

3. Discussion

A primary goal of the demonstration and exploratory research was to demonstrate the art of the possible, which is that an SWT architecture could be developed that would allow digital data to be parsed and queried in such a way that an Excel table could be created for automatic visualization. The demonstration was able to show that the IoIF SWT architecture is capable of retrieving SysML data in JSON format and then transforming it into an Excel data format. This is important not only for advancing the concept of digital thread, but also for advancing the adoption of model-based systems engineering (MBSE) by adding value to the model data [28].

Through the development of the demonstration, there were areas discovered that require further research. One such area is change management and control of data inside the IoIF—including a new area of research on how to handle the version control of single data elements in an SWT environment to enable the AST.

In this demonstration, the decision ontology was aligned with the Basic Formal Ontology (BFO) so that it was easier to integrate, expand, and reuse if needed in the future. Aligning ontologies with an

upper-level ontology like BFO was found to be a best practice for ontology development [29]. BFO was chosen for this research due to the availability of domain ontologies that were already developed with BFO, like the Common Core Ontologies. Along with the utilization of BFO from other researchers in similar domains to the present work, the availability of support from the lead developer of BFO, Dr. Barry Smith, was a factor in choosing BFO.

One of the challenges associated with applying Semantic Web to the IoIF application is that SysML/UML have metamodels, which are not ontologies. For example, SysML properties and classes behave more like different roles of the same thing, which makes reconciling difficult in the IoIF. This research used an ontological realism approach and a model ontology to recognize and assert SysML model constructs as fundamentally consisting of information, and therefore model elements and any liberties taken by the modeler would not yield inconsistencies. This type of treatment also allows the support of multiple modeling regimes. However, this type of issue should be common in the engineering domain, since the language surrounding much of engineering maps poorly to how one would pose an ontology. The current thought is to move toward discipline-specific ontologies. Terms like Product, System, Component, and Assembly are all useful to some extent, but are more or less impossible to reconcile in our current schema with Is-A-type relations without introducing undesirable complexity in taxonomy. The current proposed solution is to recognize these terms as labels applied to types of thing or roles.

Weaknesses found within the current ontology development approach are the gaps in off-the-shelf SWT tools, which require in-house development to be filled. In this demonstration, these gaps were reconciled with Python scripts. However, these gaps are projected to be filled as SWT is utilized by more domains. One of these gaps is terminology management (i.e., the replication of terminology across models), which currently must be handled in-house by unique scripts. Therefore, more research into the generic reconciliation of terminology reconciliation would be beneficial.

While this demonstration was limited to SysML data, other tools can be added together with tool-specific ontologies and queries generated for each tool, as discussed above for SysML. Then, other tool data can be imported and exported using semantic technology. This is a benefit of ontologies over APIs. Once an ontology is developed for the tool, it can be utilized. The ontology may need to be updated when the interface to the tool changes, but the previous patterns in the ontology for the tool should remain constant. Hence the flexibility of ontology versus the brittleness of the current API approach. This has been demonstrated and discussed in [30].

This research also helps to refine the understanding of AST in the context of digital artifacts. The concept of AST in this research first meant attempting to automatically create a single complete digital twin of the designed system. Over the course of the research, AST was redefined as being capable of automatically populating digital artifacts with the best and most authoritative data (at the element level) known at a given point of time in the system lifecycle. Currently, a concept of how this authoritative data would be determined uses queries to check for multiple different key parameters that would point to a data element from one source being better than the same data element from another source. For example, a data element(s) received from a model could be checked using queries and other SWT to determine if and to what extent the data element, and overall model it is associated with, is conceptually aligned with the overall system data model [31]. Through these semantic checks, the data element(s) could be labeled as the most authoritative source for that data element. This process may also require a scaling and scoring system, which will be investigated in future research. The current concept is also to keep a human in the loop so that the AST would enable computationally augmented design, as opposed to some AST research that is focused on automated design.

4. Materials and Methods

For this demonstration, the following software was utilized: SysML tool (MagicDraw), View Editor, MMS, Parser, Decision Ontology, Triple Store, Excel file generator, and Tableau.

This demonstration can be executed from a local computer or over a server. Software and other files that were generated for this demonstration can be obtained from the authors.

Certain commercial software products are identified in this material. These products were used only for demonstration purposes. This use does not imply approval or endorsement by Stevens, SERC, NAVAIR, or ARDEC, nor does it imply these products are necessarily the best available for the purpose.

5. Conclusions

The final goal of the IoIF is to notify users in a digital thread scenario when data do not pass completeness and consistency checks implemented through queries. One example of a basic check would be for correct and consistent units for data elements. The IoIF could flag data and/or be given the authority to determine the digital artifact that is the authoritative source for the specific data element to propagate the correct data to other digital artifacts. The IoIF is envisioned to have two main functions: (1) data acquisition and aggregation; and (2) semantic query and reasoning which allows for consistency and completeness checking of the data. These goals will require further development of SWT support, such as handling the mapping of terminology. Continued ontology and query development will be needed for both the domain(s) of the data and tools. The ontology suite(s) alone will not be a solution, as they are only a small piece of enabling the overall SWT (i.e., queries, reasoning, mapping, etc.). Future research will attempt to demonstrate that an SWT-enabled integration and interoperability architecture may lead to a solution for handling data in the AST.

The next demonstration will also focus on starting to manage data—including changes—inside the IoIF. While the MMS manages data externally to the IoIF, the IoIF will need to be capable of managing data, since the aggregation of data for all digital artifacts is one of the goals of the IoIF. This will start by developing how the IoIF can handle the change management of data.

Future research needs to address how the IoIF can determine authoritative data elements through semantic technology. The IoIF will need to handle and determine the different perspectives of the data elements when they were generated. This perspective will be required to ensure consistent representations of data throughout. What is handled inside the IoIF and what is expected to be an input into the IoIF needs to be distinguished and clarified. Current research is already looking at how digital signoff of data by a user can be implemented, which could be used as one point of determining the quality of data elements [32]. Other research that needs to be taken into consideration is on model theory and conceptual alignment [31,33]. Conceptual alignment and how it is applied at the model level as well as at the data element level needs further research. The current research direction is attempting to see how SWT will allow for data to be evaluated at the instance level using associated meta data such as owner, developer, signoff, change history, consistency factors, completeness factors, and other meta data that will be identified and explored in future research.

Author Contributions: Conceptualization, M.B., M.B., B.K., and T.H.; Data curation, B.K. and J.D.; Funding acquisition, M.B.; Investigation, M.B., B.K., J.D., and T.H.; Methodology, M.B., B.K., and T.H.; Project administration, M.B.; Resources, M.B. and I.G.; Software, B.K. and T.H.; Supervision, M.B. and I.G.; Visualization, B.K., J.D., and T.H.; Writing—original draft, M.B.; Writing—review and editing, M.B., M.B., B.K., and T.H.

Funding: This research was funded by Naval Air Systems Command (NAVAIR), Armament Research, Development and Engineering Center (ARDEC), and The Office of the Deputy Assistant Secretary of Defense for Systems Engineering (ODASD(SE)) with contributing research tasks (RT-48, 118, 141, 157, 168, 170, 176, 195).

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. West, T.; Pyster, A. Untangling the Digital Thread: The Challenge and Promise of Model-Based Engineering in Defense Acquisition. *INCOSE Insight* **2015**, *18*, 45–55. [CrossRef]
2. Kraft, E. Expanding the Digital Thread to Impact Total Ownership Cost. Presented at the NIST MBE Summit, Gaithersburg, MD, USA, 18 December 2013.
3. Blackburn, M.; Bone, M.; Witus, G. *Transforming System Engineering through Model-Centric Engineering*; SERC-2015-TR-109; Stevens Institute of Technology: Hoboken, NJ, USA, 2015.
4. Shani, U.; Jacobs, S.; Wengrowicz, N.; Dori, D. Engaging Ontologies to Break MBSE Tools Boundaries through Semantic Mediation. In Proceedings of the 2016 Conference on Systems Engineering Research, Huntsville, AL, USA, March 2016.
5. World Wide Web Consortium (W3C). Available online: https://www.w3.org/ (accessed on 10 December 2018).
6. Smith, T. What Are the Most Common Issues Affecting Integration and APIs? Available online: https://dzone.com/articles/what-are-the-most-common-issues-affecting-integrat (accessed on 12 September 2018).
7. Bone, M.A.; Blackburn, M.R.; Rhodes, D.H.; Cohen, D.N.; Guerrero, J.A. Transforming systems engineering through digital engineering. *J. Def. Model. Simul.* **2018**. [CrossRef]
8. Blackburn, M. *Transforming Systems Engineering through Model Centric Engineering*; Systems Engineering Research Center: Hoboken, NJ, USA, 2018; p. 240.
9. Biffl, S.; Sabou, M. (Eds.) *Semantic Web Technologies for Intelligent Engineering Applications*; Springer International Publishing: Cham, Switzerland, 2016.
10. Arp, R.; Smith, B.; Spear, A.D. *Building Ontologies with Basic Formal Ontology*; The MIT Press: Cambridge, MA, USA, 2015.
11. Robinson, P.N.; Bauer, S. *Introduction to Bio-Ontologies*; CRC Press: Boca Raton, FL, USA, 2011.
12. Smith, B.; Ashburner, M.; Rosse, C.; Bard, J.; Bug, W.; Ceusters, W.; Goldberg, L.J.; Eilbeck, K.; Ireland, A.; Mungall, C.J.; et al. The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nat. Biotechnol.* **2007**, *25*, 1251–1255. [CrossRef] [PubMed]
13. Bada, M.; Stevens, R.; Goble, C.; Gil, Y.; Ashburner, M.; Blake, J.A.; Cherry, J.M.; Harris, M.; Lewis, S. A Short Study on the Success of the Gene Ontology. *J. Web Semant.* **2004**, *1*, 235–240. [CrossRef]
14. The OBO Foundry. Available online: http://www.obofoundry.org/ (accessed on 9 September 2018).
15. Ashburner, M.; Ball, C.A.; Blake, J.A.; Botstein, D.; Butler, H.; Cherry, J.M.; Davis, A.P.; Dolinski, K.; Dwight, S.S.; Eppig, J.T.; et al. Gene Ontology: Tool for the unification of biology. *Nat. Genet.* **2000**, *25*, 25–29. [CrossRef] [PubMed]
16. Department of Defense. *DOD Dictionary of Military and Associated Terms*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2018.
17. Rockwell, J.; Grosse, I.R.; Krishnamurty, S.; Wileden, J.C. A Decision Support Ontology for collaborative decision making in engineering design. In Proceedings of the 2009 International Symposium on Collaborative Technologies and Systems, Baltimore, MD, USA, 18–22 May 2009; pp. 1–9.
18. Cilli, M.V. A New Product Development Trade-Off Analysis Case Study Using a Small UAV Example. 2017.
19. RDF Schema 1.1. Available online: https://www.w3.org/TR/rdf-schema/ (accessed on 25 October 2018).
20. SPARQL 1.1 Query Language. Available online: https://www.w3.org/TR/sparql11-query/ (accessed on 10 December 2018).
21. Home—Phoenix Integration. Available online: http://www.phoenix-int.com/ (accessed on 2 June 2017).
22. OpenMBEE—Open Model Based Engineering Environment. Available online: http://www.openmbee.org/ (accessed on 19 September 2018).
23. Cilli, M.V. *Improving Defense Acquisition Outcomes Using an Integrated Systems Engineering Decision Management (ISEDM) Approach*, 2016; unpublished.
24. Small, C.; Buchanan, R.K.; Pohl, E.; Wade, Z. A UAV Case Study with Set-based Design. *INCOSE Int. Symp.* **2018**, *28*, 1578–1591. [CrossRef]
25. Introduction—Owlready2 0.10 Documentation. Available online: https://owlready2.readthedocs.io/en/latest/intro.html (accessed on 1 November 2018).
26. Rdflib 4.2.2—Rdflib 4.2.2 Documentation. Available online: https://rdflib.readthedocs.io/en/stable/ (accessed on 1 November 2018).

27. Bone, M. July 2018 Demo Software. 2018. Available online: https://github.com/marybone/July_Demo (accessed on 10 December 2018).
28. Wang, L.; Izygon, M.; Okon, S.; Wagner, H.; Garner, L. Effort to Accelerate MBSE Adoption and Usage at JSC. In *AIAA SPACE 2016*; Long Beach, CA, USA, 2016; p. 5542.
29. Semy, S.K.; Pulvermacher, M.K.; Obrst, L.J. *Toward the Use of an Upper Ontology for U.S. Government and U.S. Military Domains: An Evaluation*; MITRE Corp: Bedford, MA, USA, September 2004; p. 43.
30. Happel, H.-J.; Seedorf, S. "Applications of Ontologies in Software Engineering". In Proceedings of the 2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, 6 November 2006.
31. Tolk, A. Multidisciplinary, Interdisciplinary, and Transdisciplinary Research: Contextualization and Reliability of the Composite. In *The Digital Patient*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2016.
32. Transforming Systems Engineering through Model Based Systems Engineering-NAVAIR | Systems Engineering Research Center. Available online: https://archive.sercuarc.org/projects/transforming-systems-engineering-through-model-based-systems-engineering/ (accessed on 28 November 2018).
33. Diallo, S.Y.; Padilla, J.J.; Gore, R.; Herencia-zapana, H.; Tolk, A. Toward a Formalism of Modeling and Simulation using Model Theory. *Complexity* **2014**, *19*, 56–63. [CrossRef]