34th Annual INCOSE international symposium
hybrid event
Dublin, Ireland
July 2 - 6, 2024

Configuration Management of Sets of Links in a Federated Tool Environment

Adriana D'Souza
Airbus
Aerospace Avenue, Filton
BS34 7PA, Bristol, UK
+44 7793905547
adriana.dsouza@airbus.com

David Hetherington
SSI
1221 Bowers St. #50
Birmingham, MI 48012, USA
+1 844 797 8369
david_hetherington@ieee.org

Géza Kulcsár
IncQuery
1065 Budapest, Nagymező u. 44., Hungary
+36 30 467 9890
geza.kulcsar@incquerylabs.com

Bryan Orozco
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr,
Pasadena, CA 91011, USA
+1 818 354 4321
bryan.d.orozco@jpl.nasa.gov

Aleksander Przybyło
Boeing
3003 W Casino Rd,
Everett, WA 98204, USA
+1 425 418 2667
aleksander.przybylo2@boeing.com

István Ráth
IncQuery
1065 Budapest, Nagymező u. 44., Hungary
+36 30 467 9890
istvanrath@incquerylabs.com

Abstract. In the era of document-centric systems engineering, large organizations developing complex systems had practical methods of associating drawings, specifications, requirements and other information with each other. Mostly these evolved methods involved part numbers and drawing numbers, although sometimes the methods included mundane techniques such as storing drawings in specific drawers in specific filing cabinets. These manual methods were very labor intensive, did not handle changes gracefully, and were somewhat error prone. As computer software and modeling tools began to displace paper drawings and filing cabinets, a "connect and forget" style of linking evolved. We can see this sort of thinking in the relationships in UML and SysML, but also in other places such as the use of Universally unique identifiers (UUIDs) in the XMI file format underlying UML/SysML, the relationships in DOORS and related requirements tools, as well as the original architecture of OSLC which depended on URIs embedded in design artifacts to establish relationships. While these "click to connect" features provided increased convenience compared to the previous manual numbering approaches, these "hard-coded" links have

introduced a new set of problems. The root of most of the new problems comes in the difficulty of managing this sort of extremely large set of direct links as configuration items in their own right. In a larger system, this sort of link set quickly grows to thousands or even millions of links. Managing change, modularity, variants, subsystem reuse, and so on all become very difficult in the presence of such a large, unplanned, and uncontrolled link set. In this paper, we will review the different approaches to creating and managing such sets of links and provide a concise best practices recommendation for the configuration management of such sets of links.

Introduction

Managing relationships between engineering artifacts is not a new discipline. Engineers have been performing this task for as long as the design of a product could not fit on a single drawing. While the product complexity has been growing so have the software system architectures to manage their design. Today, it is all but obvious that in order to bring a product to market, engineering data will have to be federated across multiple engineering software applications, since no single monolithic tool can handle all the tasks that need to be performed. The relationships between the artifacts managed in the different tools in this federated architecture are the glue that holds it all together. This paper explores the different mechanisms that exist for managing those relationships and discusses pitfalls of each one of them. Before concluding, the paper provides a list of critical capabilities a complete engineering data management solution must provide.

Evolution from Paper/Document Centric to Digital

Paper/Document-Based Era

In the document-based engineering world, through decades of lessons learned, best practices have emerged to relate content across documents. Picture sheets detailing installation plans call out parts by their base number to indicate that any part within the same family (i.e. with a matching base number) can fulfill the role regardless or its "dash number" or revision. In digital terms, the actual part number and revision are resolved by late binding at the time the part actually gets installed on the final product in the factory. The hypothetical link between the picture sheet and the drawing that this method implements would exist between the picture sheet revision and the object aggregating a hierarchy of all dash numbers and revision.

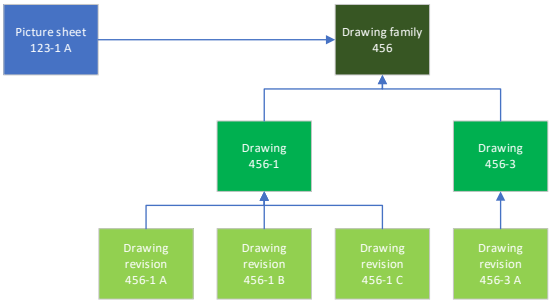

Figure 1. Managing relationship lifecycles through partial ID references

The main rationale for using this method is that revising a drawing for a detail part does not change the way the part is installed, hence there is no need to update the picture sheet and installation instructions and unnecessarily spend countless hours going through engineering change boards. Various processes have emerged for instilling semantics into part numbers and revisions (e.g. form-fit-function) that govern the business rules for when methods like this can be employed.

Those methods have not translated well into modern digital tools where engineering data is managed as a connected set of uniquely identified digital artifacts. More often than not, making minor changes causes ripple effects across the system by reconnecting links and forcing revisions to released artifacts.

The example described here is a simple one, but things quickly get complex when the relationships involve evidence used for certification, artifacts owned by different organizations, etc. The business rules built into the lifecycles of the links require establishment of thoughtful data models that support the needs and behavior expectations.

Digital Linking Era

In the evolution of digital engineering, one of the key challenges has been adapting traditional paper-based linking methods to the digital realm. This transformation has been pivotal in managing data relationships and ensuring efficient information retrieval and utilization.

Primary and Foreign Keys in Relational Databases: The initial approach in digital linking mimicked traditional index systems, utilizing primary and foreign keys in relational databases. This method provided a straightforward way to establish relationships between different sets of data. However, it was limited by the scope of a single database and often required complex joins for retrieving related information across tables.

Federated Databases and Universally Unique IDs: The advent of federated databases necessitated a shift towards universally unique identifiers (UUIDs). This development was driven by the need to reference data across multiple, distributed databases. UUIDs offered a more universal solution, enabling data to be uniquely identified regardless of its location. An example of such a strategy is elaborated in the subsequent Section on Overview of Methods, in the subsection on Smart Numbering.

Namespace-based Approaches: An alternative to the UUID approach is namespace-based thinking. This method involves organizing data into distinct namespaces, each with its own identification scheme. This approach aids in disambiguating similar identifiers and allows for more structured data organization. In this context, various identification mechanisms have emerged, including fully qualified names, URLs, URIs, and even e-mail addresses for specialized purposes. While being easily readable by humans, this comes with the drawback of links easily deteriorating or becoming dangling. A general introduction to this kind of thinking, focusing on the capacities offered by OSLC and UAF, can, again, be found in the following Section, in the subsection on Semantic Understanding of Linked Data.

Cross-referencing Data Exchange / Serialization: In scenarios involving data structure exchange, the motivation for cross-referencing shifts towards pointer serialization. This process involves converting pointers or references in data structures into a format that can be stored or transmitted, facilitating data exchange between different systems or applications. Also related to OSLC and similar uniformization initiatives (see above), further details on fostering efficient data exchange scenarios are found under Standard APIs and Schemas in the subsequent Section.

In conclusion, the adaptation of paper-based linking methods to the digital realm has undergone significant evolution, driven by the needs of federated databases, data structure exchanges, and the challenges of a distributed digital era. Issues such as dangling edges and broken links are common, resulting from the deterioration of references over time or changes in the underlying data structures. To address these challenges, *digital thread platforms* (e.g. Syndeia (Syndeia, 2023, Bajaj et al., 2017), Smartfacts (Smartfacts, 2023), IncQuery (IncQuery, 2023), OpenMBEE (OpenMBEE, 2023, Kruse & Blackburn, 2019), etc.) have emerged, elevating the status of links to that of first-class citizens. Additionally, modern Product Lifecycle Management / Engineering systems (e.g. Polarion, Jama, PTC, etc. – for conceptual details, cf. Configuration Awareness of Linking Capabilities at the end of the subsequent Section) also incorporate powerful

features to manage traceability links that connect various digital engineering artifacts, like digital thread principles. All these platforms adopt a philosophy akin to semantic web technology, inheriting both its advantages and limitations, such as costly resolution of links leading to scalability issues, highlighting the ongoing need for innovation in digital engineering.

Overview of Methods

Multiple methods of maintaining configuration management across federated systems have emerged since we started gradually moving towards more digital product data management. Those methods lie on a spectrum going from loosely coupled to tightly coupled

Smart Numbering

Of all integration methods, referencing based on smart numbering is probably the one that most loosely coupled data in a federated set of applications. Rather than creating and maintaining links between the different applications, each application references data from other applications by truncating parts of element identifiers whose changes it is assumed will not impact the validity of the reference. This is very similar to the methods used in paper days described above.

While this method is very easily established, it doesn't make use of technological advances that allow more flexibility, more efficient relationship traversal, more specificity etc. Also, in practice, this method is often abused to easily bypass costly and time-consuming engineering change boards, by "slipping through" major changes as minor (e.g. changes that should require a new part number are fast-tracked by changing the revision number).

Linking by Early/Late Binding

One way to instill more flexibility into linking mechanisms is by controlling when revisions of linked data are resolved. Early binding means that the link explicitly and persistently specifies the revision or variant of the linked data, while late binding means that the revision or variant is resolved at a specific time (e.g. when the product is baselined or when a unit is delivered to a customer). The key difference between late binding and simple smart numbering is that with late binding an artifact that specifies the revision or variant of the linked element is created and can be preserved for traceability. Early and late binding methods are well understood in software engineering and have been used for decades. Systems such as Gradle and Maven for example allow software developers to specify hard dependencies against a specific version of a library or simply use the latest one that is available. Same as in software engineering where we compile the source code in order to build a deliverable binary, in manufacturing engineering models and other engineering artifacts need to be baselined in order to establish the set of data that can be delivered to a supplier, customer or regulator.

In Model Based Systems Engineering (MBSE), modular model features in different tools often allow late or early binding but rarely both. Also, the capability to establish traceability with the baselined models when an early binding method was used is virtually always missing. What this means is that it is impossible to answer questions such as "which established baselines are affected by a requirement change?"

Centralized Configuration

On the other end of the loosely/tightly coupled spectrum is the centralized configuration method. Essentially, it requires a configuration managed structure that acts as the single index of all configurations in all

applications. This method (often called configuration hub or hub and spoke) turns engineering applications into simple authoring apps and takes away all of their configuration management responsibilities.

This approach can seem very appealing as, at least in theory, it eliminates all integration hurdles but in practice is nearly impossible to implement for multiple reasons which vary from external to internal. The primary reason is that the different applications which OEMs use are often developed by competitors who refuse to allow their data to be managed by a competing product. Another reason is that it often requires the existence of data standards that either do not exist or whose fidelity is insufficient to properly represent the data. Finally, within the OEM itself, the data is owned by different organizations or divisions who refuse to allow the data they own to be managed by others.

Smarter Linking

Smart Numbering

Smart numbering emerged as a common practice for embedding information about an object into its identifier. While almost every manufacturing company utilizes smart numbering, there is virtually no standardization on the patterns that should be used to create or decode a part number. On top of identifying where in a product structure a part is installed, what material it's made of etc., it is common to also structure the part number in a way that a change in its structure is informative of the type of change that was made to a part and its potential impact. As described above in the Paper/Document-Based Era section, linking technologies can utilize this information to automate link propagation and change event notifications.

A common challenge with this method is that the smart number patterns are rigid and require multiple "decoder rings" due to the lack of standardization. The rigidity can cause year 2K types of issues when an insufficient amount of space was given for a piece of data. The lack of standardization can cause confusion, for example when parts are collected from different suppliers with different part numbering schemes.

While smart numbering has been used for decades, it is still evolving, and a lot of research focuses on enhancing the effectiveness of its use by both standard bodies (International Organization for Standardization, 2022) and individual experts (Dullaart, 2023).

A major challenge that companies often face when using those methods is deciding which organization will own and manage the use of the numbering patterns across the company. This is a key enabler for the configuration management of the set of links for a given system.

Semantic Understanding of Linked Data

Open Services for Lifecycle Collaboration (OSLC) aim at connecting data across various platforms using a similar approach to the web

Figure 2. OSLC representation (OSLC, 2023)

Links are defined between RDF files that define in an XML format that predicates for the resource and specify the namespaces for the resource types. On one hand it is positive that we have links between various tools, however the links are hard wired in the RDF/XML format between the object types.

One challenge of the basic OSLC design is that it is designed around the peer-to-peer architecture of the semantic web. No one is in charge. There is also no inherent concept of a set of links going to a well-defined subsystem. Each link is made individually and managed individually. If a manufacturer decides to substitute a major end item component, there is no immediately obvious place where the links can be disconnected and reattached as a set. Instead, tools that are currently offered to manage large OSLC configurations end up having to query and spool all the links into a central database so that queries can be run to see what is connected to what.

The Unified Architecture Framework (UAF) is based on the Unified Modeling Language (UML), Systems Modeling Language (SysML), the Unified Profile for the U.S. Department of Defense's Architecture Framework (DoDAF) and the U.K.'s Ministry of Defence's Architecture Framework (MoDAF) (UPDM) and the North Atlantic Treaty Organization's (NATO) Architecture Framework (NAF). When these military requirements were combined with business sector requirements (90% of concepts and themes captured in the military frameworks are equally applicable in the commercial domains), the UAF was born which serves both commercial and military interests (OMG UAF, 2023).

The relationships defined in the UAF metamodel are only allowed between predefined element types. This is useful as only those types of links are allowed. The instantiation of this link type is then used in the actual model. In this way of working the links are standardized and fixed. The modelling can only instantiate the links from the generic ones. UAF does have limitations though when connecting UAF elements generated in different tools as there is no standard interfaces defined (this is intended to be solved with a future revision to UAF based on KerML).

Standard APIs and Schemas

OpenAPI provides a standard for interactions with HTTP APIs. It gives guidance for creating a specification document in which an application can document the HTTP routes available and the structure of the response in each route. When an application has a standard OpenAPI specification document, Software Development Kits can be automatically generated. Open-Source Systems Engineering Tools like OpenMBEE's Model Management System 4 have benefited from creating OpenAPI Specifications and created Python clients from this specification document. While this solution enables communication with one tool, it becomes

challenging to persist changes across different environments. One of the pitfalls in this approach is the fact that each tool implements its own schema.

Open Services for Lifecycle Collaboration (OSLC) standardized the way in which applications shared data with each other. It provided a querying standard for how to retrieve information, as well as a structure for how to interpret it. With specified structures such as those in Figure 1, a shared vocabulary was created. The interpretation of the data exposed through OSLC was defined by a given "domain". Domains are specified by OSLC working groups, such as change management, test management, requirements management and configuration management. OSLC created a common way to share information across engineering tools. Each resource in OSLC has an RDF representation, allowing for rich data structures.

The issue lies with many software tools not having OSLC implementations. Enterprise applications such as Jira and GitHub rely on their own REST API for the ingress and egress of information.

Configuration Awareness of Linking Capabilities

Product Lifecycle Engineering (PLE) has emerged as an efficient method to manage reuse of model contents across multiple product variants in a product line or family. While it greatly helps drive down maintenance cost of model data by decreasing replication, it also introduces major complexities because no two tools manage variants the same way. The impact to linking approaches lies in the fact that the linking platform is typically not aware of the applicability (range of products on which an element can be used) and effectivity (range of products on which an element is used) of the linked element and therefore cannot properly assess the level of impact of a change to a linked element. For example, a process specification could be linked to hundreds of operations, but those operations may be applicable to a product variant that very rarely is ordered by customers. Hence, a change to the process specification may be identified as a high impact, when in fact it almost never gets invoked. On the other hand, a process specification linked to a single operation may be considered low impact, even though it is linked to an operation performed on every single build unit.

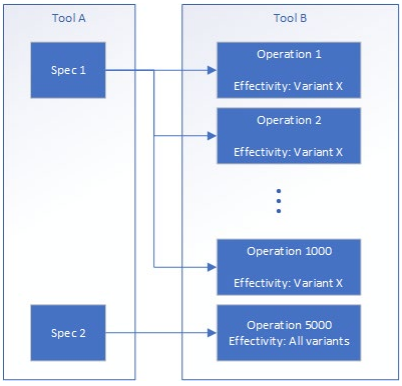

Figure 3. Links with Variant Management

A much more critical issue arises when the links are used for certification or safety assessment of a product. A part of a design may satisfy a regulatory requirement but at the same time be an optional component. When analyzing the design for completeness of requirement coverage, if the linking platform is not variant configuration aware, the analysis will not detect potential issues such as requirements not covered in specific build units where an option was not selected.

Position statement

In the last few years, a multitude of digital thread enabling applications have been released in the market. Most of them provide significant improvements in terms of user interfaces and variety of data types that can be integrated but very few focus on what actually makes the linking problem hard: maintaining configuration management across diverse data sets. For an application to properly address the challenges of linking across a digital thread, it must have the following features:

- Ability to choose between early and late binding of revisions and variants during the development phase
- Detection of suspect/broken links (i.e. links that need to be reviewed after linked data has been revised, deleted or deprecated) with the capability to provide recommendations for replacement
- Ability to break the chain reaction of change impact by stopping the impact notification propagation where the impact stops (e.g. if it has been determined that a requirement change has no effect on a function that satisfies the requirement, the owner of the logical component that performs the function does not need to be notified)
- Customization of the configuration ownership of data sets, referred to as configuration control zones (Scotto d'Apollonia, 2016), including links (e.g. a satisfy link between a function and a requirement can be placed within the configuration control zone of the function).
- Inspection of past revisions of links including revisions and variants of linked data consistently with all reference data
- Configuration management of the overall set of links in a distributed way (i.e. in the absence of a centrally managed global configuration)
- Promotion of internal model features as external interfaces for the purpose of linking models at higher abstraction levels (enables model modularity and reuse and decreases overhead of link management)
- Ability to evaluate a link's status across various internal configuration engines (PLE, variant management, effectivity etc.)

Conclusion

Managing relationships between different engineering artifacts is a crucial and critical capability every engineering organization must master. While it is by no means a new discipline or one only applicable to the digital era of engineering, there is little consensus on how it should be performed and the tools currently available on the market still lack some or most of the features required to address challenges engineers face daily. Methods developed during the paper and document centric era have first been applied as-is into the digital tools and then, in most cases, missed the opportunities provided by more advanced hardware capabilities and data management evolutions.

In order to reach their full potential, modern engineering data platforms must evolve to embrace data federation across multiple disciplines and organizations while maintaining configuration management. Some approaches to achieve this may seem good on paper but crumble when put into practice due to the volume of changes, variability of the data and number of parties involved. A robust and scalable solution must exhibit some characteristics that are listed in this paper.

Acknowledgment

Part of the research for this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

Bajaj, M., Backhaus, J., Walden, T., Waikar, M., Zwemer, D., Schreiber, C., & Issa, G. (2017). Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems. *INCOSE IS,* 27(1), 151-169.

Balslev H., & Barré, T. (2023). Common Language for Systems by the ISO/IEC 81346 Reference Model. *INCOSE International Symposium 2023*, Honolulu HI, 15-20 July 2023

Dullaart, M. (2023). *The Essential Guide to Part Re-Identification: Unleash the Power of Interchangeability and Traceability.* ISBN: 979-8395057853

IncQuery (2023). Website of IncQuery, visited 10 November 2023, https://incquery.io

Intercax (2023). Website of Intercax Syndeia, visited 10 November 2023, https://intercax.com/products/syndeia

International Organization for Standardization (2022). *IEC 81346-1:2022 Industrial systems, installations and equipment and industrial products*.

Kruse, B., & Blackburn, M. (2019). Collaborating with OpenMBEE as an Authoritative Source of Truth Environment. *Proceedings of the 17th Annual Conference on Systems Engineering Research (CSER), Procedia Computer Science,* 153, 277-284.

OpenMBEE (2023). Website of OpenMBEE, visited 10 November 2023, https://www.openmbee.org

OSLC (2023). Open Services for Lifecycle Collaboration, visited 19 September 2023, https://open-services.net

OMG UAF (2023). Introduction to 'Unified Architecture Framework' (UAF), https://www.omg.org/uaf

Scotto d'Apollonia, A. (2016). Multi-view Bill of Material. *CIMData 2016 PLM Road Map for Aerospace & Defense.*

Smartfacts (2023). Website of Smartfacts, visited 10 November 2023, https://smartfacts.com/Syndeia

Biography

Adriana D'Souza. (CSEP) is a Configuration Management Process Architect for Systems for Airbus, having previously worked as a Systems Architect and as a Design and Development Engineer on challenging and complex projects like large border security projects and air traffic management projects both at the subsystem, system and system of systems level in the Airbus Group. She was awarded an Honours M.Sc. in Computational Science and Engineering from the Technical University of Munich and a B.Sc. in Mathematics and Computer Science from the Ovidius University of Constanta.

David Hetherington is a Principal at SSI with over thirty years of experience in Software and Systems Engineering in the Automotive, Semiconductor, Oil and Gas, Enterprise IT, Server Computer, Telecom, and Electronic Publishing sectors. David has experience in automotive radar design, software safety for offshore oil rigs, infotainment software design, and ISO 26262 functional safety for automotive "System on Chip" products. David is the author of the SysML for Beginners book series and received a BA in Mathematics from the University of California San Diego and an MBA from the University of Texas McCombs School of Business.

Dr. Géza Kulcsár is a senior researcher at IncQuery Labs. He holds a PhD in computer science from the Technical University of Darmstadt, Germany. His research interest and expertise ranges from the theory and semantics of graph transformation to the methodology of systems modeling applications in the future industry.

Bryan Orozco is a Software Systems Engineer at Jet Propulsion Laboratory working on the Computer Aided Engineering Systems and Software team. He has delivered software solutions used for systems engineering in the Mars Sample Return mission. As a member of OpenMBEE, Bryan is one of the core software developers. He holds a Bachelor of Science in Computer Science from the University of California Riverside.

Aleksander Przybyło. System integrator and architect at Boeing. Specializes in Model Based Systems Engineering, Configuration Management and Data Analytics. Successfully led multiple enterprise wide deployments of software solutions for data integration, configuration management, graph-based analytics and integration platforms and data distribution systems. Holds master's degrees in Mechanical Engineering and Computer Science.

Dr. István Ráth holds a PhD in software engineering and has profound experience in several industrial domains such as aerospace, automotive and industrial automation. His vision and main drive is to build the best digital engineering tools for the future industry.