Emerging Needs and Considerations for Digital Engineering Software Tools

AIA AEROSPACE INDUSTRIES ASSOCIATION

Furnish

Share

Own

The Tech Stack

Table of Contents

Acknowledgement of contributing members of the AIA Engineering Management Committee

Jason Boss	Mechanical Engineering Sr. Manager	L3Harris Technologies
Arif Dhanidina	Advanced Programs Cybersecurity	Northrop Grumman
James Hoskins	Product Lifecycle Management Engr	The Boeing Company
Mark Ramos	Senior Principal II Systems Engineer	BAE Systems
Gery Mras	Director, Lifecycle Management	Aerospace Industries Assoc.
Paul Embry	Chief Engineer, Digital Engineering	L3Harris Technologies
Young Park	Enterprise Engineering	RTX, Collins Aerospace
Paul Segura	PLM Engineering Function Leader	The Boeing Company
Peter Pan, PhD	Manager, Mission Systems	Northrop Grumman
Brad Sumner	Head of System Design & Sub-Systems Eng.	Rolls Royce North America
Bill Luk	System Engineering Development Mgr.	BAE Systems
Jason Hill	Senior Principal II Cyber Systems Architect	BAE Systems
Jamie Wightman	Systems Architect	Lockheed Martin
Tim Walden	Chief Engineer, Digital Transformation	Lockheed Martin

1 Background & Goals

The aerospace and defense (A&D) industry has vast supply networks and diverse products necessitating a very broad portfolio of tools and processes. Now, with the increased adoption of *digital engineering and the shift to models as the record of authority,* these digital artifacts must be shared across organizational boundaries and used for many years. Tools and data will no longer exist in isolation, so we must consider how each has a place in a broader "systems of systems" context. The broad set of use cases, focused on sharing, collaboration, and reuse, will be best achieved through interoperability, which is employing tool-agnostic approaches and standards for information exchange. There is a need to accelerate the path to interoperability, as early efforts employing tool standardization methods had undesirable effects such as loss of upgradeability, higher costs, lack of flexibility, and reduced usability of data. Currently, several interoperability capabilities are lacking and need to be addressed. This document conveys these capability needs from the perspective of experts from aerospace companies that use and administer digital engineering software tools, explaining the anticipated needs and benefits. While this paper was developed from an engineering perspective, it is anticipated that similar themes are relevant to other domains. These capabilities have been broken down into nine categories summarized below.

1. ***Interoperability for Visualization & Collaboration*** – Support for industry standard digital formats for sharing, retrieving, and viewing of information.

2. ***Automation*** – Availability of Application Programming Interfaces (APIs) based on industry standards in tools.

3. ***Master Data Management*** – Enablement of authoritative models and data through interoperability with configuration and data management systems.

4. ***Digital Continuity*** – Ability to access legacy data in a manner that does not impose significant data preservation activities as tool versions and proprietary data formats evolve.

5. ***Accuracy & Usability*** – Trust that analysis and other computational methods performed by tools will be technically accurate against industry standards.

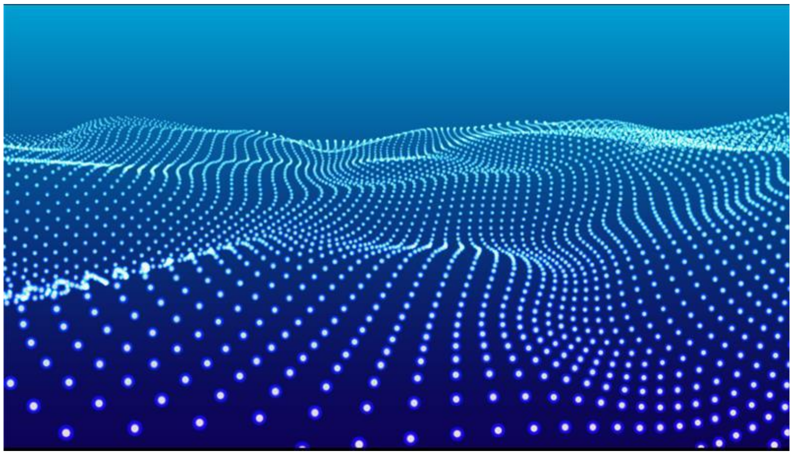
6. ***Security*** – Security requirements and security review processes that a software tool is required to be vetted against to be used in unclassified and classified applications in the A&D industry.

7. ***Flexible Licensing*** – Licensing models and administration approaches often affect how a software tool can be used, and with the diversity of con-ops and processes across companies, having options in licensing models will enable optimization of digital engineering.

8. ***Intellectual Property (IP) Management*** – Data generated often has mixed ownership and rights, and as models and data sets grow in complexity, the methods used to identify and persistently communicate data's IP status will need to evolve with it. Further, as systems can "learn" from proprietary datasets, and communicate with third-party clouds, clarity on how that learning data is managed and owned is also important.

9. ***Emerging Tech Compatibility*** – There are also several new emerging trends identified, which this paper captures at a high level; as they mature, additional considerations are anticipated in the future.

2 Guardrails



2.1 Interoperability for Visualization & Collaboration

Tools should have the capability to publish digital data into industry standard formats to enable accessibility of the data without requiring specific proprietary tools and licensing.

2.1.1 What is it?

Engineering processes require the use of a multitude of software tools, and digital engineering initiatives across industry are driving significant growth in the quantity of software tools used. Currently, each software tool has its own unique and proprietary file format required for use. While this is acceptable for inter-organizational work, there is a need for the information also to be published into a form that is agnostic to a specific software tool, based upon open standards. Standards-based data formats are vital for the scalability and long-term viability of digital engineering, and it must become the primary mode of data interoperability between organizations, software tools that perform similar functions, and storage strategies for Long-Term Archival and Retrieval (LOTAR).

2.1.2 Why is it important to digital engineering?

Standards-based interoperability is important to digital engineering because it will break down barriers between organizations that need to share data. Lack of data interoperability in early digital engineering efforts triggered many organizations to lock onto a specific software tool and version as a standard for exchange. While near-term objectives were met in these cases, the approach hindered software tool innovation and upgrades and drove up the unnecessary purchase and training of software tools for the sole purpose of data exchange. Furthermore, particularly with longer lifecycle products, unplanned costs were realized with maintaining and upgrading software tools for the sole purpose of retaining access to IP generated by the software tool.

2.1.3 What are some of the challenges?

This lack of interoperability is driving a broad array of approaches to collaborate and share, many of which drive up cycle time, cost, and confusion associated with the source of truth. This friction in data exchange processes through manual or point-to-point approaches is not desired. Some of these techniques include spending time deriving documents from models for exchange (in effect de-digitizing information to move it), attempts to standardize tools down the supply chain, or developing custom processes to convert information to source and target unique systems for each data exchange. Currently, there are standards for many digital engineering data types, each at different levels of maturity; in some cases, there are multiple standards to choose from. However, many of them are lacking in implementation in software tools for direct use in publishing data. The gaps are attributed to a few different factors and must be established case by case. In some areas, the standards need revisions to keep up with the evolving technologies they support, and in other areas, the software tool developers have not fully implemented the standards due to a lack of clarity of its intent to use or maturity, and therefore cannot justify the investment.

2.1.4 What are some recommendations and best practices?

Ideally, there would be a mature industry standard and data/file format established for each element that needs exchange, and the formats would be easily ingested into visualization, collaboration, and development tools. These neutral data formats would then be established as the primary data exchange method between organizations and would be established in agreements in lieu of calling out any specific software tool. To move more quickly towards these goals, each major data type for digital engineering needs to have an established neutral standard(s) selected, matured, and then implemented into software tools. Achieving such an objective might be established by first developing a standards list with maturity to be worked by the appropriate industry working groups and standards bodies, with software tool developers directly engaged to establish each data type with acceptable standards.

2.2 Automation

Tools should have APIs based on industry standard interfaces available without IP-sharing requirements for custom-built extensions.

2.2.1 What is it?

Automation is the use of software tools and technology to perform tasks and processes that are traditionally performed with direct human interaction. Automation is achieved through the integration and customization of multiple tools into tool chains. In this context, tool integration is the ability for different tools to operate with each other and exchange data, while customization is the ability to tailor and extend the functionality of a tool, automating workflows and processes. Extensions may also be desired to look across multiple data management systems and software tools in order to locate, identify, or create relationships between specific datasets associated with an engineering use case. Open APIs, along with extensible architectures and open data formats, are key enablers for introducing integrations and customizations.

2.2.2 Why is it important to digital engineering?

To achieve a high level of digital engineering maturity, organizations must be able to capture and automate their business, engineering workflows, and processes, and they must be able to access, connect, and use their data effectively. A typical digital engineering ecosystem is composed of a collection of disparate tools with many of

the tools uniquely qualified to perform a task. The data that flows in and out of a tool is an essential element of a process and is often not inherently compatible with other tools employed in the process, nor would it be fiscally advantageous for a software tool vendor to provide such a product or company-centric experience in their general offerings to industry. This results in the need for predictable and sustainable APIs into software tools to build out the required integrations and customizations required to achieve digital success for the unique needs of an organization.

2.2.3 What are some of the benefits and challenges?

Customizations enable an organization to automate tasks, integrate, and port data across tools, and introduce new functions and capabilities. Customizations are a catalyst for agility and innovation. They can lead to development and introduction of new products and features not originally envisioned by the tool provider, as well as significantly accelerate the development process. This enables an organization to codify its IP and discriminating capabilities for increased customer satisfaction and knowledge management. For organizations, customizations and open data formats help reduce risk by providing a means for ownership of their data and migrating to other tools when needed. Within the A&D sector, there are many complex processes that are developed to meet customer requirements and regulations that govern quality and safety of the system. While these processes are vital to overall success, they are costly and time-consuming, and having the ability to build out tailored and automated workflows to address these aspects of the acquisition process are fundamental needs of the A&D industry’s goals of moving more quickly to address emerging threats.

Along with the many benefits, customizations also pose many challenges. Vendor-defined APIs and data formats must be stable, well documented, and carefully maintained. Vendors will need to consider maintaining backward compatibility when updating their tools to avoid breaking existing customizations. Customizations will likely result in a need for increased technical support. Vendors may need to support tool uses that are outside of their control. These factors can lead to increased complexity and cost to vendors as well as constrain their own ability to deliver innovations.

2.2.4 What are some recommendations and best practices?

Vendor and user organizations can take several actions to realize the benefits of tool-supported integrations and customizations. Organizations that develop and use customizations should actively partner with tool vendors to help define APIs, data standards, and capabilities. Organizations that develop customizations should also anticipate future changes to the APIs and data formats and utilize design patterns to help mitigate the impact changes may have on their integrations and customizations. At a minimum, tool vendors should provide the capability to perform a tool’s primary tasks through a well-defined API. Tools should provide a defined data format as well as support the ability to import and export data into the tool, leveraging open standards. Following established standards can reduce the support burden on vendors as well as increase usability and portability by customization developers. Even when using standards, tool vendors should provide clear and complete documentation of their chosen APIs and data formats as well as provide usage examples. Tool vendors should also avoid imposing IP restrictions that may prevent organizations from sharing or owning the customizations that they create.

2.3 Master Data Management

Tools should support industry-standard data interface requirements to be interoperable with configuration and data management systems (i.e. PLM), with preference for data managed at the object level.

2.3.1 What is it?

To fully embrace digital engineering, having configuration and data management systems in place to store, track and verify artifacts, and thus become the authoritative source of truth, is a core requirement. This is critical for delivering information, acting on information, and reducing redundant artifacts that are generated to track information.

2.3.2 Why is it important to digital engineering?

Without clear traceability, models and data cannot be used for official business, and when this is the case, reverting to documents in a basic version control system will continue to exist and create a lack of trust in model data. When digital artifacts are actively managed and verified, they can then be used across the entire product lifecycle to extract value, either through direct use or through the derivation of new datasets from them. Data management systems also offer significant efficiency gains by providing tools to find the correct data quickly, as well as link across systems to find relevant data, as those links are traces for a full lifecycle view. Examples would include building manufacturing and quality machine programs from engineering design models or automating tests from systems engineering models. If adequately adopted across the lifecycle, more advanced concepts like Digital Twins become much more achievable and cost effective to generate and manage. Fortunately, with the current state of technology, there are many options available to put digital artifacts under configuration control; however, there is room for improvements.

2.3.3 What are some of the challenges?

One challenge with the way configuration management systems are developed and deployed in the current state is the varying level of integration between authoring tools and data management applications depending on where they are sourced from, and they are primarily file-based. The nature of the products designed in this industry often require a very diverse toolbox in order to capture the entire technical baseline. In the current state, however, there are tight couplings between specific authoring and data management systems, particularly by model type and by software supplier, and they are often specified down to specific versions of applications, which creates “configuration lock.” Data management systems must be tailored for specific model types: systems, software, hardware, etc., with rich integration between authoring and data management systems. That form of rich integration enables better management of data at the object level, as well as enhanced collaboration methods, tailored management processes, and graph-based interrogation to drive efficiencies. However, in the current state, such a rich experience is only offered when data management and authoring tools are sourced from the same software supplier, which drives an increase in application complexity for core data management processes.

2.3.4 What are some recommendations and best practices?

Addressing this key challenge, data management systems should be improved to be more open and agnostic to authoring tools. Interoperability should be inherent with all authoring applications of similar model type to enable holistic data management and allow for more flexibility for design and analysis toolboxes, etc. It is not expected or desirable that a single software tool provider will be able to build the one-size-fits-all solution for an engineering firm, given the diversity of products and needs that they have. This is not a feasible goal to standardize on; rather, it is needed to enable a suite of capability to allow data management systems to support a diverse array of customer toolboxes. To achieve this goal, the data standards for models would need to have some consistency for the data models and information passed between a model and a data management system that can be leveraged for a more common experience. The end vision would be that the data management experience for any equivalent authoring tool is the same within a data management system using consistent standards for the interfaces between them.

2.4 Digital Continuity

Tools should not force data migrations or large-scale verification efforts of legacy data when upgrading to new versions. Further, as tool versions and formats evolve, the ability to access legacy data unmodified should be retained in new versions.

2.4.1 What is it?

Software tools evolve at a rate significantly faster than the products they are used to design. As such, it is a constant challenge for organizations to keep current on the latest technologies and maintain the data associated with previously developed systems that are in operations and sustainment. It has been observed that for a generation of a software product the data stays accessible and useful. However, when generational upgrades are made and the previous software tools are obsoleted, large amounts of data are subjected to forced migration or deprecation. The owners of the data must at that point decide to either spend large sums of money investing in data migration, remastering, and verification activities, or take the risk of accessing the data through unsupported software for an indefinite duration.

2.4.2 Why is it important to digital engineering?

The long-term access and use of models and data will be foundationally required as they become the authoritative source of truth. This will become a key concern, as the timing and occurrences of these types of data obsolescence events are not well known to the end users of the software products they use. As such, these become unplanned costs or risks that have become generally accepted as “the cost of doing business.” It is extremely important for digital engineering to drive improvements in how generational upgrades to software affect existing data and gain more proactive predictability into when these events will occur in order to adequately plan for them.

2.4.3 What are some recommendations and best practices?

Organizations can consider several practices to help mitigate these concerns. First, end-user organizations should inquire and track these significant milestones for software tools that are in use that generate authoritative data sets, and plan for data risk mitigation factors. Second, exploring long-term software tool access through emulation or other legacy software tool support may be enough for some sustainment use cases. Third, software tool companies might consider offering lightweight applications or stronger backwards compatibility features in their tools as a standard offering. In any case, it should be noted that the large, resource-intensive data migration experiences across industry in the past few decades must be dealt with and prevented. It is also important to note that these use cases are focused on systems that are in active use and maintenance and employing a long time archival and retrieval method may be a factor, but that is not in itself sufficient for supporting day-to-day maintenance of work in progress on a long lifecycle system that outlives multiple software tool generations.

2.5 Accuracy & Usability

Tools should be validated to be technically accurate against industry standards (ASME 14.x, ISO STEP, etc.).

2.5.1 What is it?

There is an opportunity for software tool suppliers to demonstrate compliance and usability of the tool outputs against industry-wide available benchmarks, particularly in analysis software tools. Existing industry standards could be used for the benchmarks, or other academically accepted references that are available. While these software tools are generated with rigorous considerations for methods and accuracy, it is often difficult to extract from initial demos and inquiries where the edge cases are for their use. In the A&D industry, where a large amount of this work occurs for systems with significant human safety concerns, this difficulty triggers a lot of caution associated with using new analysis codes.

2.5.2 Why is it important to digital engineering?

As digital engineering encourages more and more virtual simulations and tests in lieu of their physical counterparts, there will be expectations to use and trust analysis tools more and more, which will require a significant amount of testing and scrutiny of commercially available tools. Rather than those procuring the software tools independently running verification checks for these algorithms to understand their margins, errors, and design envelopes for use each time, it would be ideal if these benchmarks were run against a representative set of test cases and advertised as part of the product-offering literature. This will allow organizations to more quickly understand the edge cases and predictable areas of use of a software tool, as well as reduce the overall cost and time of acquisition by mitigating manual use case development and test.

2.5.3 What are some of the benefits and challenges?

The benefit for having pre-documented benchmarks is intuitive, as they can be reused by any potential user without having to spend time and money investigating and deciding to use a specific tool; quite simply, it will speed up the process of selection and adoption. Further, it would also make limitations of software codes more transparent, reducing risk of misuse and increasing the speed at which results can be trusted. Some of the key challenges would be to accurately predict which benchmarks and test cases are the best representative ones to help inform potential buyers since end users are always finding new ways to stretch the limits of their tools. Finally, in some cases, the specifics on how an algorithm is written and works may be highly proprietary, so some aspects would require extra work by a tool provider to develop a shareable abstraction.

2.5.4 What are some recommendations and best practices?

The A&D industry should work on a summary list of test cases or benchmarks that tool vendors can pull from to perform testing to verify and accredit their tools that are established as useful in making decisions. This will help address the total cost of ownership concerns with the tools portfolio maintained for digital engineering, as tools requiring constant and manual re-verification of accuracy during new versions and new product introduction will deter adoption of new technology, impacting the advancement of the overall engineered products. In addition, end-user experience expectations and workflows for completing tasks should also be expressed as these play a role in how usable a tool is perceived in practice.

2.6 Security

During the selection or development of new engineering tools, adequate consideration should be given to the location and circumstances in which they will be used. When dealing with the various security requirements for these spaces, the tools must be able to accommodate security controls that may handle sensitive information.

2.6.1 What is it?

Embedding cybersecurity into the engineering environment begins with the selection of the engineer's tools. Cybersecurity considerations commence during the early development phases and extend throughout the engineering lifecycle. Choosing engineering tools that align the program's cybersecurity objectives lead to a more robust product line. Said another way, the engineering tools used to develop the product have a direct impact on the product's security, and they must be considered in how they will integrate in both unclassified and classified environments. Engineering tools should strive to deliver integrity as a core attribute, and they must be compatible with the requirements and protections installed in enclaves. To meet this goal, tool development and selection must incorporate source attribution or provenance. Ensuring provenance means the tool will be able to assert authorship, degree of compliance, and traceability over product changes. This section will provide further considerations in determining a tool's ability to deliver integrity to the product-line supply chain.

2.6.2 Why is it important to digital engineering?

Tool selection criteria should take into consideration how tool configurations are managed. Supply chain risk management (SCRM) provides assurance surrounding the tool's provenance or origin. Key factors of SCRM include:

- Supply chain security risk plan
- Bill of materials (BOM), including software BOM
- Source composition analysis
- Scanning and automating source composition analysis
- Configuration management enforcement policies
- Continuous monitoring of the engineering tool and dependencies

An SCRM program manages risk utilizing traceable artifacts and version control. Identifying and managing supply chain risks requires documenting software libraries, frameworks, dependencies, plugins, baselines, underlying infrastructure, and automation features.

An accurate BOM of deliverable software products is critical for applications. A BOM will describe the components included in the application, including open-source software and library components, the version of the components used, and the license type. A BOM helps security professionals, engineers and developers understand the components used in an engineering application and gauge potential security and licensing complications.

Source composition analysis, or software composition analysis (SCA), is the process of automating visibility into open-source software (OSS). SCA is utilized to reduce program risk and ensure security and license compliance. The rise of OSS across all industries has exposed the need to track components and their inherited bugs and vulnerabilities. Because most software creation includes OSS, manual tracking is difficult, requiring the need to use automation to scan source code, binaries, and dependencies. It is critical to discover and track all open source to provide integrity and maintain security.

Engineering tools should establish and enforce policies on tool configuration and provide mechanisms for centralized policy management. License compliance is critical at all levels within an organization. SCA spotlights the need to set policies, respond to license compliance and security events, and provide training and knowledge across the project. Policy enforcement solutions automate the approval process and set specific usage and remediation guidance. Engineering tools should provide mechanisms to assess and measure compliance to policies and licensee compliance.

Engineering tools should enable proactive and continuous monitoring. Tools should accommodate various patching strategies based on installation types and network connectivity.

2.6.3 What are some enclave security considerations?

Tools selection and development criteria for engineering enclaves must account for security requirements and constraints. These include the level of sensitivity pertaining to the data processed on the systems. Engineering tools must provide proper handling of IP and controlled information. This includes proper handling of data tagging, labeling, encryption, or other security mechanisms during processing of the engineering tool. In addition to the handling provisions, enclaves must consider connectivity requirements for on-premises, air-gapped, and cloud environments. The following enclave security considerations may apply for tool development and selection:

- Enclave information classification handling to include labeling, tagging, routing, and storage
- Shared enclave resource considerations pertaining to confidentiality and integrity and service-level agreements
- Privacy requirements and government regulations
- Tool authentication, feature isolation, data separation, and confidentiality management
- Tool audit capability supporting various enclave requirements
- Tool provisioning mechanisms
- Vendor interchangeability and licensing

2.6.4 What are some recommendations and best practices?

Establishing an *industry-agreed target digital environment reference architecture framework* will reduce effort for security vetting of tools by allowing both tool suppliers and users to identify gaps or additional security requirements, using a transparent accessible framework. Tool providers could quickly identify their tools as complying with a common framework, thereby increasing the likelihood of their approval for use. End users could use the common framework to enable a rapid assessment of their engineering tools. Using a common framework enables tools to be marketed as usable in government-controlled spaces. This greatly reduces the time and labor required to gain approval from governing organizations for secure engineering environments.

2.7 Flexible Licensing Options

It is preferred for software tools to offer and support a variety of licensing models.

2.7.1 What is it

A software license is a legal instrument governing the use or redistribution of software and is used as a contract between the creator and user of an application. Licenses are used to protect the author's intellectual property and establish the end user's rights to use the software. All software applications have a licensing model attached to them as a method to authorize and track utilization of the application. Licensing is also the primary method for the sale and use of a software application. There are a multitude of different licensing models available across industry, and each has pros and cons based on how an application is intended to be used across a user base. The goal would be to use a licensing model that aligns well with how an organization plans to roll out usage. Some of the most common licensing models are listed below.

- Perpetual
- Floating
- Subscription
- Metered
- Use-time
- On-demand
- Feature-based
- Fixed duration
- Trial
- Project-based
- Academic
- Offline
- Device-specific
- Named user
- Open Source

2.7.2 Why is it important to digital engineering?

Engineering teams and functions vary significantly in size and distribution based on the project, discipline, organizational con-ops, and level of classification. Therefore, there is a need for multiple licensing options to be made available for software tools to align with these diverse needs. Currently, the landscape of applications demonstrates that there are many licensing models in existence in industry. However, any single application tends to have a very limited subset of licensing types available as a choice. In most cases, the deployment approach for an application will align better to one model versus others. Misalignment will drive poor utilization statistics, increase administration costs, and ultimately erode the business case for purchases. Digital engineering will increase the quantity of applications that will be required in an engineer's toolbox, and therefore having more licensing options for an application will be critical to ensure agility and availability of the correct applications at the right time.

2.7.3 What are some of the benefits and challenges?

When a licensing model is well aligned to a using organization's con-ops, the utilization, administration, and maintenance are optimized, maximizing the value proposition for both the user and the author of the application. As an example, when a single expert has a need to run an application in multiple segregated or isolated locations, having a single named user license they can carry from location to location would be most effective. In contrast, a shared floating licensing model would be more valuable for a large, distributed workforce that needs access to an application only for a few hours a week. However, it is nearly impossible for an application author to predict all the different ways their application will be deployed, and implementing them all would be an undesirable outcome for the application providers.

2.7.4 What are some recommendations and best practices?

Ideally, there would be license allocation and tracking technologies that could offer a multitude of options and enable flexibility between the author and user to optimize the model for their specific agreement. Recognizing this may not be feasible in the near term, providing a subset menu of the options that are most likely to be used might be considered. Some early broad collaboration between the application creator and users would help identify the most desired deployment models to offer for an application. Then later, when negotiations arise for a new agreement, discussing the assumptions on the deployment approach and alignment to licensing models should be a standard best practice.

2.8 IP Management

The data used by or generated by software tools must have the ability to be clearly marked for IP ownership, and it must be clear that if tools have the ability to "learn" from proprietary datasets, how that learning data is managed and owned.

2.8.1 What is it?

The advance of digital engineering tools in recent years by software vendors has provided powerful solutions. At the same time, the proliferation of technology providers has also increased the numbers of offerings that, due to their ability to capture and convey complex data sets and models, makes it quite difficult to clearly identify aspects of the dataset that are IP restricted and which components can be more freely shared. Further, the need by some tools, particularly in the artificial intelligence and machine learning (AI/ML) areas, to learn from user data sets introduces IP concerns over sharing of the data and ownership over the insights gained from doing so.

2.8.2 Why is it important to digital engineering?

The aerospace industry offers a large opportunity space for these technologies as our products, supply chains, and operational environments are higher complexity and lower volume than some other markets. This means that gaining high levels of predictability through repetition is not feasible, whereas leveraging high-fidelity models, simulations, and AI/ML can allow for predictions to be made with the help of insights gained via other methods. However, most of the models and data generated of an end item are often a mixture of IP from multiple companies and varying levels of access rights to the end user and the company that performs the final top-level integration. Therefore, there is a need to share models and data but also to have each element clearly identified for who owns it, who can access it, and to what extent it can be used such that effective policy enforcement can be employed. When effectively done, the sharing of data drives significant value, particularly through the reuse of data for production, integration and test, operations, and sustainment.

2.8.3 What are some of the challenges?

Some of the key challenges with sharing of models and data is tied to the lack of any commonly used or standard conventions for how to mark information, where those markings are stored in digital datasets and how the markings can be persistent throughout the use of the information to mitigate inadvertent data spills and/or misuse. Many digital datasets lack the equivalent of a header or footer in a written document such as this paper, driving a wide variety of techniques that often require re-orientation or customization of software tools to employ.

A second key challenge is abstraction, where some subset of a model needs to be shared but not the full detail. In these cases, a derived or more simplified model is needed for exchange. Yet there are very limited conventions across modeling types to employ without developing tailored or custom solutions at each modeling domain level to prevent over-disclosure of information.

The third key challenge is with data transmission, as it is often difficult to track where a piece of data goes once it is transmitted to another organization, such as passing a model to a supplier for production, which is compounded by the ease of electronic sharing as compared to physical documents. At this point, the data has been replicated into another system into which the originator lacks insight, and therefore cannot assure against unauthorized use or run audits without labor-intensive collaborative processes. The data transmission methods are also highly variable, and many lack any way to detect attempts at file tampering or corrupting by systems and actors in the process.

Finally, since ML algorithms coming online are trained on data that is IP sensitive, situations where AI algorithms are designed to generate revenue from the data it gains access to drives the need to segregate and distribute data on a case-by-case basis. However, this is often counter to what makes these capabilities the most effective, as larger data sets tend to make the tools more accurate and trustworthy. The proprietary nature of the approach of software tools in this space limits the ability of the aerospace industry to take full advantage of many of these tools, and often generates concerns about who would own the insights gained from doing so.

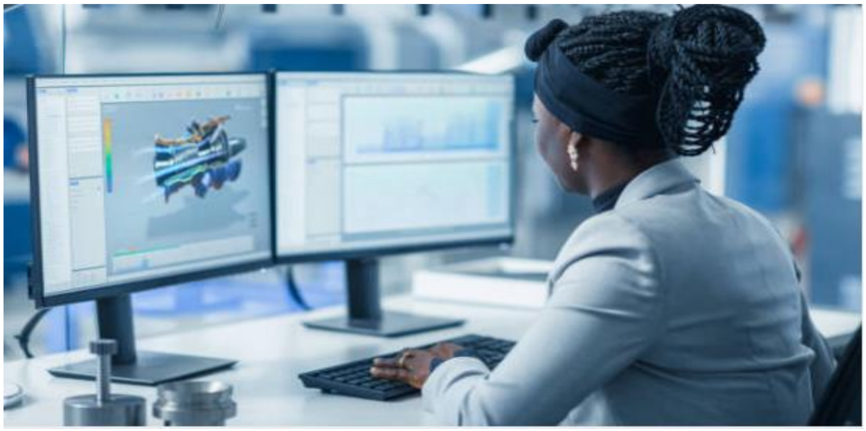
2.8.4 What are some recommendations and best practices?

There are likely a number of strategies and conventions that can be established to mitigate the four concerns highlighted in this section, but the key is some level of standardization of those conventions such that they can be consistently applied across a vast variety of organizations in a familiar, and therefore frictionless, manner. First data markings need to be designed into software tools such that modeling and data elements can be tagged at the object/feature level against a standard set of options and then persistently displayed in modeling and visualization tools. These markings or tags should also be accessible and reportable from data management systems and APIs to ensure clear and transparent usage.

Abstraction of models would benefit from some industry-established descriptions for content for some of the most commonly employed use cases for exchanging information, such that the amount of information is sufficient to perform the agreed-to tasks but obscure the additional IP or part of the agreement that is not needed. An example of this may be an “interface control” model, which depicts the overall geometry and functional behaviors, etc., of a system component but does not provide any insights into how it actually works at the lower level, or a “build to print” model, which provides enough information to manufacture a part but does not provide any design intent or analysis techniques that went into arriving at the final configuration. Development of a standard set of use cases and abstraction techniques for the most common customer and supply chain interactions that could be used as a convention might be considered for future iteroparity standardization.

There have been several techniques established for data transmission that could be further explored for feasibility and a number of them that are available today, but due to their complexity and cost to tailor these implementations, they are normally limited in use to larger program installations, technologies such as digital rights management, block chain, etc.

Finally, tools that leverage large IP data sets to be “trained” to gain insights or supply tailored results should be disclosed and optioned to not retain or feedback information outside of the owning organization’s environment when back-feeding insights to software supplier-hosted environments could impose IP or security concerns.



2.9 Emerging Technology Considerations

Digital engineering is evolving quickly. During the development of this paper, more emerging trends have surfaced that may warrant further requirements or planning activities that were not investigated in depth. The trending topics captured in this paper are ones that are still gaining definition and a full appreciation of their challenges and considerations, many of which are about a pivot towards more virtualized environments. In the current state, many engineering software tools are highly specialized and are executed on workstations that are tuned for an application(s) to run accurately and efficiently. These attributes include processors, memory, hard drive types, and sizing, graphics processing units (GPU), network cards, and specific drivers, each of which will affect how the application runs. In some cases, software applications are certified to run on a specific brand of GPU as an example. With engineers becoming more multi-disciplinary, these constraints could impose sub-optimal workstation configurations or a need for multiple high-end machines.

With the trend towards more virtualization of workstations, cloud-hosted tools, and data services, as well as containerization, many software tools will have to be re-designed to operate effectively in those environments. While it shows a lot of promise for efficiency, speed, and simplicity of hosting, for a large variety of software packages, dedicated workstations or high-performance computing clusters are the only reliable approach, which does impose some near-term limiting factors. As these trends evolve, a continuous re-evaluation of these capabilities for feasibility of adoption is required. It is also anticipated that there will be similar trends that continue for hosting and management of environments in government/public clouds across multiple classification levels for enterprise systems and data management.

3 Summary

This paper outlines engineering software tool capabilities that must be matured to accelerate adoption of digital engineering. These capabilities will reduce friction associated with data interoperability and technology flexibility, which are necessary for the goals of digital transformations. While many of the recommendations might be taken on by industry groups through direct engagement on closing a standard or technology gap, it is also anticipated that many of these requirements and considerations are directly useable in developing digital engineering ecosystems as guidance. Therefore, it is encouraged to leverage this as a guide, where applicable, in designing and selecting capabilities for your respective organization’s unique digital engineering experience and in establishing methods to interact with customers and suppliers.