


## From tears to tiers - architectural principles for federated PLM landscapes

Erik Herzog , Johan Tingström, Johanna Wallén Axehill, Åsa Nordling Larsson and Christopher Jouannet

Saab AB, Sweden

 herzog.erik@gmail.com

### Abstract

PLM systems are key enabling systems in the development of today's products. Introduction of a new PLM capability is an expensive and risky undertaking. Many implementation projects end in tears in the sense that they are frequently late or even cancelled. In this paper, a federated PLM architecture pattern – Genesis – is introduced and evaluated against prevalent PLM approaches. From an architecture perspective, Genesis with its two distinct integration tiers decrease the number of integration points and thus cost and complexity.

*Keywords: product lifecycle management (PLM), integration, systems engineering (SE), OSLC*

## 1. Introduction

Product Lifecycle Management (PLM) systems are key enabling systems in organisations developing and maintaining cyber-physical systems. With increasing expectations comes the risk for not meeting stated requirements and objectives when implementing and introducing a new PLM capability. Moreover, as it takes time for realising an intended PLM capability, be it through implementation activities alone or in combination with organisation alignment activities, there is the risk that the realised capability will be a poor fit for the current challenges facing the organisation. Publications of PLM implementation failures are rare, as noted by Singh (2020). Still, we are aware of a large number of implementation projects which has been less than successful, even outright failures. Based on verbal feedback among business colleagues, we conclude that PLM implementation projects often end in disappointments and even tears in the sense that developed solutions are never taken to live operation. In its simplicity, the Cynefin framework (Snowden and Boone, 2007) is an invaluable tool for helping to understand the best approach for problem solving, see Figure 1. This tool could also be very useful in the context of PLM, when analysing organisational needs and perspectives of the PLM capability. The model identifies four classes of problems, and suggests a solution pattern for each class:

- **Simple** – Characterised by stability and clear cause-and-effect relationships that are easily discernible by everyone. Often, the right solution is self-evident and undisputed. In this realm of “known knowns”, decisions are unquestioned because all parties share a joint understanding of the ideal solution. The analysis pattern for this class of problems is sense–categorise–respond.
- **Complicated** – May contain multiple right answers, and although there is a clear relationship between cause and effect, not everyone can see it. This is the realm of “known unknowns”. The analysis pattern for this class of problems is sense–analyse–respond.
- **Complex** – Cause and effect can only be deduced in retrospect, and there are no right solutions, each solution proposal will have its strengths and weaknesses. This represents “unknown

unknowns". The analysis pattern for this class of problems is probe–sense–respond, i.e., for the complex domain there is a risk that the context will change based on the intervention made.

- **Chaotic** – Representing situations where the relationships between cause and effect are unclear, i.e., "unknowable unknowns". At a development program level, Chaotic situations cannot be allowed to be the norm. Still, leadership need always be prepared to manage chaos when it arises. The analysis pattern in Chaotic situations is act–sense–respond.

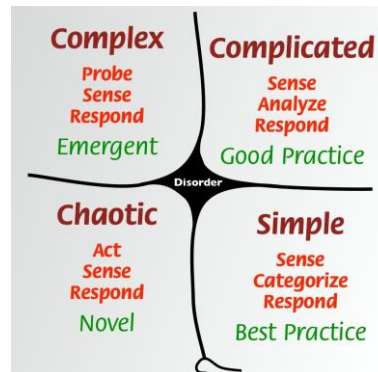


Figure 1. Cynefin decision framework (Snowden and Boone, 2007)

Over the past few years, we have realised that prevalent PLM strategies suffer from applying solution patterns which are poorly aligned with the problem class. They offer an elegant solution pattern for enterprises facing Simple and Complicated environments, while most organisations today are embedded in Complex and occasionally even Chaotic environments. Perhaps, it should not come as a surprise that many PLM system introduction and migration projects are late and grossly over budget? In the rest of this paper an overview of PLM, different PLM solution patterns and the challenges encountered by an organisation implementing a modern PLM capability are presented. This is followed by the introduction of a novel architecture pattern for realising a federated PLM capability and a discussion on implementation experiences and on the potential advantages of such a pattern over prevailing approaches.

## 2. Tensions when implementing PLM

This section outlines our understanding of PLM, prevailing solution patterns and identifies a number of tension areas for consideration, where an organisation have to make strategic choices related to their PLM realisation when introducing a new PLM capability.

### 2.1. What is PLM?

Product Data Management (PDM) and its gradual evolution to PLM has its root in mechanical engineering in the 1980'ies and the need to keep a digital trail representing the evolution of a product, firstly in the design phase and now over the complete lifecycle. What a few decades ago was systems for managing coarse-grained objects (such as drawing documents) has evolved into integrated development environments where fine-grained objects (such as an individual requirement) is managed. The front-end applications where fine-grained data is consumed and manipulated is getting tightly integrated into the PLM capability.

Today, a PLM environment typically supports managing information in multiple areas, such as:

- Requirements management
- Functional and logical systems design
- Verification and validation management
- Mechanical and electrical engineering design, production and maintenance data

PLM suppliers are also making inroads into other engineering disciplines, e.g., software engineering. In addition to the engineering discipline support, there is also support for, e.g., multi-user collaboration, configuration and change management, declaration of conformity to requirements, approvals, roles and credential management.

In short, modern PLM systems provide extensive support for almost all technical processes of organisations developing, producing and maintaining modern cyber-physical systems. Recently, engineering discipline specific integrated development environments providing an integrated PLM capability have emerged, such as [IBM ELM \(2023\)](#) for Systems Engineering. Such environments open up for organisations realising federated PLM capabilities built on multiple integrated engineering discipline specific ones.

## 2.2. PLM solution patterns

To our understanding, there are four dominating solution patterns for establishing a PLM capability, as outlined below:

### 1. Loosely integrated front-end applications with a Coarse-Grained Monolithic PLM

Under this pattern, the PLM capability operates on coarse-grained objects exported from stand-alone front-end applications. The front-end application provides the product data, while the back-end PLM capability provides the mechanisms for managing product structures, baselining and approvals. Under this approach an organisation, has the freedom to choose front-end applications freely, as long as documents can be generated, but traceability between information originating from different front-end applications is weak.

### 2. Fine-grained Monolithic PLM

The fine-grained monolith is characterised by a tight integration between front-end applications and the PLM capability. The front-end applications may be integrated directly in the PLM capability. This allows for fine-granular traceability between information elements, but limits the range of front-end applications available to the ones supported by the supplier of the PLM capability, either directly or through partnerships. Consequently, there is frequently a time penalty for the supplier to integrate state-of-the-art components.

### 3. PLM Backbone

A PLM Backbone is characterised by a dedicated database or service layer that mediate information between discrete PLM capabilities. The backbone typically does not offer any front-end application services, its sole purpose is to connect and mediate information. The objective is to enable fine-grained traceability between otherwise stand-alone PLM capabilities. With the PLM backbone being supplied by a dedicated supplier, the end result is a high dependence on that supplier. Examples of this pattern can be found in [Banaj et al. \(2016\)](#) and [Chadzynski et al. \(2018\)](#).

### 4. Service-Oriented Federated PLM

This pattern is characterised by the existence of a large number of stand-alone applications offering a single or small number of services with the capabilities normally associated with PLM systems, e.g., the ability to create versions, variants, baselines, change requests and approvals. Fine-grained traceability is enabled via the use of standard representation stateful transfer (REST) type Application Programming Interface (API). In this pattern there is neither a centralised database nor a main supplier. The members of the federation may change over time. One example of this pattern is discussed in [Hooshmand et al. \(2022\)](#).

## 2.3. PLM realisation tensions

A number of trade-offs – tensions – must be considered when introducing a new PLM capability. The areas listed below should not be interpreted as binary choices, but rather a greyscale where different solutions will provide value depending on the approach. These areas will be used to evaluate the long-term consequences for introducing a PLM capability in accordance with the solution patterns above.

- a) **Traceability** – Coarse vs. fine grain. Traditionally, PLM has offered traceability on a coarse scale, between large information elements enclosed in documents. Modern PLM systems offer fine-grained traceability, i.e., linking between individual requirement elements or linking between an individual requirement and a verification case.

- b) **Integration** – Loose vs. tight. An application is loosely integrated with the PLM capability if it comes with its own data management capability. Conversely, a tightly integrated application uses the data management capability of the PLM environment.
- c) **Operational strategy** – Out of the box or custom made. With the increased capability and scope provided by PLM suppliers, an overarching question is whether to adapt the PLM capability to the current or envisioned organisational practices, or the other way around.
- d) **Lifecycle** – PLM system life vs. the product life. An organisation’s approach to PLM is fundamentally different if the life of a PLM system is assumed to be longer than the life of products being developed, or the other way around. Also, older systems dependence on legacy IT environment needs to be incorporated in the lifecycle perspective.
- e) **Business environment** – Flexibility vs. stability. An organisation having a stable environment may optimise its PLM capability on stability, where one engaging in multiple collaborative product development activities may benefit from flexibility optimisation.
- f) **Supplier selection** – Decision simplicity vs. productivity. For an organisation introducing a new PLM capability, it might be appealing to select a single supplier for the complete capability. Alternatively, the focus may be on ensuring maximum engineering productivity by introducing loosely integrated applications from multiple suppliers.

### 3. Analysis

The PLM solution patterns in Section 2.2 are evaluated against the tension areas identified above.

**Table 1. PLM solution patterns against tension areas**

	<b>Coarse-grained monolith</b>	<b>Fine-grained monolith</b>	<b>Backbone</b>	<b>Service-oriented federated</b>
<b>Traceability</b>	Coarse grained	Fine grained	Fine grained	Fine grained
<b>Integration</b>	Loose integration, freedom of choice	Tight integration, based on supplier offer	Tight integration, based on backbone supplier offer	Loose integration, based on integrator preference
<b>Operational strategy</b>	Freedom to choose methodology based on end-user priorities	Based on supplier defined methodology and flexibility	Freedom to choose, limited by integrated applications	Freedom to choose individual tools/services, based on integrator priorities
<b>Lifecycle</b>	Front-end tools and PLM can be exchanged independent of each other. Need to consider lifecycle of integrations of front-end tools.	Dependency on lifecycle of fine-grained monolith supplier. Need to consider lifecycle of own additions and customisations.	Front-end tooling can be complemented with new ones, but high reliance on backbone supplier	Tools/services can be replaced individually
<b>Business environment</b>	Flexible, as long as document granularity is sufficient	Stable, dependence on supplier offers	Stable, dependence on supplier offers	Flexible, selection is depending on interface compliance
<b>Supplier selection</b>	Freedom to select front-end applications independently of PLM supplier	Supplier selection implies constraints on the supply of tools. Supplier integrated tools is assumed to be best fit for all users	Supplier selection implies constraints on the supply of tools. Supplier integrated tools is assumed to be best fit for all users	Freedom to select based on compliance to architectural principles

### 3.1. Evaluation

This section outlines the strengths and weaknesses of the respective PLM solution pattern and discusses in particular the conditions for making service-oriented federated PLM an attractive solution. Referring to Table 1, it can be seen that:

- For an organisation where fine-grain traceability is not beneficial or adding little value, it appears that a Coarse-grained PLM monolith would suffice. Such a solution provides the freedom to select and switch front-end applications at little extra overhead. However, the lack of fine-grained traceability makes solutions in this class non-ideal for supporting development of Complex (and safety critical) systems.
- For solutions based on a Fine-grained monolith or a dedicated Backbone, the ability to maintain fine-grained traceability is very attractive and a clear driver for improving data quality and engineering productivity. The ability to adapt an organisation to the services offered by a supplier is attractive and cost efficient in the short term. However, these solutions come with a strong alignment with the capabilities integrated in the respective solution. Beyond lock-in effects, there is a risk that the PLM capability, or parts thereof, will not age with grace, potentially leading to a desire to replace individual parts. It has also been noted by [Hooshmand et al. \(2022\)](#) that PLM capability development velocity decrease with increased size of the PLM core. It appears that the larger the supplier, the slower the integration velocity of new capabilities.
- The Service-oriented federated pattern is the one offering the highest level of freedom to end-users, but on the other hand it also places the whole integration cost solidly on the end-users. An organisation electing a federated solution must be prepared to shoulder the costs over the life of a system.

Overall, the Service-oriented federated patterns appear to be appropriate for an organisation developing Complex products and operating in a dynamic (Complex) environment. However, the integration cost must be addressed for making the pattern attractive.

## 4. Mechanisms for enabling federated PLM

In previous sections, we have reviewed and analysed architectures for realising a PLM capability. In this section we introduce an architecture pattern for federated PLM, named Genesis ([Herzog et al., 2022](#)), and the OSLC family of standards for linking across application boundaries.

### 4.1. Genesis - A 2 tier architecture for federated PLM

The Genesis architecture pattern exploits the emergence of engineering discipline development environments, providing support for all engineering and management activities within an engineering discipline. An enterprise-wide capability can be established by integrating multiple such environments, as illustrated in Figure 2. The basic idea is that all activities related to an engineering process, e.g., systems engineering, software engineering or mechanical engineering is performed within its dedicated environment. Tier 1 in the patterns are the individual engineering discipline-oriented environments, whereas the individual components within a Tier 1 environment belong to Tier 2. Obviously, the engineering discipline development environment supporting mechanical engineering will need to include support for production engineering, production, operations and maintenance.

A consequence of the architecture is that project managers, configuration managers and engineers active in a particular process will all be working in the same environment (the horizontal component in Figure 2). Another consequence of the pattern is the acceptance of dedicated development tool capabilities per process, e.g., different requirements management capabilities in the systems and software engineering environment.

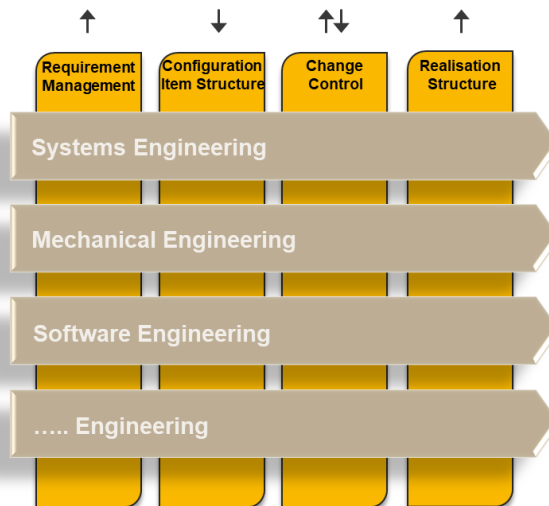


Figure 2. Genesis architectural pattern

#### 4.1.1. Traceability between Tier 1 environments

As noted above, the attractiveness of the Service-oriented federated PLM pattern depends on minimising the number of interfaces that must be maintained over time. For Tier 1, the focus is on traceability between information elements of different engineering disciplines, see Figure 3 (left). Based on Saab's internal processes there are four traceability dimensions that have to be maintained across engineering discipline boundaries, i.e., via linking information elements:

- **Requirements traceability** – Capturing the justification for the existence of each individual requirement. Requirements traceability is typically captured bottom-up, i.e., from the child requirement to its parent(s).
- **Configuration item structure traceability** – Capturing the product structure of a system. In a federated environment, parts of such a structure will be maintained in the respective discipline specific environment and traceability between product structure nodes must be maintained. This traceability dimension is typically maintained top-down, i.e., the parent configuration item node identifies its children.
- **Issue management traceability** – Capturing development planning and problem reporting traceability over development environment boundaries. This traceability dimension is top-down for forward looking development planning activities and bottom-up for capturing problems that arise during development.
- **Realisation item structure traceability** – Capturing how realised items are assembled. This structure is maintained to keep track of what has actually been realised. Traceability in this dimension is established bottom-up, i.e., realised parts are integrated to form assemblies. For a product there may be multiple realisation structures. Beyond the actual realisation there could be any number of virtual ones created with a wide range of purposes and realisation fidelities.

#### 4.1.2. Traceability between Tier 2 components within an engineering discipline

Here, there is a need for extensive traceability between information elements created within the engineering discipline process. This could be traceability from requirements over architecture to design and analyses, production engineering, production to usage of realised items, or from requirements to verification and validation of realised system elements. The engineering artifacts and traceability patterns will most likely be unique for each engineering discipline and may evolve individually over time. Tier 2 in the pattern concerns the individual components within a horizontal element, see Figure 3 (right).

### 4.1.3. Discussion

The strengths of the proposed pattern are:

- The engineering discipline focus offers the opportunity to create integrated development environments tailored for the needs of an engineering process within an organisation.
- The number of integration points between Tier 1 engineering discipline environments is low and expected to be stable over time, limiting the integration cost between, e.g., a Systems Engineering and a Mechanical Engineering environment.
- The architecture pattern is agnostic to the actual implementation architecture for an engineering discipline environment. An individual environment may be realised using a Fine-grain monolith, Backbone or a Service-oriented federation.
- Within an engineering discipline environment, there is the possibility to exchange individual Tier 2 components as they become obsolete or when there are more suitable alternatives available without upsetting the overall PLM capability.

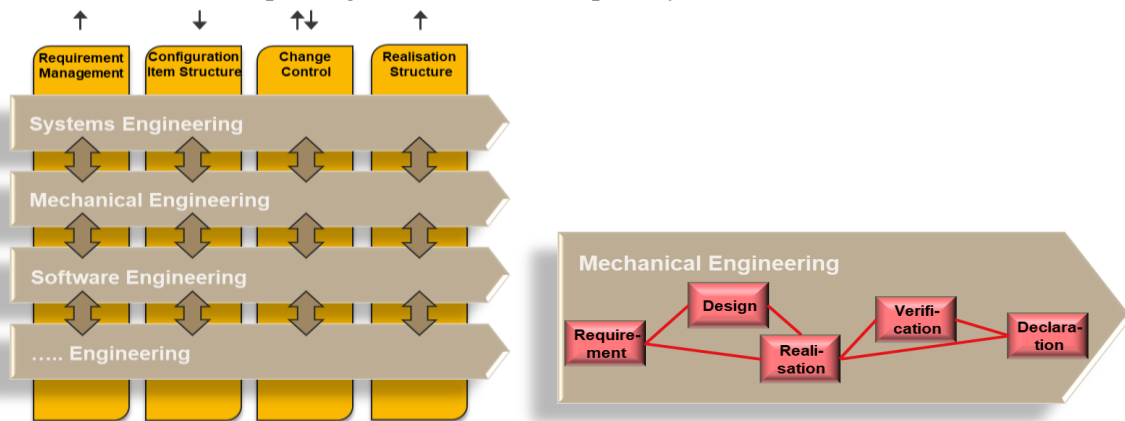


Figure 3. Tier 1 traceability (left) and Tier 2 traceability (right)

The proposed pattern allows an organisation to focus first on transitioning to a federated solution integrating Tier 1 engineering discipline environments, with the opportunity to transfer to a fully Service-oriented PLM capability at a later stage. Moreover, the proposed tier structure allows for an organisation to maintain a number of alternative engineering discipline development environment for each engineering discipline. This enables product development projects to make a selection based on their actual needs and collaboration context.

## 4.2. Enabling standards

Standards are a key element for realising a federated capability based on the Genesis architecture pattern. The OASIS Open Services for Lifecycle Collaboration (OSLC) standard framework provides the capability to capture links across tool boundaries. A technical overview and commercial status of OSLC is available in [El-khoury \(2020\)](#) and is summarised below.

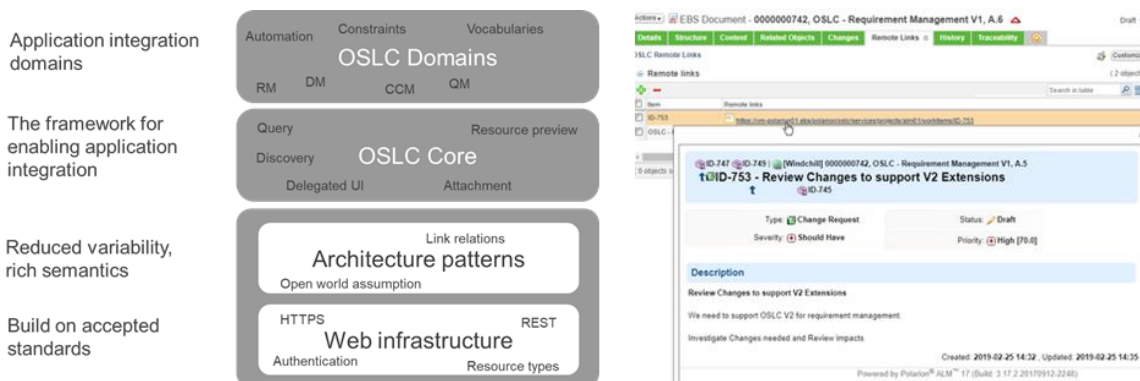


Figure 4. OSLC architecture overview (left) and an example of delegated UI (right)

Information-wise, OSLC adopts the Resource Description Framework (RDF) and accompanying standards to represent the structure and semantics of the information being exchanged between applications. See Figure 4 (left) for an architecture overview.

OSLC is built around a core specification with support for linking between version and variant aware objects and has a number of domain specific extensions (OSLC, 2023). The dedicated domain specifications for requirements and change management are perfect matches for the requirements management and issue management traceability dimension within the Genesis architecture pattern. The Configuration Management specification provide support for managing the configuration item and realisation item structures. All OSLC domains are configuration management aware meaning that linking can be made to a particular version of an object and not just the latest one.

Another interesting feature of OSLC is that of user interface delegation, which allows an application to provide a user dialogue that can be used by other web applications for creation and selection of resources, see Figure 4 (right). This feature allows end users to link information across tool boundaries using a familiar user interface. Of course, sufficient credentials are required to log in to the target tool to create links or to follow a link.

## 5. Discussion

This paper has provided an expose over the challenges in deploying a PLM capability within an organisation. Based mainly on anecdotal evidence, we postulate that a fair share of PLM implementation run into troubles or are cancelled. In many cases, such failures can be traced to the long implementation times required to field a PLM system. By the time a capability is ready for introduction within an organisation, the needs of the organisation may very well have changed.

For organisations existing in dynamic environments in terms of collaborations and/or technology development, a federated approach to PLM appears more appealing. However, an unstructured federated PLM architecture will come with high costs for implementation and maintenance, even though it offers the desired flexibility. In this respect, the proposed Genesis architecture pattern offers the following characteristics in the areas defined in Section 2.3.

- **Traceability** – Fine grain, with a restricted set of integration points between engineering disciplines (Tier 1).
- **Integration** – Loose integration in two tiers, firstly at the inter-engineering discipline tier (Tier 1) and secondly at the intra-discipline tier (Tier 2).
- **Operational strategy** – In this area the Genesis pattern offers flexibility in that different strategies can be applied for different engineering disciplines.
- **Lifecycle** – Also in this area the Genesis pattern offers flexibility as individual Tier 1 and Tier 2 components can be exchanged independently over time. This flexibility allows for the application of different PLM federations for different products or systems. In fact, during a transition period it is plausible that the replaced PLM component can be used in parallel with the replacing component.
- **Business environment** – The Genesis pattern contribute to flexibility in a dynamic business environment as individual Tier 1 and 2 components can be replaced to adapt to a changing business environment.
- **Supplier selection** – Also in this area, the Genesis pattern offers supplier selection flexibility as components in both tiers can be replaced. Hence supplier selection is not a once in the life of the PLM solution, but can be applied continuously as individual components evolve.

Moreover, the Genesis pattern offers advantages as it:

- Focuses on optimising performance per engineering discipline (from an overall perspective in Tier 1 and for optimisation for a specific Tier 2 component).
- Minimises the number of integration points between Tier 1 components. The small number of integration points implies a loose coupling, which simplifies upgrades, replacement and migration of individual components.
- Provides a structure offering an opportunity for a stepwise transition towards federated PLM.



- Allows for an organisation to offer alternative engineering discipline environment for selection by individual projects.

From a supplier perspective, Genesis opens up opportunities for smaller suppliers to provide Tier 1 environments without having to support the full range of engineering disciplines required to develop a multi-disciplinary product. One can also foresee situations where an integration-oriented supplier selects Tier 2 components for creating an integrated Tier 1 solution, or an integrated PLM capability consisting of multiple Tier 1 solutions, creating a PLM solution as envisioned by [Bleisinger et al. \(2022\)](#).

The existence of validated and acknowledged standards is a clear prerequisite for widespread adoption of federated PLM. To our best understanding, OSLC is the primary candidate in that there is an existing standardisation community and a solid technical foundation. There are at least three third-party suppliers, [Lynxwork \(2023\)](#), [Sodius-Willert \(2023\)](#) and [MID \(2023\)](#), providing services related to interface generation. Services provided range from development of OSLC interface to provisioning of tools and frameworks that automate large part of the interface development process, making the cost of creating and maintenance of interfaces manageable. Within the [Heliple-2 \(2022\)](#) research project, the authors have worked with the Lynx designer tool suite, focusing on developing interfaces for integration between Tier 1 environments. We can confirm that OSLC interfaces can be created in a matter of hours for attaining the basic functionality, with a week or so required to provide a professional service. End user performance for creating and following OSLC links depend on the setup of the integrated applications. Experience from interfaces generated is that there is a small but still acceptable performance penalty compared to application internal operations.

The weakness in the case for OSLC is that the standardisation group need to be invigorated with new members with a clear vision of where to take the application of the standard. Growing the existing community with stakeholders committed to federated PLM would be an excellent step, ensuring the opportunity to validate existing standard parts and prompt the definition of new ones.

We now return again to the title of this paper – from tears to tiers. Based on our experience in realising complex systems, the reason for tears shed in PLM development projects stem from the following factors:

- The team developing a new PLM capability is frequently not aware of the actual practises within the different engineering disciplines in the organisation, making a designed solution a poor fit for organisational needs. Often this is due to PLM developers providing solutions intended for the Simple or Complicated domains, as defined in the Cynefin framework ([Snowden and Boone, 2007](#)), where the organisation operates in the Complex domain.
- The time required to implement a PLM capability means that by the time the capability is fielded the organisational needs may have changed.
- Once fielded, traditional PLM implementations will meet the wear of time, meaning that parts, if not the complete implementation will no longer be state-of-the-art.
- Vendor lock-in effects decrease the flexibility of an organisation. The investment in a PLM solution is of such a magnitude that a decision to change to new suppliers are not taken easily.

The solution pattern advocated in this paper – federated PLM based on the two-tier Genesis architecture pattern – appears to offer a number of advantages in that the PLM domain is broken down into Tier 1 modules with a small number of well-defined integration points. Individual Tier 1 modules are interchangeable as long as they conform with the integration points. If the individual modules are realised in accordance with federation principles, then their internal components can be exchanged with no or little impact on the external functionality of the module. The two tiers of modularity increase flexibility in terms of PLM capability selection, allowing for rapid replacement of individual components be it for obsolescence or partner collaboration reasons. Overall, transitioning towards a tiered approach to PLM has the potential to minimise the risk of implementations running into tearful end results. This said, it should be underlined that while initial development and integration activities are promising, integration performance appear acceptable, it should be noted that we have not yet deployed an industry strength instance of the federated PLM environment envisioned in this paper. Large-scale, industry-strength implementation and end-user and performance measures is a topic for future work.

## 6. Conclusions

Formulating and implementing a PLM strategy within an organisation is a challenge given the size of the investment and the reliance of an organisation on such a capability. In this paper we have argued that the current predominant approach to establishing a PLM capability come with high risks. A two-tier architecture pattern, called Genesis, is proposed to increase flexibility and in conjunction with the OSLC standards minimise cost for establishing and maintaining a federated PLM capability over time. Adopting the proposed pattern appears to offer advantages also for smaller suppliers in the sense that their applications can make contributions to a larger PLM capability. All in all, we believe that the proposed framework with its two architectural tiers minimises the risk for PLM implementations ending up in tears.

## References

- Bajaj, M., Zwemer, D., Yntema, R., Phung, A., Kumar, A., Dwivedi, A. and Waikar, M. (2016), MBSE++ — Foundations for Extended Model-Based Systems Engineering Across System Lifecycle. INCOSE International Symposium, 26: 2429–2445. <https://doi.org/10.1002/j.2334-5837.2016.00304.x>
- Bleisinger O., Psota T., Masior J., Pfenning M, Roth A., Reichwein A., Hooshmand Y., Muggeo C., Hutsch M. (2022). Killing the PLM Monolith – the Emergence of cloud-native System Lifecycle Management (SysLM), Fraunhofer IESE, Mat 2022, Downloaded from: <https://publica-rest.fraunhofer.de/server/api/core/bitstreams/a09d6709-63ad-48de-b819-efad5387c2ab/content>
- Chadzynski, P.Z., Brown, B. and Willemsen, P. (2018), Enhancing Automated Trade Studies using MBSE, SysML and PLM. INCOSE International Symposium, 28: 1626–1635. <https://doi.org/10.1002/j.2334-5837.2018.00572.x>
- El-khoury J. (2020), An Analysis of the OASIS OSLC Integration Standard, for a Cross-disciplinary Integrated Development Environment: Analysis of market penetration, performance and prospects, TRITA-ITM-RP, 978-91-7873-525-9, 2020.
- Heliple-2 (2022), Heterogeneous Linked Product Lifecycle Environment – Iteration 2, Vinnova dnr 2022-03032, 2022
- Herzog, E., Nordling Larsson, Å. and Tingström, J. (2022), Genesis – an Architectural Pattern for Federated PLM. INCOSE International Symposium, 32: 782-791. <https://doi.org/10.1002/iis2.12963>
- Hooshmand, Y., Resch, J., Wischnewski, P., and Patil, P. (2022). From a Monolithic PLM Landscape to a Federated Domain and Data Mesh. Proceedings of the Design Society, 2, 713-722. <https://dx.doi.org/10.1017/pds.2022.73>
- IBM ELM, Engineering Lifecycle Management, <https://www.ibm.com/products/engineering-lifecycle-management>, visited 2023-11-07.
- Lynxwork (2023), <https://lynxwork.com/>, visited 2023-11-07.
- MID (2023), <https://mid.de/en/>, visited 2023-11-07.
- OSLC (2023), specification <https://open-services.net/specifications/>, visited 2023-11-04.
- Singh S., Misra S. and Chan F. (2019). Establishment of critical success factors for implementation of product lifecycle management systems, International Journal of Production Research, 58:4, 997-1016, <https://dx.doi.org/10.1080/00207543.2019.1605227>
- Snowden, D. J., and Boone, M. E. (2007). A leader's framework for decision making. Harvard business review, 85(11), 68.
- Sodius-Willert (2023), <https://www.sodiuswillert.com/en/solutions/linking-data-oslc>, visited 2023-11-07.