PLM Migration in the Era of Big Data and IoT: Analysis of Information System and Data Topology

Piers Barrios, François Loison, Christophe Danjou, Benoit Eynard

PLM migration in the era of Big Data and IoT: analysis of information system and data topology

Piers Barrios[1,2,[0000-0002-6850-5182]], François Loison[2], Christophe Danjou[3], Benoit Eynard[1,[0000-0002-5393-4363],*]

[1]Université de technologie de Compiègne, France
piers.barrios@utc.fr
benoit.eynard@utc.fr*
[2]Gfi Informatique, France
francois.loison@gfi.world
[3] Polytechnique Montréal, Canada
christophe.danjou@polymtl.ca
* Corresponding Author

Abstract. As product lifecycle management is starting to be a standard information system in most companies, a data manipulation issue starts to emerge. Indeed, the information contained in a system often needs to be accessible or even transferred to another one to exploit a new software's functionalities. With the Big Data era, new methods of analysis start to emerge, and the graph visualisation of data enables a better human understanding of the data itself. We propose to address the PLM migration issues with a new approach based on analysis of information system and data topology. The data is passed onto a graph and bundles are made thanks to clustering algorithms; this bundling enables a better understanding of the data and ease migration. However, PLM has links between links which complicate the transition to graphs. Finally, the described use-case proves that data bundling eases the data migration and prevents some usual pitfalls and delays.

Keywords: Product Lifecycle Management, Internet of Things, data package, group clustering

1 Introduction

Information and communication technology (ICT) systems are in constant evolution to manage more information, improve the user interface or enable new features. However, data migration remains a core problem when going from one software package to the next. It can be mitigated by keeping the same software editor and doing regular updates, but this is too costly compared to keeping the system intact for many years and doing a brutal upgrade. An upgrade often means a major change in the data model in between the source and the target. To address this complexity, most of ICT systems' upgrades are conducted by three main actors: the industrial, client of this upgrade; the software editor of the target system and the integrator, the company that makes sure that the

software will fit the industrial's needs. In the context of Big Data [1] and emergence of Internet of Things (IoT) [2], PLM data needs to be easily accessible in the context of a data migration to be efficiently used by these emerging technologies. However, until now this is not the case and we aim to reduce the gap in this domain by proposing a method to ease data migration.

An upgrade process can be split up into multiple streams (**Fig.1**):

- The applicative stream concerns the user interface, the behaviour and the data model.
- Data migration stream produces a method to migrate data as well as executing it.
- Interfaces stream ensures the link with external applications.
- Infrastructure stream tackles the IT infrastructure needs for the app to function.
- User experience and acceptance stream manages the change and the user acceptance of the new system.
- Finally, pilot and system engineering stream ensure the constraints and interactions in-between the different streams.

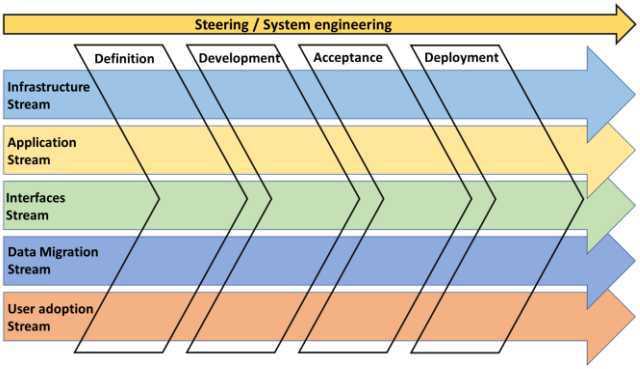

Fig. 1. ICT system upgrade decomposition into streams

Managing a product's lifecycle is quite a tenuous task as it goes from its ideation to its disappearance. In this regard, Terzi [3] defines Product Lifecycle Management (PLM) as follows: "PLM can be broadly defined as a product centric – lifecycle-oriented business model, supported by ICT, in which product data are shared among actors, processes and organisations in the different phases of the product lifecycle for achieving desired performances and sustainability for the product and related services." »

Nevertheless, ICT, only being there to support the PLM process and methodologies, has gained an increasing impact in recent years to the point where vendors are now providing "PLM software" aiming to encapsulate all the necessary components for a properly functioning process and even methodologies.

Compared to other ICT systems such as MES or ERP, PLM data migration is fundamental in PLM systems as the data needs to be available throughout the existence of the product's instances. In some cases, such as major aircraft companies, the legacy PLM system needs to remain available causing huge impact on IT costs. Indeed, the knowledgeable personnel tends to retire earlier than some of the products on the market; hence the resource gets rarer and incidentally more expensive.

In this regard, one could easily conclude that it would be cheaper for a company to do a migration than to keep legacy systems up and running. However, going from a legacy system to a top-of-the-pop system implies the successful migration of the existing data.

Overall, implementation of a data migration process is challenging because:

- The data migration strategies are plenty and dimensioning for any project. Unfortunately, theoretical reality that can be far from data's reality. To our knowledge, there is no specialised software to evaluate data migration strategy.
- Initial resource estimation tends to underestimate the complexity of data migration because of no direct or visible added value for the trade and users.
- Data migration stream has a transient state – i.e. data migration elaboration -followed by a steady state. During elaboration, complexity is very high because of impact of other streams and organisational make or buy choices. A tooled-up method intended to reduce this complexity is needed.
- Finally, the triptych of application, data migration and user acceptance are strongly interdependent and tends to destabilise the project and the global system.

Most often, the migration isn't going from one legacy system to one target system but from multiple sources to one target associated with important data volumetry. Data extraction may require to only migrate part of a system's data as that data is not relevant anymore or should be migrated towards another system. Also, on the data migration stream, there are no established methodologies to tackle such a complex task as there would be on other streams; e.g. Waterfall, Scrum.

Therefore, to address this issue, our article will be structured as follows: we will focus our state of the art on interoperability and PLM data migration. The methodology proposition will underline the issues faced and the choices made to attain a usable solution. This proposition will be illustrated by use cases inspired by our industrial experience. Finally, discussion, conclusion and future works will complete this paper.

2 State of the art

2.1 Interoperability

The European interoperability framework [4] defines three dimensions for interoperability: technical, semantic and organisational. Technical interoperability is composed mostly of all the communication protocols and infrastructure that enable various systems to communicate. Semantic interoperability is the key one here; it "is the ability to automatically interpret the information exchanged meaningfully and accurately in order to produce useful results as defined by the end users of the systems under consideration"[5]. Finally, organisational interoperability is concerned with people that wish to exchange information and may have different internal structures and processes.

PLM's ICT systems are, until now, made to address semantical interoperability as they add sense to data for the various trades working on the products; people within the

same company may achieve semantic interoperability through their access to PLM. ISO 14258 [6] says that semantic interoperability can be achieved in three ways:

- Integrated: "With integrated models there is a standard model form." The main limit there is the standardisation of models.
- Unified: A meta-model exists to represent all existing models. Hence, one can establish correspondence between the various models.
- Federated: Models are diverse and rely on the establishment of dynamic correspondence.

PLM migration from multiple initial systems to one target system can be considered as one way to achieve integrated semantic interoperability. Hence, enabling a better understanding of data migration could enable, in a near future, the transition of integrated systems towards a unified or even a federated ICT system. However, we need to better understand what has already been done concerning PLM data migration.

On data exchange and interoperability, one might look towards one of the following for extended state of the art: Rachuri et al. [7], Fortineau et al. [8] and Sriti et al. [9].

2.2 PLM data migration

Research review through Scopus and Web of Science highlights the following:

- In Müller et al. [10], "PLM migration project" is only in the abstract; in the rest of the articles, the authors talk about "PLM deployment". Unfortunately, data migration is not mentioned throughout this article. However, project management in interestingly discussed in one paragraph, providing precious insight about the management of such projects: Even if led by equal rights between the business unit and IT, PLM upgrade still had been perceived as a pure IT project.
- Singh [11] challenges only the transition to the cloud and does not address data migration itself. It is understood that this is an isomorphic migration and that systems are unlikely to evolve between on-premise PLM and cloud PLM. Unfortunately, experience has proven otherwise as PLM migration always implies upgrade.
- Renji Kuncheria [12] stresses out key points of data migration, without however tackling them, in these terms: "Migration of data from legacy enterprise data-systems is one of the important and challenging tasks for most PLM implementations. […] Moreover, the dependency of production activities on data, that is complete and accurate, stresses the importance for error free data migration. Translating data stored in formats as required or limited by legacy systems to conform to the PLM system specifications only adds to the complexity." However, throughout the article the actual data migration process is neither explicated, nor detailed.

In order to address product development, Guérineau et al. [13] proposes a four-level categorisation for product development. However, none of the listed artefacts enables us to tackle the data migration stream of an ICT system upgrade from the multiple source applications into a unified target application.

3 Methodology Proposition: Data Systemizer

To challenge the previously cited challenges, we propose a standard reusable methodology for data migration stream. The applied approach consists in decoupling a complex problem into several simpler problems, while controlling the interdependencies between the sub-problems. Applied to data transformation issues, this principle amounts to breaking down the dataset into the most independent data islands possible.

First, we will explore the data model before exploring the data clustering and the existing interface.

3.1 Concise and Generalised Data Model

As PLM systems try to encompass as much information on the product as possible, the number of entities and number of attributes per entity is staggering. As a global understanding is key to making accountable choices, only a few elements are useful to understand the system from one point of view. For a decisional tool such as Data Systemizer, the model must be restricted to entities and attributes that are mandatory for analysis, i.e. entities and attributes driving data clustering. Therefore, we will base ourselves on a concise part of the data model.

PLM software tends to consider the various instances of an object separately. For instance, one may consider that a part is spread over several entities such as engineering part and manufacturing part for instance. In this case there is no such thing as a part by itself. To better approach a model, we proceed to a reification meaning that we consider all the parts as actual parts and process information on them: all parts attributes are now considered similar. Generalising the model eases the analysis a great deal and the definition of analysis rules.

3.2 Data Model Structuring Choice

Indeed, the PLM data is more than often stored in a relational database for speed and reliability purpose. However, graph databases take benefit from these similarities to propose full-featured graph applications with interfaces including real world object/link-based data.

In this regard, we consider switching the data stored into PLM towards a graph. In the graph theory, everything is either a vertex or an edge and an edge is necessarily between two vertices. In PLM, any artefact is either an object or a link. An object is a standalone entity holding attributes and a link is an entity relaying objects, generally 2 objects identified by their role; links can hold attributes. As one may realise, PLM data model systems have strong similarities with graph theory models. However, PLM application usually push-up the limit by considering links on links. When including these, the model is no longer comparable with the graph theory as a vertex cannot connect a node and a vertex.

Fig. 2 illustrates data model. It shows a Printed Circuit Board (PCB) accepting a Surface Mount Device (SMD) chip. Both are objects. PCB contains pads for SMD chip, it is materialized by a structure link. To prevent component shortage issues, PCB has a

dual option loading, its pads can accept a SMD chip or a Dual Inline Package (DIP) chip. The ability of replacing SMD chip by DIP chip is not global but is constrained by a validation. Equivalence link enables replacement for the specific PCB/SMD chip association. This is a "link on link" as DIP chip is connected to structure link. Replacement is enabled only for this specific PCB / SMB chip assembly.

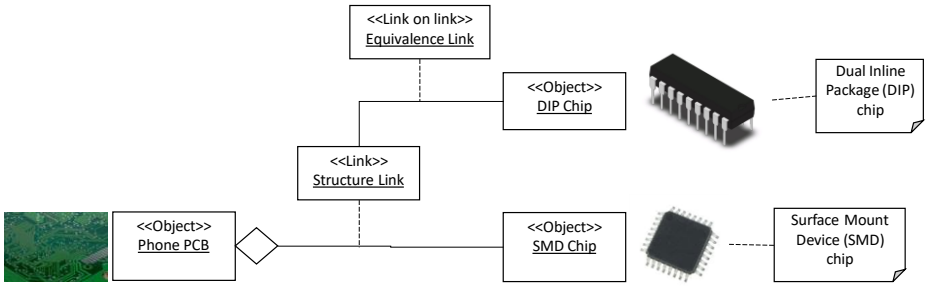

Fig. 2. Data Model illustration

Gutwenger et al. [14] and Farhadian [15] propose solutions to convert such "links on links" model to a graph. However, it requires a virtual node splitting the vertex in two, the "link on link" vertex being connected to this virtual node. This approach is suitable for presentation (sketching the model into a graph). At this point, it is not compatible with clustering as the virtual node does not belong to any cluster. However, to enhance analytics on the data, clustering is essential.

We therefore shift towards an Entity/Link model: Entity is either an object or a link and Link is a link between 2 entities. This model allows implementing links on links between objects. Recursive definition of link allows more sophisticated constructions: "links on links between objects and links" for example. The Entity/Link model has an impact on clustering and tree traversal consolidation processes (Breadth-first search, Depth-first search) needed, for example, to apply metrics on clusters; e.g. counting entities. Basically, clustering allocates entities to lots, while remaining links are stored into a link between lots called a Lot Dependency. Lots and lot dependencies are also entities that can be processed by a further clustering pass.

In the end, everything is an entity. We can distinguish different classes of entities depending on the sophistication; as follows:

Element	**Entity class**
Object or Link	$Entity^{[0]}$
Lot or Lot Dependency containing $Entities^{[0]}$	$Entity^{[1]}$
Lot or Lot Dependency containing $Entities^{[1]}$	$Entity^{[2]}$
…	…
Lot or Lot Dependency containing $Entities^{[N-1]}$	$Entity^{[N]}$

In the case where there are no links on links, lot dependencies allocation calculation and tree traversal processes are nominal. In the case of links on links, issues are raised because nominal algorithms raise precedence problems; e.g. entity accessed but not yet initialised.

We define an ordering metric called LinkLevel such as:

$$\text{LinkLevel}(\text{object}) = 0 \quad (1)$$

$$\text{LinkLevel}(\text{link}) = \text{LinkLevel}(\text{edge A entity}) + \text{LinkLevel}(\text{edge B entity}) + 1 \quad (2)$$

Links are sorted by increasing LinkLevel and are processed one after the other, starting by LinkLevel 1. This approach sorts out the precedence issues explained previously. Our model also contains Virtual Entities which are not taken into consideration by the clustering process. As circuits (loops of directed vertices) are an issue, we isolate them as such. These are present in the final graph, while clustering remains unchanged [16].

To summarise:

- We implement an Entity/Link data model which is not a graph model
- The model is completed to allow further lot clustering by a meta-model containing lots and lot dependencies; a lot is an entity and a lot dependency is a lot link
- Bridges to graph model exist for clustering entities into lots and lot dependencies & sketch graphs for presentation
- Entity/Link data model raises tree model traversal precedence issues, a metric called LinkLevel allows to correctly order links processing

3.3 Clustering

Zhong et al. [17] explored the possibilities of putting part of the PLM data into clusters in order to increase relevancy and flexibility. However, PLM data migration was not addres
sed. The most important success factor of PLM migration is choosing the right data transformation strategy. Successful data transformation strategies go as follows:

- Loosely coupled data lots to allow to parallelise data lot processing.
- Most possible functionally significant data lots: a data lot and its dependencies should contain a business product to foster validation tests. On the contrary, a lot dividing strategy based on bottom/up layers such as bolt & nuts, assemblies, modules and end items is not acceptable because it's testable only at the end of migration.
- Lots should be decomposed in a loosely coupled lot tree in order to recourse this process at lot level and build efficient test lots given a size and error rate.

Clustering is the key to achieve these needs. The clustering is realised by a "Lot Divider" which takes entities as input and output lots and lot dependencies partitioning entities. There are two categories of Lot Dividers:

- Business Lot Divider: driven by functional parameters available from data. For example divide data by program name, divide data by site name.
- Mathematical Lot Divider: driven by graph topology, requires no business parameters. E.g. Metis Lot Divider partitions a graph in sub-graphs while reducing sub-graphs dependencies; Adherence Lot Divider creates sub-lots to understand and measure lot adherence; Bill of Material (BOM) Lot Divider groups entities until an entity is re-used.

Both dividers produce lots who have a type that could be Business or Technical. Lot type drives further passes of dividing as explained in next chapters. By default, Business Lot Dividers produce Business Lots and Mathematical Dividers product Technical Lots but this default settings can be overridden by parameterization depending on clustering strategy needed.

Metis [18] is a graph partitioning tool taking as input a graph and desired number of partitions N. It produces N sub-graphs optimised to reduce number of cross-partition vertices. Metis is used by "Better than Big Bang Divider" intended to split data in most possible number of independent lots, to reduce data migration complexity.

The experimental approach is based on scenarios. A scenario is a pipe of Lot Dividers that allocates data entities$^{[0]}$ (objects, links) to entities$^{[M]}$ (lots and lot dependencies), M being the number of clustering passes. Scenario outcomes are evaluated by verifying they are compliant to rules enounced previously. Data Systemizer defines a lot type: business or technical. Whatever the Lot Dividers execution sequence is and their numbers, the lot aggregation chain is always a business lot chain followed by a technical lot chain:

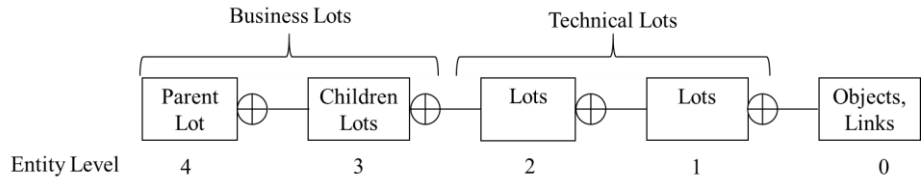

Fig. 3. Example of sequence of processing

Fig. 3 introduces lot inclusion chains given the Entity Level (EL). In this example four levels are retained. Each lot is noted as included (UML) in the previous one. The deducted properties are as follows:

$$\text{Entity}^{[EL]} = \{ \text{Object, Links} \} \text{ for EL=0} \quad (3)$$

$$\text{Entity}^{[EL]} = \{ \text{Lot, Lot dependency} \} \text{ for EL} \geq 1 \quad (4)$$

$$\text{An entity with EL=N+1 contains entities EL=N : } \text{Entity}^{[EL+1]} = \{ \text{Entity}^{[EL]} \} \quad (5)$$

The clustering pipe is then executed as following:

1. Identify min($EL_{Business}$) and max($EL_{Technical}$), the EL directly preceding (business / technical lots) break. From example: min($EL_{Business}$) = 3, max($EL_{Technical}$) = 2
2. For each Business Lot$^{[min(ELBusiness)]}$:
   a. Allocate to a dedicated working space (bench) the Lot Divider with Entities$^{[max(ELTechnical)]}$ belonging to current Business Lot
   b. Run the Lot Divider algorithm using bench entities as input. Lots produced by Lot Divider are either Business or Technical (functional parameter of scenario). They are attached as direct children of current business lot.
3. Run the Lot Divider with Entities$^{[max(ELTechnical)]}$ as input

Throughout this section, we set out all the necessary artefacts and tools necessary to dive into PLM systems' data; we shall now consider the use cases this methodology can be applied to.

Needless to say, Data Systemizer is data format agnostic. I.e. a data format such as XML imposes a structure whereas the analysis detailed here is agnostic and able to explore multiple structures. Hence, the flatter the format, the better. The contribution of a format such as XML in this case is grammar and completeness verification, which is nice but remains optional.

4 Use-case

4.1 Optimizing an ETL Data Load

ETL (Extract/Transform/Load) tools aim to migrate data from source to target system by applying a pattern. However, it is flow driven and unaware of data topology, e.g. decomposition of data in loosely coupled lots. Some approaches to ETL processes such as [19] in the data warehouse and big data contexts might be graph-based and semantic however these do not fit the PLM needs as expressed here.
During ETL elaboration, loading error rate may be dramatic for PLM data structures because of the re-use object factor - an object is re-used by other objects through a link. If the re-used object loading is broken, all dependent links and all recursively dependent links will fail, a.k.a. explosive errors spreading. As a result, data is not business testable and the error analysis task is complex because valuable error root causes are flooded by useless dependent errors. Data Systemizer can extract a "successful transformation data strategy" and drive ETL.

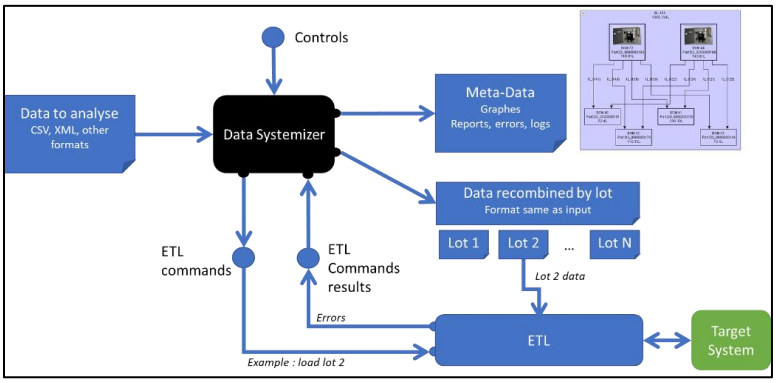

Fig. 4. Interfaces of Data Systemizer in the ETL Use-case

Fig. 4 presents interfaces in the black-box manner. Data is input via CSV or XML files, clustered by Data Systemizer and then output. On this interface, the various clusters are going to have their own files; hence enabling smooth input to ETL solution for the new PLM system. ETL is fed by data lots respecting the original data format, not full data from the input. After lot loading is processed by ETL, Data Systemizer gets objects, link errors and inspects impacts of errors. If explosive errors spreading phenomena are detected, loading is stopped for dependent lots while independent lots continue to load.

Moreover, output can be a graph; this is helpful as it enables meaningful insight of data; it enables company and project leaders to understand and anticipate possible issues; they will be able to evaluate the right complexity of the data transfer. Also, it sometimes underlines the fact that the theoretical truth about the data is far from reality.

4.2 Optimizing a migration strategy

A data migration project needed to evaluate migration strategy of data package containing 10 million objects and links. ETL direct approach had failed because of "explosive errors spreading" phenomenon and the duration of a test increment which was too long to bear iterative development mode.

Data is extracted from a PLM source system. Data extraction has been made by source object types into a set of CSV flat files. Files data sets are not organized by programs nor any business organization. Loading data type by type in a Big Bang migration strategy is not acceptable because this loading strategy would be testable only at end of process and subjected to cascading errors phenomenon. The question raised was: Is there a more optimal than the Big-Bang migration strategy?

Data Systemizer was parameterised to perform two BOM Dividing and Metis BOM Dividing:

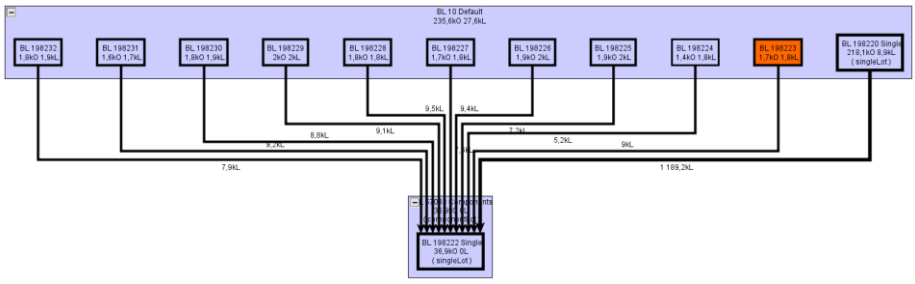

Fig. 5. Result of Data Systemizer application: data is separated in 12 different lots

Fig. 5 shows input data which has been clustered into 12 lots; 11 separated lots all depending on a single component lot. Data Systemizer can restitute this monolithic data package in 12 lots. The lot in orange contains some data referential integrity errors; i.e. orphan links. Some elaboration specific test data lots can be provided given needed volumetry and error rate. Test data lots are functionally significant, they contain testable end-to-end business data.

4.3 Choosing right migration strategy

An aircraft company needs to migrate a certification application from a PLM system to an ERP system. Volumetry is low but data model is complex. Drastic system changes and model complexity requires important end users testing hence big-bang migration is not acceptable because on one hand, it would retain tests at the end of the migration project and on the other end, users would redo costly tests.

To enable a fine-grained migration strategy, two lot dividing strategies are studied: at Family level (top level) and at Baseline level (middle level). Business users would like to know what level is best.

As data model is complex, some specific algorithms have been developed to extract Family and Baseline information. Two graphs are produced to compare both migration strategies.

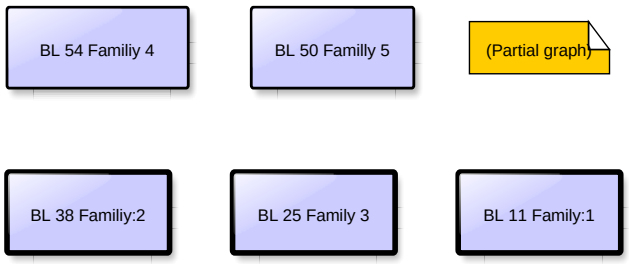

Fig. 6. Partial graph of Family level migration strategy

Fig. 6 shows graph corresponding to Family level migration strategy. Family lots are totally separated, this migration strategy is efficient.

To see if data could be split at lower level, the baseline level analysis has been run.

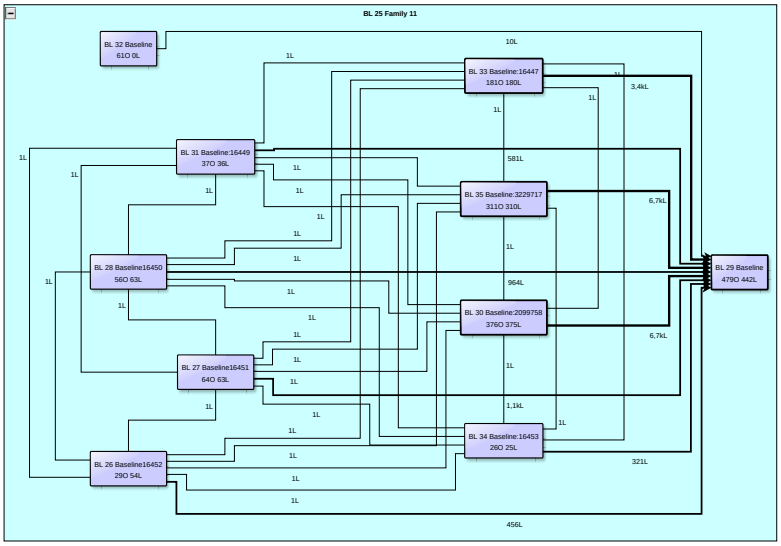

Fig. 7. Partial graph of Baseline level migration strategy

Fig. 7 shows partial graph corresponding to Baseline level migration strategy. Outer box is a family lot, inner boxes are baseline lots contained in this family lot. Clearly baseline lots are highly coupled. Baseline lots should be migrated altogether, but this

wastes the benefit of baseline-level migration strategy. In this case, the study shows that Family level migration is best choice.

5 Conclusion and future work

This article has outlined the issues related to PLM data migration as well as why this is crucial regarding being able to tackle one company's own data in the current context of Big Data. Methodology was described and use cases improving existing were outlined.

Discussion focus on possible reusability of the underlined methodology as well as possible improvements. Indeed, the method outlined could be reused in any information system whose object representation is similar, such as ERP, MES or IoT platforms. Improvements to the method could be made regarding the treatment of circuits. Indeed, they are currently temporarily put away to avoid blocking the process. However, in some scenarios, their number would be too non-negligible, and one would need to tackle them head on. Also, improvements would be made on the implementation and focus on enabling user-friendly interfaces.

Future works should tackle measuring the effectivity increase thanks to the use of this method which would be necessary to leverage further deployments.

Current work could we improved according to the following directions:

- Some real-life examples such as **Fig. 7** show data graphs with lots highly coupled. The raw question is to know whether highly coupling is intrinsic to data topology or to weakness of clustering algorithm. A tooled-up method able to inspect graph coupling quality would be helpful.
- This article deals with data loading use cases. They are other interesting use cases: modification and purge/archive. Modification use case implies a 2 ways data analysis: data to process versus data present in the system. Purge/archive use case is interesting in PLM systems because data are highly linked to other data. Scoping the purge/archive target is challenging, executing purge/archive is even more complicated. Deletion use case is a simpler subcase of purge/archive, still not achievable manually.
- Data verification after migration. Current industry practice is based on end users' tests. This practice should be improved. Current directions are: probability and statistics-based method to provide a more controlled confidence level to tests executed, mode-based approaches relaying entity counts to reconciliate number of entities before and after migration, brute force export after migration and comparison with source data.

References

1. Dekhtiar, J., Durupt, A., Bricogne, M., Eynard, B., Rowson, H., & Kiritsis, D. Deep learning for big data applications in CAD and PLM – Research review, opportunities and case study. Computers in Industry, 100, 227–243. (2018). 10.1016/j.compind.2018.04.005

2. Barrios, Piers & Eynard, Benoit & Danjou, Christophe. Towards a Digital Thread Between Industrial Internet of Things and Product Lifecycle Management: Experimental Work for Prototype Implementation. (2020).10.1007/978-3-030-42250-9_26.
3. Terzi, S., Bouras, A., Dutta, D., Garetti, M., Kiritsis, D.: Product lifecycle management - from its history to its new role. International Journal of Product Lifecycle Management. 4, 360 (2010).
4. Chen, D., Doumeingts, G., Vernadat, F.: Architectures for enterprise integration and interoperability: Past, present and future. Computers in Industry. 59, 647–659 (2008).
5. Özacar, T., Öztürk, Ö., Ünalır, M.O.: ANEMONE: An environment for modular ontology development. Data & Knowledge Engineering. 70, 504–526 (2011).
6. ISO 14258: Industrial automation systems — Concepts and rules for enterprise models. International Organisation for Standards, Geneva (1998)
7. Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S.J., Foufou, S., Sriram, R.D.: Information sharing and exchange in the context of product lifecycle management: Role of standards. Computer-Aided Design. 40, 789–800 (2008).
8. Fortineau, V., Paviot, T., Lamouri, S.: Improving the interoperability of industrial information systems with description logic-based models-The state of the art. Computers in Industry. 64, 363–375 (2013).
9. Sriti, M.F., Assouroko, I., Ducellier, G., Boutinaud, P., Eynard, B.: Ontology-based approach for product information exchange. International Journal of Product Lifecycle Management. 8, 1 (2015).
10. Müller, P., Muschiol, M., Stark, R.: PLM-Based Service Data Management in Steam Turbine Business. In: Rivest, L., Bouras, A., Louhichi, B. (eds.) Product Lifecycle Management. Towards Knowledge-Rich Enterprises. pp. 170–181. Springer, Heidelberg (2012)
11. Singh, S., Misra, S. Migration of PLM systems to cloud. International Journal of Communication Systems. 31 (2018).
12. John, R. K..: PLM implementation at MAN Diesel A/S: a case study. In: Garetti, M., Terzi, S., Ball, P., Han, S. (eds.) Product Lifecycle Management. Assessing the industrial relevance. pp.199-204 Inderscience, Geneva (2007)
13. Guérineau, B., Rivest, L., Bricogne, M., Durupt, A.: Agile and Project-Planned Methods in Multidisciplinary Product Design. In: Harik, R., Rivest, L., Bernard, A., Eynard, B., Bouras, A. (eds.) Product Lifecycle Management for Digital Transformation of Industries. pp. 108–118. Springer International Publishing, Cham (2016).
14. Gutwenger, C., Mutzel, P., Weiskircher, R.: Inserting an Edge into a Planar Graph. Algorithmica. 41, 289–308 (2005).
15. Farhadian, A.: A Coordinate System for Graphs. arXiv:1701.02443 [math]. (2018)
16. Johnson, D.: Finding All the Elementary Circuits of a Directed Graph. SIAM Jounal on Computing 4, 77–84 (1975).
17. Zhong, H., Yan, G., Lei, Y.: Evolution Supporting Class-Cluster Data Model for PLM. In: Jin, D. and Lin, S. (eds.) Advances in Electronic Commerce, Web Application and Communication. pp. 191–196. Springer, Heidelberg (2012)
18. Abou-Rjeili, A., Karypis, G.: Multilevel algorithms for partitioning power-law graphs. In: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. p. 10 pp. IEEE, Rhodes Island, Greece (2006)
19. Simitsis, A., Vassiliadis, P., Terrovitis, M., & Skiadopoulos, S. Graph-Based Modeling of ETL Activities with Multi-level Transformations and Updates. Lecture Notes in Computer Science, 43–52. (2005)