Generating Layered Enterprise Architectures with Conceptual Structures

Matt Baxter[1], Simon Polovina[2], Wim Laurier[3], and Mark von Rosing[4]

[1] Conceptual Structures Research Group, Sheffield Hallam University, UK
a7033771@my.shu.ac.uk
[2] Conceptual Structures Research Group, Sheffield Hallam University, UK
S.Polovina@shu.ac.uk
[3] Université Saint-Louis, Belgium wim.laurier@usaintlouis.be
[4] LEADing Practice, France mvr@leadingpractice.com

Abstract. Enterprise Architecture (EA) uses metamodels to document and align organisations' business, information, and technology domains. This structure then enables these domains to work in harmony. Layered Enterprise Architecture Development (LEAD) builds upon EA by introducing layered metaobjects connected by semantic relations that make up LEAD's layered metamodel. Previously, an algorithm was developed to elicit active semantic relations to achieve a view highlighting the metaobject dependencies. Subsequently, CG-FCA (Conceptual Graph and Formal Concept Analysis) and a LEAD case study were used to develop an enhanced algorithm that also generates the LEAD layers. The resulting layered FCA lattice shows a way to discover the hitherto hidden insights in LEAD, including the relationship between business and information technology.

Keywords: Business Problem Solving · Conceptual Graphs and Formal Concept Analysis · Layered Enterprise Architecture.

In Press: ICCS 2021 (https://iccs-conference.org/), Springer LNAI

1 Introduction

Enterprise Architecture (EA) promotes the alignment of business, information, and technology domains within organisations by a principled framework, method, and approach that draws on holistic thinking. The Layered Enterprise Architecture Development (LEAD) Ontology adds to this rigour by structuring that thinking according to 91 metaobjects arranged in business, information, and technology layers and categorised further via various sublayers [4, 7]. These metaobjects are linked by semantic relations both within and across layers, thus describing how a metaobject views itself in relation to another. Consequently, the interdependencies within organisations can be elucidated and used to the benefit of industry decision-makers.

Conceptual Graphs (CG) represent knowledge via the formalised ordering of concepts and their relations [6]. Formal Concept Analysis (FCA) uses a principled approach to ascertain the conceptual hierarchy of a set of objects and their attributes [3], meaning LEAD's metaobjects can be ordered using their shared attributes. Ergo, the CG-FCA application allows us to convert the ternary semantic relations of LEAD into binary relations, which are then mathematically validated as a conceptual hierarchy by the FCA element of the application [1]. By complementing CG-FCA with an algorithm that identifies the active semantic relations (defined as those relations whereby a metaobject directs another), we aim to create a Formal Concept Lattice (FCL) that is ordered according to LEAD's three business, information, and technology layers.

2 The Metamodel Diagram

Figure 1 is a metamodel created using the Enterprise Plus (E+) software (`www.enterpriseplus.tools`). LEADing Practice (`www.leadingpractice.com`) is a not-for-profit group of LEAD practitioners, and are the developers of E+. This software captures LEAD's comprehensive reference content, including its metaobjects, semantic relations, and supporting artefacts. Figure 1 reflects the warehouse pick pack process of a UK manufacturer, based on the LEAD Enterprise Ontology (LEAD ID#-ES20001ALL) [7]. Figure 1's semantic relations are two-way, revealing how a metaobject views itself in relation to another and vice versa. The metamodel in the figure includes 30 two-way semantic relations, with a modeller-imposed limit of one relation between each adjacent sublayer, and zero instances of relations that span more than one sublayer. These choices were made with two objectives in mind. Firstly, preservation of readability—the LEAD metamodel contains 147 two-way semantic relations for the chosen metaobjects—and secondly, the promotion of accessibility for inexperienced EA users within the case study organisation. Thus, the salient semantic relations were selected to develop a robust and digestible narrative for those users. The metamodel was created to facilitate the deconstruction of its abstract concepts and their subsequent relation to known real-world organisational elements.

3 Activating And Layering The Metamodel

The CGtoFCA algorithm implemented in the CG-FCA application translates the ternary relations of CGs into binary relations, thereby preparing them for processing by FCA [1]. The concepts are then presented in a Formal Concept Lattice (FCL), which visualises the mathematical rigour of FCA. CG-FCA was augmented with an algorithm to create a graph of active verbs that supports a chain of command, highlighting objects that act upon other objects [2]. However, this approach took little account of LEAD's layers, adversely impacting the readability of the active direction graph. The proposed revised algorithm introduces LEAD layering, delivering a more lucid, mathematically validated representation of the metamodel.

3.1 Methodology

The proposed algorithm is depicted in Figure 2. The stages are identification of active semantic relations, resolution of cycles, then the introduction of layering. The variables

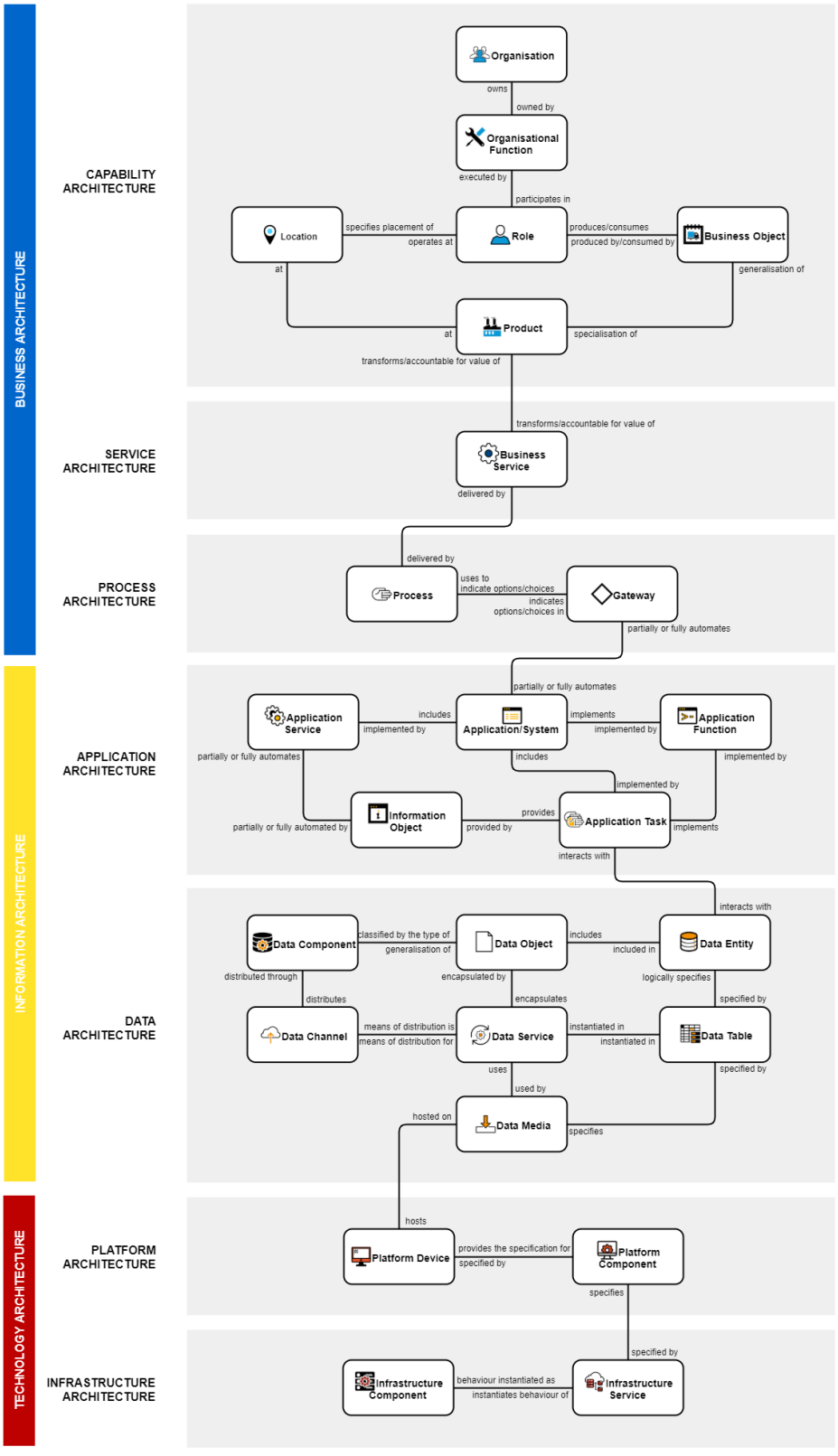

Fig. 1. Warehouse pick pack metamodel (based on LEADing Practice Meta Model)

are A active model, B bidirectional model, o triple object, s triple subject, and v triple verb. While the human interpretation required by the algorithm renders it more of a 'pseudo-algorithm' at present, it remains suitable for our current purposes.

```
1  begin
2      A = ∅
3      foreach ((o, v, s), (s, v′, o)) ∈ B do
4          if isPassive(v) then
5              A = A ∪ (s, v′, o)|((o, v, s), (s, v′, o)) ∈ B
6          else
7              A = A ∪ (o, v, s)|((o, v, s), (s, v′, o)) ∈ B
8      C = TriplesInCycles(A)
9      foreach (o, v, s) ∈ C do
10         if inMultipleCycles(o, v, s)) or isImplicityPassive(v)) then
11             A = A\(o, v, s)
12             A = A ∪ (s, v′, o)|((o, v, s), (s, v′, o)) ∈ B
13         if isTransitive((o, v, s)) then
14             A = A\(o, v, s)
15     S = ConceptsInSupremum(A)
16     foreach (o, v, s) ∈ A|o ∈ S and Count((o, α, β) ∈ A) > 1 do
17         A = A\(o, v, s)
18         A = A ∪ (s, v′, o)|((o, v, s), (s, v′, o)) ∈ B
19     I = ConceptsInInfimum(A)
20     foreach (o, v, s) ∈ A|s ∈ I and Count((α, β, s) ∈ A) > 1 do
21         A = A\(o, v, s)
22         A = A ∪ (s, v′, o)|((o, v, s), (s, v′, o)) ∈ B
23 end
```

Fig. 2. Layered active semantic relations algorithm

The initial steps mirror an earlier study, whereby an active direction graph is created by determining the active relations and resolving any unwanted semantic cycles [2]. Subsequently, LEAD layering was introduced by examining and modifying the compositions of the supremum (the top-most formal concept) and infimum (the bottom-most formal concept) in the FCL. Beginning with the supremum, the triples containing the incorrectly layered metaobject were reviewed so that the triple determined to be the least active could be inverted. Encompassed within this stage was the consideration of the wider data set. We were mindful that the number of instances of a metaobject could be either a source or a target in a triple. Therefore, we could avoid a scenario whereby resolving one metaobject's erroneous presence in the supremum or infimum would see it replaced by another. Such a situation could lead to a cycle of layering issues (incidentally, not to be confused with a cycle defined by the CG-FCA application). In situations where a least active triple could not be determined (e.g., due to identical semantic relations), the relation furthest down the LEAD index would be inverted. We

now describe our findings from the algorithm in Figure 2 according to the EA layering and usability principles highlighted in Figure 1.

3.2 Findings

The 30 active semantic relations were selected from the two-way relations included in the metamodel and compiled in the 00ActiveAll.csv file, which was then processed by the CG-FCA application. The subsequent '00ActiveAll_report' file presented two cycles, shown below:

1. Cycle: Data Object - includes - Data Entity - logically specifies - Data Table - instantiated in - Data Service - encapsulates - Data Object
2. Cycle: Data Table - instantiated in - Data Service - uses - Data Media - logically specifies - Data Table

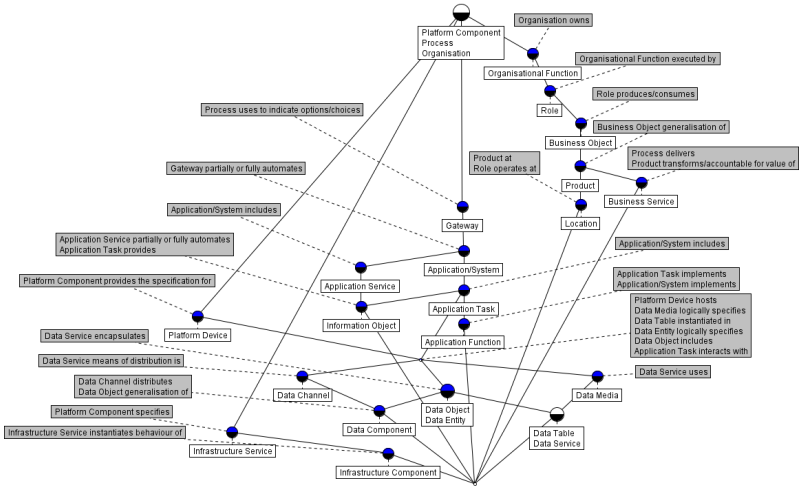

Fig. 3. 00ActiveAll lattice

Figure 3 shows the 00ActiveAll lattice, with a cycle indicated by the Data Table + Data Service object with no attributes. Furthermore, the Application Task, Platform Device, Data Media, Data Channel, and Data Object + Data Entity concepts are linked to an infimum formal concept that is empty, i.e. one with no objects.

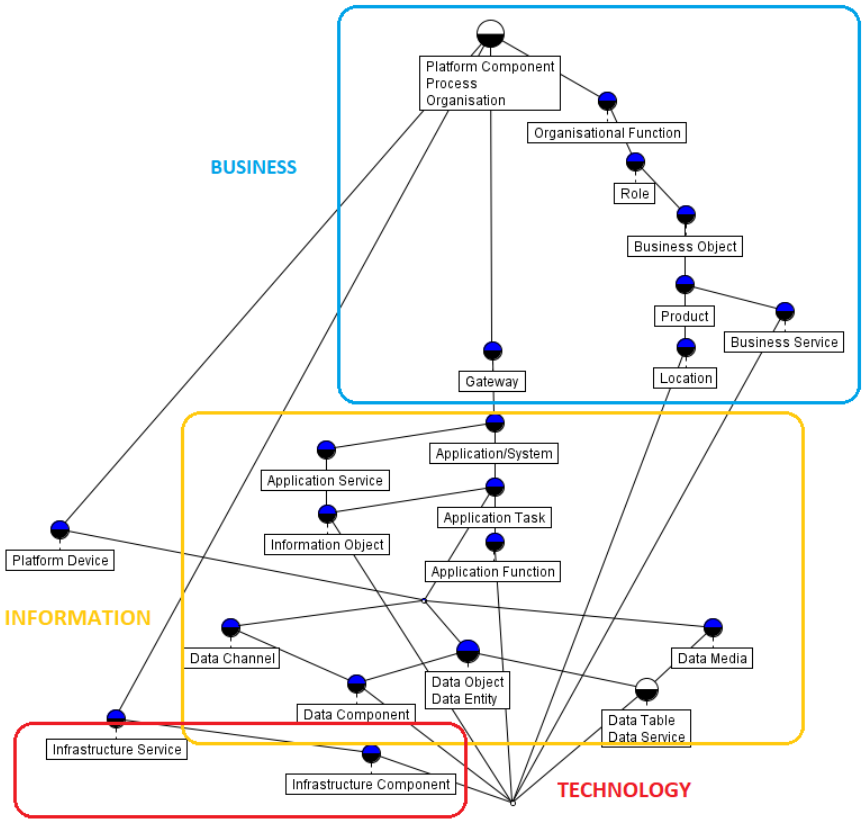

Fig. 4. 00ActiveAll lattice without Attributes, LEAD layers indicated

Figure 4 reveals that LEAD layering is not a distant proposition for the 00ActiveAll lattice, although Platform Component and Platform Device are not presented in the appropriate LEAD layers. We note that while the 00ActiveAll lattice is (mostly) visually displayed in the LEAD layers, it is not so mathematically, e.g., Location and Business Service are positioned preceding the infimum at the bottom of the lattice (along with several other concepts). This outcome is evident by their direct line to the bottom-most empty concept. Overall, the 00ActiveAll lattice appears to suggest that LEAD layering is, to an extent, included in the activation of the semantic relations.

In Table 1, we initiate the refactoring of the lattice, whereby the 00ActiveAll file is broken down and rebuilt step-by-step to remove unwanted cycles[1].

[1] Due to space considerations and only two files presenting cycles, not all the steps are included in this paper.

Table 1. Refactoring the Capability sublayer of the metamodel – Active Organisation and Active Role.

File, Operation, & Outcomes
File: 01ActiveOrganisation.csv **Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Organisation to empty file **Outcome:** No semantic cycles in 01ActiveOrganisation_report.txt
File: 02ActiveRole.csv **Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Role to 01ActiveOrganisation.csv **Outcome:** No semantic cycles in 02ActiveRole_report.txt

Table 2. Refactoring the Data sublayer of the metamodel – Active Data Table.

File, Operation, & Outcomes
File: 18ActiveDataTable.csv **Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Data Table to 17ActiveDataEntity.csv **Outcome:** One semantic cycle in 18ActiveDataTable_report.txt
File: 18v2ActiveDataTable.csv **Operation:** Replacing the implicitly passive 'Data Object - includes – Data Entity' with 'Data Entity – included in – Data Object' **Outcome:** No semantic cycles in 18v2ActiveDataTable_report.txt

Table 3. Refactoring the Data sublayer of the metamodel – Active Data Service.

File, Operation, & Outcomes
File: 19ActiveDataService.csv **Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Data Service to 18v2ActiveDataTable.csv **Outcome:** One semantic cycle in 19ActiveDataService_report.txt
File: 19v2ActiveDataService.csv **Operation:** Replacing the implicitly passive 'Data Table – instantiated in – Data Service' with 'Data Service – instantiated in – Data Table' **Outcome:** No semantic cycles in 19v2ActiveDataService_report.txt

18ActiveDataTable.csv and 19ActiveDataService.csv were the only two files to present any cycles, which were resolved using the operations documented in Tables 2 and 3.

Figure 5 displays the FCL for 25ActiveInfrastructureService, highlighting the effects of the refactoring process. Namely, both the Data Table + Data Service object without attributes and the empty formal concept linking various other formal concepts have been resolved. However, while the operation carried out in 19v2ActiveDataService (replacing 'Data Table – instantiated in – Data Service' with 'Data Service – instantiated in – Data Table') resolved an unwanted cycle, it also displaced Data Service from its appropriate position in the LEAD layers to within the supremum. This outcome suggests that the refactoring of the 00ActiveAll file has caused us to take a step back-

wards, away from our goal of an active, layered lattice. This position is supported by the continued presence of seven concepts (from all three layers) preceding the infimum, but with the addition of Data Service in the supremum. Furthermore, Figure 5 overlays the 25ActiveInfrastructureService lattice with the LEAD layers, highlighting the absence of Data Service and Platform Component in the Information and Technology layers, respectively.

The absence of a metaobject as either a target or source metaobject in the triple composition—where the first triple element is the source, and the third is the target—corresponds with its presence in the FCL as either supremum or preceding infimum, respectively. To maintain the LEAD layer hierarchy, reorienting these metaobjects is the driving force behind the operations carried out and documented in Tables 4 and 5.

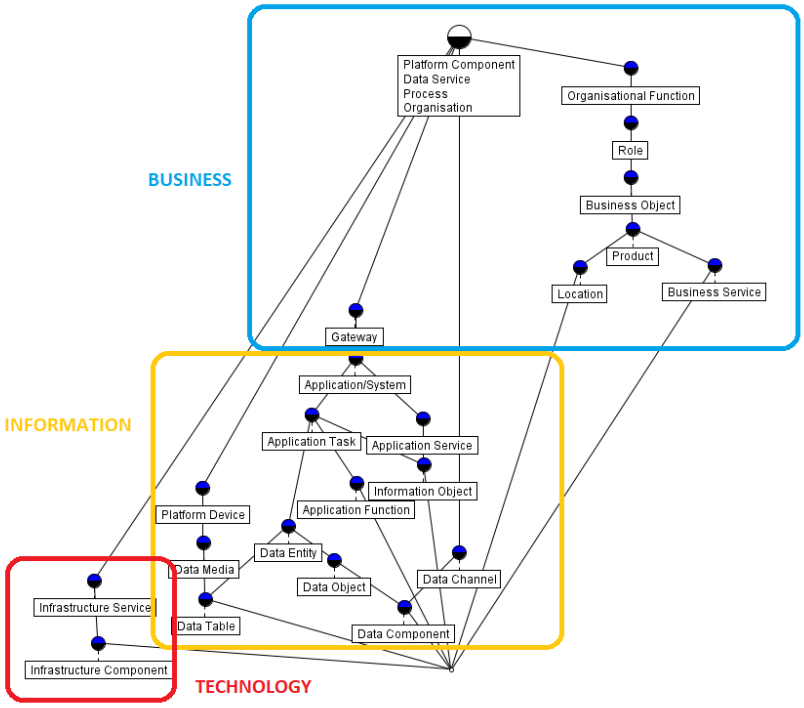

Fig. 5. 25ActiveInfrastructureService lattice without Attributes, LEAD layers indicated

Table 4 displays the operations performed to reposition Data Service and Platform Component to their appropriate LEAD layers while also removing Process from the supremum (as its presence there suggests it being equal to Organisation as a driving force). 25v2ActiveInfrastructureService and 25v7ActiveInfrastructureService are examples of operations that had active relations replaced with the passive relations ('Data Service – uses – Data Media' replaced with 'Data Media – used by –

Data Service' and 'Process – delivers – Business Service' replaced with 'Business Service – delivered by – Process'). We also reversed the identical semantic relations in 25v6ActiveInfrastructureService (where distinguishing active/passive relations was not possible).

Table 4. Refactoring 25ActiveInfrastructureService lattice to achieve LEAD layering – supremum focus.

File, Operation, & Outcomes
File: 25v2ActiveInfrastructureService.csv **Operation:** Replacing 'Data Service – uses – Data Media' with 'Data Media – used by – Data Service' **Outcome:** Data Service no longer in supremum
File: 25v3ActiveInfrastructureService.csv **Operation:** Replacing 'Platform Component – provides the specification for – Platform Device' with 'Platform Device – specified by – Platform Component' **Outcome:** Platform Component no longer in supremum. Platform Device now in supremum
File: 25v4ActiveInfrastructureService.csv **Operation:** Replacing 'Platform Device – hosts – Data Media' with 'Data Media – hosted on – Platform Device' **Outcome:** Platform Device no longer in supremum. Data Media now in supremum
File: 25v5ActiveInfrastructureService.csv **Operation:** Replacing 'Data Media – logically specifies – Data Table' with 'Data Table – specified by – Data Media' **Outcome:** Data Media no longer in supremum. One semantic cycle in 25v5ActiveInfrastructureService_report.txt Cycle: Data Service - instantiated in - Data Table - specified by - Data Media - used by - Data Service
File: 25v6ActiveInfrastructureService.csv **Operation:** Replacing the implicitly passive 'Data Service – instantiated in – Data Table' with 'Data Table – instantiated in – Data Service' **Outcome:** No semantic cycles in 25v6ActiveInfrastructureService_report.txt
File: 25v7ActiveInfrastructureService.csv **Operation:** Replacing 'Process – delivers – Business Service' with 'Business Service – delivered by – Process' **Outcome:** Process no longer in supremum

Table 5 shows how we approached positioning a Technology-layer concept as the infimum. Active relations were again substituted for the passive relation. There were two further instances of reversing identical semantic relations (25v8ActiveInfrastructureService and 25v14ActiveInfrastructureService). Interestingly, the first operation performed is also the last, whereby reversing 'Data Service – uses – Data Media' to remove Data Service from the supremum is then itself reversed to complete the LEAD layered lattice. Other modellers could conceivably make different operation choices consistent with the algorithm, which provides the overarching context under which such decisions can be made.

Table 5. Refactoring 25ActiveInfrastructureService lattice to achieve LEAD layering – infimum focus.

File, Operation, & Outcomes
File: 25v8ActiveInfrastructureService.csv **Operation:** Replacing 'Product – at – Location' with 'Location – at – Product' **Outcome:** Location no longer preceding infimum
File: 25v9ActiveInfrastructureService.csv **Operation:** Replacing 'Application Task – implements – Application Function' with 'Application Function – implemented by – Application Task' **Outcome:** Application Function no longer preceding infimum
File: 25v10ActiveInfrastructureService.csv **Operation:** Replacing 'Application Task – provides – Information Object' with 'Information Object – provided by – Application Task' **Outcome:** Information Object no longer preceding infimum
File: 25v11ActiveInfrastructureService.csv **Operation:** Replacing 'Data Channel – distributes – Data Component' with 'Data Component – distributed through – Data Channel' **Outcome:** Data Component no longer preceding infimum. Data Channel now preceding infimum
File: 25v12ActiveInfrastructureService.csv **Operation:** Replacing 'Data Service – means of distribution is – Data Channel' with 'Data Channel – means of distribution for – Data Service' **Outcome:** Data Channel no longer preceding infimum. One semantic cycle in 25v12ActiveInfrastructureService_report.txt Cycle: Data Object - generalisation of - Data Component - distributed through - Data Channel - means of distribution for - Data Service - encapsulates - Data Object
File: 25v13ActiveInfrastructureService.csv **Operation:** Replacing the implicitly passive 'Data Service – encapsulates – Data Object' with 'Data Object – encapsulated by – Data Service' **Outcome:** Data Service now preceding infimum. No semantic cycles in 25v6ActiveInfrastructureService_report.txt
File: 25v14ActiveInfrastructureService.csv **Operation:** Replacing 'Data Table – instantiated in – Data Service' with 'Data Service – instantiated in – Data Table' **Outcome:** Data Service no longer preceding infimum. One semantic cycle in 25v14ActiveInfrastructureService_report.txt Cycle: Data Service - instantiated in - Data Table - specified by - Data Media - used by - Data Service
File: 25v15ActiveInfrastructureService.csv **Operation:** Replacing the implicitly passive 'Data Media – used by – Data Service' with 'Data Service – uses – Data Media' **Outcome:** No semantic cycles in 25v15ActiveInfrastructureService_report.txt

3.3 Layered Formal Concept Lattice

Figure 6, the visualisation of 25v15ActiveInfrastructureService, shows a successfully layered lattice. While the lattice is predominantly active, it includes various passive relations due to the operations performed to introduce LEAD layering.

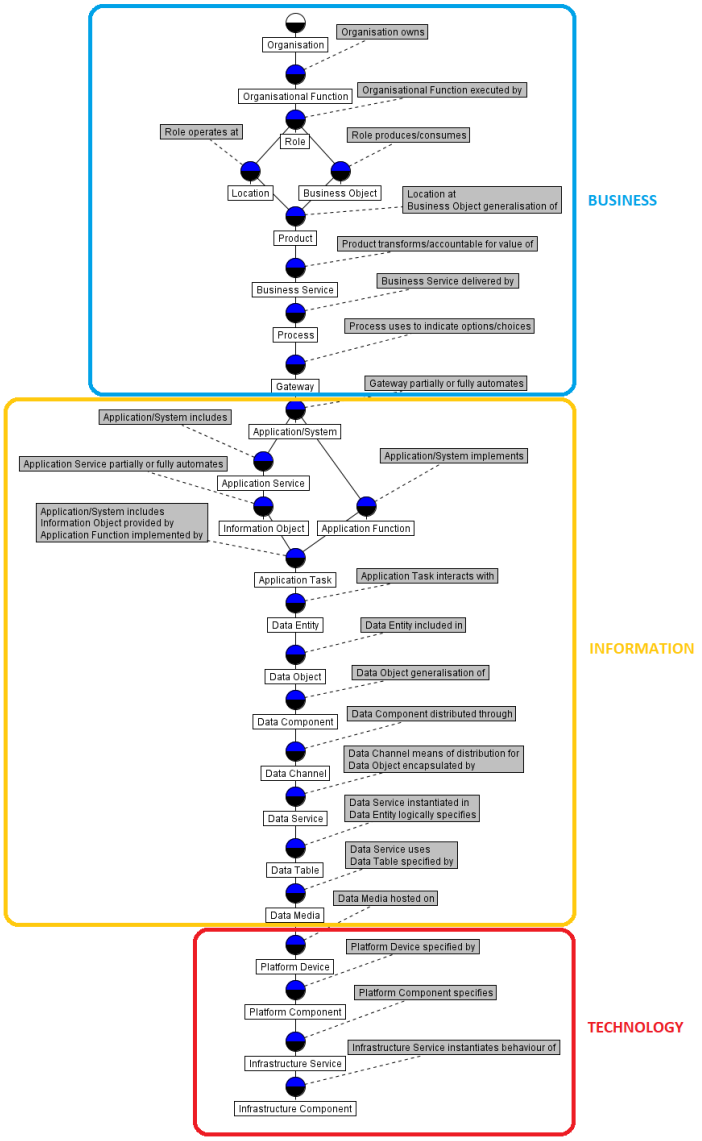

Fig. 6. 25v15ActiveInfrastructureService lattice, LEAD layers indicated

For example, 'Business Service – delivered by – Process' is passive compared to 'Process – delivers – Business Service'. However, the other option was inverting 'Process – uses to indicate options/choices – Gateway', which would have removed Process from the supremum but replaced it with Gateway. This route would have an adverse effect

on the link between the Business and Information layers ('Gateway – partially or fully automates –Application/System'), hence disrupting the layering.

4 Discussion

4.1 Implications

We have shown that an active direction graph can be rebuilt and visualised to display LEAD layering. While the initial 00ActiveAll graph and refactored 25ActiveInfrastructureService graph were largely layered, this demonstrates the contrast between how humans process information and how a computer does. Certain concepts in the 00ActiveAll and refactored 25ActiveInfrastructureService FCLs (Location, Business Service, Application Function, Information Object, Data Component, and Data Table), while not mathematically appropriately layered, could be positioned in a way that exhibits LEAD layering to the human eye. Our principled approach introduced this mathematical layering to narrow the gap between human and computer interpretation of the metamodel[2]. Readability is patently improved when contrasting the layered 25v15ActiveInfrastructureService graph with the purely active graphs (00ActiveAll and 25ActiveInfrastructureService), which has value for business decision-makers when identifying the levers required to effect change in an organisation. By attempting to create a directed graph that is both active and layered, we could elucidate that layering can be included in the remit of semantic relations. We also observed that LEAD's overall active flow is primarily top-down, nudged along by the occasional bottom-up active flow, suggesting that business mostly drives technology, with some instances where technology demonstrates that the reverse is possible.

4.2 Current Limitations

While a layered lattice was attained, there remains room for ongoing development of our approach. Firstly, the manual nature of the current approach means that there is room for interpretation and that the rigour in the execution of the pseudo-algorithm depends on the human executor. As the size of the data set increases, a human modeller can inadvertently stray from the proposed refactoring logic. While this challenge could be overcome by executing the logic in a computer-implemented algorithm, there are various considerations before such an approach can be developed. For example, the logical conclusion of selecting the 'least active' triple when applying FCL layering is that the semantic relations would need to be ranked to be interpreted by a computer, such as by using a passivity scale. However, other criteria could be included, such as the impact of the proposed operation on the lattice's intended (layered) structure. We created a layered lattice without introducing further relations to the metamodel portrayed by Figure 1, but including other relations could add value. Within the omitted 117 two-way semantic relations possible with Figure 1, other active relations may have resolved the supremum and infimum issues by other means while also minimising or even eradicating the need for the inclusion of semantically passive relations. Furthermore, there are more than 3,300 semantic relations in the LEAD ontology, thus fertile ground to explore.

[2] Thus aligning information processing in mind and machine in accordance with the subtitle of Sowa's seminal text on Conceptual Structures [5].

4.3 Future Research

Given the above considerations, by inserting LEAD's integrated ontology elements that are only implicit in the case study metamodel, it would be possible to highlight further the levers that can be pulled to effect the desired change. This future work would also tease out more of LEAD's hitherto hidden insights. Furthermore, refinement of the algorithm could be pursued to integrate the activation and layering stages of the approach. We seek ways it can be more automated. One such method is by considering layering (and supremum and infimum composition) at the point of concept introduction. We could then conceivably resolve undesirable, complex-to-handle situations whereby the operations of one stage adversely impact another, contributing to expressive lattices dynamically modified according to any given organisational situation.

5 Conclusion

We have demonstrated a principled approach that successfully introduces LEAD layering to an active direction graph. Consequently, the outcome is an improved value proposition for industry, in the form of an inherently more readable illustration of change levers for business decision-makers. It would promote understanding of Layered Enterprise Architecture Development (LEAD) and the LEAD Enterprise Ontology, allowing organisations to exploit their information technology assets instead of being exploited by them.

References

1. Simon Andrews and Simon Polovina. Exploring, Reasoning with and Validating Directed Graphs by Applying Formal Concept Analysis to Conceptual Graphs. In Madalina Croitoru, Pierre Marquis, Sebastian Rudolph, and Gem Stapleton, editors, *Graph Structures for Knowledge Representation and Reasoning*, volume 10775 LNAI, pages 3–28, Cham, 2018. Springer International Publishing.
2. Matt Baxter, Simon Polovina, Wim Laurier, and Mark von Rosing. Active semantic relations in layered enterprise architecture development. In Michael Cochez, Madalina Croitoru, Pierre Marquis, and Sebastian Rudolph, editors, *Graph Structures for Knowledge Representation and Reasoning*, pages 3–16, Cham, 2021. Springer International Publishing.
3. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis*. Springer Berlin Heidelberg, 1999.
4. Simon Polovina, Mark von Rosing, and Georg Etzel. Leading the Practice in Layered Enterprise Architecture. In *CEUR Workshop Proceedings*, volume 2574, pages 62–69, 2020.
5. John F Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley Longman Publishing Co., Inc., 1984.
6. John F. Sowa. *Conceptual graphs*, chapter 5, pages 213–237. Elsevier, 2008.
7. Mark von Rosing and Wim Laurier. An introduction to the business ontology. *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*, 3(1):20–41, Jan 1, 2015.