Active Semantic Relations in Layered Enterprise Architecture Development

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

[1] Conceptual Structures Research Group, Sheffield Hallam University, Sheffield, UK
a7033771@my.shu.ac.uk, S.Polovina@shu.ac.uk

[2] Université Saint-Louis, Brussels, Belgium
wim.laurier@usaintlouis.be

[3] LEADing Practice, Dronningmølle, Denmark
mvr@leadingpractice.com

Abstract. Enterprise Architecture (EA) metamodels align an organisation's business, information and technology resources so that these assets best meet the organisation's purpose. The Layered EA Development (LEAD) Ontology enhances EA practices by a metamodel with layered metaobjects as its building blocks interconnected by semantic relations. Each metaobject connects to another metaobject by two semantic relations in opposing directions, thus highlighting how each metaobject views other metaobjects from its perspective. While the resulting two directed graphs reveal all the multiple pathways in the metamodel, more desirable would be to have one directed graph that focusses on the dependencies in the pathways. Towards this aim, using CG-FCA (where CG refers to Conceptual Graph and FCA to Formal Concept Analysis) and a LEAD case study, we determine an algorithm that elicits the active as opposed to the passive semantic relations between the metaobjects resulting in one directed graph metamodel. We also identified the general applicability of our algorithm to any metamodel that consists of triples of objects with active and passive relations.

Keywords: Enterprise architecture frameworks · Layered enterprise architecture development · Business-IT alignment · Ontology · Semantics and reasoning · Conceptual structures · Model verification and validation

1 Introduction

Enterprise Architecture (EA) is a comprehensive approach to the documentation and understanding of organisational composition to promote alignment of its business, information and technology assets [9]. The Layered Enterprise Architecture Development (LEAD) Ontology includes a metamodel that is underpinned by building blocks consisting of 91 metaobjects organised in layers and sub-layers [7,14]. Semantic relations link the metaobjects thereby integrating all aspects of business, information, and technology for any organisation. These multiple relations highlight the inbuilt interconnections and the interdependencies between the elements in an enterprise. Conceptual Graphs (CG) are a formalised method of knowledge representation based on concepts and their relations [11,12]. Formal Concept Analysis (FCA) is a principled approach to determining a conceptual hierarchy of objects and their attributes [15]. FCA interrelates objects through their related attributes, thus enabling FCA to determine and visualise a conceptual hierarchy [3]. A CG can visually display LEAD's metaobjects and their semantic relations by linking each concept to another via these relations; however, validation can be difficult due to the manual nature of the task [1]. Subsequently, processing these 'triples' (metaobject–relation–metaobject) via FCA can highlight gaps in the model, revealing an organisational gap or human error in the modelling process. Thus, while a manual review of the LEAD artefacts can identify organisational gaps, an element of mathematical rigour can be applied to the process thereby complementing LEAD through the application of CG and FCA [6,8].

2 The Metamodel Diagram

To illustrate the contribution of CG and FCA, Fig. 1 acts as our starting point. This figure represents the metamodel of a warehouse pick pack process of a UK manufacturer, based on the LEAD Enterprise Ontology referred to earlier (i.e. LEAD ID#-ES20001ALL) [13]. The metamodel was created using the Enterprise Plus (E+) software (www.enterpriseplus.tools) from LEADing Practice, a not-for-profit body of LEAD industry practitioners (www.leadingpractice.com). E+ is a comprehensive repository of LEAD reference content, including its artefacts, metaobjects, and semantic relations. The semantic relations in Fig. 1 go in two directions between each metaobject. This duality is intended in many EA metamodels, including LEAD. That is because it reveals how each metaobject views itself in relation to each other directly, and indirectly through intermediate metaobjects; hence LEAD metamodels are two-way directed graphs [9].

3 Activating the Metamodel

The *CGtoFCA* algorithm converts the inherent ternary relations of CGs to the binary relations required for FCA [1]. This algorithm can also apply to other directed graph triples, including LEAD metamodels as illustrated by Fig. 1 [9]. The formal concepts can then appear in a Formal Concept Lattice (FCL). The CG-FCA software based on *CGtoFCA* thus facilitates an improved understanding of LEAD metamodels in tandem with highlighting human errors in the manual modelling process [1,9]. Further to that previous work, and in search of the metaobjects' dependence on each other, the proposed algorithm shown in Fig. 2 distinguishes the active and passive semantic relations. An active relation depicts a situation whereby a metaobject directs another, with the latter metaobject dependent on it, i.e. the passive relation. Following the identification

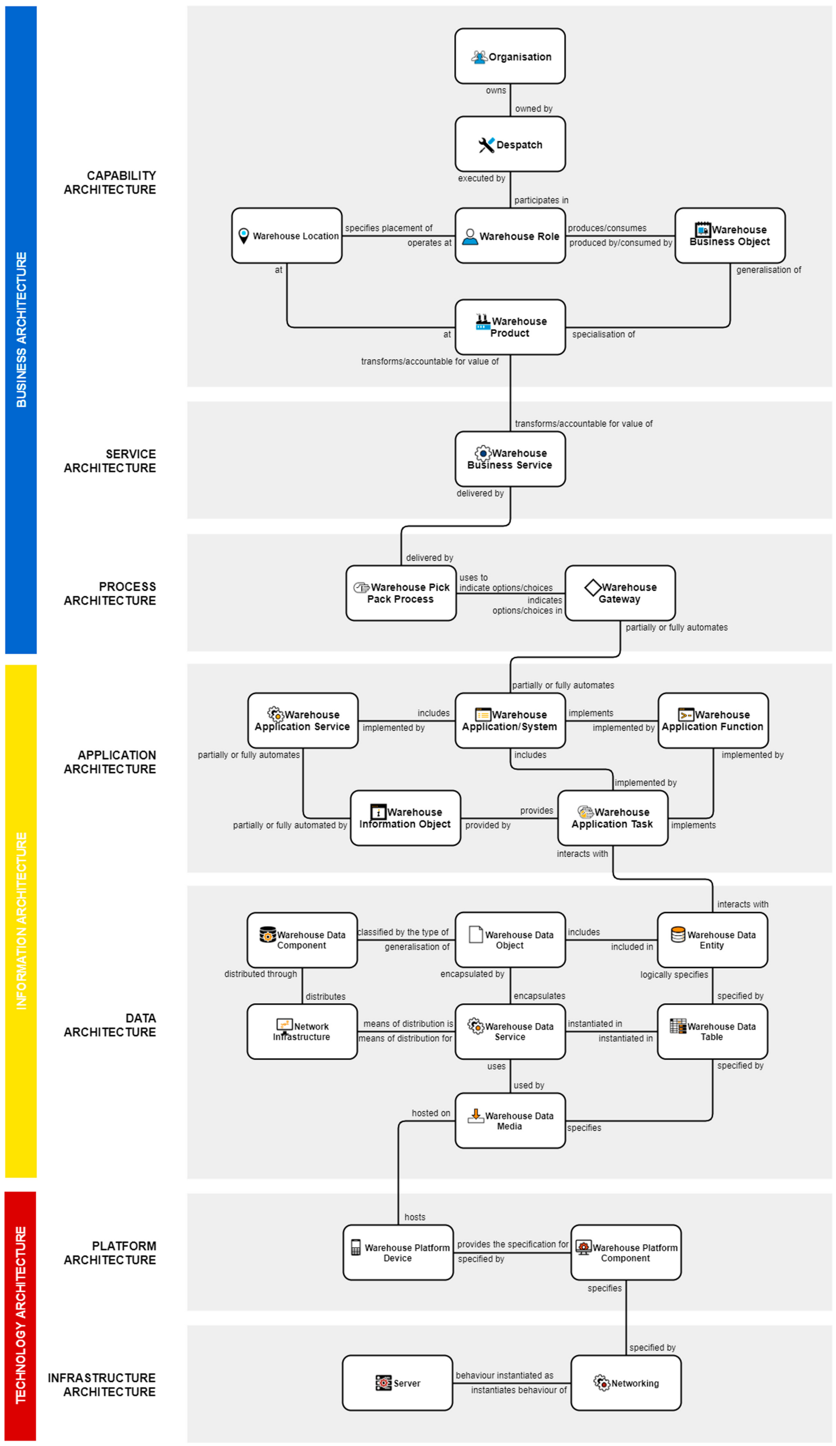

Fig. 1. Warehouse pick pack metamodel (from LEAD ID#-ES20001ALL)

of all the active relations, the algorithm incrementally rebuilds the model and removes unwanted semantic cycles before being visualised in an FCL.

3.1 Methodology

Using the algorithm depicted by Fig. 2, we identify and analyse the active semantic relations towards our goal of attaining an active direction graph, thus highlighting the metaobject dependencies. Strictly-speaking, our algorithm is presently more of a 'pseudo-algorithm' as it requires human interpretation. For example, in line 19 $isTransitive(v)$ we could debate this step, with one possibility that we should just invert the relation. Formalising the algorithm so that it can be computer-executed is the subject of our ongoing work. Meanwhile, Fig. 2 fits the present purpose of our claims.

```
1  begin
2      $A = \emptyset$
3      $M = \emptyset$
4      foreach $(o, v, s) \in B$ do
5          $\neg\exists o \in M | M = M \cup o$
6          $\neg\exists s \in M | M = M \cup s$
7      foreach $o \in M$ do
8          foreach $(o, v, s) \in B$ do
9              if $isPassive(v)$ then
10                 $\neg\exists(s, v', o) \in A | A = A \cup (s, v', o)$
11             else
12                 $\neg\exists(o, v, s) \in A | A = A \cup (o, v, s)$
13         $C = triplesInCycles(A)$
14         if $C \neq \emptyset$ then
15             foreach $(o, v, s) \in C$ do
16                 if $isImplicityPassive(v))$ then
17                     $A = A \backslash (o, v, s)$
18                     $\neg\exists(s, v', o) \in A | A = A \cup (s, v', o)$
19                 if $isTransitive(v))$ then
20                     $A = A \backslash (o, v, s)$
21                 if $inMultipleCycles(o, v, s))$ then
22                     $A = A \backslash (o, v, s)$
23                     $\neg\exists(s, v', o) \in A | A = A \cup (s, v', o)$
24 end
```

Fig. 2. Active semantic relations algorithm

Following Fig. 2, we reviewed each two-way semantic relation to determine which should be assigned active or passive status and created an initial active model. We examined the semantics in the narrative of the relations and identified which metaobject was directing the other and vice versa. We then rebuilt the model by reviewing each concept in turn to remove semantic cycles [9]. Where both a direct and indirect pathway exists between two metaobjects, we removed the former, as the latter illustrates the mediating metaobjects. This step enabled a deeper understanding of the interdependencies. The ternary relations were compiled as 3-column CSV files and processed by the CG-FCA application to create the binary concepts. The operations and outcomes for each metaobject CSV file were recorded in a table to document the steps taken. After successfully refactoring each concept, we generated the FCL.

3.2 Findings

Following the selection of the active semantic relations in the one hundred forty-seven pairs of relations, the 00ActiveAll.csv file was unable to be processed by the CG-FCA application despite multiple attempts. The final attempt was aborted with the '00ActiveAll_report' file having amassed a size of over 10 GB after nearly eighty-eight hours of processing time. This first experiment prevented the creation of an FCL for the initial active model.

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

File	Operation & Outcomes
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
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
03ActiveOrganisationalFunction.csv	**Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Organisational Function to 02ActiveRole.csv **Outcome:** No semantic cycles in 03ActiveOrganisationalFunction_report.txt

Identifying the source of this seemingly infinite processing run was therefore attempted by employing an iterative approach and gradually increasing the number of triples included in 00ActiveAll.csv; however, we then encountered further issues. For example, in the case of 00ActiveAllDataObject1.csv (comprised of all 00ActiveAll triples up to and including the first instance of a Data Object triple), the processing time totalled just over twelve hours. Hence, there exists an issue of practicality in attempting to identify the triple that is causing the seemingly infinite compilation. We thus judged when to abort the processing due to uncertainty surrounding whether the processing run will not complete or whether it is only taking longer than expected compared to the previous iteration. The difficulty of the decision became exacerbated as processing time appears dependent on both the triple inserted and existing triples in the file, in the sense that one triple could cause a minimal increase in processing time while the impact of another could be significant. This intractability could reflect a combinatorial explosion: the number of input values increases exponentially with the number of potential outputs [2]. Nonetheless, and in light of the above experiences, we were able to proceed.

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

File	Operation & Outcomes
16ActiveDataObject.csv	**Operation:** Adding all active (o, v, s) ϵ 00ActiveAll.csv with o or s = Data Object to 15ActiveDataComponent.csv **Outcome:** Two hundred thirty-five semantic cycles in 16ActiveDataObject_report.txt
16v2ActiveDataObject.csv	**Operation:** Deletion of transitive relation 'Data Object - influences the design of - Application Service' **Outcome:** One hundred twelve semantic cycles in 16v2ActiveDataObject_report.txt
16v3ActiveDataObject.csv	**Operation:** Deletion of transitive relation 'Data Service – encapsulates – Data Object' **Outcome:** Three semantic cycles in 16v3ActiveDataObject_report.txt
16v4ActiveDataObject.csv	**Operation:** Deletion of transitive relation 'Data Object - influences the design of - Application Task' **Operation:** Deletion of transitive relation 'Data Object - assumes or specifies - Platform Component' **Outcome:** No semantic cycles in 16v4ActiveDataObject_report.txt

The first five metaobject CSV files contained no cycles, three of which are detailed in Table 1. Subsequently, five cycles appeared in 06ActiveLocation.csv. The decision to replace 'Product - at - Location' with 'Location - at - Product' resolved all cycles[1].

We also encountered cycles in the LEAD Data sublayer, with cycles ranging from one to two hundred and seventy-nine. Table 2 shows the three iterations

[1] Not all the metaobjects and semantic relations appear in Fig. 1, including these two-way metaobjects and semantic relations, to maintain the figure's readability.

required to resolve all cycles initially presented in 16ActiveDataObject.csv. Due to space considerations, we do not list these cycles. We identified 'Platform Component – serves – Location' as a common triple across cycles; however, an alternative pathway remained undiscovered. 'Location –has – Process – produces/consumes – Data Object' exists as a more indirect pathway. However, we deleted it as part of an operation for 08v2ActiveProcess.csv, which highlights the cumulative effect of the decisions made at each stage of refactoring. Consequently, we made alternative choices. Considering the vast number of initial cycles presented (two hundred and thirty-five) and the manual nature of the activity, it is possible that a more indirect pathway does exist but overlooked by a human modeller.

3.3 Formal Concept Lattice

To visualise the output of CG-FCA, we created the FCL for 25ActiveInfrastructureService.csv, displayed in Fig. 3. The FCL lucidly exhibits the dependencies and driving metaobjects. A salient example is Product illustrated as being dependent on Process, which in turn is dependent on Role. In the context of the warehouse pick pack process, this dependency suggests that the product that is picked and packed is dependent on the process for doing so, which in turn is dependent on the employee that executes the process. Perhaps the most initially striking element of the FCL is the presence of Platform Component within the top-most formal concept, implying all objects below it in the diagram, i.e. its extent, are in some way dependent on it. While we might expect that technology ought to be driven by business, technology can drive business. For example, in recent years, the rise of cloud computing (a Platform Component) has driven a proliferation of decentralised business models. Accordingly, remote working is the norm and the presence of physical business components (Business Object, Location) is either minimised or eschewed entirely dependent on the industry.

A further interesting element elucidated in the FCL is 'Platform Device – hosts – Application/System', which implies that an Application/System is dependent on a Platform Device. This active pathway suggests that Platform Devices are the starting points, with the Application/System developed based on the specifications, constraints, and existence of the Platform Devices. While this makes sense, so does the opposing view, whereby Platform Device should be dependent upon Application/System because without an application to run, for what purpose does the device exist?

The presence of an empty formal concept close to the top of the lattice is also notable, and several potential explanations exist. Firstly, it could merely be a mistake in the modelling process, a probability which is heightened by the vast number of cycles encountered at some stages of the refactoring. Secondly, it could also be that the empty formal concept is irrelevant, as it exists purely as a vehicle for the facilitation of human understanding. Thirdly, and most speculatively, it could be pointing to a hitherto unnamed formal concept object, which in turn could potentially indicate a new metaobject arising from the other metaobjects and semantic relations, already validated by the LEADing Practice community.

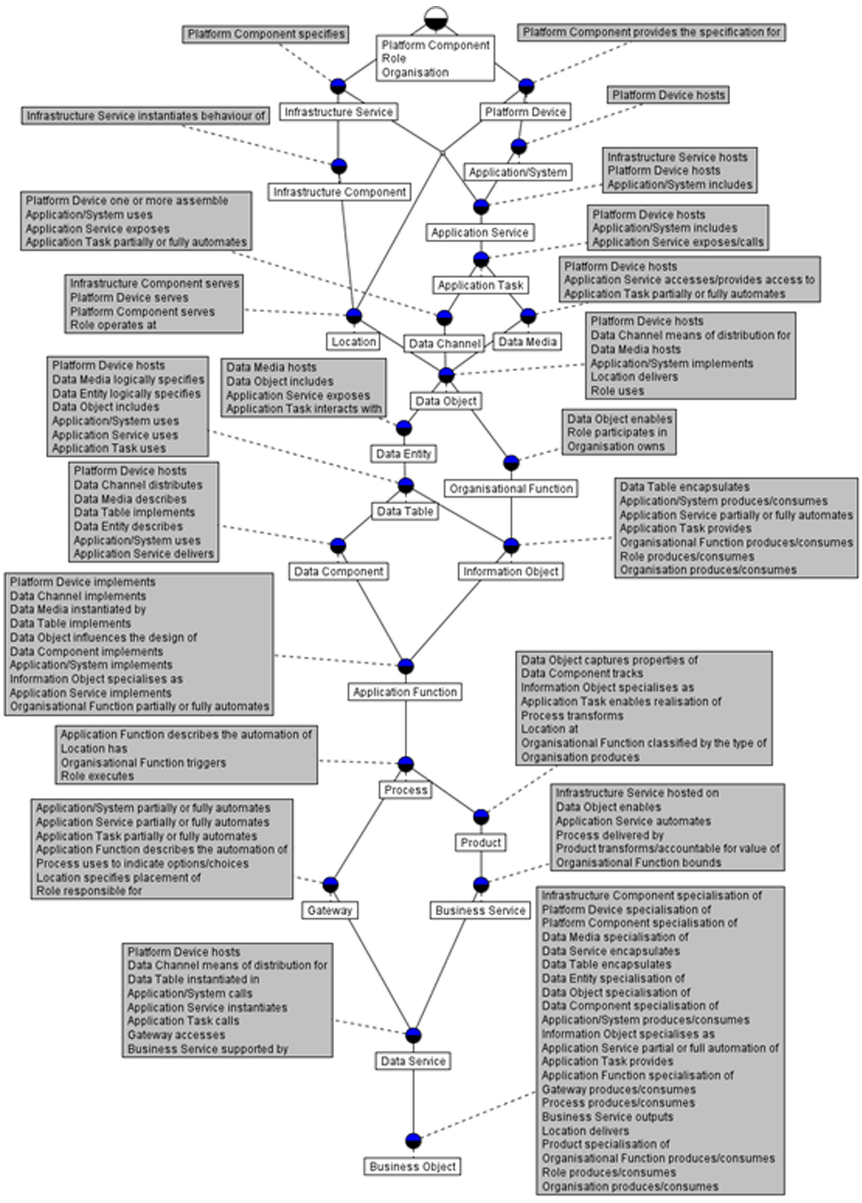
Platform Component
Role
Organisation
Platform Component specifies
Platform Component provides the specification for
Infrastructure Service
Platform Device
Platform Device hosts
Infrastructure Service instantiates behaviour of
Application/System
Infrastructure Component
Infrastructure Service hosts
Platform Device hosts
Application/System includes
Platform Device one or more assemble
Application/System uses
Application Service exposes
Application Task partially or fully automates
Application Service
Platform Device hosts
Application/System includes
Application Service exposes/calls
Application Task
Platform Device hosts
Application Service accesses/provides access to
Application Task partially or fully automates
Infrastructure Component serves
Platform Device serves
Platform Component serves
Role operates at
Location
Data Channel
Data Media
Platform Device hosts
Data Channel means of distribution for
Data Media hosts
Application/System implements
Location delivers
Role uses
Platform Device hosts
Data Media logically specifies
Data Entity logically specifies
Data Object includes
Application/System uses
Application Service uses
Application Task uses
Data Media hosts
Data Object includes
Application Service exposes
Application Task interacts with
Data Object
Data Object enables
Role participates in
Organisation owns
Data Entity
Organisational Function
Platform Device hosts
Data Channel distributes
Data Media describes
Data Table implements
Data Entity describes
Application/System uses
Application Service delivers
Data Table
Data Table encapsulates
Application/System produces/consumes
Application Service partially or fully automates
Application Task provides
Organisational Function produces/consumes
Role produces/consumes
Organisation produces/consumes
Data Component
Information Object
Platform Device implements
Data Channel implements
Data Media instantiated by
Data Table implements
Data Object influences the design of
Data Component implements
Application/System implements
Information Object specialises as
Application Service implements
Organisational Function partially or fully automates
Application Function
Data Object captures properties of
Data Component tracks
Information Object specialises as
Application Task enables realisation of
Process transforms
Location at
Organisational Function classified by the type of
Organisation produces
Application Function describes the automation of
Location has
Organisational Function triggers
Role executes
Process
Infrastructure Service hosted on
Data Object enables
Application Service automates
Process delivered by
Product transforms/accountable for value of
Organisational Function bounds
Product
Application/System partially or fully automates
Application Service partially or fully automates
Application Task partially or fully automates
Application Function describes the automation of
Process uses to indicate options/choices
Location specifies placement of
Role responsible for
Gateway
Business Service
Infrastructure Component specialisation of
Platform Device specialisation of
Platform Component specialisation of
Data Media specialisation of
Data Service encapsulates
Data Table encapsulates
Data Entity specialisation of
Data Object specialisation of
Data Component specialisation of
Application/System produces/consumes
Information Object specialises as
Application Service partial or full automation of
Application Task provides
Application Function specialisation of
Gateway produces/consumes
Process produces/consumes
Business Service outputs
Location delivers
Product specialisation of
Organisational Function produces/consumes
Role produces/consumes
Organisation produces/consumes
Platform Device hosts
Data Channel means of distribution for
Data Table instantiated in
Application/System calls
Application Service instantiates
Application Task calls
Gateway accesses
Business Service supported by
Data Service
Business Object

Fig. 3. 25ActiveInfrastructureService lattice

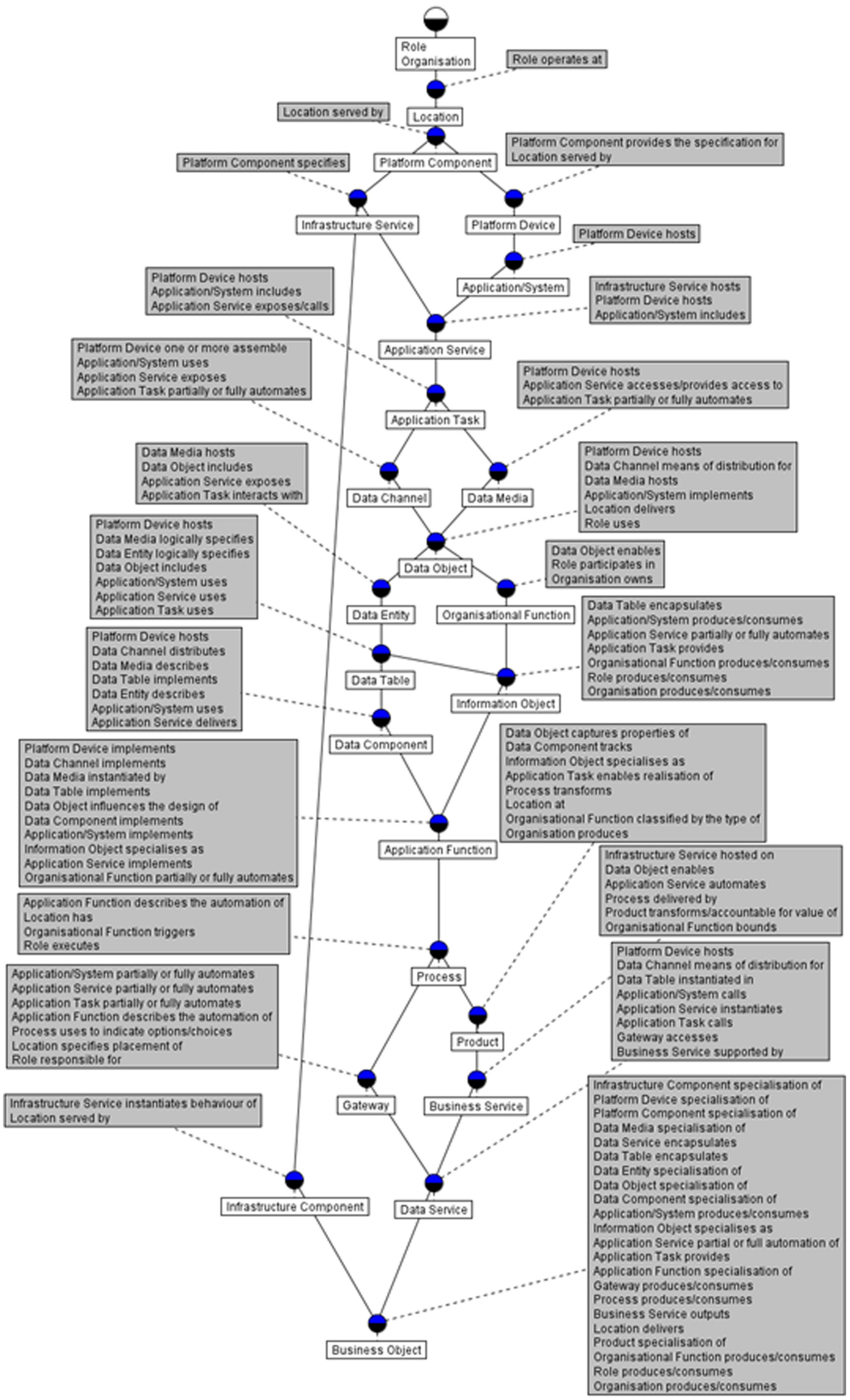
Role
Organisation
Role operates at
Location served by
Location
Platform Component specifies
Platform Component
Platform Component provides the specification for
Location served by
Infrastructure Service
Platform Device
Platform Device hosts
Platform Device hosts
Application/System includes
Application Service exposes/calls
Application/System
Infrastructure Service hosts
Platform Device hosts
Application/System includes
Application Service
Platform Device one or more assemble
Application/System uses
Application Service exposes
Application Task partially or fully automates
Platform Device hosts
Application Service accesses/provides access to
Application Task partially or fully automates
Application Task
Data Media hosts
Data Object includes
Application Service exposes
Application Task interacts with
Platform Device hosts
Data Channel means of distribution for
Data Media hosts
Application/System implements
Location delivers
Role uses
Data Channel
Data Media
Platform Device hosts
Data Media logically specifies
Data Entity logically specifies
Data Object includes
Application/System uses
Application Service uses
Application Task uses
Data Object
Data Object enables
Role participates in
Organisation owns
Data Entity
Organisational Function
Data Table encapsulates
Application/System produces/consumes
Application Service partially or fully automates
Application Task provides
Organisational Function produces/consumes
Role produces/consumes
Organisation produces/consumes
Platform Device hosts
Data Channel distributes
Data Media describes
Data Table implements
Data Entity describes
Application/System uses
Application Service delivers
Data Table
Information Object
Data Component
Data Object captures properties of
Data Component tracks
Information Object specialises as
Application Task enables realisation of
Process transforms
Location at
Organisational Function classified by the type of
Organisation produces
Platform Device implements
Data Channel implements
Data Media instantiated by
Data Table implements
Data Object influences the design of
Data Component implements
Application/System implements
Information Object specialises as
Application Service implements
Organisational Function partially or fully automates
Application Function
Infrastructure Service hosted on
Data Object enables
Application Service automates
Process delivered by
Product transforms/accountable for value of
Organisational Function bounds
Application Function describes the automation of
Location has
Organisational Function triggers
Role executes
Process
Platform Device hosts
Data Channel means of distribution for
Data Table instantiated in
Application/System calls
Application Service instantiates
Application Task calls
Gateway accesses
Business Service supported by
Application/System partially or fully automates
Application Service partially or fully automates
Application Task partially or fully automates
Application Function describes the automation of
Process uses to indicate options/choices
Location specifies placement of
Role responsible for
Product
Gateway
Business Service
Infrastructure Service instantiates behaviour of
Location served by
Infrastructure Component specialisation of
Platform Device specialisation of
Platform Component specialisation of
Data Media specialisation of
Data Service encapsulates
Data Table encapsulates
Data Entity specialisation of
Data Object specialisation of
Data Component specialisation of
Application/System produces/consumes
Information Object specialises as
Application Service partial or full automation of
Application Task provides
Application Function specialisation of
Gateway produces/consumes
Process produces/consumes
Business Service outputs
Location delivers
Product specialisation of
Organisational Function produces/consumes
Role produces/consumes
Organisation produces/consumes
Infrastructure Component
Data Service
Business Object

Fig. 4. 25v2ActiveInfrastructureService lattice

To remedy Platform Component's presence in the top-most formal concept, we reviewed the FCL and identified the source as 'Platform Component – serves – Location'. For convenience, the triple was substituted with the passive triple, as were the two further triples containing the 'serves' semantic relation. Figure 4 displays the resultant FCL.

The revised FCL arguably presents a more intuitive model in the context of the warehouse pick pack process, with Location preceding Platform Component and much of the lattice being dependent upon the former. As pick pack represents the physical process of picking and packing goods at a location – a concept that pre-dates technology platforms, the revised interpretation offers a more lucid model. However, we note that due to the manual and interpretative nature of the exercise, other modellers could feasibly reach different conclusions.

4 Discussion

4.1 Implications

We have demonstrated that an active direction graph can be attained via the identification of active semantic relations, rebuilding of concepts, and visualisation via an FCL. The proposed algorithm depicted by Fig. 2 enabled us to elaborate on the identification and rebuilding stages, supported by the *CGtoFCA* algorithm implemented in the CG-FCA application. The ensuing FCL presented a clear view of metaobject dependency and driving forces, consequently providing a deeper understanding of the LEAD framework both generally and in the context of a warehouse pick pack process.

Furthermore, the presence of an unnamed concept in the 25ActiveInfrastructureService lattice could prompt a further, deeper examination of the semantics, potentially leading to refined semantic relations or a new metaobject. These enhancements would underpin the rigour of LEAD, by revealing which metaobjects are consistently driving others due to their active and passive semantic relations. It is in this scenario where the active-directed graphs visualised as FCLs provide value, due to their explicit ordering of driving forces and dependencies. It is conceivable that such diagrams could, due to their facilitation of more in-depth understanding, provide business users with direction when attempting to resolve issues or enact continuous improvement. For example, for an organisation wishing to improve the KPIs of a Business Service, the active FCL outlines all other metaobjects on which the Business Service is dependent, and highlighted by Fig. 5.

In the context of the warehouse pick pack process, if we consider the 'picking' Business Service, the active FCL suggests this is dependent upon Process. Review of the decomposition of the Process metaobject shows the various process steps undertaken by the Warehouse Admin. Many of these steps must be completed before the Picker can begin picking, which supports the notion that the picking Business Service's KPIs, e.g. picks per hour, could be adversely impacted by the process on which it depends.

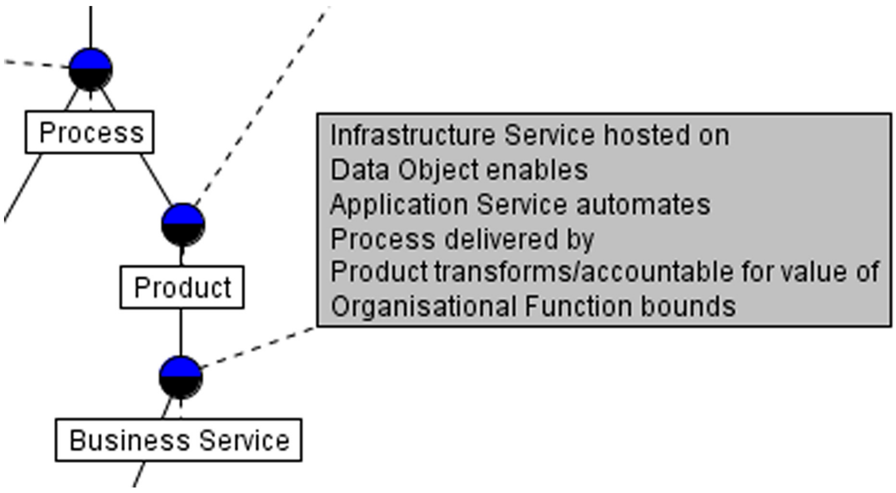
Process
Product
Business Service
Infrastructure Service hosted on
Data Object enables
Application Service automates
Process delivered by
Product transforms/accountable for value of
Organisational Function bounds

Fig. 5. Activated Business Service object and attributes

4.2 Current Limitations

We are aware that our choice of semantic relations from E+ might question the external validity of the work. From our experiments, we can quantify the scale of absent semantic relations as fifty-four out of two hundred ninety-four for the selected metaobjects. However, the number of incorrectly identified semantic relations (e.g. process – delivered by – Business Service) is unknown at this time. Both issues affect the selection process, as potentially erroneous assumptions for the former and the latter are uncertain by nature. These considerations are pertinent as they influence the active vs. passive selection, which in turn impacts all pathways associated with the triple. Inclusion of a triple from all one hundred forty-seven pairs of semantic relations potentially contributed to the issues with the CG-FCA application, reflecting the combinatorial explosion.

Similarly, the inclusion of triples with identical two-way semantic relations, e.g. Application Task and Data Table, increased the complexity of the task, subsequently increasing the likelihood of errors. While we based our approach on the proposed algorithm and selecting the TDV relation in these instances based on sound logic, alternative methods may exist. The omission of all identical two-way semantic relations would provide consistency but also prevent the explication of all pathways containing those triples. The manual nature of the exercise should also be considered, especially in the case of where many cycles occur. Determining which triple is most common across cycles by eye is imprecise when reviewing such a substantial data set.

Furthermore, we chose pathways based upon our intuitive knowledge of the LEAD framework. For example, during refactoring of 08ActiveProcess.csv, three triples were deleted (‘Process – produces/consumes – Data Object’, ‘Process – produces/consumes – Information Object’, and ‘Application Task – partially or fully automates – Process') based on the assumption that other pathways with more mediating metaobjects existed. This decision was based on the distance between the metaobjects in the LEAD layers and was later validated with the discovery of 'Data Object – influences the design of – Application Task – uses – Data Table – encapsulates – Information Object – specialises as – Application Function – describes the automation of – Process' in 16ActiveDataObject_report.

However a more precise approach might be preferable, such as a tool that accepts an input and output metaobject in addition to all other metaobjects within the set, before returning a list of pathways in descending length order. If an algorithm comprises both logic and control, we can improve its control element [5]. The modeller acting as a 'manual' control by 1) being aware of the effect of a more significant number of triples and therefore limiting them, and 2) determining triple commonality across cycles by eye, is not optimal. As we have demonstrated, the proposed algorithm significantly assisted, thus based on our experiences, there are routes to refine it further. Therefore, the approach could be improved if the refined version complemented the CGtoFCA algorithm implemented in the CG-FCA application. Hence, the refined version duly implemented alongside CG-FCA can account for one or both these issues.

4.3 Future Research

We started with a(n) (ontology-based) metamodel, composed of concepts that were related by two-way, or bidirectional, relationships. The large majority of these bidirectional relationships seemed to be active in one direction and passive in the other. The LEAD metamodel reveals which aspects of business (the concepts) act upon or impact on others. In the context of change management (but also of the day-to-day management of a company) it is important to be able distinguish between the causes (active) and the effects (passive) of management issues (in day-to-day management) and identify the levers (active) needed to "pull" in order to realise the wanted change, while accounting for the passive effects that pulling the levers might have.

In case semantic relationships were two-way active or two-way passive, we needed to evaluate whether they could be reformulated as active-passive couples, i.e. the presently pseudo-algorithm (Fig. 2) into one that can be computer-implemented. With help from software libraries or web services that for example allow us to identify and rephrase passive and active relationships—e.g. Grammarly (www.grammarly.com) or DeepL (www.deepl.com)—the pseudo-algorithm could be automated as real executable code.

Our formal analysis of the metamodel has two main objectives. First, optimising the hands-on nature of the metamodel as a management tool: by separating the active from the passive semantics it is easier to find causes of a management issue and the levers that act upon this problem (that needs to be addressed) using the active semantics. Additionally, the passive semantics allow for identifying the effects of this management issue (and building the business case for the change). Moreover, the passive semantics will allow for identifying the (positive and negative) side-effects of the change, as the levers that are chosen or pulled will have an impact on the change goal, but also on other aspects of management that are actively affected. As such this clear "chain of command" is expected to both help identify the levers to obtain a desired change and minimise its adverse effects. Second, in ontology engineering there is an expectation that directed graphs with active and passive semantic relations should be isomorphic, i.e. a passive directed graph is the flip side of an active one. However, where they are not, there needs to be an elaboration. Is the "chain of command" thus asymmetric, and why, or are there missing concepts? As such this formal approach could be combined with OntoClean, METHONTOLOGY or other ontology engineering approaches [4,10].

5 Conclusion

We have shown that by distinguishing the active semantic relations in bidirectional (two-way directed) graphs that we can identify the dependencies in metamodels from their metaobject and semantic relation building blocks. Furthermore, we outlined how our approach provides value to industry practice, thus promoting a deeper and more widespread understanding of Layered Enterprise Architecture Development (LEAD) and the LEAD Enterprise Ontology.

References

1. Andrews, S., Polovina, S.: Exploring, reasoning with and validating directed graphs by applying formal concept analysis to conceptual graphs. In: Croitoru, M., Marquis, P., Rudolph, S., Stapleton, G. (eds.) GKR 2017. LNCS (LNAI), vol. 10775, pp. 3–28. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78102-0_1
2. Butterfield, A., Ngondi, G.E., Kerr, A. (eds.): A Dictionary of Computer Science, 7th edn. Oxford Quick Reference. Oxford University Press, Oxford, England (2016)
3. Formica, A.: Ontology-based concept similarity in formal concept analysis. Inform. Sci. **176**(18), 2624–2641 (2006)
4. Guarino, N., Welty, C.A.: An Overview of OntoClean, pp. 151–171. Springer, Berlin Heidelberg, Berlin, Heidelberg (2004)
5. Kowalski, R.: Algorithm = logic + control. Commun. ACM **22**(7), 424–436 (1979)
6. Polovina, S., Scheruhn, H., Weidner, S., Von Rosing, M.: Highlighting the gaps in enterprise systems models by interoperating CGS and FCA. In: Andrews, S., Polovina, S., (eds.), 22nd International Conference on Conceptual Structures (ICCS 2016), 5th-7th July, pp. 46–54. Tilburg University, 12 Jul 2016
7. Polovina, S., von Rosing, M., Etzel, G.: Leading the practice in layered enterprise architecture. CEUR Workshop Proc. **2574**, 62–69 (2020)
8. Polovina, S., von Rosing, M., Laurier, W.: Conceptual structures in LEADing and best enterprise practices. In: Hernandez, N., Jäschke, R., Croitoru, M. (eds.) ICCS 2014. LNCS (LNAI), vol. 8577, pp. 293–298. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08389-6_25
9. Polovina, S., von Rosing, M., Laurier, W., Etzel, G.: Enhancing layered enterprise architecture development through conceptual structures. In: Endres, D., Alam, M., Şotropa, D. (eds.) ICCS 2019. LNCS (LNAI), vol. 11530, pp. 146–159. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23182-8_11

10. Sawsaa, A., Lu, J.: Building information science ontology (OIS) with methontology and protégé. J. Internet Technol. Secur. Trans. (JITST) **1**(3/4) (2012)
11. Sowa, J.F.: Conceptual graphs for a data base interface. IBM J. Res. Develop. **20**(4), 336–357 (1976)
12. Sowa, J.F.: Conceptual graphs. Found. Artif. Intell. **3**, 213–237 (2008)
13. von Rosing, M., Laurier, W.: An introduction to the business ontology. Int. J. Concept. Struct. Smart Appl. (IJCSSA) **3**(1), 20–41 (2015)
14. von Rosing, M., von Scheel, H.: Using the business ontology to develop enterprise standards. Int. J. Concept. Struct. Smart Appl. (IJCSSA) **4**(1), 48–70 (2016)
15. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts Ordered Sets. In: Proceedings of the NATO Advanced Study Institute held at Banff, Vol. 83, Canada, August 28 to September 12, 1981, pp. 445–470 (1982)