Using conceptual structures in enterprise architecture to develop a new way of thinking and working for organisations

POLOVINA, Simon <http://orcid.org/0000-0003-2961-6207> and VON ROSING, Mark

Using Conceptual Structures in Enterprise Architecture to develop a new Way of Thinking and Working for Organisations

Simon Polovina[1,2] and Mark von Rosing[2]

[1] Conceptual Structures Research Group, Communication and Computing Research Centre & Department of Computing, Sheffield Hallam University, Sheffield, UK
[2] Global University Alliance, Chateau Du Grand Perray, La Bruere Sur Loir, France
S.Polovina@shu.ac.uk, mvr@globaluniversityalliance.org

Abstract. Enterprise Architecture (EA) is a discipline that provides generic patterns that any organisation can reuse throughout its own business, informatics and technical components. However, EA's current way of thinking and working to achieve this aim is not standardised. EA thus continues to "reinvent the wheel" that causes mistakes or wastes resources on rediscovering what should already be known. We, therefore, represent the specific business, information and technology meta-models as patterns that can be fully reintegrated in one repeatable meta-model for the whole organisation. The outcome is a new agile way of thinking and working, highlighted by how EA works better in enterprise layers, sub-layers and levels of abstraction. To test the meta-models, two forms of Conceptual Structures known as Conceptual Graphs (CGs) and Formal Concept Analysis (FCA) are brought together through the *CGtoFCA* algorithm. The algorithm identifies how the layered meta-models can share meaning and truth and without having to recombine them into one large, unwieldy meta-model as the repeatable structure.

1 Introduction

Organisations can draw upon leading and best practices to gain insight into how best to fulfil their value and purpose. This insight ranges from understanding the external forces (e.g. the marketplace or non-profit environment) and internal forces (e.g. career ambitions of their employees) from which the organisations derive their strategy. The insight ranges to the operational behaviour (e.g. business processes), computer-based applications and data needed to implement that strategy most effectively.

Enterprise Architecture (EA) is a discipline that provides generic patterns that any organisation can reuse throughout its own business, informatics and data models in fulfilment of that organisation's overall purpose. The organisation thus avoids "reinventing the wheel" which causes it to make mistakes or waste resources on rediscovering what is already known.

To reduce misinterpretation, the patterns are intended as formal models of the models—i.e. meta-models—that each business can specialise according to

their specific needs. Computer science and informatics contributes to the expressibility in these meta-models through its advances in ontology and semantics; together they capture the objects and relations that describe the interplay and effects of business in a formal, computable model [2, 3].

There are however multiple EA frameworks and methods, each with their own meta-models and associated approaches revealing a lack of mutual understanding what the meta-models should consist of and how they ought to be used. The content within and interconnections between the meta-models for the 'architectural domains'—i.e. business, information and technology—that make up the organisation are also interpreted differently according to the EA framework. The inconsistencies in the meta-models, and how to think and work with them undermine our conceptual understanding of organisations with potentially damaging effect. Consequently, organisations still end up reinventing the wheel.

The classical way of thinking and working in EA's architectural domains with a linear waterfall approach is counterproductive to the aims of EA. We evidence that representing the architectural domains as 'layers' enables us to think and work simultaneously within and across these multiple domains. We test this approach through Conceptual Structures, namely Conceptual Graphs and Formal Concept Analysis. As well as offering an agile way of thinking and working with EA, organisations can thus better draw upon the suggested best and leading practices to gain insight into how best to fulfil their value and purpose.

2 Understanding Architectural Layers in Organisations

Independent of their size or industry, organisations share a common underlying structure that consists of the following enterprise layers identified from previous work [8]:

- Business: Such as the purpose and goal, competencies, processes, and services aspects;
- Information: Such as the application systems, as well as the data components;
- Technology: Such as the platform and infrastructure components.

These layers represent the three perspectives by which organisations are viewed. They are called layers because the business layer sits on top of the information layer that in turn sits on top of the technology layer. It epitomises that organisations are driven by their business needs that are enabled by their information systems (applications and data), which require the underlying technology to run these systems.

The organisation thus has to align its way of thinking with its way of working within and across all these perspectives. The Global University Alliance (GUA, www.globaluniversityalliance.org) is a non-profit body run and supported by academics who have researched and developed these layers, as further detailed in Figure 1. This Figure illustrates that the three layers are decomposed into eight sub-layers.

The layers and sub-layers are an abstraction that represents and considers the enterprise as a whole [8]. For example, a policy, act, regulation or even a strategy is a part of the business layer, while the application systems and data aspects are a part of the information layer. It also highlights that organisational requirements cut across all the layers, and organisational transformation and innovation draws on the layers too. The Layered Enterprise Architecture Development (LEAD) that Figure 1 depicts has been embodied by the industry practitioners' enterprise standards body LEADing Practice (www.leadingpractice.com) [7].

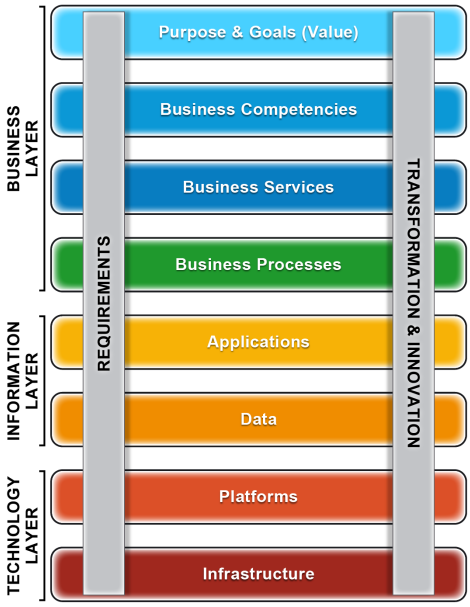

Fig. 1. The Layers, and Sub-Layers

Figure 1 is further dimensioned by Figure 2, which explicates how the architectural domains (the layers through their sub-layers) are further decomposed into architectural views—i.e. contextual, conceptual, logical and physical. These views are called 'levels' and described shortly. The figure is now a matrix structure, where the layers and sub-layers are the rows, and the levels are the columns.

2.1 An Illustration

To illustrate Figure 2, Table 1 populates the layer and level structure[3] with meta-model entities. These entities are accordingly referred to as 'meta-objects' which, for simplicity, we shall call objects.

The illustration (which supersedes previous work [4]) will be used to show how we can traverse through the layers and sub-layers, thinking and working simultaneously within and between domains through the decomposition and composition of the objects on all the levels. In so doing, we effortlessly integrate the right concept—i.e. object—across the different sub-layers when interlinking the EA for an organisation [8].

Returning to the table, 'Application Function' is an object under Application in the Information layer at level 2. Likewise, 'Objective' is a business layer Value object at level 3. The table lacks the technology layer but demonstrates the principle of layering, at least for the business and information layers. Also in line with Figure 2, Table 1's level 1 is the contextual view, level 2 is the conceptual

[3] The table has the layers in columns not rows, and the levels in rows not columns. This layout allows the table to best fit on the page; it should be the other way round.

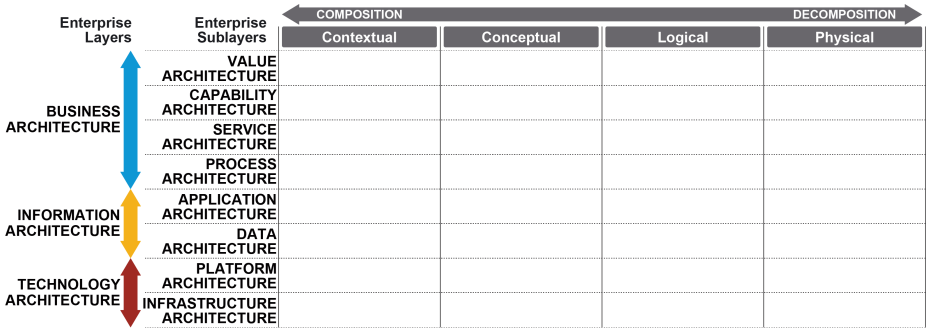

Fig. 2. Layers with Levels (Contextual, Conceptual, Logical and Physical)

view, level 3 is the logical view, and level 4 is the physical view in architectural view terms.

Many of the objects can exist in the business, information and technology layers. These objects hence can both be repeated or related at more than one level (e.g. vision, mission, strategy, goal, business function, and business service), but are scoped according to their level of abstraction. For example, the strategy object at level 3 reflects an implementation of the strategy set by the highest-level and most abstract contextual depiction of the strategy object at level 1 that in turn is mapped to level 3 through the intermediate conceptual strategy object at level 2. Level 4 is the physical form of the three levels above it. The table shows, for example, a performance indicator so that strategy and other value elements can be measured thereby to determine their effectiveness. Likewise, the most physical form of data sub-layer appears at level 4 (e.g. data table, key/foreign-key/attributes). The other level 4 objects for these and the other sub-layers (competency, service, process and application) can be viewed from the table. A more detailed discussion beyond illustrating the principle as we have described can be found elsewhere [7]. We will, however, explore the objects and their interrelationships as illustrated by Table 1 through Conceptual Structures.

3 Conceptual Structures

Conceptual Graphs (CGs) are a system of logic that express meaning in a form that is logically precise, humanly readable, and computationally tractable. CGs are a conceptual structure that serve as an intermediate language for translating between computer-oriented formalisms and natural languages. CGs graphical representation serve as a readable, but formal design and specification language [6]. CGs can thus powerfully represent the formal structure of meta-models while allowing them to be human-readable. A CG (Conceptual Graph) was therefore produced for each sub-layer in Table 1.

Although CGs provide a logical level of rigour, their constituent concepts and relations are essentially put together by hand according to the human's subjec-

Table 1. The meta-model matrix with meta-objects (objects)

Layer & Sub-layer:		Business			Information	
	Value	Competency	Service	Process	Application	Data
Level						
1	Vision, Mission	Business Function	Business Service	Business Process	Application Module	Enterprise Data Cluster
	Strategy, Goal	Organizational unit			Organizational unit	
2	Vision, Mission	Business Function	Business Service	Process Step	Application Function	Department Data Cluster
	Strategy, Goal	Organizational unit			Organizational unit	
3	Vision, Mission	Business Function	Business Service	Process Activity	Application Task	Workplace Data Entity
	Strategy, Goal				Transaction Code, System organizational Unit	
	Objective	Business Object		Event	Business Object	Dimension
					Data Entity Event	Data Entity
		Business Media / Accounts		Data Object	Data Object (Media)	
		Business Roles	Services Roles	Process Role	Application Roles	
		Business Roles	Service Rules	Process Rules	Application Rules	
4	Performance Indicator	Business Compliance	Service Level Agreement (SLA)	Process Performance Indicator (PPI)	IT Governance	Fact Table Customizing Data Table
						Master Data Table / View
						Transaction Data Table
		Revenue/ CostFlow			System Measurements	Key Foreign Key Describing Attributes

tive interpretation of the real-world phenomena for it to be captured in a logical structure. This procedure is akin to how the meta-models are produced in practice, using for example Class Diagrams in UML, which CGs can help model [9]. A second form of conceptual structure known as Formal Concept Analysis (FCA) which is used in information science [5]. FCA provides an objective mathematical interpretation of CGs' logical but subjective human interpretations and is brought to bear through the *CGtoFCA* algorithm [1]. A Formal Concept in FCA is the result of when certain conditions are met in a formal *context*:

- A formal context is a triple $\mathbb{K} = (G, M, I)$, where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary (true/false) relation that expresses which objects have which attributes.
- (A, B) is a formal concept precisely when:
  - every object in A has every attribute in B,
  - for every object in G that is not in A, there is some attribute in B that the object does not have,
  - for every attribute in M that is not in B, there is some object in A that does not have that attribute.

The Formal Concepts can then be presented in a lattice, namely a Formal Concept Lattice (FCL), as will be demonstrated shortly.

3.1 The Business Layer

The top layer, the Business Layer, establishes the connections of the enterprise to the environment through the identification of objects that describe the purpose and goal and therefore points both to the source of value and to concerns about the trade-offs necessary to optimise the ability to pursue this value. It further identifies the competencies needed to execute the functions, processes, and services within the environment. These are then used, in conjunction with business functions and other primitives, to organise and aid in the decomposition and organisation of the logical view and physical implementation of the business services and processes. In the following, we will elaborate on the individual sub-layers of the business layer.

Value The Value architecture sub-layer captures ideas about the vision, mission, strategy policy, act and regulations as well as all the purpose, goal and value that the organisation seeks to create.

The CG (Conceptual Graph) for the Value sub-layer is shown in Figure 3. It reveals how CGs follow an elementary concept → relation → concept structure that describes the ontology and semantics of the meta-model as explained earlier. The Value CG shows each object name (i.e. Vision, Mission, Strategy, Goal) as a CG type label. To instantiate it as a particular object, a unique identifier appears in the referent field, hence making use of CGs `[Type-Label: Referent]` structure. For example, `v1V` denotes that a object that is Vision (v), Level 1 (1), and V (Value sub-layer). Likewise, `g3V` for example describes Goal,

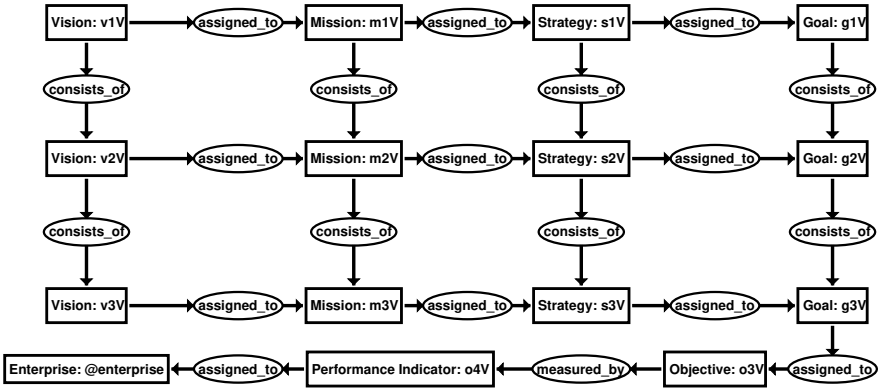

Fig. 3. Value, CGs

Level 3, Value and so on. The `[Enterprise: @enterprise]` concept follows an alternative pattern where `@enterprise` is a CGs' measure referent [6].

The key significance of the `[Enterprise: @enterprise]` concept is that all the activities that make up an enterprise ultimately point to the enterprise, even though Enterprise is absent in Table 1. This follows EA's holistic perspective. To draw from a building architect's analogy, architecture ranges "From the blank piece of paper to the last nail in the wall." EA follows the same principle, bringing all the objects at all the levels within a sub-layer to the same single point i.e. `[Enterprise: @enterprise]` being the organisation (that in EA terms is the enterprise, which accounts for all kinds of organisations not just profit-making enterprises).

The relations (e.g. `(assigned_to)`) describe the interrelationships between the objects in the table. Essentially the `(assigned_to)` relation refers to a horizontal relation usually in the same sub-layer while `(consists_of)` is a vertical relation between the levels in the sub-layers. (There is no associated layer, sub-layer or level for Enterprise as it reflects the above-described culmination of all the sub-layers, and—as we shall see—all the levels). The relation `(measured-by)` has its usual meaning.

Figure 4 shows the FCL (Formal Concept Lattice) for the Value sub-layer. It is the result of the *CGtoFCA* algorithm transforming the object→ relation → object triples in the CG of Figure 3 to object⌢relation→ object binaries[4]. An example binary is `Vision:v1V⌢assigned_to→Mission: m1V`. The neatly displayed lattice shows that `[Enterprise: @enterprise]` is bottommost i.e. at the *infimum* of the FCL, and highlighted by the bold rectangle in Figure 4. The topmost formal concept in a FCL is the *supremum*. In this case the supre-

[4] The CGs are drawn in CharGer (http://charger.sourceforge.net/) as it has support for the CGIF (CG Interchange Format) in ISO/IEC 24707 Common Logic. The CGIF is passed through *CGtoFCA* then visualised as an FCL in FCA Concept Explorer (http://conexp.sourceforge.net/).

mum is represented by `[Vision: v1V]`, which traverses downwards through the lines (pathways) connecting the intermediate concepts in the Value FCL case culminating in `[Enterprise: @enterprise]`.

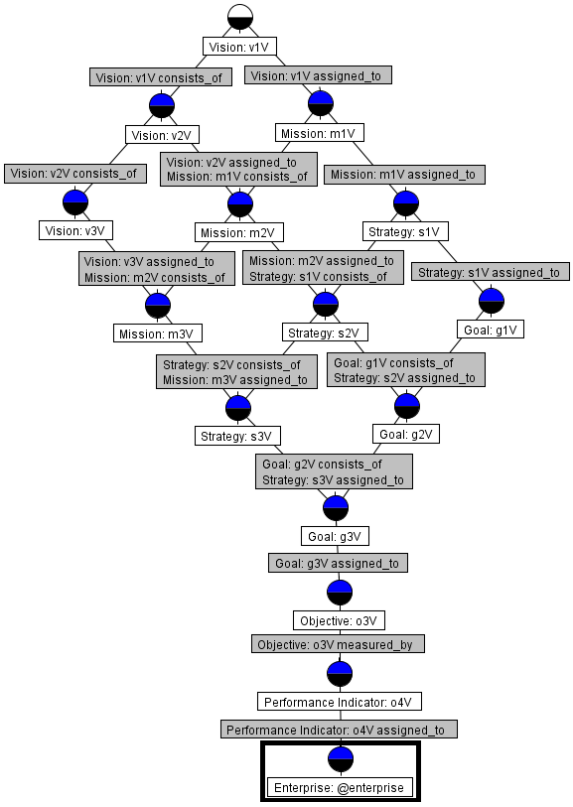

Fig. 4. Value, FCL

The above downward traversal is denoted as that formal concept's *extent*, meaning all the FCA objects below it. As `[Enterprise: @enterprise]` is at the infimum that means *all* the lattice's objects are in `[Vision: v1V]`'s extent. This is because in *CGtoFCA* a CG concept becomes an FCA object. Given each CG concept represents a meta-model object in our case, that object in effect becomes an FCA object. Hence why in the Value FCL all the objects are in the extent of `[Vision: v1V]`.

The object⌒relation (e.g. `Vision: v1V assigned_to`) part of the binary in *CGtoFCA* likewise becomes an FCA attribute. The upward (as opposed to downward) traversal is denoted as that formal concept's *intent*, meaning all the FCA attributes (as opposed to objects) that are through the lines (pathways) above it (as opposed to below it). Hence in our case all the object⌒relation attributes going upwards from a formal concept are in that formal concept's intent. If the formal concept is the infimum then that would mean *every* attribute in the FCL. As the infimum in the Value FCL is populated by the `[Enterprise: @enterprise]` FCA object then all the attributes are in its intent including `Vision: v1V assigned_to`.

Competency Figure 5 shows the CG for the Competency architecture sub-layer. Some of the CG concepts in Figure 5 are shaded to highlight where they appear in the other sub-layers shown by the sub-layer and level Figures 1 and 2 shown earlier. The referent can be inspected to reveal the other sub-layer

(e.g. The '`..V`' in the referent for `Goal` shows it is in the Business Layer, under the Value sub-layer; the '`..A`' in the referent for `Business Object` shows it is in the Information Layer, under the Application sub-layer). There is an `(occurrence_copy)` relation too. Essentially, this relation describes two concepts that are similar but are not co-referent (i.e. do not have the same referent), which would make them the same. For example, `[Business Function: bf1C]` → `(occurrence_copy)` → `[Business Service: bs1S]`. The rationale for such a relationship is further detailed elsewhere [4].

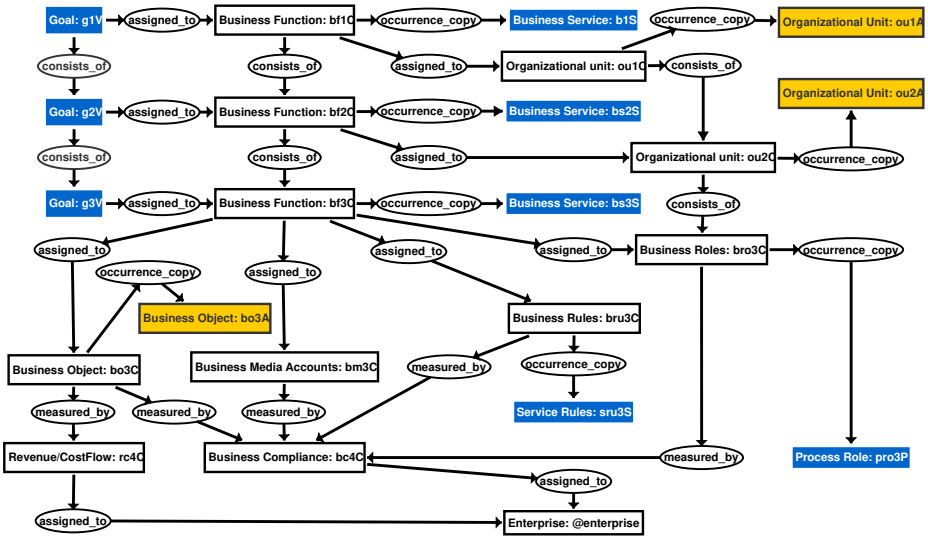

Fig. 5. Competency, CGs

Again the same mapping through *CGtoFCA* is applied and Figure 6 shows the resulting FCL. This time `[Enterprise: @enterprise]` is not bottommost, highlighted by the bold rectangles in Figure 6. The outcome is due to the concepts in the CG, such as `[Business Service: b1S]`, `[Service Rules: sru3S]`, `[Process Role: pro3P]` that have their identical concept in another business sub-layer (e.g. S for Service, P for Process). These concepts do not have `[Enterprise: @enterprise]` in their extent for the Competency meta-model. Likewise `[Business Object: bo3A]`, and `[Organizational Unit: ou1A]` in A the Application sub-layer do not end up at the CG concept `[Enterprise: @enterprise]` unlike the Value CG Figure 5 above. This again is because the `[Enterprise: @enterprise]` is not in their extent.

The Competency meta-model henceforth has dependencies with the other sub-layers that will only be resolved when the CGs in this sub-layer are joined with the identical, corresponding CG concepts in those other sub-layers. This is possible through the CGs join operation, where the CG concepts have the same

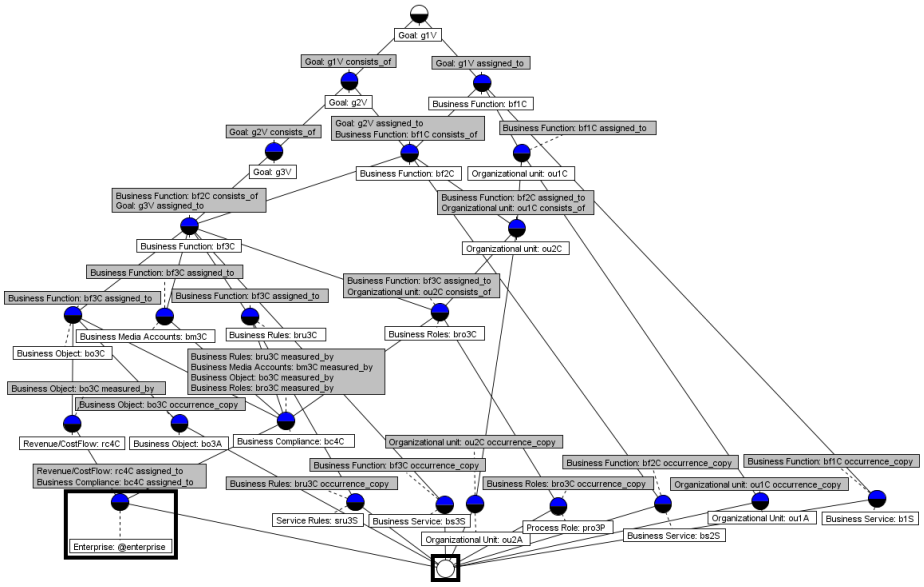

Fig. 6. Competency, FCL

referent i.e. are co-referent [6]. For example the Value sub-layer concept `[Goal: g1V]` can join with its counterpart in Competency as they share '`g1V`'. This operation applies to all matching referents (co-referents) across all the sub-layers. If, when all the sub-layers are thus joined, all the paths lead to `[Enterprise: @enterprise]` then, together, the (Enterprise) Architectural principle of arriving at that 'last nail in wall—i.e. `[Enterprise: @enterprise]`—is achieved.

While a simple inspection of the CG for this sub-layer without the FCL reveals the incomplete arrival to `[Enterprise: @enterprise]`, the FCL—which is computer generated rather than hand-drawn—horizontally lays out the objects according to their levels. Where they do not eventually point to `[Enterprise: @enterprise]`, the levels look more uneven. Compare the horizontal layout of the levels in Figure 6 with Figure 4 for example. As well as highlighting that the levels look uneven, the concepts in Figure 5 that do not eventually point to `[Enterprise: @enterprise]` are further highlighted by being shown at the same bottom level part of the lattice as `[Enterprise: @enterprise]`. Manual inspection of the CG makes identifying these much harder, especially for the much more comprehensive meta-models encountered in practice than the simple illustration given in Table 1. And we haven't accounted for discovering and rectifying errors in drawing the CGs, such as the arrows in the CGs pointing in the wrong direction or mistyped concepts as simple examples from other work have demonstrated [1, 4].

3.2 The Information Systems Layer

The Information Architecture Layer describes the objects, semantic relations and deliverables within the Application and Data sub-layers, and are the main components for both Application Architecture, Data Architecture and Information Architecture. The maps, matrices and models used within the Application and Data sub-layers illustrate how their objects such as data goals, data flows, data services, data requirements and data components are linked to application goals, information flows, information services, application requirements, application flows and applications components.

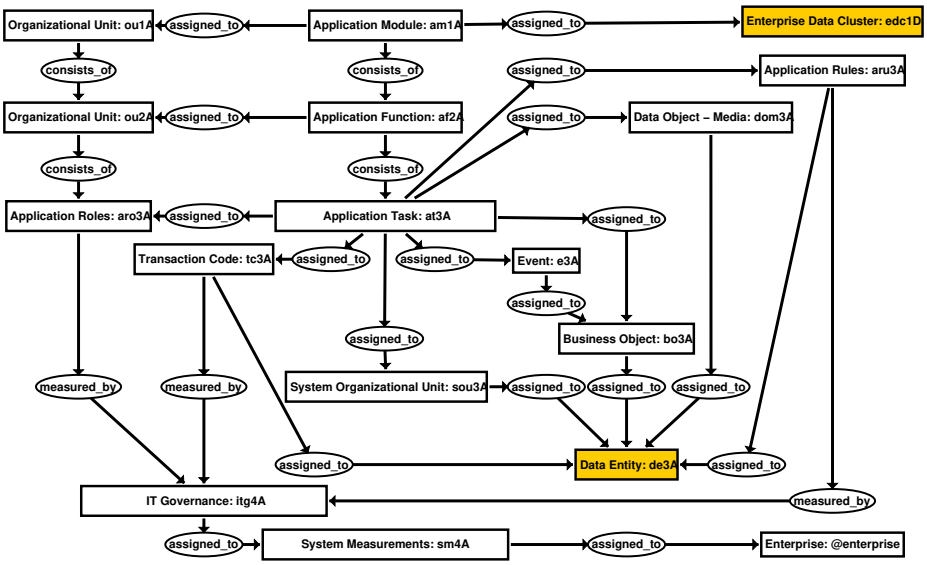

Fig. 7. Application, CGs

Application Due to space constraints the CG and FCL for the **Service** and **Process** architecture sub-layers are not shown. They have appeared in earlier work [4]. Figure 7 depicts the Application architecture sub-layer CG. Figure 8 evidences that `[Enterprise: @enterprise]` is not bottommost i.e. not at the infimum. That is because of the object⌢relation attributes that are outside the intent of the level 4 key performance indicator (KPI) object `[System Measurements: sm4A]`, which evaluates the Application sub-layer. Also emerging in the middle of the lattice is another formal concept without its own object. This formal concept appears as a while circle, and is highlighted in Figure 8 by a bold rectangle. So far such a formal concept has only occurred at the infimum when `[Enterprise: @enterprise]` is not bottommost, which also happens to

be the case in Figure 8. We can follow the intent and extent from and to the highlighted formal concept to get a sense of what name we might give this object, or confirm that it's simply warranted thus doesn't need its own object. Pertinent to us however is what appears at the infimum.

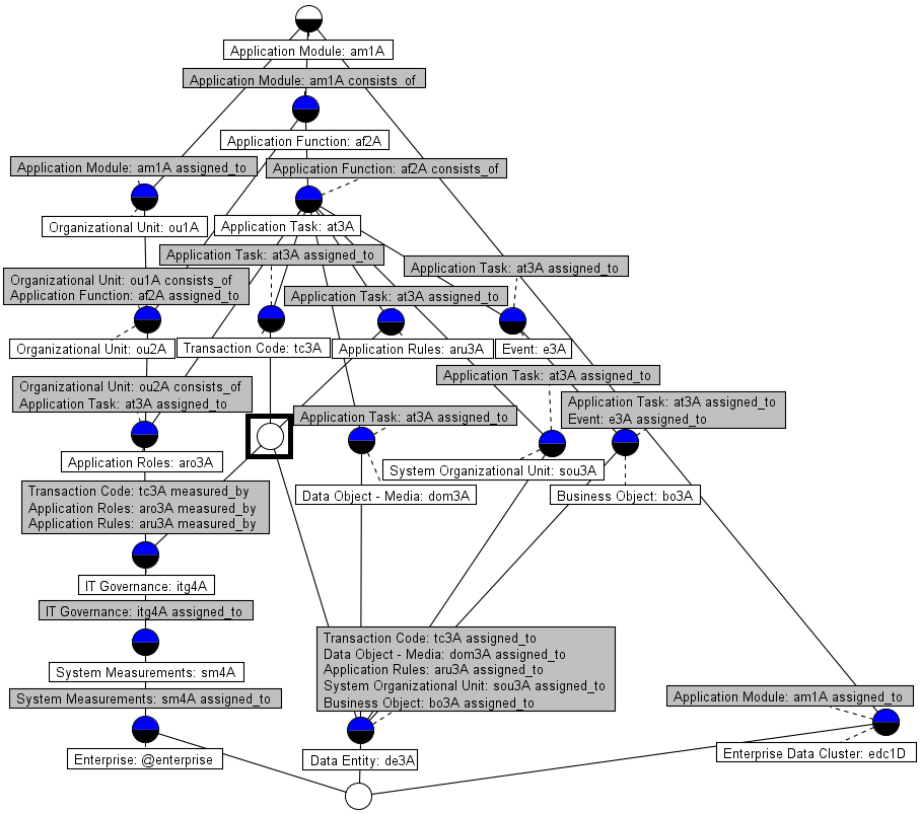

Fig. 8. Application, FCL

Data Figure 9 depicts the Data architecture sub-layer CG. Figure 10's FCL evidences that `[Enterprise: @enterprise]` is bottommost i.e. at the infimum. Like Value, the extent of all the attributes from the topmost formal concept i.e. the supremum is `[Enterprise: @enterprise]` including from all the relevant KPIs (level 4 objects) including `[System Measurements: sm4A]`. In this sub-layer all its concepts (objects) extend to the enterprise, remembering that it includes objects from other sub-layers (i.e. process, service and application) as highlighted by the shaded CG concepts in Figure 9. Working back, the dependencies started at the Value architecture sub-layer, and `[Vision: v1V]` in

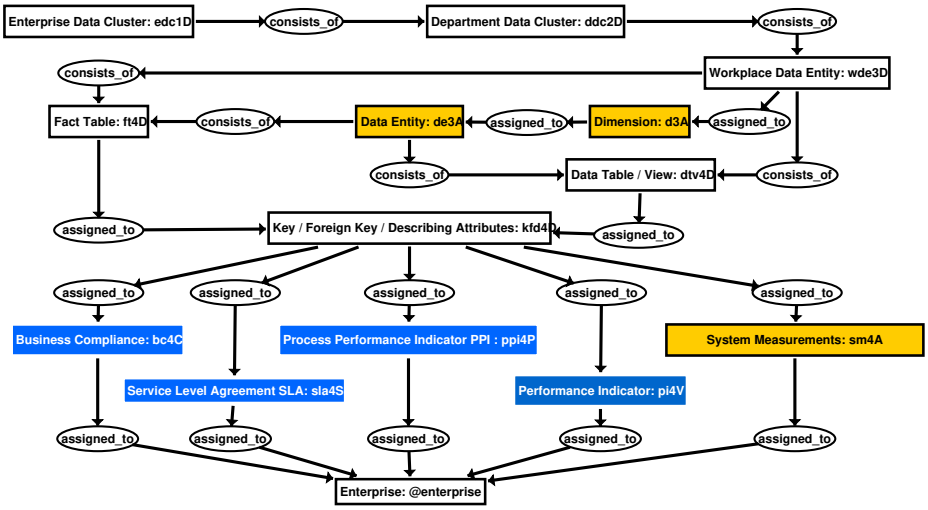

Fig. 9. Data, CGs

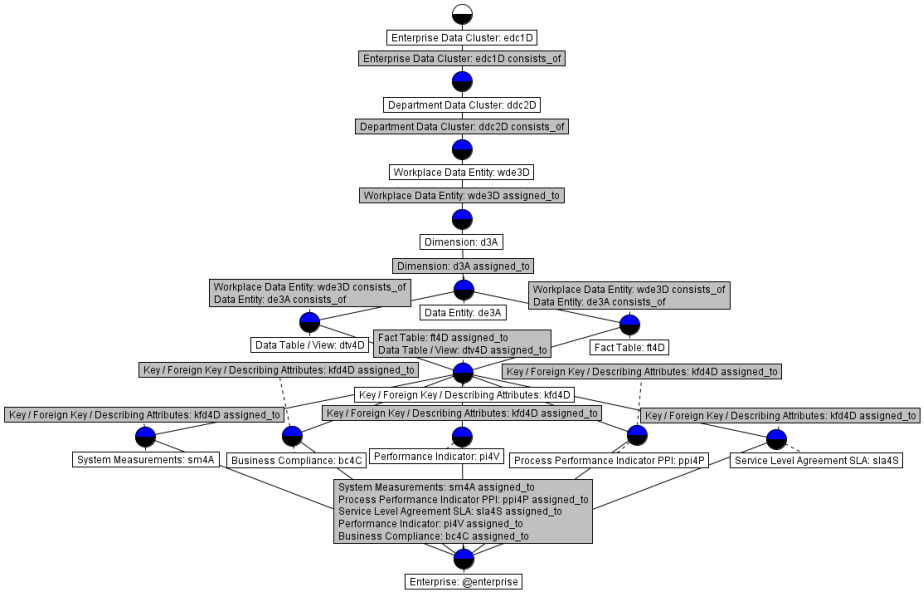

Fig. 10. Data, FCL

particular. The meta-models for each sub-layer make up the parts of the whole meta-model for the organisation.

4 The Whole Meta-model

Figure 11 shows the FCL when each CG meta-model for its architecture sub-layer (Value, Competency, Service, Process, Application and Data) are all combined through the co-referent links. Though not shown, it can be appreciated that even for our elementary illustration that would be one, huge, unwieldy CG.

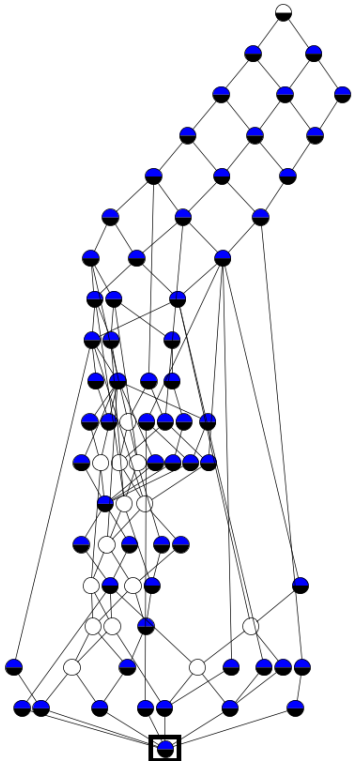

Fig. 11. Combined FCL

Indeed the FCL Figure 11 that is generated from this CG looks complex, and the names of the FCA objects and attributes are omitted to avoid cluttering the lattice. (Also though not remarked on here, a number of formal concepts that appear in the middle do not have their own objects, denoted by the clear circles like the circle that appeared for Application.) What is evident nonetheless is that its infimum is a solid circle, highlighted by the bold rectangle in Figure 11. That is our single point of interest, thus obviating the need to visualise Figure 11 at all. If its infimum object was shown it would be `[Enterprise: @enterprise]`. The layered meta-models thereby demonstrate that the organisation's way of thinking and its way of working across all its layers through the sub-layers are aligned.

Even though the meta-model for each sub-layer apart from Value and Data did not have `[Enterprise: @enterprise]` as its infimum, when combined into Figure 11 `[Enterprise: @enterprise]` became the infimum. It reminds the organisation that it parts (e.g. departments, business, informatics and technical experts) depend on each other, and through the formal concepts explicates through which shared objects that they have to align and communicate through. A good meta-model, as our elementary illustration shows must, in the combined meta-model have an supremum that is the architectural 'blank piece of paper' and an infimum that represents the architectural 'last nail in the wall'. In our illustration that was `[Vision: v1V]` and `[Enterprise: @enterprise]` respectively.

5 Conclusions

In EA, thinking and working in architecture domains alone is counterproductive. Through reconsidering the domains as architecture layers and sub-layers, and using levels of contextual, conceptual, logical and physical abstraction within these sub-layers, we have been able to open up the objects' multiple interaction points within and across the various layers, sub-layers and levels that better architect the organisation. From this agile approach, we can think and work with an EA in which the meta-models can repeatedly point to a single truth as opposed to the divergent meta-models that have characterised EA frameworks. Thereby, the organisation is less likely to reinvent the wheel needlessly. We have portrayed how the layered EA can be enhanced by Formal Concepts. The use of CGs (Conceptual Graphs) and FCA (Formal Concept Analysis) through the *CGtoFCA* algorithm provided a formal underpinning to the meta-models, pinpointing the direction of the interdependencies throughout the architecture layers, sub-layers and levels. Through the co-referent links it revealed how meta-models could be aligned towards that single truth and, along the way, without having to generate one large, unwieldy meta-model.

References

1. Simon Andrews and Simon Polovina. *Exploring, Reasoning with and Validating Directed Graphs by Applying Formal Concept Analysis to Conceptual Graphs*, volume LNAI 10775 of *GKR 2017, Revised Selected Papers*, pages 3–28. Springer, 2018. http://www.springer.com/gb/book/9783319781013.
2. The Open Group. 30. content metamodel, 2018. http://pubs.opengroup.org/architecture/togaf92-doc/arch/chap30.html.
3. Daniel Oberle. How ontologies benefit enterprise applications. *Semantic Web Journal*, 5(6):473–491, 2014.
4. Simon Polovina, Hans-Jurgen Scheruhn, and von Rosing Mark. *Modularising the Complex Meta-Models in Enterprise Systems Using Conceptual Structures*, pages 261–283. Developments and Trends in Intelligent Technologies and Smart Systems. IGI Global, Hershey, PA, USA, 2018. ID: 189437.
5. Uta Priss. Formal concept analysis in information science. *Annual Rev. Info. Sci & Technol.*, 40(1):521–543, December 2006.
6. John F. Sowa. *Conceptual Graphs*, pages 213–237. Handbook of Knowledge Representation, Foundations of Artificial Intelligence. Elsevier, Amsterdam, volume 3 edition, 2008.
7. Mark von Rosing. Using the business ontology to develop enterprise standards. *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*, 4(1):48–70, 2016. ID: 171391.
8. Mark von Rosing, Bonnie Urquhart, and John A. Zachman. Using a business ontology for structuring artefacts: example - Northern Health. *International Journal of Conceptual Structures and Smart Applications (IJCSSA)*, 3(1):42–85, 2015. ID: 142900.
9. B. Wei, H. S. Delugach, E. Colmenares, and C. Stringfellow. A conceptual graphs framework for teaching UML model-based requirements acquisition. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 71–75, April 2016.