Re-engineering Data with 4D Ontologies and Graph Databases

Sergio de Cesare[1], George Foy[1], and Chris Partridge[1,2]

[1] Department of Information Systems and Computing,
Brunel University, Uxbridge, UB8 3PH, U.K.
{firstname.lastname}@brunel.ac.uk
[2] BORO Solutions Ltd. London, U.K.
partridgec@borogroup.co.uk

Abstract. The amount of data that is being made available on the Web is increasing. This provides business organisations with the opportunity to acquire large datasets in order to offer novel information services or to better market existing products and services. Much of this data is now publicly available (e.g., thanks to initiatives such as Open Government Data). The challenge from a corporate perspective is to make sense of the third party data and transform it so that it can more easily integrate with their existing corporate data or with datasets with a different provenance. This paper presents research-in-progress aimed at semantically transforming raw data on U.K. registered companies. The approach adopted is based on BORO (a 4D foundational ontology and re-engineering method) and the target technological platform is Neo4J (a graph database). The primary challenges encountered are (1) re-engineering the raw data into a 4D ontology and (2) representing the 4D ontology into a graph database. The paper will discuss such challenges and explain the transformation process that is currently being adopted.

Keywords: 4D ontology, perdurantism, foundational ontology, semantic transformation, graph databases, Neo4J, Big Data.

1 Introduction

The amount of data that is currently made available on the Web is growing thanks primarily to the Linked Open Data (LOD) initiative. While LOD data tends to be in formats such as the Resource Description Framework (RDF) and Microformats, there is also an enormous amount of data made available in other structured formats and more often even semi-structured or unstructured. This is the case of data that is privately sold (e.g., by credit risk companies) or made publicly available by Government Agencies (e.g., open government data initiatives) [1].

In order to more effectively process and integrate data from a multitude of sources as well as make it semantically consistent with the existing enterprise data architecture, we have chosen to adopt ontologies. More specifically we adopt a 4D foundational ontology [2, 3] to drive the interpretation of such data sources, improve the semantic expressiveness of the data and harmonise it in a consistent manner. The approach that we are adopting, therefore, begins with the raw data, ontologically interprets and transforms the data in order to extract its semantics and express such semantics in 4D ontologies. These 4D ontologies are then mapped to a graph-based data architecture. Neo4J [4] is the implementation technology that we have currently chosen to adopt. The main reasons for adopting a graph database to persist the ontological models are: (1) the flexibility that a graph structure provides in implementing any modelling paradigm and (2) the scalability it provides in terms of organising and accessing massive amounts of data.

The paper is organised as follows. Section 2 defines the problem in more detail. It also explains the reasons for underpinning the data re-engineering with a foundational ontology and why a graph database was chosen as the implementation technology. Section 3 provides an overview of the 4D foundational ontology adopted to underpin the re-engineering effort. Section 4 explains the challenges of mapping the foundational ontology to a graph and Section 5 presents a few mapping patterns discovered to date. Section 6 describes related work and Section 7 concludes the paper and discusses future work.

2 The Research Problem

This research investigates the problem of integrating large datasets from different sources into one common data repository or into an existing corporate database. While this research focuses on large datasets acquired externally, it must be noted that the approach described in this paper can also be applied by organisations to examine their own vast transactional datasets from which to glean potential competitive information.

The datasets that are being referred to here are, for example, those made available by authorities such as Companies House (the U.K. Company Registration Office). These datasets come in a variety of formats. For example, datasets available at data.gov.uk are currently provided as CSV (Comma-Separated Values) or JSON (JavaScript Object Notation) files. In essence the underlying original structure is that of a spreadsheet. Normally such files and their corresponding JSON representations are direct format conversions from legacy spreadsheets or flat files. This is normally apparent from the denormalised form that such data assumes. While syntactically structured (in terms of rows and columns), much of the semantics of these datasets is implicit and cannot be readily integrated with other datasets. The integration of multiple datasets is not a mere technical problem, but it also represents a business opportunity for organisations to exploit in this new era of the Digital Economy [5]. As stated by the U.K. Cabinet Office, the aim is to create “an information marketplace for entrepreneurs and businesses; releasing valuable raw data from real-time transport information to weather data” [6].

The overall research problem also has another aspect to it, which is performance. In fact, since the amount of data processed can easily be in a range that runs from hundreds of gigabytes to tera/petabytes, there is also a technical challenge of processing so called Big Data [7]. This paper however will only focus on the problem of semantic transformation, which represents the part of the research carried out to date. Future work, as documented in Section 6, will explore the other aspects of the research.

Re-engineering data can be viewed as essentially a problem of semantic interpretation, in other words a process of interpreting the raw data and identifying the things that the data refers to in the real world (or any possible world). This realist approach is greatly simplified by the adoption of a foundational ontology to drive the re-engineering. A foundational (or upper-level) ontology defines the kinds of existence that things can have (i.e., a categorical theory). Categorical theories are studied in Philosophy.

In Philosophy, Ontology, as a discipline, is the study of existence and of the kinds of things that (can) exist. One aspect of existence is change over time, and in this area there are two predominant ontological theories: endurantism and perdurantism [8]. In endurantism a three-dimensional object is wholly present at any given instant and persists by 'sweeping' through a region of space-time (in the words of Sider [8]). Another aspect of ontology is identity; and a key question is whether there is any criterion of identity and what it is. One endurantist approach is to say that while wholly present at all moments of its existence, an object preserves its identity via a set of essential attributes (for example, a person's DNA). The perdurantist approach sees an object as a four-dimensional extension (or extent) in the universe (i.e., occupying a region of space-time) and it is not therefore totally present at any given instant, but instead only partially present. A common perdurantist criterion of identity is the object's four-dimensional extension. In its lifetime an object goes through states (or stages). For example, a person goes through the stages of childhood and adulthood. In perdurantism change is explained via successive temporal parts. Therefore, while an endurantist object persists in three-dimensional space and entirely shifts from one point in time to the next, in perdurantism an object exists in four-dimensional space-time and can be regarded as partially present at any time or portion of its spatiotemporal extension.

In this research we adopt BORO, a 4D foundational ontology, described in the following section. The adoption of a perdurantist ontology is motivated by it being particularly suitable to model the enterprise context and its continuously changing nature. Perdurantism models change by representing stages of a particular object as temporal parts (examples include changes of address, changes of legal status and changes of a company's primary type of activity). Perdurantism and extensionalism naturally allow to model particular objects with intersecting spatiotemporal extents, for example, between a person (*Bill* Gates) and a company position (*CEO of Microsoft*). These aspects of 4D ontologies (along with others) provide more explicit and accurate representations of change in terms of a succession of different temporal parts. Greater accuracy in the models produced can lead to greater levels of flexibility and reusability when evolving information systems (IS) as more thoroughly explained by Partridge [2].

In order for ontologies to provide concrete and visible benefits to IS engineering it is essential to take the ontological models beyond the modelling/design stage and attempt to use them not only to influence the implementation of technological artefacts (e.g., databases and software), but preferably to realise the ontologies in the technology itself. This means being able to take a foundational ontology with the modelled domain ontologies and create a database or software implementation that maintains high levels of direct traceability to the ontology. With most traditional paradigms (e.g., relational databases and object-oriented languages), aligning the technological implementation to the ontology is possible, but given the paradigm mismatch, development normally occurs with 'workarounds' that may have a negative impact on ontological alignment.

In graph databases [9] representations assume a graph form with nodes and edges. Edges represent relations between nodes. Properties can be defined for both the nodes and the edges. Graph databases are schemaless; this means that, unlike relational databases in which data must be represented and stored in a rigid structure with tables, rows and columns, the only structural constraint that graph databases dictate is the graph structure (a network of nodes connected by edges). This allows the modeller/developer significantly increased flexibility in the way the data is represented. From a metamodelling perspective this implies that the metamodel can be treated as data and represented as a graph and combined with its model instantiations also represented in the same graph. In our case the foundational ontology represents the metamodel and the domain ontologies represent the metamodel instantiations. As a consequence, our working research 'hypothesis' is that a schemaless database would enable us to implement the database in a form that more closely resembles the 4D ontology.

3 A Perdurantist Foundational Ontology

BORO, developed by Partridge [2], is a perdurantist upper level ontology strongly based on extensionality. BORO influenced the ISO 15926 standard and inspired the upper level ontology of the International Defence Enterprise Architecture Specification for exchange Group [10], adopted by the U.S. Department of Defense Architecture Framework (DoDAF). BORO has been applied in various industrial sectors including finance, oil and gas, and defence.

The aim of this section is to present the BORO foundational ontology and provide the reader with the fundamental knowledge to understand the work described in the remainder of the paper. It is beyond the scope of this paper to provide an exhaustive explanation and definitions of the whole foundational ontology. For an in depth presentation of BORO the reader is invited to refer to Partridge [2] in its original form or IDEAS [10] for a slightly modified, yet still detailed, version.

From a philosophical perspective the BORO foundational ontology explicitly addresses a set of metaphysical choices. BORO has adopted: (1) a realist stance towards ontology, that is it takes for granted a mind-independent real world; (2) a revisionary stance – accepting that if we want better models, we need to change the ways we look at the world; (3) completeness categories based upon extensional criteria of identity and (4) a 4D and possible worlds approach as these fit best with its commitment to extensionalism [11].

Figure 1 presents a graphical representation of the foundational ontology. The names of the foundational objects are prefixed with 'F_'. The notation is that of the Unified Modelling Language (UML).

At its highest level the BORO foundational ontology represents:

- *Objects*: Anything that exists. (In IDEAS the term *Thing* is used in place of *Objects*.)
- *Elements*: An element is a physical body with a spatiotemporal extent (i.e., particulars).
- *Types*: A type is a set or class of objects (i.e., universals). The extension of a type is given by all the objects of that type. Objects of a certain type are said to be instances of that type. Types can have individual instances (*ElementTypes*), type instances (*Powertypes*) or tuple instances (*TupleTypes*). Only *TupleTypes* are explicitly represented in Figure 1.
- *tuples*: A tuple is a relationship between two (in the case of *couples*)or more objects. Examples of subtypes of *tuples* include *typeInstances*, *superSubTypes*, *powertypeInstances* and *wholeParts*.
- *TupleTypes*: A type whose instances are tuples. There is a *powertypeInstance* relation between *TupleTypes* and *Tuples*.
- *TemporalParts*: A temporal part is an individual whose spatiotemporal extent is part of another individual.
- *Events*: An event is an individual temporal part that does not persist through time (i.e., an event has zero 'thickness' along the time dimension). Events represent temporal boundaries that either create (*CreationEvents*) or dissolve (*DissolutionEvents*) individuals (e.g., a person) or individual temporal parts that persist through time (i.e., states).
- *States*: A state is a temporal part of an individual that persists through time. States (and elements in general) are bounded by events. A state can have further temporal parts (i.e., states and events).
- *happensTo*: This tuple type relates an event with one or more individuals affected by the event. *happensTo* has two subtypes:
  - *creates*: Relates a creation event with the element(s) whose creation is triggered by the event.
  - *dissolves*: Relates a dissolution event with the element(s) whose dissolution is triggered by the event.
- *happensAt*: This tuple type relates an event with a time instant or interval (*TimeInstantsOrIntervals*) and it indicates the time at which an event takes place.
- *temporalPartOf*: This tuple type relates an individual with its temporal parts (states and/or events).

To visually clarify how BORO as a perdurantist ontology models the real world including change, let us consider a simple example of a company (*Acme Company Ltd.*) who during the course of its life changes its primary business activity from the production of paper to the production of mobile phones. Such information is normally legally required by Company Registration Offices. This is represented in Figure 2. As the figure shows *Acme* (as a 4D element) extends through space-time. A portion of *Acme*'s extension has a temporal part named '*Business Activity 1'* representing the company's paper manufacturing stage (or state) and another temporal part named

'*Business Activity 2'* representing the mobile phone manufacturing stage. Both stages have extents that are physically part of *Acme*'s overall spatiotemporal extension. There are also three events implicitly represented by the lines that bound the states. The temporal boundary on the left of '*Business Activity 1'* represents the event creating that stage, the boundary lying between '*Business Activity 1'* and '*Business Activity 2'* represents an event that dissolves the first state and creates the second state. The boundary to the right of '*Business Activity 2'* dissolves this state.

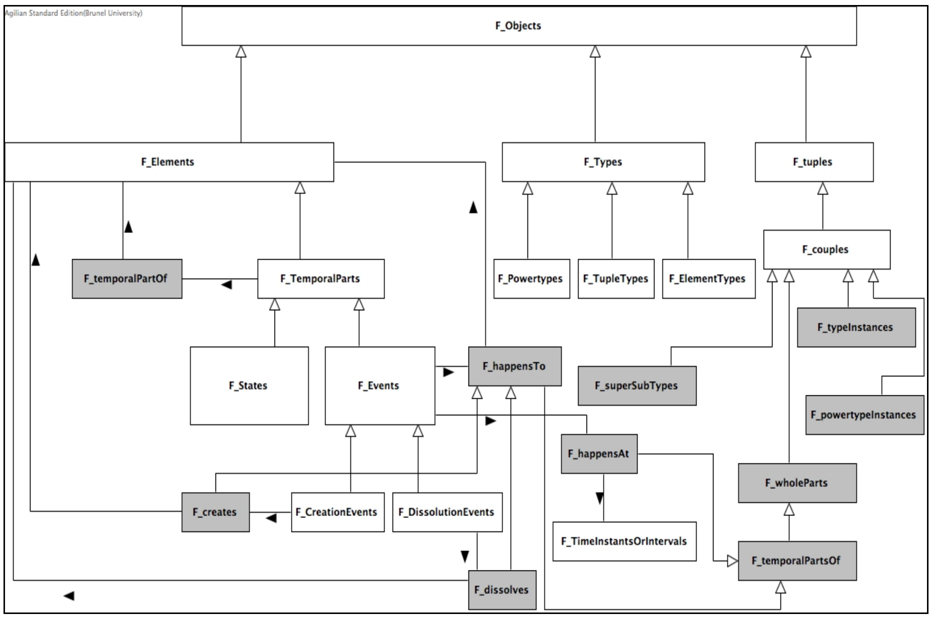

Fig. 1. BORO foundational ontology (partial view). (*TupleTypes* are represented in light grey).

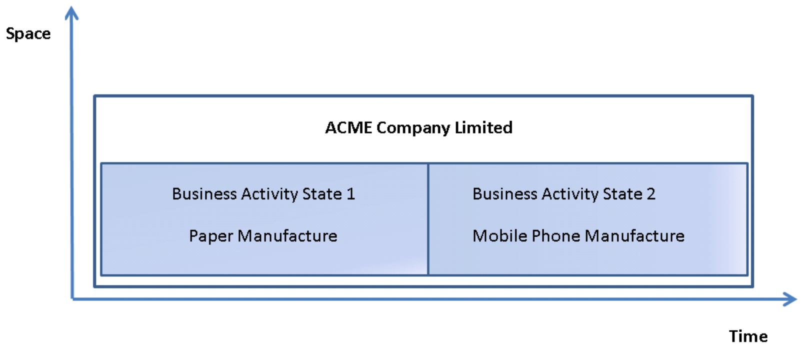

Fig. 2. Example space-time map

4 Implementing the Upper Ontology

4.1 The Foundational Graph

In order to represent ontological models in a database (in our case a graph database) it is necessary to represent and load the foundational ontology first of all. It is the foundational ontology that enables the representation of the domain ontology derived from the raw data. In essence things in the domain ontology will instantiate or subtype the high-level types of the upper level (i.e., the types represented in Figure 1). Since all models are a graph, the model in Figure 1 should ideally be transposed as it stands into the graph database. However a few considerations must be made.

In BORO *tupleTypes* and *tuples* are first-class objects. As Figure 1 shows *tupleTypes*, like *F_happensTo* and *F_creates*, are not simply drawn with the UML association notation but explicitly represented with the UML class symbol. This is necessary in order to allow the ontologist to describe the *tupleTypes* and *tuples* themselves. For example, subtyping a *tupleType* as in the case of *F_temporalPartsOf* and *F_wholeParts*.

In the graph implementation we have therefore chosen to maintain the same explicit representation of *tupleTypes* and *tuples*. Therefore when representing a relation (*R1*) between two things (e.g., *Prince William* and *Prince Charles*) in Neo4J, the relation is not simply represented with two nodes and an edge (i.e., [William → Charles]), but with three nodes and two edges. This enables us to say the following:

[*William* → *R1* → *Charles*] (1)
[*childOf* → *typeInstances* → *R1*] (2)
[*tuples* → *superSubTypes* → *childOf*] (3)
[*tupleTypes* → *typeInstances* → *childOf*] (4)

While it is important to explicitly represent relations, there are three upper-level relations that are unsurprisingly very widely used: *typeInstances*, *superSubTypes* and *powertypeInstances*. In these three cases we have decided to simply name the edges (as shown in the above listing) rather than reify the relations. The name would be implemented as a property of the edge. This makes the graph more compact with, in this context, losing required explanatory power. Otherwise a relation like (3) would become:

[*tuples* → *R2* → *childOf*] (5)
[*superSubTypes* → *typeInstances* → R2] (6)

With *typeInstances* relations there is another reason for using this compact form. The problem in Philosophy is known as the Third Man Argument (or Bradleyian Regress) and leads to an infinite regress of reified relations. For example, in (5) of the above listing the reification of the typeInstances relation leads to the following:

[*tuples* → *R2* → *childOf*] (7)
[*superSubTypes* → *R3* → *R2*] (8)
[*typeInstances* → *R4* → *R3*] (9)
[*typeInstances* → *R5* → *R4*] (10)
ad infinitum …

R3, *R4* and *R5* are all instances of *typeInstances* leading to an infinite chain of relations.

4.2 Graphs of Domain Patterns

Besides being a foundational ontology, BORO also defines a method for discovering general ontological patterns from existing systems and data. These general patterns enforce reusability and enable the ontologist to apply existing semantic models to the knowledge discovered from the interpreted data. As with the foundational ontology, these general patterns must also be loaded into the graph database before they can be used; however unlike the foundational layer, such patterns can be loaded in parallel with the semantic interpretation of the data as long as they are present in the database prior to their use.

An example of a general pattern is the Naming Pattern (in the model the prefix used is ‘N_’) represented in graph form in Figure 2. The *N_Names* type represents the set of all possible names in the world. *N_Names* is a type and not an element. For example, the name *John* is considered as the set of all utterances (written, oral, etc.) that name people called *“John”*. A naming space is a set of names; for example, the set of registration numbers that Companies House issues to uniquely identify a company.

It is important to note that while all nodes of the graph are labelled by a ‘name’ property in Neo4J, this property merely names the modelling element and it is not meant to be a name for the thing being modelled. The only exception is the name

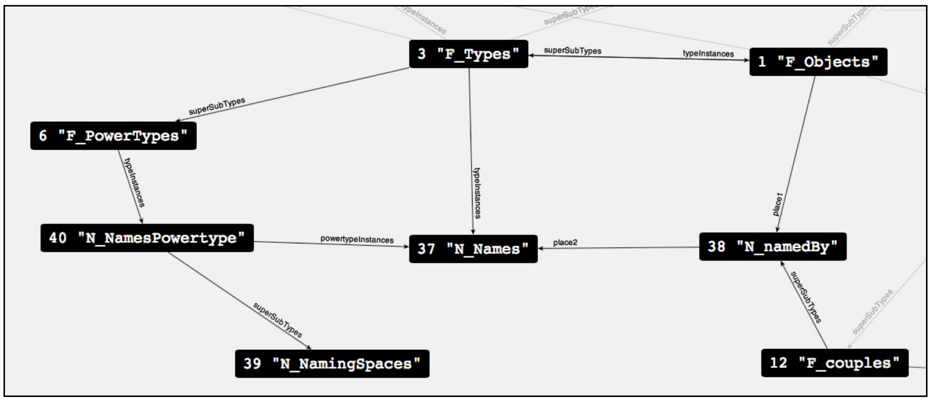

Fig. 3. Naming Pattern in Neo4J

property of *N_Names*. Names of real world objects must be represented as instances of the *N_Names* type to which the named object is related via the N*_namedBy* tupleType.

For reasons of space other patterns are not shown here but will be referred to later in the paper.

5 Semantic Transformation of the Data

This research uses a dataset acquired from Companies House in the U.K. The dataset was provided as a CSV file of approximately 3 GB corresponding to a spreadsheet of 173 columns and approximately 3.5 million rows. An extract of the column headings is provided in Table 1. The data was managed with custom-built code written in Python importing the standard CSV module. For the creation of the graph database the py2neo API was used in order to send REST requests to the Neo4J server.

Table 1. Extract of the column headings of the data file

registration_ number	legal_status_ code	date_inc	latest_accounts_date	latest_ar_date

After implementing the foundational ontology and those patterns deemed relevant to the domain (e.g. names, persons and intentionally constructed objects), the semantic interpretation of the data proceeded as follows.

First, the set of rows was interpreted. This implies understanding what a row refers to and the type it is an instance of. In this case each row refers to an individual U.K. company. The type instantiated is named *UKCompanies*. Once the meanings of the elements and the type have been determined, these objects must then be related to existing patterns and via the patterns to the foundational ontology. If no existing pattern appears to be relevant then this may be a sign that a new pattern may be hidden in the data and possibly discovered through further analysis. In this instance the Persons pattern previously loaded can be reused. In fact *UKCompanies* is a subtype of *LegalPersons* which in turn is a subtype of *Persons*.

The next (and most significant) step is to iterate through the columns of the spreadsheet and semantically interpret them. While the interpretation of the set of rows was relatively straightforward (at least in our case), interpreting column data presents some interesting challenges. There are a few mapping patterns (MP) that have emerged and summarised as follows:

MP1: If r_i represents the specific element that a row refers to and R_i its type, then one can think of a column as representing a type (C_i) and the intersection with the row (i.e., each cell) explicitly representing an instance of C_i (or c_i). Implicitly represented are also a tuple type (T_i) and a tuple (t_i). For example, with the first column (named "registration_number") the mapping in Table 2 emerges.

Table 2. Example of Mapping Pattern MP1

Spreadsheet Type	Value	Refers to	Referent	BORO Foundation
Column name (C_i)	registration_number	Set of all registration numbers assigned by Companies House	CHRegNumbers	F_Types
Cell value (c_i)	"0000006"	Individual registration number assigned to a company	"0000006" (instance of CHRegNumbers)	F_Elements
Implicit relation (T_i)	n/a	Set of all relations between UK Companies and Registration Numbers	namedByCHReg-Number	F_TupleTypes
Implicit relation (t_i)	n/a	Relation between a specific UK company and the registration number "0000006"	The tuple: (Company6, 0000006)	F_tuples

MP2: In many cases the columns cannot be interpreted in isolation because their values represent elements that have relations between them. For example, there are columns representing the different parts of a company's address (street and number, city, county, etc.). In this case there exists a *wholeParts* relation between them respectively. As a consequence the rules in MP1 are applied along with another interpretation rule which maps (and makes explicit) the implicit relations between the types represented by the columns and the pairs of elements represented by the cell values of those columns on the same row.

MP3: Some cells contain values that encode and map to more than one real world element or even to an entire classification structure. An example of the former is a composite address (e.g., 123 Main Street). In this case '123' refers to a specific building while 'Main Street' refers to a whole street. The latter case (which actually may be a mapping pattern in its own right) is exemplified by the U.K. Standard Industry Codes (SIC). In this case a specific code, expressed as a string (of numeric characters) in the spreadsheet, codifies a structure in which the code can be broken down into parts, with each part representing successive groupings of companies. In SIC terminology these groupings are called sector, division, group, class and subclass. Coding schemes like SIC are classification systems, which BORO can handle well with *PowerTypes* (or the set of all subtypes of a given type) in conjunction with *superSubTypes* and *powertypeInstances* (two tuple types of the foundational ontology). This type of semantic transformation allows us to explicitly model and refer to an entire classification system (i.e., SIC), relate it to other classification systems (for example, successive versions of SIC) and relate it to a naming space (in this case U.K. SIC codes). This is an effective example of how BORO is capable of transforming a set of simple codes (e.g., "0311") into a complex ontological structure. Thanks to BORO's strong extensionality principle a clear and explicit distinction is made between a classification system and its naming space.

MP4: This mapping pattern builds on MP3 and relates to cases in which there is a succession of columns that represent a type of change that a company may undergo in its lifetime. Examples include a change of address, change of name, change of SIC code, etc. While there are subpatterns that are not discussed here for limitations of space, a typical case is one in which there are columns representing the current status (e.g., current address) and the previous four statuses (addresses) plus further columns specifying when the various changes (of address) occurred. The mapping to a 4D ontology translates into a new subtype of *F_States* (e.g. *CompanyX_at_AddressY*), a new subtype of *F_Events* (e.g., *AddressChange*) and all related subtuples/types that this entails as modelled in the foundational ontology. While the original data was limited to the representation of only four previous addresses, the 4D graph model can record an unlimited number of changes. This pattern is a clear example of the effectiveness of the 4D approach in modelling change. In fact, with each change of address the new state represents a temporal part of the company, which can be itself related to its corresponding address, thus providing a more objective model of what occurs in the real world.

The above mapping patterns are a non-exhaustive list. As the research progresses and more data is semantically analysed, we expect to discover further mapping patterns with more clearly defined rules, as well as refine the existing rules. At this stage the mapping is being carried out manually and the translation rules are being encoded in Python on a case-by-case basis. We envisage that once these patterns are tested against a greater amount of data, we will be able to develop generic APIs for each consolidated and tested mapping pattern so as to gradually proceed to a much higher level of automation for the further 4D re-engineering of data.

6 Related Work

The focus of the work presented in this paper is on the semantic transformation of large amounts of data (e.g., Big Data) acquired typically from heterogeneous sources and in semi-structured raw formats (e.g., CSV files). This problem is part of a broader research area related to the re-engineering of data and systems. In fact the challenges encountered and described here are typical of most projects that adopt processes similar to what is known as Extraction-Transformation-Loading (ETL) [12]. While ETL is mostly applied in Data Warehousing, the general problems are common. In our case, however, the main differences lie in the transformation phase, which is entirely driven by a 4D foundational ontology. In brief, for us transformation consists of Semantic Interpretation (translation into the 4D paradigm), Semantic Improvement (generalise to existing patterns or identify general reusable ontological patterns), and Semantic Harmonisation (consistently integrate the new ontologies and patterns into the existing ontological graph). Moreover previous work carried out on ontology-driven ETL (e.g., [13]) normally adopts a Semantic Web (OWL/RDF) approach rather than be driven by a philosophically grounded foundational ontology.

Further literature (for example, see [14] and [15]) also investigates ontologies in the context of public data and its use for providing information services. However, even here, as with most ontology related ETL work, the focus is on Semantic Web Technologies (primarily Linked Data) with no grounding in philosophical foundational ontologies.

An example of previous research that has also investigated the use of foundational ontologies to derive domain models and ontology patterns from Web resources is

SmartWeb Integrated Ontology (SWIntO) [16]. However in this case the authors adopted a foundational ontology (merger of DOLCE and SUMO), which focused on linguistic-cognitive aspects. This was fully justified in this project due to SmartWeb system's heavy reliance on linguistic information. In fact one of the main objectives of the research was to "produce domain-specific ontologies that are relevant for mobile and intelligent user interfaces to open-domain question-answering and information services on the Web". In our case, a linguistic-cognitive based ontology would not have been suitable since our aim is for the re-engineered models to ultimately be fully integrated with an organisation's corporate knowledge assets, hence our decision to adopt a foundational ontology that is context-independent and mind-independent. Such an approach appears, in our view, more appropriate toward facilitating the integration of multiple data sources with the existing corporate data.

7 Conclusion

This paper summarised the initial findings of research-in-progress aimed at re-engineering large amounts of raw data with a 4D ontology and implementing the ontology in a graph database (Neo4J). The research thus far uncovered some of the major challenges related to (1) implementing a 4D foundational ontology in a graph database, (2) semantically interpreting raw data in a spreadsheet format to a 4D ontology and (3) identifying mapping patterns which, with further testing, can help to move from a manual data translation into a more automated mapping to the 4D ontology and consequent loading of the semantically interpreted model into Neo4J.

Once the semantic problem of translating the data from its raw form to a 4D ontology and a 4D graph is proven effective our next step will be to integrate further data sets which besides being grounded in the 4D foundational ontology must be harmonised with the previously reengineered models. We will also explore developing an automated solution that will adopt the discovered mapping patterns to transform the data into 4D representations. Performance will also be fundamental for information retrieval and general usage once the system goes into the production stage. This means that while the database will be semantically rich and expressive thanks to its strong ontological foundation, it must also run on an architecture that guarantees high accessibility and performance [7].

Acknowledgement. The authors wish to thank James Dobree, CEO of Level Business Ltd., for funding this research.

References

1. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)
2. Partridge, C.: Business Objects: Re-Engineering for Re-Use. Butterworth-Heinemann (1996)

3. Partridge, C., Mitchell, A., de Cesare, S.: Guidelines for developing ontological architectures in modelling and simulation. In: Tolk, A. (ed.) Ontology, Epistemology, & Teleology for Model. & Simulation. ISRL, vol. 44, pp. 27–57. Springer, Heidelberg (2013)
4. Neo4J, `http://www.neo4j.org`
5. McAfee, A., Brynjolfsson, E.: Big Data: The Management Revolution. Harvard Business Review (October 2012)
6. Minister of State for the Cabinet Office and Paymaster General, Open Data White Paper: Unleashing the Potential, `http://data.gov.uk/sites/default/files/Open_data_White_Paper.pdf` (last accessed on April 03, 2013)
7. Pereira, A.L., Appel, A.P.: Modeling and Storing Complex Network with Graph-Tree. New Trends in Databases and Information Systems. Advances in Intelligent Systems and Computing 185, 305–315 (2013)
8. Sider, T.: Four-Dimensionalism: An Ontology of Persistence and Time. Oxford University Press, USA (2002)
9. Angles, R., Gutierrez, C.: Survey of Graph Database Models. ACM Computing Surveys 40(1) (2008)
10. IDEAS Group: The IDEAS Model, `http://www.ideasgroup.org/foundation/`
11. Partridge, C., Mitchell, A., de Cesare, S.: Guidelines for Developing Ontological Architectures in Modelling and Simulation. In: Tolk, A. (ed.) Ontology, Epistemology, & Teleology for Model. & Simulation. ISRL, vol. 44, pp. 27–57. Springer, Heidelberg (2013)
12. Vassiliadis, P.: A Survey of Extract-Transform-Load Technology. International Journal of Data Warehousing and Mining 5(3), 1–27 (2009)
13. Skoutas, D., Simitsis, A.: Ontology-based Conceptual Design of ETL Processes for both Structured and Semi-structured Data. International Journal on Semantic Web and Information Systems 3(4) (2007)
14. Shadbolt, N., O'Hara, K., Berners-Lee, T., Gibbins, N., Glaser, H., Hall, W., Schraefel, M.C.: Linked Open Government Data: Lessons from Data.gov.uk. IEEE Intelligent Systems (May/June 2012)
15. Alani, H., Dupplaw, D., Sheridan, J., O'Hara, K., Darlington, J., Shadbolt, N.R., Tullo, C.: Unlocking the Potential of Public Sector Information with Semantic Web Technology. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC/ISWC 2007. LNCS, vol. 4825, pp. 708–721. Springer, Heidelberg (2007)
16. Oberle, D., Ankolekar, A., Hitzler, P., Cimiano, P., Sintek, M., Kiesel, M., Mougouie, B., Baumann, S., Vembu, S., Romanelli, M., Buitelaar, P., Engel, R., Sonntag, D., Reithinger, N., Loos, B., Zorn, H.P., Micelli, V., Porzel, R., Schmidt, C., Weiten, M., Burkhardt, F., Zhou, J.: DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). Web Semantics: Science, Services and Agents on the World Wide Web 5, 156–174 (2007)