Grounding for an Enterprise Computing Nomenclature Ontology

Chris Partridge[1,2(✉)], Andrew Mitchell[1], and Sergio de Cesare[2]

[1] BORO Solutions Ltd., London, UK
{partridgec,mitchella}@borogroup.co.uk
[2] University of Westminster, London, UK
s.decesare@westminster.ac.uk

Abstract. We aim to lay the basis for a unified architecture for enterprise computer nomenclatures by providing the grounding ontology based upon the BORO Foundational Ontology. We start to lower two significant barriers within the computing community to making progress in this area; a lack of a broad appreciation of the nature and practice of nomenclature and a lack of recognition of some specific technical, philosophical issues that nomenclatures raise. We provide an overview of the grounding ontology and how it can be implemented in a system. We focus on the issue that arises when tokens lead to the overlap of the represented domain and its system representation – system-domain-overlap – and how this can be resolved.

Keywords: Enterprise computing nomenclature ontology · Nomenclature · Identifier · Identifier inscription · Identifying space · Foundational Ontology · BORO · System-domain-overlap · Type-token distinction · Type-token-occurrence distinction · Use-mention distinction · Paper tools

1 Introduction

An examination of a legacy enterprise system's data schemas will usually reveal, among other things, that a substantial proportion of the data are identifiers; many of the field/attribute names will have a tell-tale suffix; such as code, short name, and identifier. Typically, the identifier fields in the data schemas, such as 'Alpha Currency Code', mark out a column of identifiers in (what we call) a nomenclature, a formalised system of identifiers. When one starts to examine the ways in which the nomenclatures have been implemented, there seem to be a variety of patterns, driven by a combination of established practices and requirements, with little theoretical foundation.

Given the scale and ubiquity (and, as we shall show below, importance) of these nomenclatures, it is not surprising that there have been attempts to try to organise them across the enterprise (e.g. [1]). However, none of these have yet produced either a general, unified architecture or an ontology for computer nomenclatures. In this paper, we aim to lay the basis for such a unified architecture by providing the grounding for an nomenclature ontology.

A. H. F. Laender et al. (Eds.): ER 2019, LNCS 11788, pp. 1–9, 2019.
https://doi.org/10.1007/978-3-030-33223-5_40

We believe that there have been two significant barriers within the computing community to making progress in this area. Firstly, a lack of a broad appreciation of the nature and practice of nomenclature. In the first part of the paper, we develop this through a brief historical review. Secondly, a lack of recognition of some specific technical, philosophical issues that nomenclatures raise. In the second part of the paper, we outline these issues. Next, we provide an overview of the ontology and how it can be implemented in a system. In the third part, we describe a nomenclature ontology that addresses the issues identified. In the fourth part, we describe a significant issue that arises when implementing the ontology in a system and how to address it.

2 A Brief History of Nomenclatures

In this paper, our focus is on nomenclatures in enterprise computer systems. However, these are the result of a long and broad evolution that started well before computing technology or enterprises emerged. We look at this evolution to provide a better understanding of their general nature and see more clearly where and how concerns have shaped them. Nomenclatures are dependent upon classifications. Initial classifications were developed without the formality and structure of nomenclatures, which emerged when classifications start to scale. The combination of classifications and nomenclatures, particularly in biology, are often called taxonomies – the Linnaean taxonomy being the original example.

There is ample evidence of classification from early history; examples include Ancient Egyptian onomastica (such as the Onomasticon of Amenope), Aristotle's Categories and Theophrastus's Historia Plantarum. In the 15th century, the invention of printing introduced the technology to support larger classification systems. In the 16th and 17th centuries, this led to an increase in works classifying plants and animals. In Species Plantarum (1753), Carl Linnaeus started modern formal binomial nomenclature with two types of names; genus and species – these are early examples of what we call *identifying spaces*. Over time, nomenclature management was formalised. For many scientific nomenclatures, a two-part process evolved; firstly, providing an accessible example of the identified entity – a type specimen – and then secondly publishing its name, publicly providing an example of the name. Both the type specimens and published names were exemplars provided for examination and comparison. This formalisation aimed to make the process more efficient, allowing the nomenclature to scale effectively. This formalisation was part of a broader emergence of bureaucracy, which aims for a rational and efficient way of organising human activities. As Weber [2, Chap. 6] noted, this mechanises the organisation. As well as being subjected to bureaucratisation, nomenclatures played a significant role in enabling it. As they developed more formal processes, enterprises found their efficiency depended heavily upon standardised classifications and their associated standardised nomenclatures; modern examples include ISO 3166 Country Codes and ISO 4217 Currency Codes.

In many ways, computing technology offered an opportunity to create the ultimate bureaucracy – a living embodiment of the 'iron cage' (Weber's characterisation of bureaucracy). It enables far more efficient rational calculation and control than its predecessor, writing technology. Nomenclatures seem a natural fit for computing, whether storing the codes or handling the rules for managing them. Though there is a recognition that the paper-based rules might need tightening up to take advantage of computing's capabilities – see, for example [3].

3 Nomenclature's Type-Token Architecture

For sound pragmatic reasons nomenclatures are streamlined. For example, within an identifying space, the identifiers typically aim to be unique (no duplicates) and distinct (no object with two identifiers), making algorithmic identification feasible. It is the identifiers that are unique rather than the individual inscriptions. When the algorithmic rules are executed, the process of identifying and reidentifying does not involve the identifier directly. Let's say we are matching an identifier in an article with the nomenclature's list. We match two inscriptions, a portion of text in a copy of the article with an entry in a copy of the list. Clearly, neither inscription is the identifier (if it were, we could simply destroy it and then the identifier would no longer exist). The processes work with the identifiers indirectly through their distinct member inscriptions.

This distinction between the identifier and the inscription is known as the *type-token distinction*. It was introduced by Peirce [4, sec. 4.537]; where (roughly) types are general, and tokens are their concrete instances; typically written (inscriptions) or spoken (utterances). So, for example, in this sentence – Rose is a rose is a rose is a rose – one could say that there are three different word-types ('rose', 'is' and 'a') and ten different word-token inscriptions. (There are also three occurrences of 'rose', but we don't have enough space to discuss this here). From a nomenclature's perspective, identifiers are word-types and identifier inscriptions are word-tokens. The nomenclature rules are typically framed in terms of word-types, though the execution of the rules typically involve word-tokens. However, the design, particularly of computer nomenclatures, does not usually make explicit the type-token distinction.

4 Outlining a Nomenclature Ontology

Here we provide an example of what a general nomenclature ontology would look like. We base this upon the BORO Foundational Ontology [5]. Hopefully, it will inspire alternative nomenclature ontologies based upon different foundations. There is a good argument that a nomenclature ontology should be included within a foundational ontology as it spans multiple (if not most) domains.

One of the critical requirements is to capture the type-token distinction, to clearly identify identifier inscriptions (tokens) and separate them from identifiers (types). This is done by showing that they are different types of objects. There is also a less obvious requirement to characterise what the third type of nomenclature component, identifying spaces, are. Identifier inscriptions (tokens) manually written on paper, carved in stone or metal are visibly concrete, existing in space and time. Identifier inscriptions (tokens) written by the computer into computer storage (whether a magnetic tape or disk or solid-state memory), while not directly visible, are also plainly concrete and spatio-temporal. The identifier inscriptions (tokens) belong to an identifier (type). Under the

BORO extensional view, these types are the set of tokens (all possible tokens in all possible worlds). For example, all GB inscriptions that are ISO Country Code inscriptions make up the type that is the GB ISO Country Code. This extensional view is common, though not universal, in philosophy [6–11]. On this view, identifier inscriptions (tokens) and identifiers (types) are clearly different kinds of thing. Identifiers (types) belong to identifying spaces. For example, the GB ISO Country Code belongs to the ISO 3166-1 alpha-2 codes identifying space. Under the extensional view, these identifying spaces are the set of appropriate identifiers (types).

In this view, the objects in the nomenclature ontology are recursively grounded in a series of levels. As shown in Fig. 1, at ground level, the identifier inscriptions are of spatio-temporal particulars. At the next level, identifiers are grounded in their member inscriptions; then, the identifying spaces are grounded in their member identifiers.

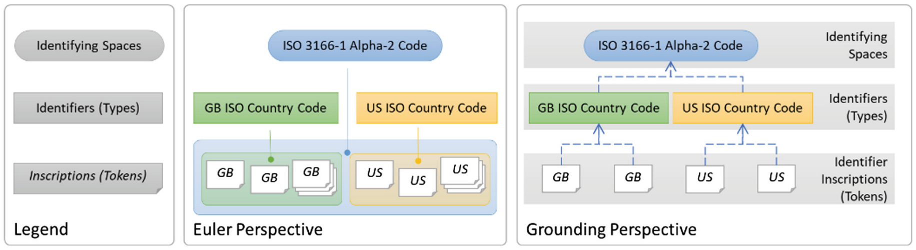

Fig. 1. Grounding levels

The purpose of an identifier is to refer to an object – the identified object. Furthermore, every identifier (type) and all its identifier inscriptions (tokens) refer to the same object. From an engineering perspective, there are two main architectural options for how this reference could work; firstly, that only the identifiers (types) refer directly and the identifier inscriptions (tokens) refer indirectly via their types or secondly, that both identifiers (types) and the identifier inscriptions (tokens) refer directly. The second option has the disadvantage of needing a mechanism to ensure that all the identifier inscriptions (tokens) consistently refer to the same object as their identifier (type) – a typical example of data redundancy.

5 Implementing a Nomenclature Ontology System

When using the BORO methodology (bCLEARer) to mine the ontologies from legacy systems, the relevant data, as well as their data schemas, are extracted and transformed into the ontology. Typically, the legacy systems contain domain nomenclatures, so these are extracted, transformed and stored in the ontology. Our experience is that this raises a theoretical issue with design implications. The underlying issue is that the represented domain and its representation overlap (as shown graphically later in Fig. 3); the actual real identifier inscriptions (parts of the domain, and not some representation of them) are stored and processed in the system. We call this a ‘system-domain-overlap'. The situation repeats in models of the system, where tokens are typically used in naming – a kind of a 'model-domain-overlap' or more generally a 'representation-represented – overlap'.

5.1 Lessons Learnt from Mention and Use

The issues that this kind of overlapping raise have been recognised in philosophy, though mostly in the context of written text (so writing technology). Quine [12, pp. 23–26] describes a *use-mention distinction*. He contrasts two sentences; (1) Boston is populous and (2) 'Boston' is disyllabic. He notes that in the first sentence the name is being used, and in the second sentence it is being mentioned – and suggests we follow the practice of always using quotation to distinguish use and mention, saying it is more convenient but needs "special caution" as a "quotation … designates its object not by describing it in terms of other objects, but by picturing it."

While Quine is surely correct to distinguish between use and mention, his proposal for the use of quotation has been questioned. Tarski [13] and Church [14, Chap. 8] examine the use of quotations in natural language and decide against using them. Davidson [15] also examines this, noting several issues arising from how to interpret the use of a name-token inside the quotation marks. He proposes an alternative 'demonstrative theory' in which quotation marks help to refer to a name-type by pointing to (showing) a token of one. He suggests one reads them as 'the expression a token of which is here' – a 'word-type-reference to word-token-reference to word-token' pattern. The benefit with this approach is that anything that looks like a token is one. There is no need to view some tokens as pictures. A good lesson here is to let tokens be tokens.

Quine's quotation analysis assumes that each inscription of a name can be categorised as a use or a mention – this is necessary if mentions are to be enclosed in quotation marks. Davidson [15] suggests that one could both use and mention at the same time. The lesson here seems to be that one cannot usefully assume that a token in the published nomenclatures is exclusively a use or a mention. So, Quine's quotational approach cannot be applied directly to nomenclatures – they are more about ways of using their identifiers. One final consideration is technology. One can write quotation marks, but there is no easy way of directly pronouncing them. The possibility of quotation marks is created by writing technology; emerging technologies create new opportunities for representation. Both Quine and Davidson's proposals are plainly tied to writing technology. This suggests that there may be opportunities for a new mode of nomenclature representation suitable for computing technology.

5.2 The Current Patterns of Implementation – A Baseline

Current system implementations contain tokens and successfully work with them. Examination of enterprise systems shows two main patterns illustrated in Fig. 2 (Identifiers as Attributes and Identifiers as Objects), neither of which resemble Quine and Davidson's approaches mentioned above. It is not uncommon to find examples of both patterns in a single system. The figure shows the pattern for a single entity type – Countries. In an enterprise system, this pattern is repeated for each entity type –

resulting in a multiplicity of nomenclature infrastructures. Our goal here is to find a way of unifying them.

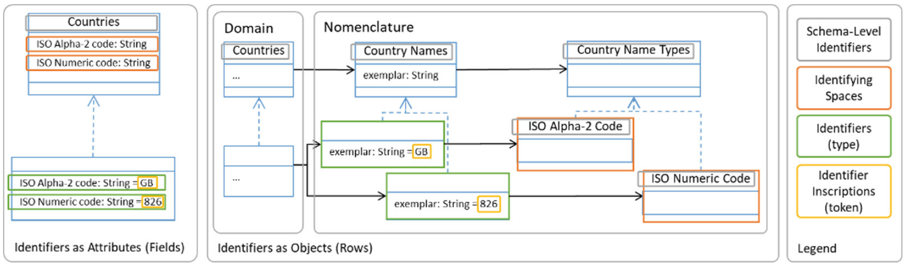

Fig. 2. Patterns of current identifier implementation for an entity

In Fig. 2, the main data-level nomenclature components have been marked out, as far as it is feasible. Hopefully, this makes clear that this classification is implicit, that there is no way to work it out from just the explicit structure. Also, the components are all, in a way, second class citizens [16]; in that, as attributes or objects they do not have access to the same range of resources as classes; for example, they cannot be super- or sub-typed. In the models of the domain, the representations of the three components of the nomenclature are explicitly marked out. They also clearly separate the "pure" domain and the nomenclature. As we use this as a basis for our implemented system, we keep this architecture.

5.3 A Proposed Implementation

Figure 2 makes clear that the system contains 'real' identifier inscriptions (tokens). These play a critical role in the operation of the system as they are exemplars of their types, used to identify and reidentify tokens of the same type. The nomenclature ontology needs to be extended with tokens. Given the earlier analysis, we want to introduce the tokens as just tokens, with no additional commitments to sometimes treating the tokens as pictures of themselves. We also want to avoid committing to a token being either exclusively for use or mention. The simplest design is to add the token as a new kind of representation and connect them to their representation in the model. This is similar in some ways to Davidson's 'word-type-reference to word-token-reference to word-token' pattern (mentioned above), in that the word-type and word-token are referred to and the actual token demonstrated. One outstanding issue is that the non-token picture uses of the inscriptions remain in the representations. The next step is to remove these, leaving the representations as bare nodes.

When humans wish to review the model, it is useful to present the nodes with labels. To achieve this, one needs first to make a clear distinction between what is stored internally and what is viewed (a kind of ANSI-SPARC or model-view architecture). In the view layer, one presents a framed copy of the token (in a similar manner to Quinean quotations) with agreed framing glyphs. The view can recover the names/labels of these nodes algorithmically by navigating from the bare node to their 'real' identifier inscriptions. One can present the name/label using a bare copy of the 'real' identifier inscriptions; we have done this for the schema level representations in our models – nodes such as 'Identifiers' and 'Countries'.

This gives us a system such as that modelled in Fig. 3, which illustrates both ways of presenting the tokens; firstly, reflecting the way they are stored in the ontology data structure with tokens and bare nodes and secondly, as bare nodes that are adorned with an inferred name. In the latter case, this notation needs to be read as shorthand for the fuller first case.

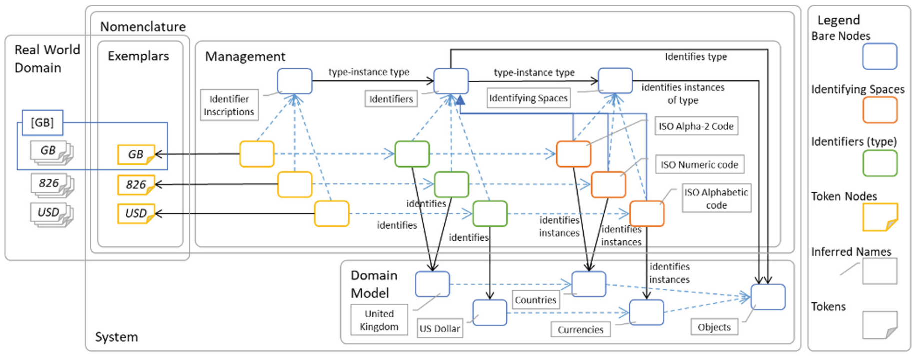

Fig. 3. Proposed implementation structure

In this approach, all the nomenclature management is handled in the same way; there is no schema-data names distinction as in Fig. 2. Moreover, all the representations of nomenclature components are bare nodes and so are first-class citizens; they have access to the same resources as the domain's bare nodes. Tokens are always tokens, as there is no need for framing devices such as quotation marks; hence, there are no pictures of tokens. This resolves the issues identified earlier.

This way of managing tokens has a history. For example, in the semantic network SNePS, [17, 18], inscriptions are a special kind of thick node joined to the rest of the semantic network of thin bare nodes by LEX arcs (see also [19]). Schapiro [18] rightly compares the resulting separation of thick (token) and thin (bare) nodes with Carnap's [20, sec. 14] example of structural definite descriptions. However, the thick inscription nodes are held at arm's length via the LEX arc, without access to the resources of the bare nodes, making them second class citizens [16].

6 Conclusions

We believe that the grounding for an enterprise computing nomenclature ontology, described above, lays the basis for a unified architecture for nomenclatures in computer systems. We provided the grounding in four ways. Initially, we made clearer what a nomenclature is (and so its requirements). Then we looked at the critical technical issues faced when specifying such an ontology. We focused on the distinction between type (identifiers) and token (inscriptions). Then, we provided an example of a nomenclature ontology based upon the BORO Foundational Ontology that explicitly makes the distinction between inscriptions, identifiers and identifying spaces and specifies their identity criteria. It also clearly distinguishes between the domain and the nomenclature. Finally, we provided an example of an implementation of a nomenclature ontology. We identified the treatment of tokens as a significant issue, illustrating this with the analysis of the proposed use-mention distinction. We related this to system-domain-overlap and showed how this can be accommodated by treating tokens as token.

A theme running through the paper was the way in which nomenclatures are tools both shaped by and shaping the prevailing technology. In the era of printing technology, nomenclatures in lists and tables were ‘paper tools’ deployed alongside scientific taxonomic and bureaucratic classifications. These tools were subsequently embedded in computer enterprise systems. In this narrative, the nomenclature ontology can be used to design computer tools for a computer-based nomenclature, unconstrained by writing technology. We hope this paper will both encourage a unified approach to nomenclatures though the development of alternative nomenclature ontologies based upon different foundational ontologies and an increasing number of implementations of these ontologies in systems.

Acknowledgements. We want to thank Salvatore Florio and Mesbah Khan for all their help in developing this paper.

References

1. Business Scenario: Identifiers in the Enterprise. The Open Group (2006)
2. Weber, M.: Economy and Society. Bedminister Press, New York (1922)
3. McMurry, J.A., et al.: Identifiers for the 21st century: how to design, provision, and reuse persistent identifiers to maximize utility and impact of life science data. PLoS Biol. **15**, e2001414 (2017)
4. Peirce, C.S.: Collected Papers of Charles Sanders Peirce. Harvard University Press, Cambridge (1932)
5. De Cesare, S., et al.: BORO as a foundation to enterprise ontology. J. Inf. Syst. **30**, 83–112 (2016)
6. Ramsey, F.P.: Foundations of Mathematics and Other Logical Essays. Routledge, Abingdon (1931)
7. Whitehead, A.N., et al.: Principia Mathematica, to *56. Cambridge University Press, Cambridge (1925)
8. Quine, W.V.: Quiddities: An Intermittently Philosophical Dictionary (1987)
9. Haack, S.: Philosophy of Logics. Cambridge University Press, Cambridge (1978)
10. Hugly, P., et al.: Expressions and Tokens. Analysis **41**, 181–187 (1981)
11. Ayer, A.J.: Language, Truth and Logic. Courier Corporation, Mineola (1946)
12. Quine, W.: Mathematical Logic. Harvard University Press, Cambridge (1940)
13. Tarski, A.: The concept of truth in formalized languages. Log. Semant. Metamathematics **2**, 152–278 (1956)

14. Church, A.: Introduction to Mathematical Logic. Princeton University Press, Princeton (1996)
15. Davidson, D.: Quotation. Theory Decis. **11**, 27–40 (1979)
16. Strachey, C.: Fundamental concepts in programming languages. High.-Order Symb. Comput. **13**, 11–49 (2000)
17. Maida, A.S., et al.: Intensional concepts in propositional semantic networks. Cogn. Sci. **6**, 291–330 (1982)
18. Shapiro, S.C., Rapaport, W.J.: SNePS considered as a fully intensional propositional semantic network. In: Cercone, N., McCalla, G. (eds.) The Knowledge Frontier. SYMBOLIC, pp. 262–315. Springer, New York (1987). https://doi.org/10.1007/978-1-4612-4792-0_11
19. Partridge, C.: Business Objects: Re-engineering for Re-use (1996)
20. Carnap, R.: The Logical Structure of the World: Pseudoproblems in Philosophy (1967)