Grounding for an Enterprise Computing Nomenclature Ontology

Chris Partridge[1,2] [0000-0003-2631-1627], Andrew Mitchell[1] [0000-0001-9131-722X] and Sergio de Cesare[3] [0000-0002-2559-0567]

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

Abstract. We aim to lay the basis for a unified architecture for nomenclatures in enterprise computer systems by providing the grounding for an ontology of enterprise computing nomenclatures within a foundational ontology. We look at 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 paper we develop an ontology that can be used as a basis for nomenclature 'computer tools' engineered for computing technology.

Keywords: Enterprise Computing Nomenclature Ontology, Nomenclature, Identifier, Identifier Inscription, Identifying Space, Foundational Ontology, BORO, Type-Token Distinction, Type-Token-Occurrence Distinction, Use-Mention Distinction, Paper Tools.

"Now the LORD God had formed out of the ground all the wild animals and all the birds in the sky. He brought them to the man to see what he would name them; and whatever the man called each living creature, that was its name." Genesis 2:19

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, identifier and so on. 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, i.e. a 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 a general, unified architecture (or ontology) for computer nomenclatures. In this paper we aim to lay the basis for such a unified architecture by providing the grounding for an ontology of nomenclature.

We believe that there have been two main barriers to making progress in this area. Firstly, within the computing community, 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, starting with the nomenclatures represented in lists and tables that emerged at the end of the 16th Century as 'paper tools' alongside scientific taxonomic classifications and their development in bureaucracy and eventual embedding in current enterprise systems. Secondly, a lack of recognition of the technical, philosophical (in the sense that these as are discussed in the philosophy of words and names) issues that nomenclatures raise. In the second part of the paper, we examine key issues, which are typically raised in the context of ordinary language and review them in the context of nomenclatures exposed in the first part.

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 second part. In the fourth part, we describe a major issue that arises when implementing the ontology in a system and how to address it. The fifth and final part is the conclusion.

2 A Brief History of Nomenclatures

In this paper, our focus is on nomenclatures in enterprise computer systems. However, nomenclatures have been and are used more widely; they had a long and broad history before computing technology emerged. We look at this history to help give us a better understanding of their general nature and see more clearly where and how concerns have shaped them.

Nomenclatures are dependent upon classifications, though classifications do not necessarily need nomenclatures. Earlier, simpler classifications worked without the formality and structure of nomenclatures, which emerged when classifications start to scale. The combination of classifications and nomenclatures are often called taxonomies.

There is ample evidence of classification from early history; examples include Ancient Egyptian onomastica (such as the Onomasticon of Amenope containing 610 items organised hierarchically), Aristotle's Categories (ten items in a flat list) and Theophrastus's Historia Plantarum (a nine-volume work).

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 (genus/species) nomenclature. This contains two types of names; genus names and species names – we call these types *identifying spaces*. The Linnaean nomenclature introduced a couple of key innovations. Firstly, the names had a formal, fixed structure. The genus identifying spaces contain single word names, whereas the species identifying spaces contain binomial, two-part, names – where the first part is a genus name. Secondly, the names were intended to be identifying labels not descriptions; so, for example, some names honoured individuals rather than described the species.

Over time the process of managing the nomenclature 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. These processes, for both classification and nomenclature, were governed by rules, which tended to be made as algorithmic as possible, so that they could be followed without interpretation or discussion and so avoid contentious debate. This formalisation aimed to make the process more efficient, allowing the nomenclature to scale effectively.

This formalisation was part of a wider emergence of bureaucracy, which aims for a rational and efficient way of organising human activities. Where activities are formalised into a hierarchical structure with clear lines of authority and rigid division of labour, with decisions and powers specified and restricted by regulations. As Weber [2, chap. 6] noted, this mechanizes the organization: "The fully developed bureaucratic apparatus compares with other organisations exactly as does the machine with the non-mechanical modes of production." As well as being subjected to bureaucratisation, nomenclatures played an important role in enabling it. As they developed more formal processes, enterprises found the efficiency of these processes depended heavily upon standardised classifications and their associated standardised nomenclature; modern examples include ISO 3166 Country Codes and ISO 4217 Currency Codes.

The nomenclature's identifiers exhibit a range of composition structures. ISO 3166 Country Codes are simple and atomic with no internal structure. ISO 4217 Currency Alphabetic Code has an internal structure as it uses the first two letters of the ISO 3166-1 alpha-2 country codes. For example, Pound Sterling is GBP – GB for the United Kingdom and P for pounds. The format structure can get quite complex; for example, the NORSOK standard [3] developed for the Norwegian petroleum industry has quite intricate formats with multiple components.

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 is far more capable of 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 [4].

3 Nomenclature's Type-Token Architecture

We start by characterising nomenclatures as socially constructed through rules. We then examine how the detailed rules raise the issue of type-token distinction. Finally, we look at how accepting this distinction raises the question of what counts as a word-type – in other words, which tokens fall under which type.

Nomenclatures are socially constructed following agreed, specified rules, though historically agreeing on the design and implementation of these can take a while – see, for example, [5]. Many of the written rules spell out detailed algorithmic processes; for example, rules that eliminate punctuation marks such as apostrophes, quotation marks and full stops from names. What motivates these rules? Strawson, in [6, pp. 31–32], characterises the main driver. He describes how we need to be able to both identify something and to reidentify it. This is clearly the work the type specimens and identifiers are designed to do. However, the identifiers also need to be identified and reidentified. Many rules are aimed at this last requirement. Badly designed rules can make the names more difficult to identify or reidentify. Restricting the character set used in the names (for example, eliminating punctuation marks) makes them easier to recognise: not only are there fewer characters to check, but it also removes tricky edge cases, such as whether two names that differ by a final full stop are the 'same' name.

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 – for example, the species name *Homo sapiens* – that are unique rather than the individual inscriptions. However, 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 [7, 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.

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. This is an understandably common position in a variety of fields, given how it apparently simplifies things. But it turns out that identifying the word-type is not straight-forward. Linguists recognize different types of words: for example, McArthur 1992's The Oxford Companion to the English Language [8, pp. 1120–1121] lists eight different types of word: orthographic, phonological, morphological, lexical, grammatical, onomastic, lexicographical and statistical, but notes more can be found.

The problem of what counts as a word and what does not is discussed in the philosophical literature on words, for example, [9, 10]. Peirce [7, sec. 4.537] says that tokens are "one happening or a Single object or thing which is in some single place at any one instant of time". Like Peirce, the literature generally accepts that word-tokens are spatio-temporal, concrete particulars. But, it also recognizes that it is difficult to say exactly which properties one should use to allocate these tokens to types. It considers what properties could be used to characterise words. There is much discussion of the problems with simple properties. For example, should spelling be the benchmark? But there are different dialectal spellings (written inscriptions) – for example, ‘colour’ and ‘color’. And there are similarly spelled written inscriptions with different meanings in different languages – for example, ‘sensible’ in English and French.

From the perspective of nomenclature, it is important to recognise that the linguists and philosophers are examining natural language which is not subject to the same bureaucratic pressure as nomenclatures. What examination of the practices of nomenclature shows is that their design aims to regiment away many of these issues. They restrict their focus to identifying spaces within which the types and tokens are managed. Most nomenclatures, given the current state of speech recognition technology, focus exclusively on written inscriptions. They typically mandate a restricted character set and a single spelling of each identifier within their identifying space; enabling a simple character comparison algorithm to be used to establish identity. 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.

4.1 Based Upon a Foundational Ontology

We base this upon the BORO Foundational Ontology [11]. 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. A key metaphysical choice [12] this ontology inherits from the foundational ontology is extensionality as the criterion of identity, and this is used to characterise the objects in the ontology. One of the important 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 characterize 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 ISO Country Code – GB – inscribed on the screen in front of me as I am writing this (or the inscription in front of you the reader as you read it) are plainly concrete. Tokens are commonly regarded as concrete in philosophy [10].

In both the case of manual and computer written inscriptions, the identifier is composed of inscribed characters in a linear order. These inscribed characters are also spatio-temporal particulars. One could decompose characters – Goodman and Quine [13] explore how to build characters from shapes – but this is outside our current scope.

The identifier inscriptions (tokens) belong to an identifier (type). Under the extensional view, these types are the set of tokens (all possible tokens in all possible worlds). For example, the GB inscription above and all other GB inscriptions that are ISO Country Code inscriptions make up the class that is the GB ISO Country Code. This extensional view is common, though not universal, in philosophy [14–19]. On this view identifier inscriptions (tokens) and identifiers (types) are clearly different, separated as different kinds of thing. For this paper, we will use a shorthand for labelling types and tokens in models that clearly distinguishes the two: angle brackets to frame an identifier inscription (token) – e.g. <USD> – and square brackets to frame an identifier (type) – e.g. [USD].

Identifiers (types) belong to identifying spaces. For example, the GB ISO Country Code belongs to the ISO 3166-1 alpha-2 codes identifying space. ISO 3166-1 standardises two other identifying spaces, ISO 3166-1 alpha-3 and ISO 3166-1 numeric codes. Under the extensional view, these identifying spaces are the set of appropriate identifiers (types).

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

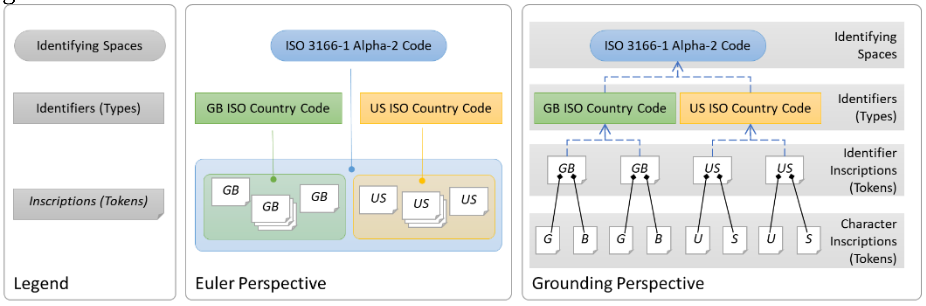

Fig. 1. Grounding Levels

BORO uses a powertype mechanism to ascend levels [20]. As shown in **Fig. 2**, multiple levels are needed. The ascending Elements powertype chain constructs the various set levels, where the Element n-type powertype constructs the sets – as its members – at level n (described here [21]). **Fig. 2**, also makes a clear distinction between the domain that is being identified and the nomenclature. In practice this is usually a reasonably sharp distinction. However, the ontology can easily support cases where one wants a nomenclature of a nomenclature – in these cases, the boundaries overlap.

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. **Fig. 2** illustrates the first, less redundant, option.

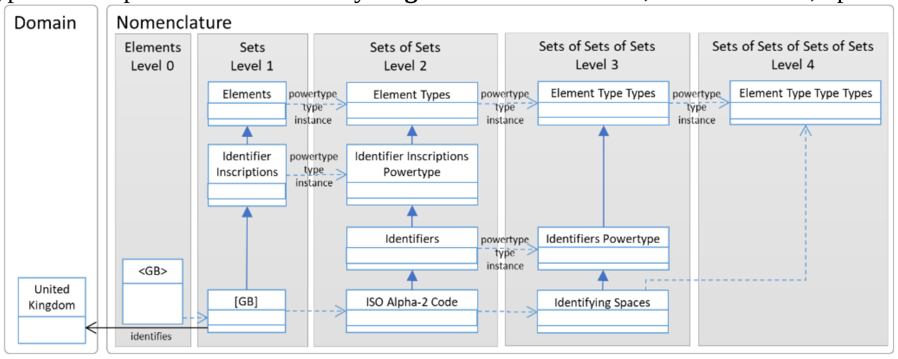

Fig. 2. Powertypes Ascending Levels

The grounding analysis gives a picture of how the objects are composed, of what they are. An analysis of how they are constructed gives a picture of how they work. Historically, nomenclatures have evolved in a variety of ways, that has tended to end up in a similar pattern.

This evolution has been shaped by the fact that general rules that span the identifying space make better sense from the point of view of simplicity and efficiency. Several common types of rule have emerged. For example, there will typically be a rule that all identifiers in the space identify an instance of some type or types, this is the identified type. For example, the International Committee on Systematics of Prokaryotes regulates binomial names that identify prokaryotes – so this is the identifying type. Sometimes the names of nomenclatures can be false friends; ISO 4217 may be entitled Currency Codes, but it includes precious metals, such as gold and platinum, which are not normally regarded as currencies. Typically, there are common rules across the identifying space. These specify the syntactic algorithm used to identify whether tokens are of the same type. This is often a simple orthographic rule – same characters. Though what sameness means needs to be made clear; for example, whether it is sensitive to differences in letter case or font.

In the ideal situation, the physical creation of the inscription also creates the identifier. In the case of scientific nomenclatures, the date the species name (in a correct form) appears in a suitable publication marks the ‘birth’ of the species name (type) – which, in turn, baptises the set of inscriptions that appear in various copies of the publication as species name tokens. In a computer system, the successful input of the identifier – its acceptance by the system – marks again the (intentional) creation of the identifier and that the stored identifier inscriptions belong to it.

Though there would appear to be a mutual dependence between the coming into existence of inscription(s) and their identifier, this is not necessary. Botanical classification allows for cases where a species name-type can be accepted without there being a corresponding inscription. Instead a formula is used, where the name of the genus is reused giving a repeated name; for example, *Magnolia magnolia* – known as an autonym.

There is a similar kind of common algorithmic process that plays an important role in many nomenclatures – where identifiers are built out of components that include other identifiers. One way of describing this is to say that the component identifier occurs in the composite identifier. For example, the genus name '*Falco*' occurs in the species name '*Falco buteo*' and the ISO Country Code 'GB' occurs in the ISO Currency Code 'GBP' (as shown in **Fig. 3**). There can be quite complex composition hierarchies, with components within components – for example, see the asset tags formats in [3].

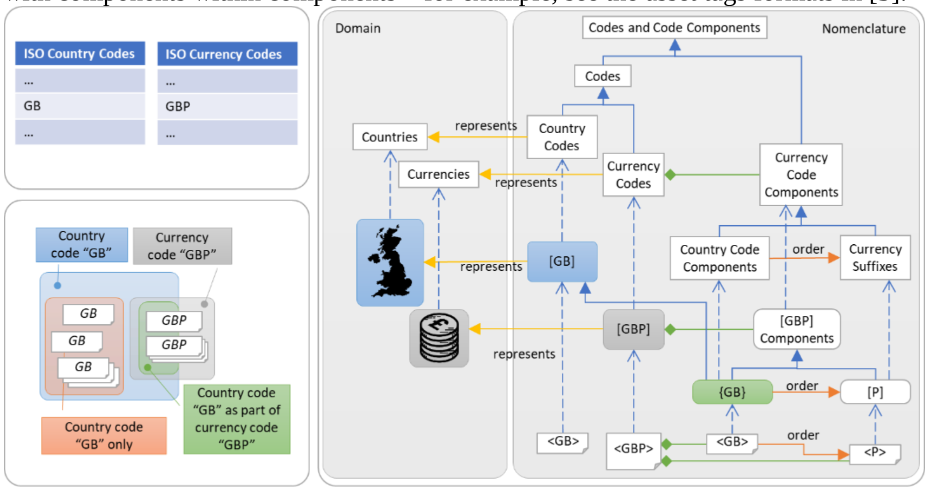

Fig. 3. Example and Model of Occurrences

This apparently simple process raises a problem when one tries to tie down its ontology. It turns out we need to explain how, at the type level, one and the same identifier can occur multiple times. Consider the SWIFT Bank Codes (defined by ISO 9362) where the fifth and sixth characters are the ISO 3166-1 alpha-2 country code. Consider two SWIFT Codes (identifier - word-types) in which the ISO code 'GB' occurs, one would say that there are two occurrences of the single GB code (word-type). These two occurrences are not identical to the one word-type. Neither are they word-tokens as there can be multiple inscriptions of the same SWIFT code with the same GB occurrence. What are these occurrences?

A more general version of this problem is mentioned in Quine [22, p. 297] and taken up by Simons [23] – who notes similarities to structural universals [24]. It is further discussed by, among others, [25], [26] and more recently in terms of form [27].

Wetzel [26] raises several useful points (it also provides an analysis of occurrences using sequences). She notes occurrence is transitive: if *a* occurs in *b* and *b* occurs in *c*, then *a* occurs in *c*. The obvious general case is that characters occur in words and words occur in sentences. This raises the question of whether and how *a*'s occurrence in *b* is a different occurrence from *a*'s occurrence in *c*, even if the same item, *a*, is involved. She also notes that it is not clear that the occurrence of a word-type is a word-type and that it might make good philosophical sense to distinguish word-types from occurrences

– extending the *type-token distinction* to *type-token-occurrence*. We take note and add a shorthand for representing occurrences in models: curly brackets to frame an occurrence – e.g. {USD}.

In the extensional ontology adopted here, occurrences in identifiers are the sub-sets of the identifiers that occur as components in a specific order. As **Fig. 3** shows, if one considers the identifier GBP, then for each of its instances, take the initial GB part and make these into a set then this is a subset of the identifier GB (which is a set of GB inscriptions). This set is not an identifier, and so does not directly identify anything. Then there is a sub-set of the character P, which has as members those characters that are in the p-position of GBP. Instances of the first set are ordered before instances of the second set.

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 show graphically later in **Fig. 5**); the actual real identifier inscriptions (a part of the domain, and not some representation of them) are stored and processed in the concrete ontology system.

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 [22, 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 "each whole quotation must be regarded as a single word or sign, whose parts count for no more than serifs or syllables. A quotation is not a description, but a hieroglyph; it designates its object not by describing it in terms of other objects, but by picturing it."

In a later paragraph he writes ''Boston' has six letters' as an example. As Kaplan [9] points out this count is incorrect if one counts character types instead of tokens – as there are two character-tokens of the single character-type 'o'. More generally this illustrates a problem with quotation as a paper tool, it does not make explicit whether it is being used to refer to the type or token.

While Quine is surely correct to distinguish between use and mention, his proposal for the use of quotation has been questioned. Tarski [28] and Church [29, chap. 8] examine the use of quotations in natural language and decide against using them. Davidson [30] also examines the problems, many 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. So - 'Boston' is disyllabic – should be read - the expression a token of which is here, Boston, is disyllabic. 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 or hieroglyphs. 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. This may seem plausible if one considers individual speech acts, but if one considers a (paper) reference copy of the Linnaean nomenclature, it is not clear that each identifier inscription (token) on its pages is either exclusively as a use or mention. Strawson [6] when discussing identifying and reidentifying (see above) remarks that it is a particular speaker or hearer that does this work. Davidson [30] also suggests that one could both use and mention at the same time. Perhaps something similar is happening with the Linnaean inscriptions; that the tokens are paper tools that one can either use or mention as required. The lesson here seems to be that one cannot usefully assume that the token in the published nomenclatures are exclusively uses or mentions. 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. 4** (Identifiers as Attributes and Identifiers as Objects), neither of which resemble the 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.

A likely driver for the different patterns of nomenclature is identifying space volatility. Where change is unlikely, the Identifiers as Attributes pattern has the advantage of being simple and it is unlikely that the identifying space attributes baked into the schema will need to be changed. Where change is likely, the Identifiers as Objects pattern, though less simple, has the advantage that identifying spaces are objects and can be added without any schema change.

In many ways these are a direct implementation of the nomenclature tables (paper tools). Just as tables divide the header rows from the rest of the rows, so there is a division here between schema-level identifiers (such as Countries) and the data-level identifiers. Typically, in the implementation, these are under different governance. For simplicity, at this stage we ignore the schema level identifiers.

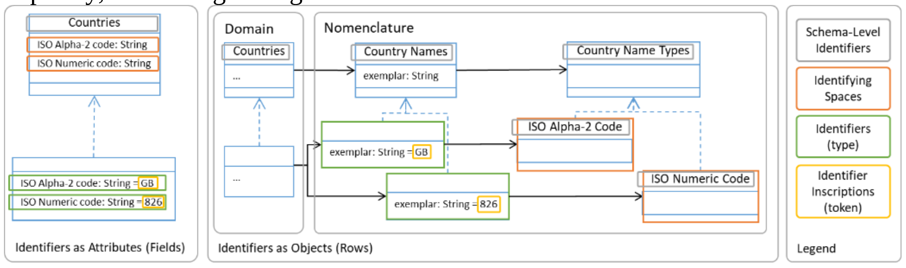

Fig. 4. Patterns of Current Identifier Implementation for an Entity

In **Fig. 4**, 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 [31]; 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 in **Fig. 2** and **Fig. 3** (unlike **Fig. 4**) 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

Fig. 4 makes clear that the system contains 'real' identifier inscriptions (tokens). These play a critical role in the operation of the system as they are specimens of their types, used to identify and reidentify tokens of the same type. As the nomenclature ontology (shown in **Fig. 2** and **Fig. 3**) does not, it 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 a 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, as shown in **Fig. 2** and **Fig. 3**, 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 to firstly 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 non-token picture uses of the inscriptions in **Fig. 2** and **Fig. 3**, such as <GB> and [GB], can then be seen as labels for bare nodes, artefacts generated by the view – not a representation of what is actually stored. 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. 5**, 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.

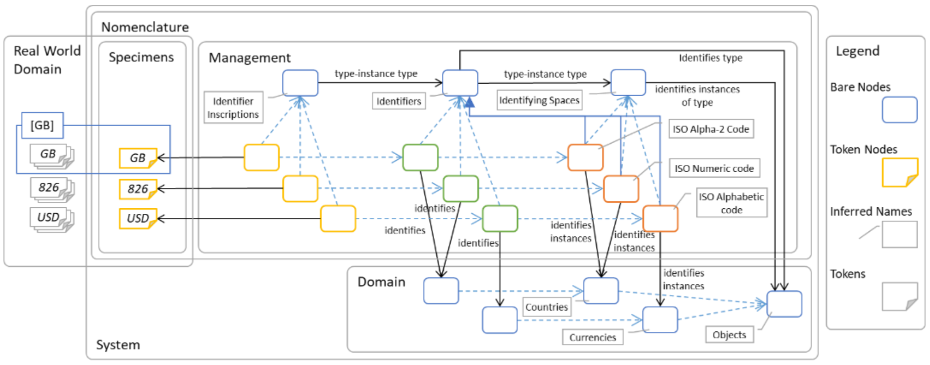

Fig. 5. Proposed Implementation Structure

In this approach, all the nomenclature management is handled the same way – there is no schema-data names distinction as in **Fig. 4**. And 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 devises 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, a similar approach (in one respect) can be found in the semantic network SNePS, [32, 33], where inscriptions are a special kind of thick node joined to the rest of the semantic network of thin bare nodes by a special type of link, the LEX arc. (See also [34]). Schapiro [33] rightly compares the resulting separation of thick (token) and thin (bare) nodes with Carnap's [35, sec. 14] example of structural definite descriptions; where names are removed from a map of the Eurasian railroad network to reveal its bare underlying structure. However, the thick inscription nodes are the full extent of the SNePS semantic network's nomenclature infrastructure. These are held at arm's length via the LEX arc, without access to the resources of the bare nodes, making them second class citizens.

6 Conclusions

We believe that the grounding for an ontology of enterprise computing nomenclature within a foundational 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 the nomenclature is (and so its requirements). Then we looked at the key technical issues specifying such an ontology face. We focused on the distinction between type (identifiers) and token (inscriptions) and, within this, the difficulties in specifying what types are. Then, we provided an example of a nomenclature ontology that explicitly makes the distinction between inscriptions, identifiers and identifying spaces and specifies exactly what their identity criteria are. It also clearly distinguishes between the domain and the nomenclature, while allowing for there to be recursive nomenclatures of nomenclatures. We introduce the notion of an occurrence, its distinction from types and the difficulties in specifying what they are – and how the ontology deals with them. Finally, we provided an example of an implementation of a nomenclature ontology. We identified the treatment of tokens as a major issue, illustrating this with the analysis of the proposed use-mention distinction – and showed how our solution avoided it.

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 as way to envisage 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 would like 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 (1922).
3. NORSOK: Z-DP-002: Design Principles - Coding System. (1995).
4. 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 Biology. 15, e2001414 (2017).
5. Tipton, K. et al.: History of The Enzyme Nomenclature System. Bioinformatics. 16, 34–40 (2000).
6. Strawson, P.F.: Individuals. Routledge (1959).
7. Peirce, C.S.: Collected Papers of Charles Sanders Peirce. Harvard University Press (1932).
8. McArthur, T.: The Oxford Companion to The English Language. (1992).

9. Kaplan, D.: Words. Proceedings of the Aristotelian Society, Supplementary Volumes. 64, 93–119 (1990).
10. Wetzel, L.: Types and Tokens. In: Edward N. Zalta (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University (2018).
11. de Cesare, S. et al.: BORO as a Foundation to Enterprise Ontology. Journal of Information Systems. 30, 83–112 (2016).
12. Partridge, C.: LADSEB-CNR - Technical report 06/02 - Note: A Couple of Meta-Ontological Choices for Ontological Architectures. (2002).
13. Goodman, N. et al.: Steps Toward a Constructive Nominalism. The Journal of Symbolic Logic. 12, 105–122 (1947).
14. Ramsey, F.P.: Foundations of Mathematics and Other Logical Essays. Routledge (1931).
15. Whitehead, A.N. et al.: Principia Mathematica, to *56. Cambridge University Press (1925).
16. Quine, W.V.: Quiddities: an Intermittently Philosophical Dictionary. Harvard University Press (1987).
17. Haack, S.: Philosophy of Logics. Cambridge University Press (1978).
18. Hugly, P. et al.: Expressions and tokens. Analysis. 41, 181–187 (1981).
19. Ayer, A.J.: Language, Truth and Logic. Courier Corporation (1946).
20. Partridge, C. et al.: Formalization of the Classification Pattern: Survey of Classification Modeling in Information Systems Engineering. Software & Systems Modeling. 1–37 (2016).
21. Partridge, C. et al.: Developing an ontological sandbox: investigating multi-level modelling's possible Metaphysical Structures. MULTI-4th International Workshop on Multi-Level Modelling. 2019, 226–234 (2017).
22. Quine, W.: Mathematical logic. Harvard University Press (1940).
23. Simons, P.M.: Token Resistance. Analysis. 42, 195–203 (1982).
24. Lewis, D.: Against Structural Universals. Australasian Journal of Philosophy. 64, 25–46 (1986).
25. Wetzel, L.: What Are Occurrences of Expressions? Journal of Philosophical Logic. 22, 215–219 (1993).
26. Wetzel, L.: Types and Tokens: On Abstract Objects. MIT Press (2009).
27. Fine, K.: Form. Journal of Philosophy. 114, 509–535 (2017).
28. Tarski, A.: The Concept of Truth in Formalized Languages. Logic, Semantics, metamathematics. 2, 152–278 (1956).
29. Church, A.: Introduction to Mathematical Logic. Princeton University Press (1996).
30. Davidson, D.: Quotation. Theory and Decision. 11, 27–40 (1979).
31. Strachey, C.: Fundamental Concepts in Programming Languages. Higher-order and Symbolic Computation. 13, 11–49 (2000).
32. Maida, A.S. et al.: Intensional Concepts in Propositional Semantic Networks. Cognitive Science. 6, 291–330 (1982).
33. Shapiro, S.C. et al.: SNePS Considered as a Fully Intensional Propositional Semantic Network. The Knowledge Frontier. pp. 262–315. Springer (1987).
34. Partridge, C.: Business Objects: Re - Engineering for Re - Use. Butterworth Heinemann (1996).
35. Carnap, R.: The Logical Structure of the World: Pseudoproblems in Philosophy. University of California Press (1967).